IMPLEMENTATION OF A MECHANISTIC-EMPIRICAL PAVEMENT DESIGN METHOD FOR URUGUAYAN ROADWAYS

Martín Scavone Lasalle

A thesis submitted to the faculty of the Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
IN
CIVIL ENGINEERING

Gerardo W. Flintsch.
Samer Katicha.
Linbing Wang.

May 10th, 2019
Blacksburg, Virginia

Keywords: Mechanistic-Empirical, Flexible, Pavement, Design

Implementation Of A Mechanistic-Empirical Pavement Design Method For Uruguayan Roadways

Martín Scavone Lasalle

Abstract

Mechanistic-Empirical (M-E) methods are the cornerstone of current pavement engineering practice because of their enhanced predicting capabilities. Such predicting power demands richer input data, computational power, and calibration of the empirical components against distress measurements in the field.

In an effort to spearhead the transition to M-E design in Uruguay, the aim of this Project is twofold: **(1) develop an open-source, MEPDG-based, simplified M-E tool for Uruguayan flexible pavements [Product-One], and (2) compile a library of Uruguayan input data for design [Product-Two].**

A functional, Matlab-based beta version of Product-One with default calibration parameters and a first collection of Uruguayan input data are presented herein. The Product-One beta is capable of designing hot-mix asphalt (HMA) structures over granular bases on top of the subgrade. Product-Two features climate information from the INIA weather station network, traffic distribution patterns for select Uruguayan highways, standard-based (Level-3) HMA properties, and Level-3 and Level-2 unbound materials' parameters. Product-One's outcomes were against other available M-E software, as a means to test the code's performance: Product-One reported a distress growth similar to CR-ME (MEPDG-based) on default calibration parameters but different to MeDiNa (calibrated core).

In conclusion, Product-One managed to perform like another MEPDG-based software under the same design inputs and constraints, accomplishing one of this Thesis' objectives. However, Product-Two could not be created to the initially-desired extent. Nevertheless, the author remains confident that significant leaps forward can be made with little extra effort and further research on M-E design can be encouraged from this project.

Implementation Of A Mechanistic-Empirical Pavement Design Method For Uruguayan Roadways

Martín Scavone Lasalle

General Audience Abstract

The design of pavement structures historically relied on methodologies developed after experimental results, the so-called “empirical methods”. Advances in technology over the recent years allowed for more complex but more reliable methods – the mechanistic-empirical (M-E) methods – to finally be adopted by practitioners both in the US and abroad.

In an effort to encourage the transition to M-E design in Uruguay, this project aimed at developing an open-source M-E design tool for Uruguayan flexible pavements based on the American MEPDG design methodology [Product-One], and assembling a library of Uruguayan data necessary for design with such an M-E method [Product-Two].

In this project, a beta version of the Product-One design tool for the design of asphalt-surfaced pavements and a collection of climate, traffic distribution, and materials’ properties data from Uruguayan sources for design is presented (load information was not available for this project); this Thesis is the log of the data collection effort as well as the guide to using and understanding all the components of Product-One. In addition, Product-One has been tested by comparing its pavement design results for a given Uruguayan highway against other M-E design software tools: MeDiNa and CR-ME. Product-One’s outcomes resembled the results given by CR-ME (also MEPDG-based) but differed with those from MeDiNa (crafted specifically for design of Brazilian roads).

In conclusion, Product-One managed to perform like another MEPDG-based design tool under the same inputs and constraints, accomplishing one of this Thesis’ objectives. However, some of the Uruguayan information sought for could not be retrieved and so added to Product-Two. Anyway, the author remains confident that both Products can be significantly improved with little extra effort and that this project may encourage further research on M-E design.

Acknowledgments

I wish to express his truest gratitude to the following people and organizations, who contributed either directly or indirectly to shaping this Project and bringing it to completion: Virginia Polytechnic Institute and State University (Virginia Tech), for receiving me in their truest Hokie spirit and letting me become a member of the Hokie Community; the Fulbright Commission, the *Asociación Nacional de Investigación e Innovación,* the Institute of International Education, Virginia Tech's Civil Engineering department and the Virginia Tech Transportation Institute [VTTI], the staff and leaders of these Entities, for granting me the possibility of pursuing my Master's Degree at Virginia Tech turning such a dream-like prospect into reality; to Dr. Gerardo Flintsch, my Academic Advisor, and my Thesis Committee Members, Dr. Linbing Wang and Dr. Samer Katicha, for their most valuable insight and feedback as this Project was underway, and for their help when progress halted because of challenging issues; to my colleagues at VTTI's Center for Sustainable Transportation Infrastructure, for their feedback as I made progress on this project; to Mrs. Guadalupe Tiscornia and Mr. Carlos Schiavi [INIA, Uruguay] and to Mrs. Jimena Alonso [IMFIA-Fing-UdelaR] for their contribution on meteorological data for the input data library; to Ms. Paola Rodriguez and Mrs. Adriana Varela [DNV MTOP], and Mr. José Calviño and Mr. Agustín Casares [CSI] for furnishing the Uruguayan traffic data that has been used in this Project; to Mrs Patricia Enrich, Ms. Paula Morales [DNV MTOP], Mr. Santiago Kröger [Bitafal], Mr. Aníbal Bentos and Mr. David Fontáns [Grinor] for their insight on material characterization and raw lab test data contributions to the input data library; to Dr. Barry J. Dempsey [UIUC] for his most valuable insight and advice during the early stages of the development of Product-One's climate effects model; to Dr. Samer Katicha [VTTI], Dr. Felipe Franco [UFRJ – Brazil], and to Mr. Piero Laurent [Lanamme-UCR – Costa Rica] for their contributions to the programming and testing of Product-One's core MLE calculator and for providing access to the M-E software suites used during Product-One's beta testing.

Yet last but not least, I am most thankful to my family, all my friends and beloved ones (fortunately a large bunch of people), and to Nati, My Dear, for their continuous support and encouragement (either from close by or over the distance from long miles away) to set sail and complete my Master's Degree since Day One. To you all, THANK YOU SO MUCH, IT WAS TOTALLY WORTH IT!

Martin S.; April 25th, 2019; Go Hokies!

TABLE OF CONTENTS

APPENDICES

LIST OF FIGURES

LIST OF TABLES

IMPLEMENTATION OF A MECHANISTIC-EMPIRICAL PAVEMENT DESIGN METHOD FOR URUGUAYAN ROADWAYS

CHAPTER 1—INTRODUCTION: PROBLEM STATEMENT AND RESEARCH OBJECTIVE

1.1 Background

Mechanistic-Empirical (M-E) pavement design methods rely on the combination of theoretical engineering mechanics and models developed from field or laboratory tests to predict how a pavement structure would perform under the application of traffic loads and how it would be affected by climatic and environmental external factors (AASHTO, 1993). For this proposal, the term 'pavement performance' refers to the degree of distress that may develop throughout the pavement structure as a consequence of its response to the application of loads – as defined by AASHTO (1993).

AASHTO (1993) and the NCHRP Synthesis 457 (Pierce and McGovern, 2014) point out that the use of M-E methods in pavement design will deliver increased design reliability; such a claim is based on three key features of these design methods:

- Since the M-E design methods rely on mechanics of materials' models, these are applicable to design scenarios beyond the models' initial calibration/verification efforts. This allows modeling a wider range of load patterns, climatic effects, and material types than those used in the model development phase.
- M-E models provide a more accurate prediction of the pavement response to applied load than a fully empirical model.
- M-E models are able to predict how distresses would develop and grow within the pavement structure.

The enhanced design reliability of M-E models may, however, come at a cost. For the user to obtain the most accurate prediction of local pavements performance, the models must be calibrated and validated against actual pavement performance measurements. In addition, these models require inputs representative of the local traffic, climate, soils, and materials' properties to produce reliable design results (NCHRP, 2004; Huang, 2004). The retrieval and processing of these data can become a cumbersome task, as pointed out by Li et al. (2011).

Nevertheless, (NCHRP, 2004) highlights that the economic savings accruing from more reliable designs obtained through an M-E method (especially when designing heavily-trafficked highways) would pay off the additional investment in model calibration and local input characterization. Therefore, because of these many advantages, the Mechanistic-Empirical Pavement Design Guide (MEPDG) software was developed under NCHRP Project 1-37A, and launched in 2004 (NCHRP, 2004).

The MEPDG is a comprehensive M-E pavement design method aimed at becoming standard practice in pavement design in the United States (NCHRP, 2004). This design model is capable of performing the analysis of a broad range of pavement structures (flexible and rigid, new and rehabilitated, and composed of different types of materials), subject to the application of different types of loads and the effect of diverse climate and subgrade conditions. As of 2014, 49 road agencies located in the US, Puerto Rico, and Canada either adopted the model or were migrating towards its adoption as standard practice in pavement design (Pierce and McGovern, 2014).

Contrary to previous models, such as the AASHTO ’93 model (AASHTO, 1993), which determined the minimum thickness of each pavement layer to withstand a given traffic level, under certain subgrade, and pavement materials; in the MEPDG the user must provide a trial pavement structure (detailing materials and layer thicknesses), along with the inputs for the other design methods (e.g. climate, traffic volume and load, and existing pavement condition). Furthermore, the user must set the maximum acceptable distress level for the projected structure and desired reliability level. Once all the required inputs are provided, the trial design is assessed and the user must either accept the trial design if the predicted distress level after the design period complies with the design constraints or modify it if not. (Pierce and McGovern, 2014).

Table 1 (AASHTO, 2008; Pierce and McGovern, 2014), summarizes the flexible and rigid pavement distresses the MEPDG is capable of modeling:

Table 1: Pavement Distresses Modeled by MEPDG

Asphalt Pavements	Concrete Pavements
Rut depth total, asphalt, unbound aggregate layers, and subgrade (inches).	Transverse cracking (JPCP) (percent slabs).
Transverse (thermal) cracking (non-load-related) (feet/mile).	Mean joint faulting (JPCP[1]) (inches).
Alligator (bottom-up fatigue) cracking (percent lane area).	Punchouts (CRCP) (number per mile).
Longitudinal cracking (top-down) (feet/mile).	IRI predicted based on other distresses (JPCP and CRCP) (inches/mile).
Reflective cracking of asphalt overlays over asphalt, semi-rigid, composite, and concrete pavements (percent lane area).	
IRI predicted based on other distresses (inches/mile).	

Source: AASHTO (2008).

Additionally, the MEPDG defines three levels of hierarchy for traffic, climate, and materials' data inputs, which relate to the accuracy of the design results. The user may select a lower or higher hierarchy depending on the importance of the project and the difficulty in obtaining high-level input data. The three levels of input are defined as follows (NCHRP, 2004):

Level 1 inputs: Data inputs for high-reliability designs. These inputs would most accurately represent traffic features, climate records, and materials' properties. Level-1 materials and subgrade parameters are obtained through direct laboratory testing of samples of the actual materials to be utilized in the pavement project, while traffic weight and volume data are obtained from on-site data collection.

Level 2 inputs: Design inputs at this level allow for a fair level of reliability and are often obtained from local historical databases. Level-2 material properties inputs are often obtained from correlations with other properties that the involved local agency is capable of measuring either in a laboratory or in the field.

Level 3 inputs: Recommended only for design where a high level of design reliability is not a concern. This level uses broad average values for material properties -such as default values often found in the literature.

[1] JPCP = jointed plain concrete pavements; CRCP = continuously reinforced concrete pavements .

1.1.1 Review of MEPDG calibration efforts

The first version of the MEPDG contained a series of default values for materials' properties and calibrated model coefficients which provide a nation-wide Level 3 input dataset for the United States, obtained from field measurements taken as part of the Long Term Pavement Performance (LTPP) program. Nonetheless, each roadway agency that uses the MEPDG software is encouraged to pursue local model calibration and deploy the capabilities needed to produce Level 1 and 2 inputs (NCHRP, 2004). AASHTO (2010) provides the framework to conduct the calibration, which can be carried out using newly constructed pavement sections or historical data from a pavement management system (FHWA, 2010).

Many efforts have been undertaken to calibrate the built-in pavement distress models to obtain distress predictions that resemble what actually would happen under given traffic, load, climate, materials, and construction practices. Kim et al. (2014) and Abu-Sufian (2016) provide comprehensive lists of research studies conducted throughout the US and Canada toward the calibration of the MEPDG distress models. The map in Figure 1 summarizes these lists and reflects the research work done more recently in this subject, showing the US states for which published results exist and are available for use by practitioners.

California, Kentucky, Illinois, and Minnesota utilize locally-developed M-E tools for the design of flexible pavements. Indiana adopted the MEPDG for professional practice (Pierce and McGovern, 2014). Harvey and Bansheer (2011) point out that the MEPDG had been tested for use in California and recommended to use it only for the design of concrete pavements only and to use CalME (Ullditz et al., 2010) for the design of flexible pavements. Tarefder and Rodriguez-Ruiz (2013) reported a calibration effort for the MEPDG to be used for the design of flexible pavements in New Mexico; Oh and Fernando (2008) published design charts based on the MEPDG and calibrated for Florida local conditions; Haider et al (2016) undertook calibration of the MEPDG pavements for Michigan conditions; Smith and Nair (2015) calibrated the distress equations for flexible and rigid pavements to be used in Virginia; Wu et al. (2014) conducted a similar calibration effort for rigid pavement structures to be designed in Louisiana; and, recently, Muftah et al. (2019) reported their efforts in calibrating the MEPDG distress models for flexible pavements in Idaho.

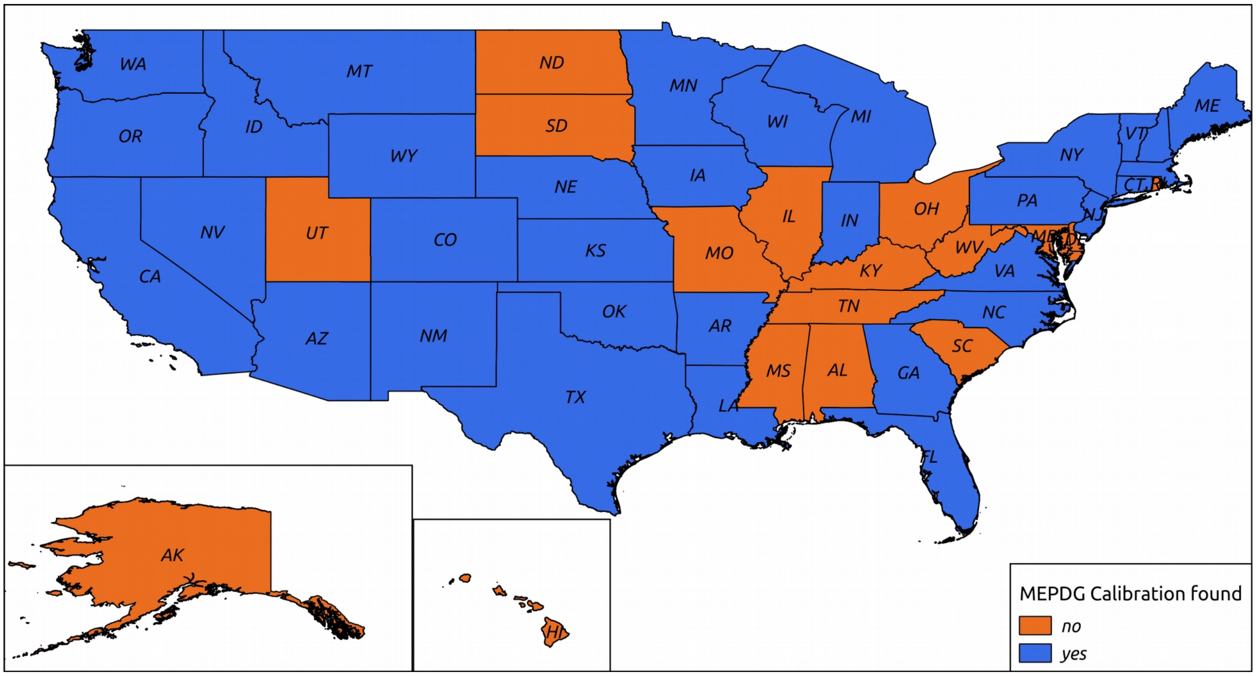


Figure 1: Availability of MEPDG distress model calibration efforts by state

Map data source: gadm.org.

Kim et al. (2014) found that the predicting capability of the MEPDG distress methods for both flexible and rigid pavements can be improved by means of calibration. However, the calibration often does not guarantee that the distress level predicted with the enhanced model would match the degree of distress measured in pavement sections used during validation, such a case was reported by Kim et al. (2011) when calibrating the rutting and fatigue cracking models for flexible pavements in North Carolina.

1.1.2 Use of the MEPDG or other M-E design methods in other countries

An international user who wishes to adopt the MEPDG to their local conditions must go beyond the calibration of the distress models. The user must devise a means to sort the design data inputs (traffic, climate, and materials' inputs) in a manner compatible with the MEPDG. For instance, the traffic data must be converted from the local customary classification to the FHWA vehicle classification. Similarly, climate data drawn from local records must be stored in a specific file format, that of US-based weather stations.

Following are some examples of calibration efforts conducted outside the US:

- Cordo et al. (2008), and Bustos et al. (2011) calibrated the MEPDG JPCP distress models for central Argentina. The authors collected weather information from the surroundings of the test sections from the Argentinean National weather service and devised a computer software to convert those weather records to a MEPDG-compatible format. Additionally, they developed a table to turn their traffic inputs (classified in the 24 standard categories defined by the Argentinean Highways Bureau) to the 13 FHWA categories. The authors successfully calculated calibration parameters for the MEPDG's joint faulting and slab cracking models, but they claim that further validation efforts were needed.

- Delgadillo et al. (2011) published a trial run of the MEPDG for the design of HMA and JPCP and calibration of the distress equations under typical Chilean conditions. The retrieval of the necessary traffic and material input data was straightforward, as the authors obtained Level-1 traffic inputs and Level-2 HMA mix properties inputs. However, the authors found that obtaining Chilean weather data (from weather stations located at airports) and importing them to the MEPDG proved arduous. Since the tested version of the MEPDG would not accept weather stations outside the United States, the MEPDG's file with the record from an US weather station had to be replaced with local data before launching the software.
- In Brazil, the design of flexible pavements with M-E models received a major boost with the launch of SisPavBR (Franco, 2007), which now evolved as the MeDiNa software package (released 2018) [2]. The core motivation behind the software development was merging the recent breakthroughs in pavement modeling and asphalt material characterization into a user-friendly pavement design solution that would replace the empirical DNER 66 methodology (DNIT, 2006; Franco, 2007'; Dambros Fernandes, 2016). Besides asking the user to provide trial pavement structures in an iterative manner until all the design constraints are met, SisPavBR also performs iterative designs automatically – the user must input the acceptable distress thresholds (namely extent of rutting, alligator cracking, and permanent deformation of the subgrade). In addition, SisPavBR (and its successor, MeDiNa), feature a library of physical and strength properties of bituminous and unbound pavement materials and Brazilian soils, which were retrieved from archived laboratory test results, and a built-in meteorological records database, whose data has been retrieved from the Brazilian weather service (Franco, 2007). Franco (2007) highlighted that SisPavBR produced results similar to the Shell flexible pavement design method, but could not compare it withthe MEPDG because of the difficulties encountered when attempting to load Brazilian inputs into the software. However, Dambros Fernandes (2016) compared these two methods in terms of final thickness for a given stretch of Brazilian highway and found that the

[2] The MeDiNa package can be downloaded from: http://ipr.dnit.gov.br/medina.png/view

final pavement thicknesses reported by the MEPDG were twice the values given by the SisPavBR. The researcher hypothesized that such difference may be due to the SisPavBR distress models being calibrated to the Brazilian environment and materials.

- Caliendo (2012) published a trial calibration of the MEPDG fatigue-cracking model for Italian flexible-pavements. His research consisted of adjusting the model's calibration coefficients based on the distress growth predictions given by other models known to represent correctly how Italian pavements deteriorate. Additionally, the author compared the design results obtained with the MEPDG and the AASHTO (1993) method for new HMA surfaces with typical base and sub-base materials. However, the author had to devise tools to convert the Italian traffic counts and to add a weather record similar to Italian weather. The author searched for the US-based station that best resembled best the Italian weather, arriving at the conclusion that a station in Alabama is the most suitable one.
- The National Materials and Structural Models Laboratory (LanammeUCR) at the *Universidad de Costa Rica* developed and released a local M-E pavement design tool – CR-ME[3] (Trejos-Castillo et al., 2016).,This software has been tailored for the design of both flexible and rigid structures with a mechanistic-empirical methodology and considers the local climate, traffic, and available paving materials (Loría Salazar, 2013). The authors decided to develop a local M-E design guide from scratch because of difficulties faced when attempting to utilize the MEPDG for design purposes. Problems included obtaining the computer software itself, characterizing Costa Rica's tropical climate and heavy vehicle traffic, and accurately modeling the dynamic behavior of the locally available materials.

Chang Albitres et al. (2013) conducted a survey among Latin American professional engineers, professors, and students to capture the extent of the use of the MEPDG for pavement design. The majority of the respondents (77.5%) acknowledged the existence of the design method and praised its extended capabilities compared to the widespread AASHTO (1993) and PCA (1984) methods. However, they have not implemented the MEPDG in their home countries. Only 12.5% of the respondents reported having used the MEPDG for design purposes.

[3] The CR-ME software suite for the design of flexible and rigid structures can be retrieved from the LANAMME's website: http://www.lanamme.ucr.ac.cr/index.php/centro-de-descarga/cr-me/descripci%C3%B3n-cr-me-flexibles.html

The authors highlight that the key limitations to the adoption of the MEPDG in Latin American countries are:

- Lack of insight on the underlying models,
- Distress equations' calibration not performed,
- Lack of material testing equipment,
- Difficulties to purchase the commercial software bundles in which the MEPDG is packed,
- Lack of hourly climate data to feed into the software,
- Lack of vehicle weight records, and
- The MEPDG not being yet accepted as a valid design procedure by the local road agencies.

1.2 Problem Statement

Based on the experiences described, implementation of the MEPDG – or a similar M-E pavement design procedure – in Uruguayan engineering practice would necessarily involve the following preliminary work:

- Compile a library of climate, traffic volume, load, soils' properties, and material strength parameters to provide the inputs the M-E design procedure requires to run. This inputs library may qualify as Level-2 inputs in MEPDG jargon, and it should contain the parameters that represent design conditions and pavement materials used in local engineering practice.
- Study the MEPDG to understand how the design inputs are loaded into the program, and how these are processed to obtain results. Also, an input conversion procedure must be developed to re-write the local design inputs (such as traffic and climate records) in a manner compatible with the MEPDG programming.
- Calibrate the distress models contained within the MEPDG – or any M-E substitute design software – to adapt them to the way local pavements deteriorate. The calibration procedure can be conducted using newly constructed test sections or historical records from pavement management systems.

1.3 Research Objective

The core objective of this Thesis is to develop an MEPDG-based Mechanistic-Empirical pavement design procedure valid for the design of pavements for Uruguayan roads. The effort is aimed at (1) compiling a library of level-2 inputs from Uruguayan sources (climate, traffic, and pavement materials), and (2) implementing a simplified MEPDG-based M-E procedure. This procedure follows the general MEPDG framework but disregards the submodels that simulate phenomena that do not occur on Uruguayan pavements. This simplified M-E model shall be capable of handling the aforementioned level-2 inputs as well as level-1 inputs from local sources (field or laboratory tests) and its output must be easily understandable by local practitioners.

1.4 Methodology

Work toward the completion of this Project has been distributed in five main tasks, a brief description of each follows:

1. **Data collection:** Information on climate, traffic, and pavement materials from Uruguayan Agencies and/or practitioners has been requested in order to develop a library of Level-2 inputs for future designs. Hourly climate records have been requested from INIA's[4] *Banco de Datos Agroclimático*, stored in the input library, and made available for the designer upon any execution of the M-E design procedure. The Directorate of Highways at the Uruguayan Ministry of Transportation and Public Works (*MTOP*) has been contacted in order to retrieve traffic information (average daily traffic, traffic distribution patterns, and heavy-vehicle load spectra at weighing stations), and an inquiry on data concerning properties of pavement aggregates and subgrade soils has been filed against local geotechnical engineers, laboratory managers, and contractors. The results of this task are documented in Chapter 2,
2. **MEPDG literature review:** The main scope of this activity has been twofold: First and foremost, study the MEPDG-related literature to find out which are the program's featured built-in models and how these operate. It is almost needless to highlight that the original MEPDG report (NCHRP, 2004) has been the departure point for this review task. Secondly, I would conduct an additional literature search for reports on distress models that were calibrated on Uruguayan roads. The research results have been distributed based on their parent topic over Chapter 3.
3. **Software development:** Develop a software package to run a simplified version of the M-E design procedure based on the MEPDG, able to receive the design inputs in a straightforward manner. The underlying design philosophy is to relieve the user from converting the design inputs to any other format and load them into the software as received. This simplified M-E design suite may incorporate the empirical pavement performance models found in point #2 (if any), while it

[4] INIA is the Spanish acronym for *Instituto Nacional de Investigación Agropecuaria* – the Uruguayan Institute for Agricultural Research.

will exclude those models within the MEPDG that represent phenomena that don't take place on Uruguayan roads (like subgrade frost heave or cold-temperature thermal cracking). A detailed description of this M-E design tool is available in Chapter 3.

4. **Software validation:** Perform a set of test runs of the M-E design tool, once coding is completed. These tests consisted of comparing its design results (using MEPDG default calibration parameters for the distress equations) against the AASHTO (1993) design method for flexible design – which Uruguayan practitioners use without any supplemental calibration parameters – and versus other available M-E software tools. The test run results are given in Chapter 4.

5. **Thesis report:** The final Thesis Report – this very document – has been written with the goal of both serving as a log all the effort that has been carried out to develop a functional beta version of the M-E design tool and its ancillary library of input values for design and becoming the software's user's guide. It has been laid out in five chapters:

   - This first chapter states the Project's objectives, the author's motivation to fulfill it, and what does he expect upon completion;
   - Chapters 2 through 4 elaborate on the research and development activities involved in this Project: The results of the data collection and reduction efforts are logged in Chapter 2, Chapter 3 is entirely devoted to the development of the M-E design tool, providing discussion on its components, what calculations are done within them, how do they interact with each other and with the end-user, and most importantly how the end-user carries a design on using this piece of software. Chapter 4 portrays the results of the comparison between this newly developed M-E software versus other commercially available similar packages and against its parent model predecessor, the AASHTO (1993) methodology.
   - Finally, Chapter 5 closes this report, furnishing a final summary of the Project results, evaluating to what extent were its objectives met, and what possible future work may be done to further improve this Project's products.

1.5 Results

Two main products are being hereby presented as results of this Thesis:

1.5.1 Product-One: Simplified MEPDG-based M-E design software

A computer program that runs the simplified M-E design procedure (from now on referred to as Product-One) has been written in Matlab language and is included as part of this Thesis. The program is compatible with both Matlab[5]® and GNU-Octave[6]. The program's user interface is spreadsheet-based.

Below is a succinct list of Product-One features:

- Product-One adopts the input level logic used by the MEPDG: both level 1 and level 2 inputs are accepted, the are provided in a companion inputs library (Product Two). It will also make use of the MEPDG data flow paradigm, in which the user provides a trial pavement structure to the design software, which is subject to the simulated traffic and climate effects. The user must then evaluate if the level of distress the software predicts at the end of the design period is adequate or whether the trial structure must be revised.
- Product-One loads the hourly climate inputs from the INIA weather stations database (figure 2) from the inputs library (level 2 input) and estimates the weather at the pavement project's location by inverse reciprocal distance method. Otherwise, the user may provide a weather record from the actual project's site.
- Concerning asphalt mixtures properties, Product-One accounts for the variability of their strength properties with temperature and frequency of load: for any asphalt concrete layer placed on the trial structure, Product-One calculates its dynamic modulus from its predicted temperature and load frequency. The temperature of the asphalt mixtures is computed from the estimated site's weather by combining the heat transfer model proposed by Solaimanian and Kennedy (1993) and the Bells equations (FHWA, 2000). The frequency of load is obtained from the traffic speed with the default MEPDG formula (NCHRP, 2004; Al Qadi et al., 2008).
- Granular materials and subgrade soils are characterized by their resilient modulus in soaked (saturated) state. The Witczak's formula, which has been programmed in the MEPDG climatic module (NCHRP, 2004), is used to compute the layers' resilient moduli on unsaturated

[5] Matlab® is a registered trademark of The Mathworks, Inc.

[6] Octave is Copyright © by John W. Eaton (1998-2019), and distributed under the terms of the GNU GPL License.

conditions. The degree of moisture of the granular layers over time is estimated using the theory of flow on porous materials and the rates of rainfall infiltration on cracked pavements provided by FHWA (1993).

- Traffic volume and load spectra analysis is conducted on the data obtained from the Uruguayan Ministry of Transportation and Public Works (MTOP). This task involves getting monthly, daily, and day/night distribution factors for all the Uruguayan vehicle categories (figure 3). Product-One also converts the input AADT records into axle counts by weight. This conversion code distributes the axle count into light-weight single axles (cars), 2-wheel heavy single axles, 4-wheel heavy single axles, 4-wheel tandem axles, 6-wheel tandem axles, 8-wheel tandem axles, and 12-wheel tridem axles; and sort the axles by weight.
- Pavement performance simulation in Product-One is conducted in a 12-hour cycle, observing day/night variations in traffic flow (for which day/night distribution factors are needed) and materials' behavior changes as a consequence of temperature and moisture variations.
- Pavement response analysis in Product-One consists of calculating stresses, strains, and settlements after the application of different load types. The MEPDG's simplification for level-2 materials properties' input conditions were also adopted in Product-One: All pavement materials are regarded as elastic and linear and they can be characterized by their resilient/dynamic modulus and Poisson coefficient (NCHRP, 2004). The procedure described by Huang (2004) and Caicedo (2019) to calculate stresses and strains in multi-layer linear elastic systems was programmed as a core component of Product-One.
- The MEPDG transfer functions and distress prediction equations were coded as is with the default US-wide calibration parameters. The underlying reason for such a decision is that the calibration of the distress models exceeds the scope of this Thesis project, but can anyway become the topic for future research. Also, should any other Uruguayan practitioner undertakes the calibration of the AASHTO MEPDG software packages, little additional effort may be required to obtain the calibration factors for Product-One. However, it must be highlighted that any transfer function to model distresses due to sub-zero temperature conditions has been disregarded from Product-One.

In an effort to further encourage the use of the M-E methods in design and/or to enable further research or improvements to this original version of Product-One – as pointed out by Chang Albitres et al. (2013), the program's source code to will be released for free distribution. *I believe that by leaving Product-One as* *open-source software further improvement of it could be carried out by others to an extent I may not be able to achieve.*

1.5.2 Product Two: Library of level-2 design inputs for M-E design

A library of level-2 and default level-3 inputs for Product-One has been bundled as part of this Thesis. This library of inputs would contain (but is not limited to) the following data:

- Hourly climate data from Uruguayan weather stations (air temperature, humidity, rainfall, solar radiation, wind speed) for the climate effects' models. Hourly weather data was requested from the INIA's *Banco de Datos Agroclimático,* which possesses records for eight weather stations (currently still in service) scattered throughout the Uruguayan territory (Figure 2).

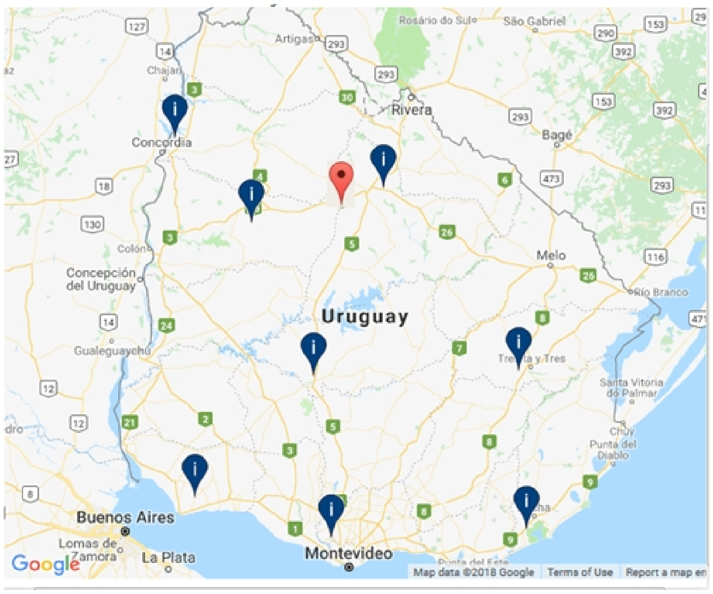
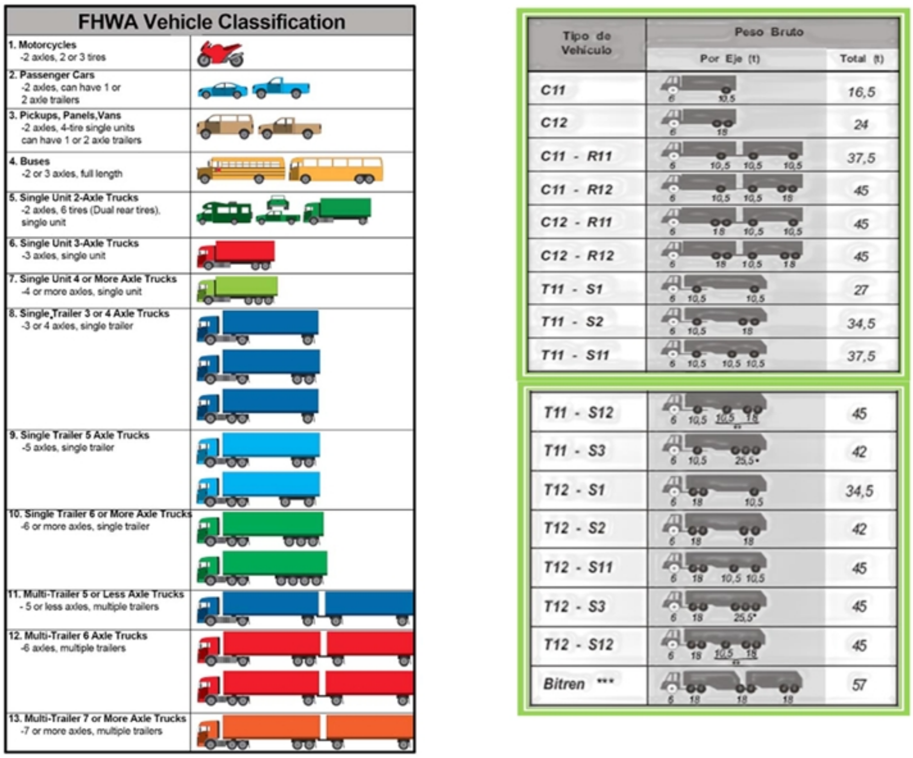


Figure 2: INIA weather stations

Source INIA website

- Traffic volume distribution patterns to convert AADT records to hourly vehicle volume. Monthly, daily, and hourly distribution factors for highways with different functional characteristics – commuter roads, roads subject to seasonal traffic, and mixed-traffic highways are available for each Uruguayan vehicle category and load level. These were retrieved from the MTOP records. Figure 3 shows the Uruguayan categories for heavy trucks plus their maximum legal load. The load categories used in Product-One are unloaded vehicle (less than 40% of maximum legal load), partially-loaded vehicle (40-80% of legal load), loaded vehicle (80-100% of legal load) and overloaded vehicle (over 100% of legal load).



FHWA Vehicle Classification
1. Motorcycles -2 axles, 2 or 3 tires
2. Passenger Cars -2 axles, can have 1 or 2 axle trailers
3. Pickups, Panels,Vans -2 axles, 4-tire single units can have 1 or 2 axle trailers
4. Buses -2 or 3 axles, full length
5. Single Unit 2-Axle Trucks -2 axles, 6 tires (Dual rear tires), single unit
6. Single Unit 3-Axle Trucks -3 axles, single unit
7. Single Unit 4 or More Axle Trucks -4 or more axles, single unit
8. Single Trailer 3 or 4 Axle Trucks -3 or 4 axles, single trailer
9. Single Trailer 5 Axle Trucks -5 axles, single trailer
10. Single Trailer 6 or More Axle Trucks -6 or more axles, single trailer
11. Multi-Trailer 5 or Less Axle Trucks - 5 or less axles, multiple trailers
12. Multi-Trailer 6 Axle Trucks -6 axles, multiple trailers
13. Multi-Trailer 7 or More Axle Trucks -7 or more axles, multiple trailers

Tipo de Vehículo	Peso Bruto	
	Por Eje (t)	Total (t)
C11	6 10,5	16,5
C12	6 18	24
C11 - R11	6 10,5 10,5 10,5	37,5
C11 - R12	6 10,5 10,5 18	45
C12 - R11	6 18 10,5 10,5	45
C12 - R12	6 18 10,5 18	45
T11 - S1	6 10,5 10,5	27
T11 - S2	6 10,5 18	34,5
T11 - S11	6 10,5 10,5 10,5	37,5
T11 - S12	6 10,5 10,5 18 **	45
T11 - S3	6 10,5 25,5*	42
T12 - S1	6 18 10,5	34,5
T12 - S2	6 18 18	42
T12 - S11	6 18 10,5 10,5	45
T12 - S3	6 18 25,5*	45
T12 - S12	6 18 10,5 18 **	45
Bitren ***	6 18 18 18	57

Figure 3: American and Uruguayan heavy vehicle categories.

Source ODOT (US categories), MTOP (Uruguayan categories)

- Collection of strength-related parameters for materials frequently used in Uruguayan practice. The Uruguayan construction and materials specifications (MTOP, 1990, 2003) and other sources (Waltham, 2002; FHWA, 2006) were reviewed and the different approved granular materials and asphalt concrete mixes were plugged into the library as default level-3 inputs.

1.6 Significance

The author expects that the development and release of Product-One as open-source software would enhance the use of M-E design procedures in local Uruguayan engineering practice, as it would be free from the licensing limitations common to commercial software which at times hindering the spreading of such methods (Chang Albitres et al., 2013). In addition, Matlab language interpreters and spreadsheet applications exist for almost any personal computer operating system, meaning that no hardware nor operating system limitation would constrain the use of Product-One.

Additionally, once Product-One gains users, demand for more detailed, richer input data would appear; and that would trigger further pavement-related research in Uruguay – e.g. conducting more detailed materials' analysis, or developing enhanced deterioration models from field measurements on local roads.

CHAPTER 2 — INPUT DATA COLLECTION AND REDUCTION

2.1 Introduction

The design of a pavement structure with a (M-E) method is strongly reliant on the input data (AASHTO, 2015). The higher the quality of the data inputs – or the more accurately these describe the behavior of any phenomenon affecting the performance of the pavement structure –, the more reliable the design process' results. According to AASHTO (2015), the inputs to the MEPDG can be distributed in six categories: general project information, design criteria, traffic, climate, structure layering, and material properties (including the design features). The collection of such design inputs can become a daunting task (Pierce and McGovern, 2014; Chang Albitres et al., 2013; AASHTO, 2015). Therefore, in order to simplify obtaining all necessary design inputs, Product-One includes a level-2 input data library (Product-Two) featuring data on Uruguayan climate, traffic, soils, and pavement materials. This chapter documents the development of the library by acquiring and reducing raw data from various local sources.

The following sections cover the various data collection tasks. The first section discusses the acquisition and reduction of climate data from Uruguayan weather stations. The second section covers the traffic-related data needs, namely traffic volume, distribution over time, axle characteristics, and weight records. The last section documents the implementation of the pavement materials' properties library compliant with Product-One data needs.

2.2 Climate data acquisition and reduction

Pavement modeling in the MEPDG requires records on select project site's weather variables in order to compute the climate-related effects on the pavement structure. More specifically speaking, the MEPDG requires air temperature, moisture, wind speed, percent sunshine, and rainfall records for such calculation, and the MEPDG's Enhanced Integrated Climate Module (EICM) module performs all the necessary preliminary calculations (NCHRP, 2004, Johanneck and Khazanovich, 2010). The EICM was developed by Lytton et al. (FHWA, 1993), and has been updated in 1997 (Larson and Dempsey, 1997) and in 1999. The MEPDG uses an even further-improved version of the EICM (NCHRP, 2004). Product-One uses local climate data and quantifies its effects on the pavement structure using some of the EICM's built-in models.

North American users of the MEPDG designing pavement structures in the US or Canada can make use of the MEPDG's built-in climate records database. This database includes data from over 850 weather stations scattered throughout these two countries and contains data spanning up to 10 years (NCHRP, 2004; Johanneck and Khazanovich, 2010). Conversely, non-American users have either to create a virtual weather station with their site's meteorological records (such as Bustos et al, 2011) or either select the US weather station that resembles the site's climate the most – as reported by Caliendo (2012).

2.2.1 Climate data availability

The MEPDG requires weather data be provided on an hourly basis (NCHRP 2004, Johanneck and Khazanovich, 2010). The following Uruguayan sources provide weather data with such aggregation level:

- The Uruguayan weather stations within the international ASOS network, which contains 14 weather stations located at Uruguayan airports and airfields. This database includes records of the five climate variables needed for pavement design[7].
- The INIA weather stations database[8], which contains records from the Institute's eight weather stations, spanning from as early as 2010 to present day. These stations provide hourly records of temperature, moisture, humidity, rainfall, and solar radiation.

Figure 4 compares the spread of these two networks over the Uruguayan territory.

For this Thesis, the INIA database was preferred over the ASOS network because it provides actual solar radiation data (input for the pavement temperature prediction module), even though fewer stations are comprised within this network. This relieves the user from estimating the amount of solar radiation at any given time from the percent sunshine records, e.g. the Baker and Haines equations, as done by one of the EICM components (FHWA, 1993). Additionally, this dataset complies with the EICM minimum length restriction – each eligible station must have at least 24 months of weather data records (Larson and Dempsey, 1997; Johanneck and Khazanovich, 2010).

[7] The ASOS network of weather stations can be queried from the Iowa Environmental Mesonet website, located in: http://mesonet.agron.iastate.edu/request/download.phtml?network=UY_ASOS

[8] The INIA weather record database can be accessed from: http://www.inia.uy/gras/Clima/Banco-datos-agroclimatico

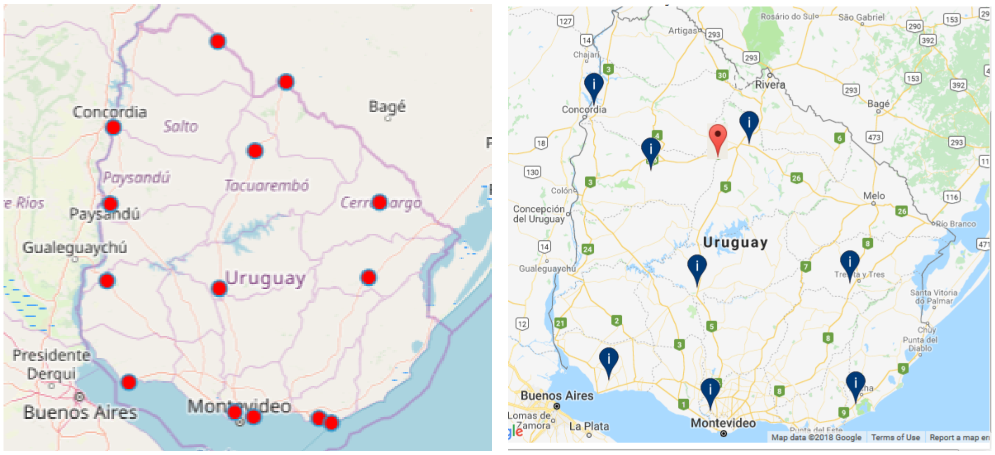


Figure 4: Weather stations within the ASOS and INIA networks

Sources: Iowa Environmental Mesonet (left), INIA (right)

2.2.2 Climate data acquisition and reduction

The raw records from INIA's eight weather stations were retrieved and reduced to match a predefined uniform template. The retrieved data spans from 2010 to 2018 and includes the five variables of interest (namely air temperature, humidity, wind speed, solar radiation, and rainfall) have been kept. Unfortunately, not all stations had full datasets (Appendix A).

These meteorological data were reduced to a standard layout template: a 10-column table arrangement, the first four columns reserved for each entry's timestamp, and the actual weather variables stored following a fixed order – as shown in table 2. The data were subject to a basic quality control procedure in order to remove outliers and foul records. For instance, some of the oddities that were removed from the reduced time series are the following:

- Humidity values greater than 100% (these were replaced by values equal to 100%)
- Wind speed records greater than 150 km/h [41.5 m/sec]. According to Severova (1997), wind speeds above 100 km/h are tied to rare extreme events, so these should only appear seldom on recorded data, and such a speed reading on a windless day (Figure 6) should be regarded as a foul reading (WMO, 2011).
- Temperature records greater than 45 degrees Celsius. The maximum temperature reading (50-year-long record) is 44 degrees Celsius (111 F), allowing to disregard records from greater values.
- Solar radiation records greater than 1394 Watt/m^2 (maximum solar radiation that reaches the Earth, excludes losses due to the planet's atmosphere, mentioned by Solaimanian and Kennedy, 1993).

- Negative values for rainfall, humidity, wind speed, and solar radiation, as such values lack physical meaning (WMO, 2011), and can therefore only be tied to either instrument malfunction or lack of record.

Figures 5 through 7 portray examples of anomalous weather records in context, that is plotted over time, doubtful records can so be detected with ease. Once detected, two different treatments were applied over such faulty values:

- If a single foul value was detected amid correct (or reasonable) values, the foul value was replaced by a linearly interpolated estimation computed with its nearest neighbors (WMO, 2011), or
- If a long stretch of foul values were detected, all the affected entries were disregarded, and that month marked as a "partial data" month

The reduced time series were stored in a Matlab Structure data file [*INIAClimate.mat*] to be used by the weather variables interpolation routine (*climateSimiulator.m,* sections 3.6 and 3.7).

Table 2: Reduced Weather Station Record (Abridged). INIA Glencoe Station.

TIMESTAMP				AirTemp	RH	Rain mm	WS ms	SlrMJ Tot	Sl Rad inst
TS				Deg C	%	mm	M/sec	MJ/m^2	Watt/m^2
mo	day	year	hour	Avg	Smp	Tot	Avg	Tot	inst
1	1	2010	0	18.67	73.57	0	5.029	0	0
1	1	2010	1	17.72	77.49	0	4.38	0	0
1	1	2010	2	16.87	80.3	0	4.388	0	0
1	1	2010	3	16.34	82.2	0	3.89	0	0
1	1	2010	4	15.87	82.8	0	3.396	0	0
1	1	2010	5	15.49	85.5	0	3.306	0.00	0.00
1	1	2010	6	14.59	88.6	0	2.988	0.01	3.23
1	1	2010	7	15.3	82.1	0	2.895	0.37	101.70
1	1	2010	8	17.45	72.29	0	4.408	1.09	302.00
1	1	2010	9	20.44	63.02	0	4.317	1.86	516.70
1	1	2010	10	23.33	59.73	0	4.221	2.57	714.00
1	1	2010	11	25.04	53.47	0	3.692	3.15	874.80
1	1	2010	12	26.46	54.1	0	2.095	3.53	981.70
1	1	2010	13	27.38	53.89	0	1.485	3.70	1,029.00
1	1	2010	14	28.79	44.47	0	1.269	3.68	1,022.00
1	1	2010	15	29.18	46.92	0	1.786	3.47	963.30
1	1	2010	16	29.43	48.58	0	2.688	2.90	805.50
1	1	2010	17	29.35	50.33	0	2.192	1.93	535.70
1	1	2010	18	28.48	49.64	0	2.527	1.00	278.30
1	1	2010	19	27.51	65.27	0	1.47	0.43	119.00
1	1	2010	20	25.58	73.18	0	1.208	0.13	35.13
1	1	2010	21	24.07	79.89	0	1.361	0.00	1.13
1	1	2010	22	22.91	83.5	0	1.914	0	0
1	1	2010	23	22.27	82.9	0	1.439	0	0

Note: The above weather record has been inserted with the only intention to show the standardized format in which the reduced records where stored.

Legend:

Columns 1-4: Timestamp (day, month, year, hour 24-hr cycle)
Column 5: Instant air temperature (deg. C)
Column 6: Instant air humidity (%)
Column 7: 1-hr accumulated rainfall (mm)
Column 8: Instant wind speed (m/sec)
Columns 9-10: Solar radiation (1-hr accumulated, MJ/m^2, and instant value, Watt/m^2)

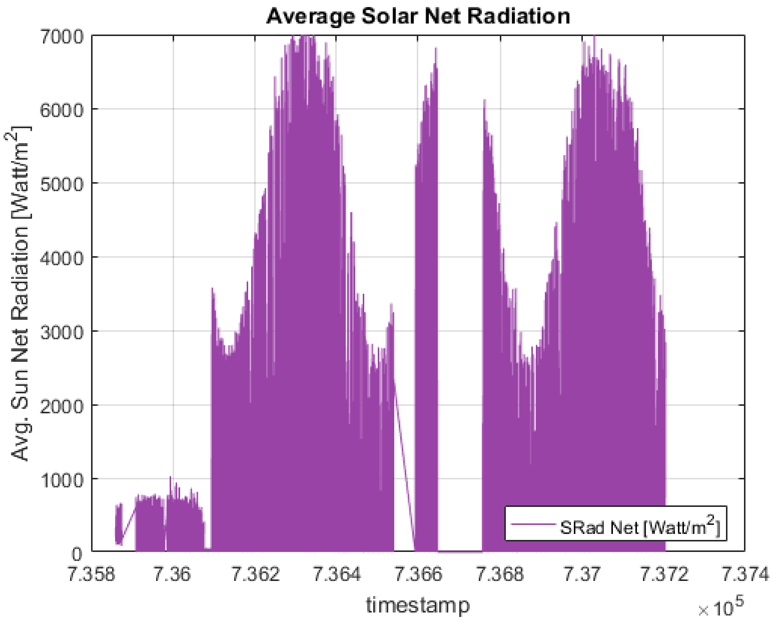


Figure 5: Example of foul solar radiation data records

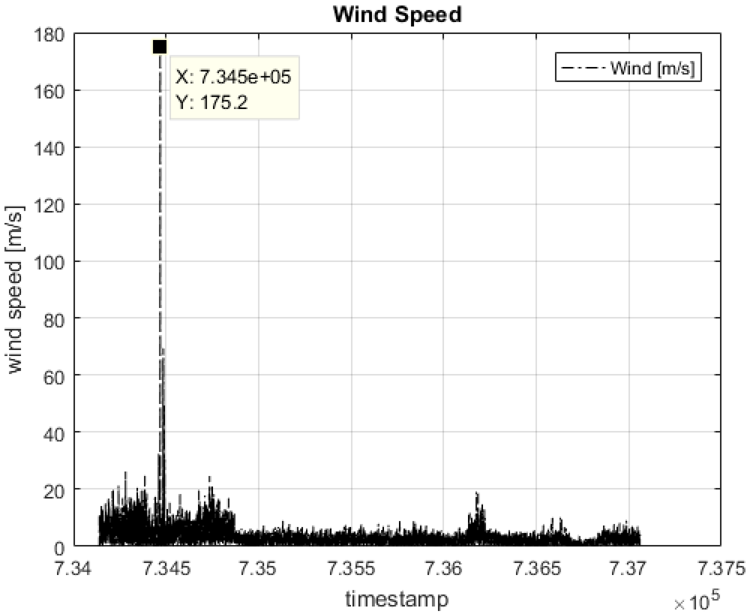
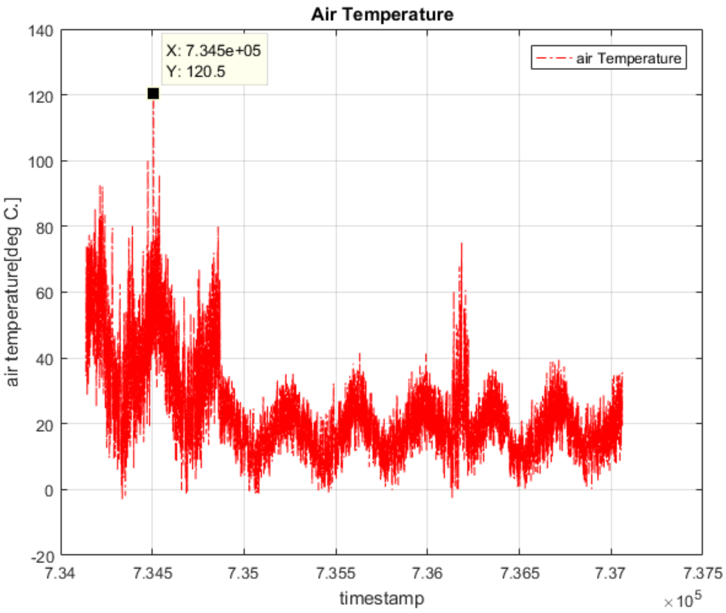


Figure 6: Example of wind speed records with a foul entry

Figure 7: Air temperature records with foul values



2.3 Traffic data acquisition and reduction

Traffic volume and load are key input variables to any pavement design methodology. M-E design methods are capable of calculating the dynamic effects different load configurations impose on the pavement structure, which eliminates the need for load conversion factors (Huang, 2004). In other words, the M-E design methodologies assess the performance of the pavement structures under the different axle types that actually run over its surface – single axles (2 and 4-wheel), double axles (4, 6, and 8-wheel), and tridem axles.

The MEPDG requires the following traffic-related data for design (AASHTO, 2015):

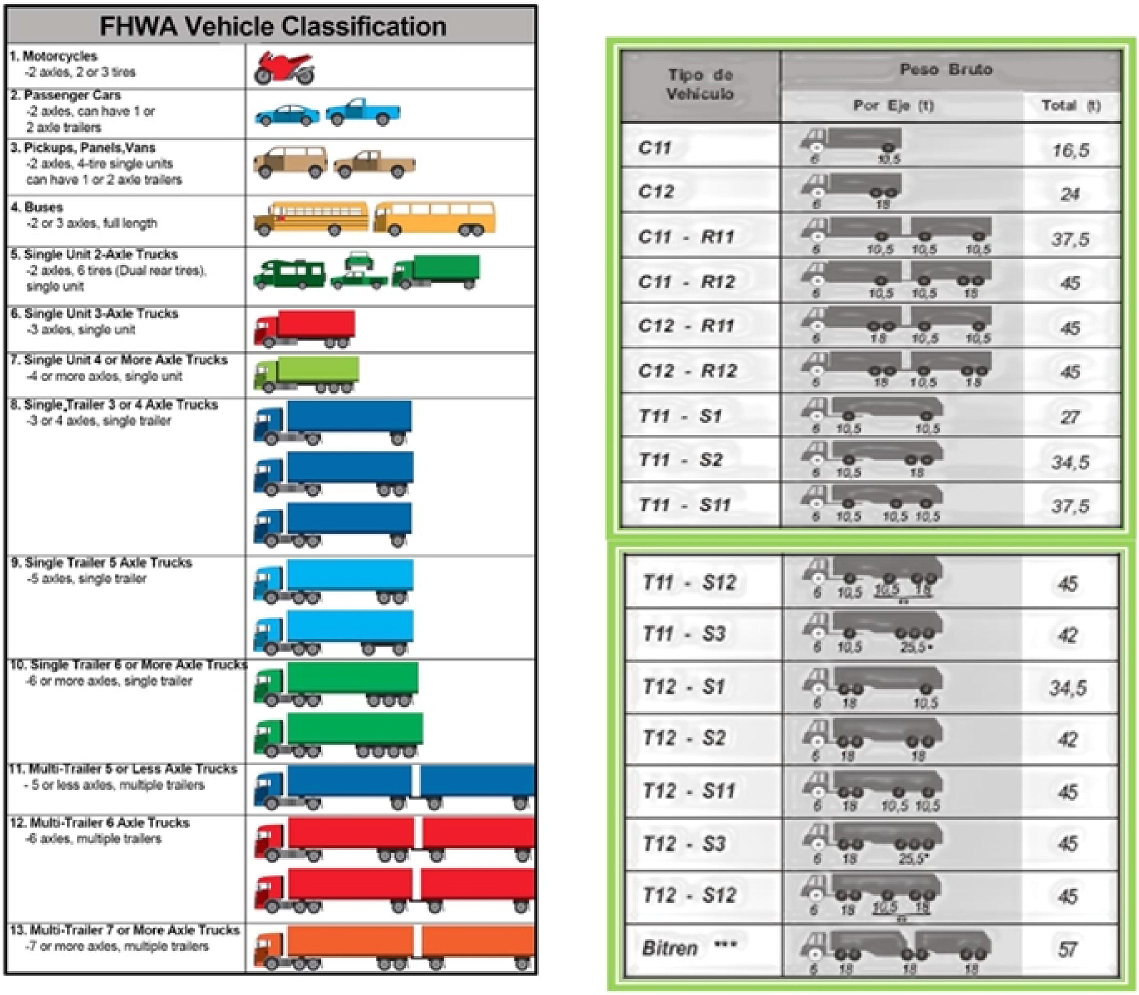
- *Base year truck-traffic volume (the year used as the basis for design computations).*
- *Vehicle (truck) operational speed.*
- *Truck-traffic directional and lane distribution factors* (the latter for multi-lane pavement design)
- *Vehicle (truck) class distribution.*
- *Axle load distribution factors.*
- *Axle and wheelbase configurations.*
- *Tire characteristics and inflation pressure.*

- *Truck lateral distribution factor.*
- *Truck growth factors.*

In the United States, all traffic-related information is recorded, stored, and published according to the FHWA standards, which use thirteen vehicle categories (FHWA, 2016). These are reproduced in the left part of figure 8. Traffic volume and weight records in the United States are typically logged and published as W-4 forms (NCHRP, 2004), and the FHWA's Traffic Monitoring Guide (FHWA, 2016) provides templates of other customary traffic data reports.

The Uruguayan Ministry of Transportation and Public Works (MTOP) uses different traffic data categories, as shown in the right part of figure 8. It is noted that the Uruguayan chart presented in figure 8 only depicts the local standard categories for trucks. There are three additional categories for buses (O11, O12, O22, where 1 stands for single axle and 2 for tandem axle), and the code A11 is used to denote light passenger vehicles, vans, and pick-ups.

Figure 9 depicts a sample AADT record for an Uruguayan highway, crafted to match the local 25 categories. A sample vehicle speed report is reproduced in Figure 10, and a truck weight report (denoting the percentage of unloaded, partially-loaded, fully-loaded, and overloaded vehicles) is reproduced in Figure 11.

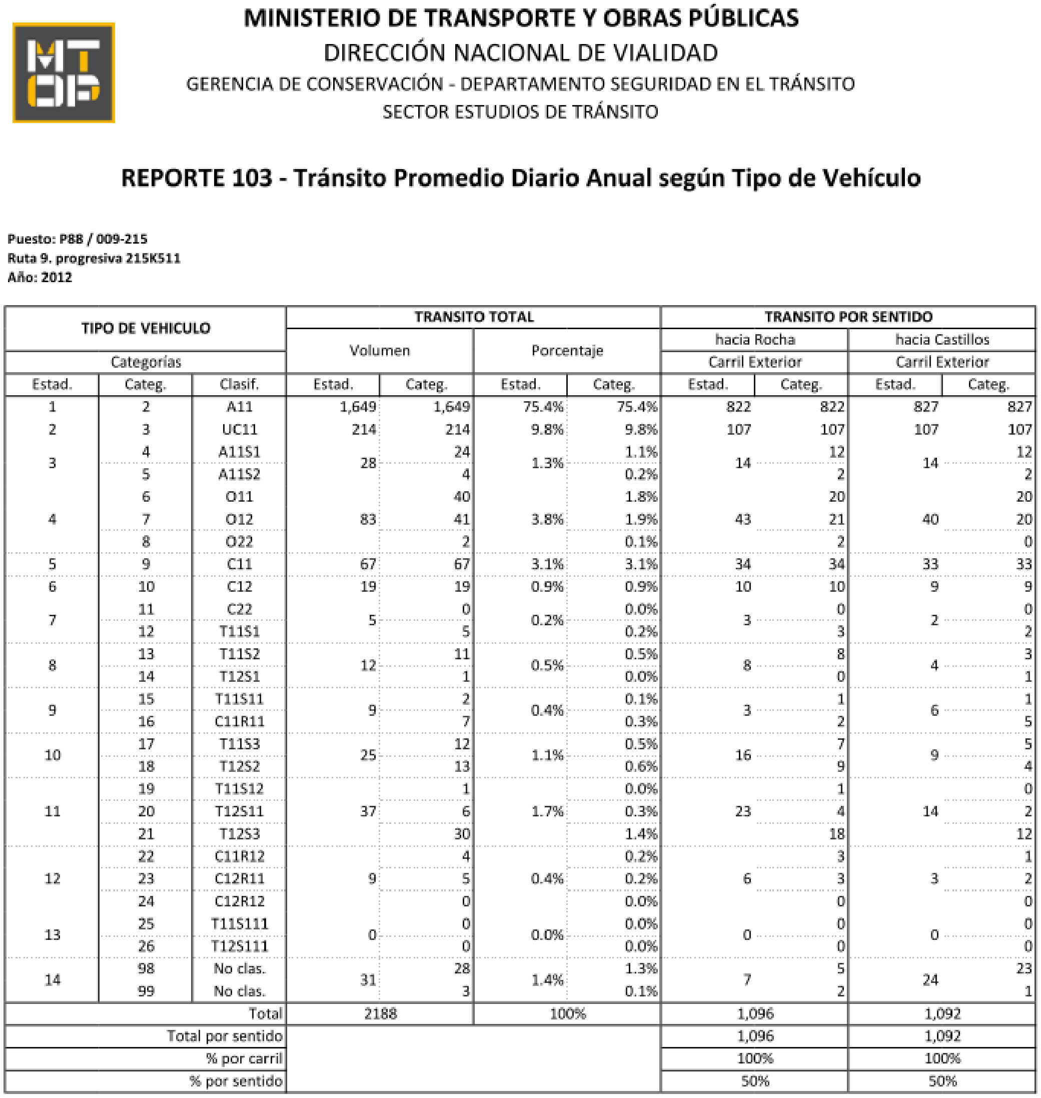


FHWA Vehicle Classification
1. Motorcycles -2 axles, 2 or 3 tires
2. Passenger Cars -2 axles, can have 1 or 2 axle trailers
3. Pickups, Panels,Vans -2 axles, 4-tire single units can have 1 or 2 axle trailers
4. Buses -2 or 3 axles, full length
5. Single Unit 2-Axle Trucks -2 axles, 6 tires (Dual rear tires), single unit
6. Single Unit 3-Axle Trucks -3 axles, single unit
7. Single Unit 4 or More Axle Trucks -4 or more axles, single unit
8. Single,Trailer 3 or 4 Axle Trucks -3 or 4 axles, single trailer
9. Single Trailer 5 Axle Trucks -5 axles, single trailer
10. Single Trailer 6 or More Axle Trucks -6 or more axles, single trailer
11. Multi-Trailer 5 or Less Axle Trucks - 5 or less axles, multiple trailers
12. Multi-Trailer 6 Axle Trucks -6 axles, multiple trailers
13. Multi-Trailer 7 or More Axle Trucks -7 or more axles, multiple trailers

Tipo de Vehículo	Peso Bruto	
	Por Eje (t)	Total (t)
C11	6 10,5	16,5
C12	6 18	24
C11 - R11	6 10,5 10,5 10,5	37,5
C11 - R12	6 10,5 10,5 18	45
C12 - R11	6 18 10,5 10,5	45
C12 - R12	6 18 10,5 18	45
T11 - S1	6 10,5 10,5	27
T11 - S2	6 10,5 18	34,5
T11 - S11	6 10,5 10,5 10,5	37,5
T11 - S12	6 10,5 10,5 18 **	45
T11 - S3	6 10,5 25,5*	42
T12 - S1	6 18 10,5	34,5
T12 - S2	6 18 18	42
T12 - S11	6 18 10,5 10,5	45
T12 - S3	6 18 25,5*	45
T12 - S12	6 18 10,5 18 **	45
Bitren ***	6 18 18 18	57

Figure 8: American and Uruguayan vehicle categories

Source Oregon DOT (US categories), MTOP (Uruguayan categories).



MINISTERIO DE TRANSPORTE Y OBRAS PÚBLICAS
DIRECCIÓN NACIONAL DE VIALIDAD
GERENCIA DE CONSERVACIÓN - DEPARTAMENTO SEGURIDAD EN EL TRÁNSITO
SECTOR ESTUDIOS DE TRÁNSITO

REPORTE 103 - Tránsito Promedio Diario Anual según Tipo de Vehículo

Puesto: P88 / 009-215
Ruta 9. progresiva 215K511
Año: 2012

TIPO DE VEHICULO			TRANSITO TOTAL				TRANSITO POR SENTIDO			
			Volumen		Porcentaje		hacia Rocha		hacia Castillos	
Categorías							Carril Exterior		Carril Exterior	
Estad.	Categ.	Clasif.	Estad.	Categ.	Estad.	Categ.	Estad.	Categ.	Estad.	Categ.
1	2	A11	1,649	1,649	75.4%	75.4%	822	822	827	827
2	3	UC11	214	214	9.8%	9.8%	107	107	107	107
3	4	A11S1	28	24	1.3%	1.1%	14	12	14	12
	5	A11S2		4		0.2%		2		2
4	6	O11	83	40	3.8%	1.8%	43	20	40	20
	7	O12		41		1.9%		21		20
	8	O22		2		0.1%		2		0
5	9	C11	67	67	3.1%	3.1%	34	34	33	33
6	10	C12	19	19	0.9%	0.9%	10	10	9	9
7	11	C22	5	0	0.2%	0.0%	3	0	2	0
	12	T11S1		5		0.2%		3		2
8	13	T11S2	12	11	0.5%	0.5%	8	8	4	3
	14	T12S1		1		0.0%		0		1
9	15	T11S11	9	2	0.4%	0.1%	3	1	6	1
	16	C11R11		7		0.3%		2		5
10	17	T11S3	25	12	1.1%	0.5%	16	7	9	5
	18	T12S2		13		0.6%		9		4
11	19	T11S12	37	1	1.7%	0.0%	23	1	14	0
	20	T12S11		6		0.3%		4		2
	21	T12S3		30		1.4%		18		12
12	22	C11R12	9	4	0.4%	0.2%	6	3	3	1
	23	C12R11		5		0.2%		3		2
	24	C12R12		0		0.0%		0		0
13	25	T11S111	0	0	0.0%	0.0%	0	0	0	0
	26	T12S111		0		0.0%		0		0
14	98	No clas.	31	28	1.4%	1.3%	7	5	24	23
	99	No clas.		3		0.1%		2		1
Total			2188		100%		1,096		1,092	
Total por sentido							1,096		1,092	
% por carril							100%		100%	
% por sentido							50%		50%	

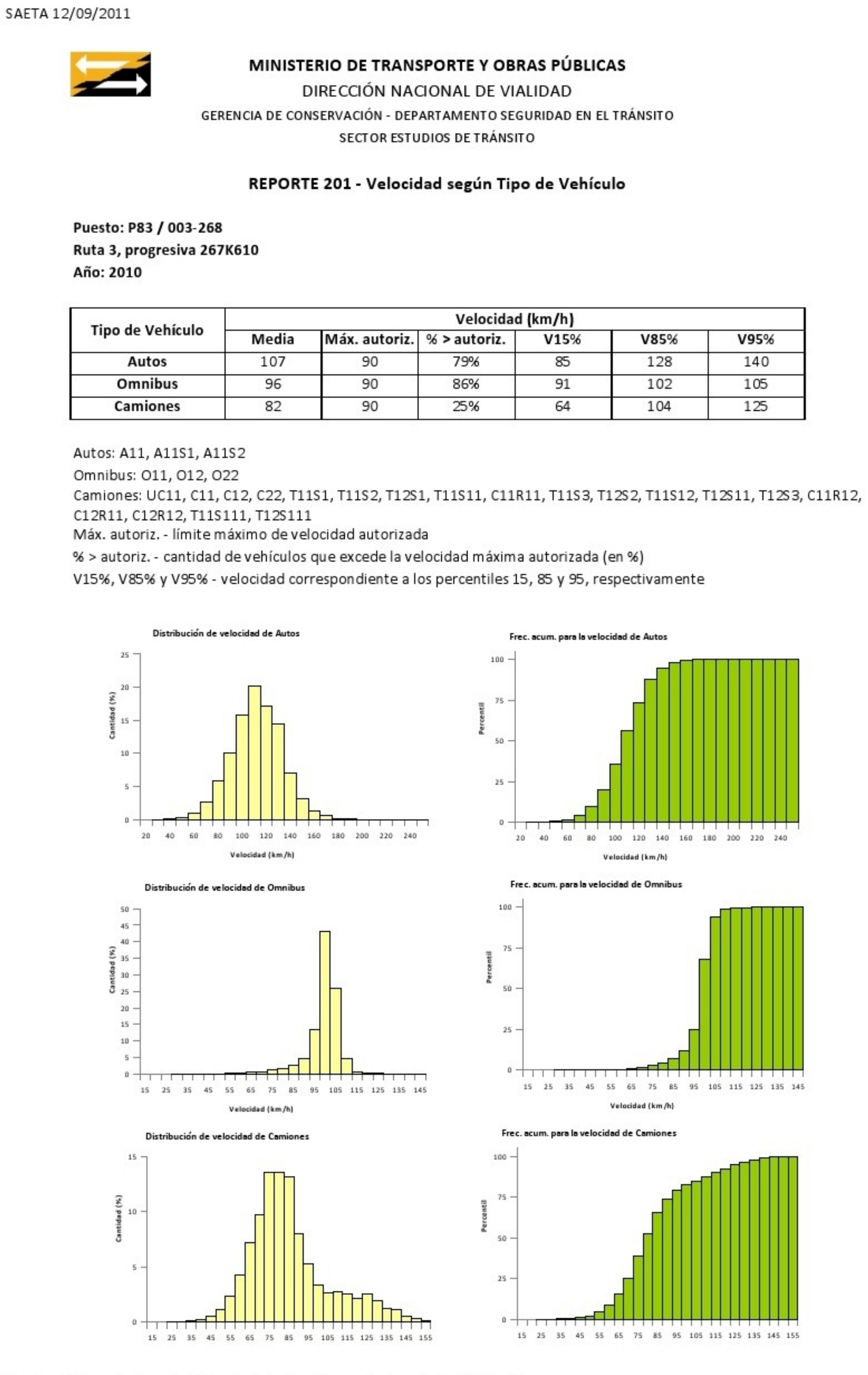
Figure 9: Sample Uruguayan Traffic count report (Reporte 103)

Legend:

Columns 1-3 of the table above denote the different vehicle categories used by the Uruguayan highway authority to classify vehicles

Columns 4-7 are the two-way AADT record

Columns 8-11 are the vehicle counts for each way of traffic.



SAETA 12/09/2011

MINISTERIO DE TRANSPORTE Y OBRAS PÚBLICAS
DIRECCIÓN NACIONAL DE VIALIDAD
GERENCIA DE CONSERVACIÓN - DEPARTAMENTO SEGURIDAD EN EL TRÁNSITO
SECTOR ESTUDIOS DE TRÁNSITO

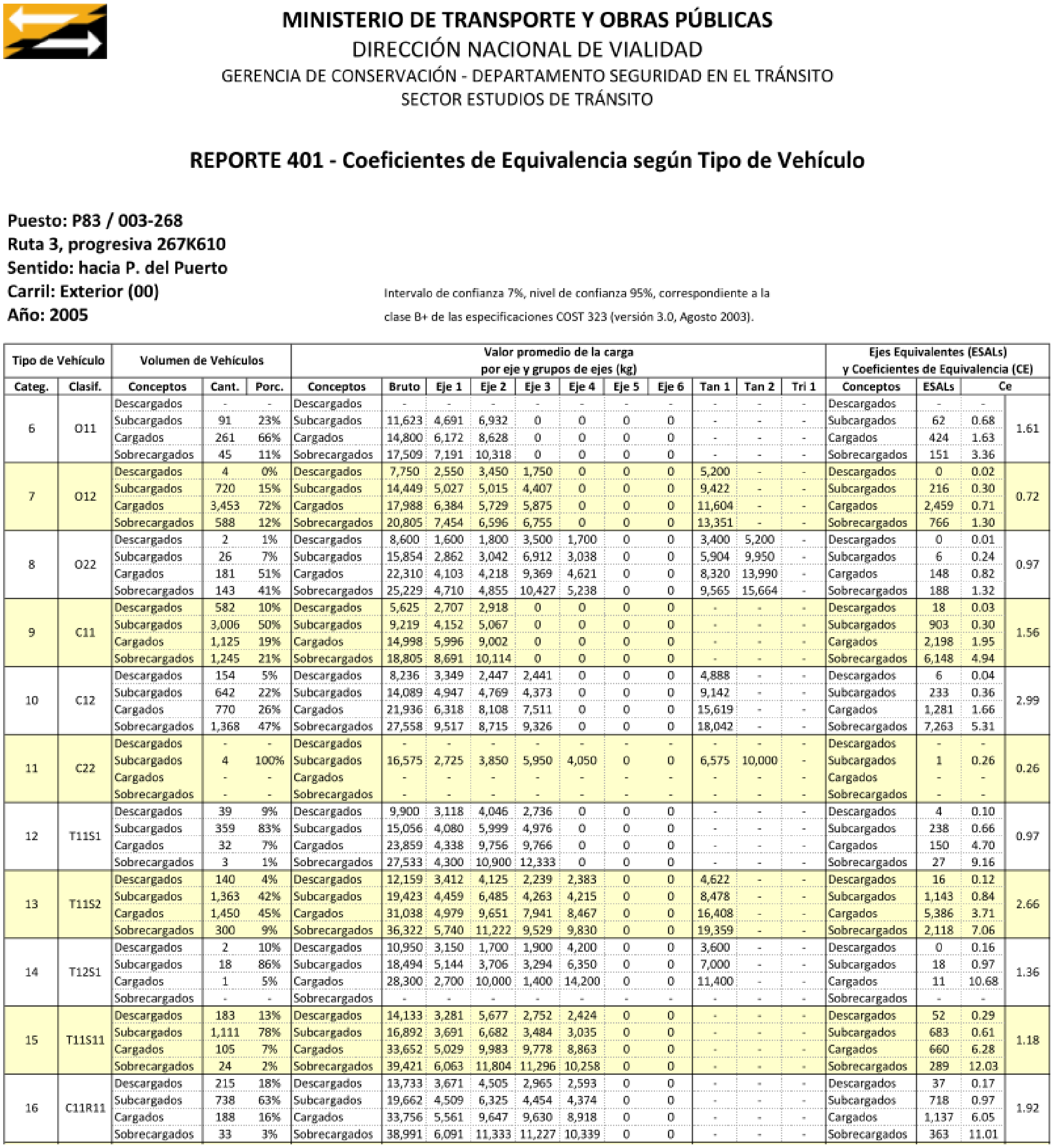
REPORTE 201 - Velocidad según Tipo de Vehículo

Puesto: P83 / 003-268
Ruta 3, progresiva 267K610
Año: 2010

Tipo de Vehículo	Velocidad (km/h)					
	Media	Máx. autoriz.	% > autoriz.	V15%	V85%	V95%
Autos	107	90	79%	85	128	140
Omnibus	96	90	86%	91	102	105
Camiones	82	90	25%	64	104	125

Autos: A11, A11S1, A11S2
Omnibus: O11, O12, O22
Camiones: UC11, C11, C12, C22, T11S1, T11S2, T12S1, T11S11, C11R11, T11S3, T12S2, T11S12, T12S11, T12S3, C11R12, C12R11, C12R12, T11S111, T12S111
Máx. autoriz. - límite máximo de velocidad autorizada
% > autoriz. - cantidad de vehículos que excede la velocidad máxima autorizada (en %)
V15%, V85% y V95% - velocidad correspondiente a los percentiles 15, 85 y 95, respectivamente

Figure 10: Sample Uruguayan traffic speed report (Reporte 201).



MINISTERIO DE TRANSPORTE Y OBRAS PÚBLICAS
DIRECCIÓN NACIONAL DE VIALIDAD
GERENCIA DE CONSERVACIÓN - DEPARTAMENTO SEGURIDAD EN EL TRÁNSITO
SECTOR ESTUDIOS DE TRÁNSITO

REPORTE 401 - Coeficientes de Equivalencia según Tipo de Vehículo

Puesto: P83 / 003-268
Ruta 3, progresiva 267K610
Sentido: hacia P. del Puerto
Carril: Exterior (00)
Año: 2005

Intervalo de confianza 7%, nivel de confianza 95%, correspondiente a la clase B+ de las especificaciones COST 323 (versión 3.0, Agosto 2003).

Tipo de Vehículo		Volumen de Vehículos			Valor promedio de la carga por eje y grupos de ejes (kg)											Ejes Equivalentes (ESALs) y Coeficientes de Equivalencia (CE)			
Categ.	Clasif.	Conceptos	Cant.	Porc.	Conceptos	Bruto	Eje 1	Eje 2	Eje 3	Eje 4	Eje 5	Eje 6	Tan 1	Tan 2	Tri 1	Conceptos	ESALs	Ce	
6	O11	Descargados	-	-	Descargados	-	-	-	-	-	-	-	-	-	-	Descargados	-	-	1.61
		Subcargados	91	23%	Subcargados	11,623	4,691	6,932	0	0	0	0	-	-	-	Subcargados	62	0.68	
		Cargados	261	66%	Cargados	14,800	6,172	8,628	0	0	0	0	-	-	-	Cargados	424	1.63	
		Sobrecargados	45	11%	Sobrecargados	17,509	7,191	10,318	0	0	0	0	-	-	-	Sobrecargados	151	3.36	
7	O12	Descargados	4	0%	Descargados	7,750	2,550	3,450	1,750	0	0	0	5,200	-	-	Descargados	0	0.02	0.72
		Subcargados	720	15%	Subcargados	14,449	5,027	5,015	4,407	0	0	0	9,422	-	-	Subcargados	216	0.30	
		Cargados	3,453	72%	Cargados	17,988	6,384	5,729	5,875	0	0	0	11,604	-	-	Cargados	2,459	0.71	
		Sobrecargados	588	12%	Sobrecargados	20,805	7,454	6,596	6,755	0	0	0	13,351	-	-	Sobrecargados	766	1.30	
8	O22	Descargados	2	1%	Descargados	8,600	1,600	1,800	3,500	1,700	0	0	3,400	5,200	-	Descargados	0	0.01	0.97
		Subcargados	26	7%	Subcargados	15,854	2,862	3,042	6,912	3,038	0	0	5,904	9,950	-	Subcargados	6	0.24	
		Cargados	181	51%	Cargados	22,310	4,103	4,218	9,369	4,621	0	0	8,320	13,990	-	Cargados	148	0.82	
		Sobrecargados	143	41%	Sobrecargados	25,229	4,710	4,855	10,427	5,238	0	0	9,565	15,664	-	Sobrecargados	188	1.32	
9	C11	Descargados	582	10%	Descargados	5,625	2,707	2,918	0	0	0	0	-	-	-	Descargados	18	0.03	1.56
		Subcargados	3,006	50%	Subcargados	9,219	4,152	5,067	0	0	0	0	-	-	-	Subcargados	903	0.30	
		Cargados	1,125	19%	Cargados	14,998	5,996	9,002	0	0	0	0	-	-	-	Cargados	2,198	1.95	
		Sobrecargados	1,245	21%	Sobrecargados	18,805	8,691	10,114	0	0	0	0	-	-	-	Sobrecargados	6,148	4.94	
10	C12	Descargados	154	5%	Descargados	8,236	3,349	2,447	2,441	0	0	0	4,888	-	-	Descargados	6	0.04	2.99
		Subcargados	642	22%	Subcargados	14,089	4,947	4,769	4,373	0	0	0	9,142	-	-	Subcargados	233	0.36	
		Cargados	770	26%	Cargados	21,936	6,318	8,108	7,511	0	0	0	15,619	-	-	Cargados	1,281	1.66	
		Sobrecargados	1,368	47%	Sobrecargados	27,558	9,517	8,715	9,326	0	0	0	18,042	-	-	Sobrecargados	7,263	5.31	
11	C22	Descargados	-	-	Descargados	-	-	-	-	-	-	-	-	-	-	Descargados	-	-	0.26
		Subcargados	4	100%	Subcargados	16,575	2,725	3,850	5,950	4,050	0	0	6,575	10,000	-	Subcargados	1	0.26	
		Cargados	-	-	Cargados	-	-	-	-	-	-	-	-	-	-	Cargados	-	-	
		Sobrecargados	-	-	Sobrecargados	-	-	-	-	-	-	-	-	-	-	Sobrecargados	-	-	
12	T11S1	Descargados	39	9%	Descargados	9,900	3,118	4,046	2,736	0	0	0	-	-	-	Descargados	4	0.10	0.97
		Subcargados	359	83%	Subcargados	15,056	4,080	5,999	4,976	0	0	0	-	-	-	Subcargados	238	0.66	
		Cargados	32	7%	Cargados	23,859	4,338	9,756	9,766	0	0	0	-	-	-	Cargados	150	4.70	
		Sobrecargados	3	1%	Sobrecargados	27,533	4,300	10,900	12,333	0	0	0	-	-	-	Sobrecargados	27	9.16	
13	T11S2	Descargados	140	4%	Descargados	12,159	3,412	4,125	2,239	2,383	0	0	4,622	-	-	Descargados	16	0.12	2.66
		Subcargados	1,363	42%	Subcargados	19,423	4,459	6,485	4,263	4,215	0	0	8,478	-	-	Subcargados	1,143	0.84	
		Cargados	1,450	45%	Cargados	31,038	4,979	9,651	7,941	8,467	0	0	16,408	-	-	Cargados	5,386	3.71	
		Sobrecargados	300	9%	Sobrecargados	36,322	5,740	11,222	9,529	9,830	0	0	19,359	-	-	Sobrecargados	2,118	7.06	
14	T12S1	Descargados	2	10%	Descargados	10,950	3,150	1,700	1,900	4,200	0	0	3,600	-	-	Descargados	0	0.16	1.36
		Subcargados	18	86%	Subcargados	18,494	5,144	3,706	3,294	6,350	0	0	7,000	-	-	Subcargados	18	0.97	
		Cargados	1	5%	Cargados	28,300	2,700	10,000	1,400	14,200	0	0	11,400	-	-	Cargados	11	10.68	
		Sobrecargados	-	-	Sobrecargados	-	-	-	-	-	-	-	-	-	-	Sobrecargados	-	-	
15	T11S11	Descargados	183	13%	Descargados	14,133	3,281	5,677	2,752	2,424	0	0	-	-	-	Descargados	52	0.29	1.18
		Subcargados	1,111	78%	Subcargados	16,892	3,691	6,682	3,484	3,035	0	0	-	-	-	Subcargados	683	0.61	
		Cargados	105	7%	Cargados	33,652	5,029	9,983	9,778	8,863	0	0	-	-	-	Cargados	660	6.28	
		Sobrecargados	24	2%	Sobrecargados	39,421	6,063	11,804	11,296	10,258	0	0	-	-	-	Sobrecargados	289	12.03	
16	C11R11	Descargados	215	18%	Descargados	13,733	3,671	4,505	2,965	2,593	0	0	-	-	-	Descargados	37	0.17	1.92
		Subcargados	738	63%	Subcargados	19,662	4,509	6,325	4,454	4,374	0	0	-	-	-	Subcargados	718	0.97	
		Cargados	188	16%	Cargados	33,756	5,561	9,647	9,630	8,918	0	0	-	-	-	Cargados	1,137	6.05	
		Sobrecargados	33	3%	Sobrecargados	38,991	6,091	11,333	11,227	10,339	0	0	-	-	-	Sobrecargados	363	11.01	

Figure 11: Sample Uruguayan load chart (Reporte 401, abridged)

Legend:

"Descargados" denotes vehicles loaded at no more than 40% of their maximum legal weight
"Subcargados" denotes vehicles loaded between 40% and 80% of their maximum legal weight
"*Cargados*" denotes vehicles loaded between 80% and 100% of their maximum legal weight
"*Sobrecargados*" denotes vehicles heavier than their maximum legal weight

2.3.1 Traffic data availability

Traffic counts are readily available for Uruguayan practitioners. The highway authority (MTOP) publishes these over the Internet as GIS datasets[9]. Figure 12 portrays an AADT map of Uruguay's highways for the year 2017, last record available. However, other traffic-related inputs such as load charts, speed information, and traffic distribution factors are not publicly available. That being said, an effort has been made to gather as many of these inputs as possible and create a library of default (level 2) values for design purposes, one of Product-Two components.

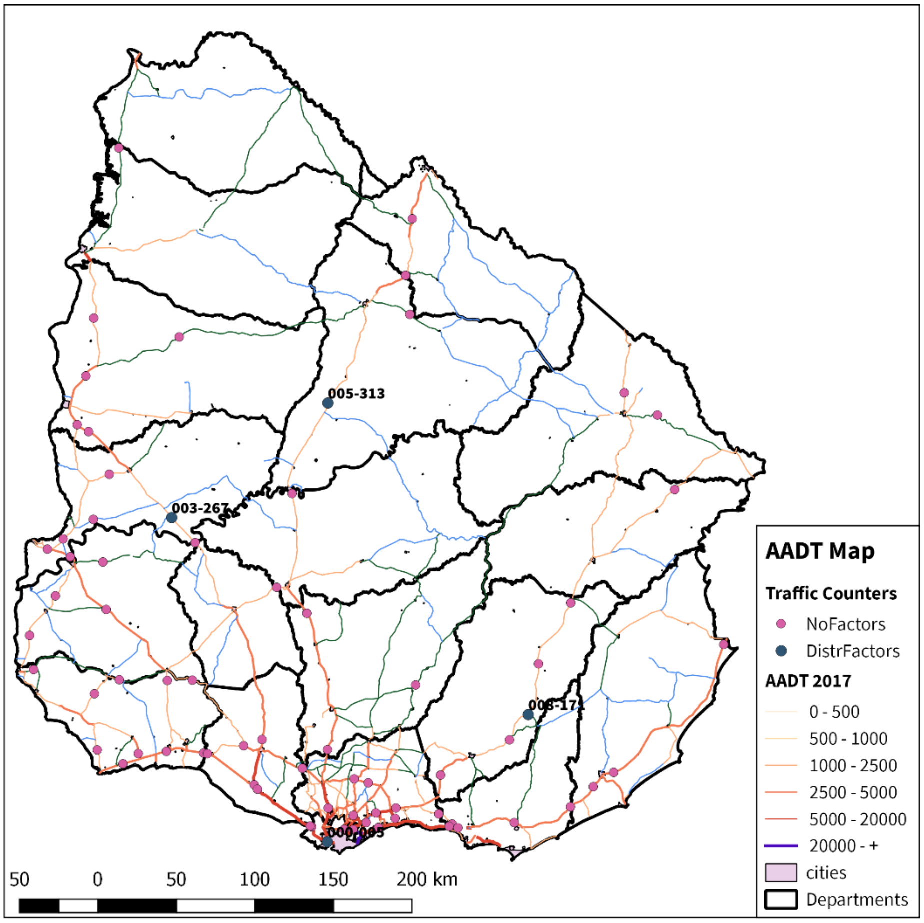


Figure 12: AADT on Uruguayan highways. Year 2017

Sources: IDE_UY (National and province boundaries), geoportal MTOP (traffic information)

A request of information has been filed with the MTOP, in which load records, speed records, and traffic distribution factors (monthly, daily, and hourly) for a set of Uruguayan highways were asked for. The set of selected highways comprised routes mostly used by either commuter traffic, heavy-haul freight (all-

[9] The aforementioned AADT records bank can be accessed at geoportal.mtop.gub.uy

year round and seasonal), tourists, and other mixed-traffic routes. A response has been received, and traffic distribution factors for a subset of Uruguayan primary roads that belong to different functional classes were provided. However, the load record data were kept from disclosure. A log on how these data have been processed for use within Product-One follows.

2.3.2 Processing of traffic distribution factors for Uruguayan roads

The Uruguayan highway authority provided the following traffic distribution factors for selected primary roads: (1) a primary network corridor rich with international freight traffic (Route 3, km 267), (2) a suburban arterial featuring mixed heavy traffic plus commuter traffic (Route 1 km 5), (3) a mixed-traffic trunk road, (Route 8 km 171), and (4) a primary highway with seasonal peaks of heavy haul traffic (Route 5 km 313). The traffic counting stations at these locations were tagged on figure 12. The information provided included:

- Monthly distribution factors for the year 2017
- Daily distribution factors for the twelve months of 2017
- Hourly distribution factors for weekdays and weekends for each month of 2017

Furthermore, the distribution factors have been provided separately for different vehicle types: cars, buses, light trucks, mid-size trucks, and heavy-haul trucks.

The calculation methodology adopted for Product-One relies on a single set of monthly, daily, and hourly distribution factors for each vehicle category, which must be stored in Product-Two. The monthly distribution factors were loaded directly into Product-Two. However, the average of the daily for all months and a weighted average of the hourly factors for all months accounting for the number of days in the weekend and weekday periods were computed and taken as representative of the entire set. The final collection of factors is available in Appendix B.

CHAPTER 3 — M-E DESIGN SOFTWARE IMPLEMENTATION

3.1 Introduction

This chapter presents the implementation of a working Matlab-based, MEPDG-inspired, simplified M-E pavement design tool (Product-One). The main motivation behind such a goal is providing Uruguayan pavement engineers with an M-E design tool whose built-in models could be unveiled and studied in depth, thus overcoming one of the obstacles Chang Albitres et al. (2013) reported that prevent M-E methods from being utilized at their fullest extent.

The software development task started by understanding the MEPDG's data flow process and detecting which of its built-in models, functions, or routines need to be adapted to become compatible with Uruguayan available data, and which would not be applicable. For instance, importing Uruguayan climate data from the INIA weather stations and modeling a subgrade's freeze-thaw cycle are perfect examples of these two categories of components, the first one must be programmed while the latter can be disregarded. Once all the software's components of interest were identified, an enhanced literature review was conducted to obtain their mathematical formulations, as presented throughout this chapter.

The following step consisted of bundling all these models together into Product-One's source code and optimizing its running time. Figure 13 depicts Product-One's conceptual flow diagram.

Finally, Product-One's results for a default trial design were compared against other available M-E design tools as a quality control measure. Additionally, the design results were also compared with the AASHTO (1993) model, to assess how much these change as a consequence of changing the design methodology while leaving all the design inputs the same. This is discussed in depth in Chapter 4.

The following sections of this chapter provide a thorough explanation of the models that have been programmed into Product-One, their mathematical formulas, and how these have been coded up. This chapter has been written as a standalone user's guide to Product-One. It includes instructions on how to load data inputs to Product-One, how to execute the program, and how to pull the design results once the calculations are completed. Nevertheless, it is assumed here that the end-user is somewhat familiar with the Matlab language and how to execute scripts from a Matlab interpreter, whether it's Matlab ® or GNU-Octave ©.

3.2 Product-One M-E Pavement Design program – Overview

The conceptual flow chart depicted in Figure 13 provides a bird-eye view of Product-One, all of its modules and the data exchange among them.

Product-One's modules have been distributed in four categories: *Input data importing, Pre-processing, pavement performance simulation, and results reporting*. All these modules are run in sequence from a script named ***mainCode,*** which is the only script the end-user needs to interact with. Either matlab or GNU-Octave (denominated "interpreters" from now on) is needed to execute Product-One. The user must run the ***mainCode*** from the interpreter's console to launch the calculations. Additionally, a spreadsheet has been devised for the end-user to intuitively load all the input variables into the design software (***default name: dataImport.xlsx*****).** When Product-One is launched, the software will ask the user the name of this spreadsheet so that the design inputs can be retrieved, the user must type it on the interpreter's command line and Product-One will resume execution instantly.

3.3 Product-One Components

This section provides an in-depth description of each of Product-One components, i.e. scripts and functions that execute the mathematical models adopted as part of the M-E pavement design program, and the front-end interface. Each sub-section describes a single Product-One component, and the referenced literature used in its development. The source code to each component of Product-One is available in Appendix E. Finally, instructions are given on how to use the module (if it requires any action by the end-user) along with an explanation of the module source code.

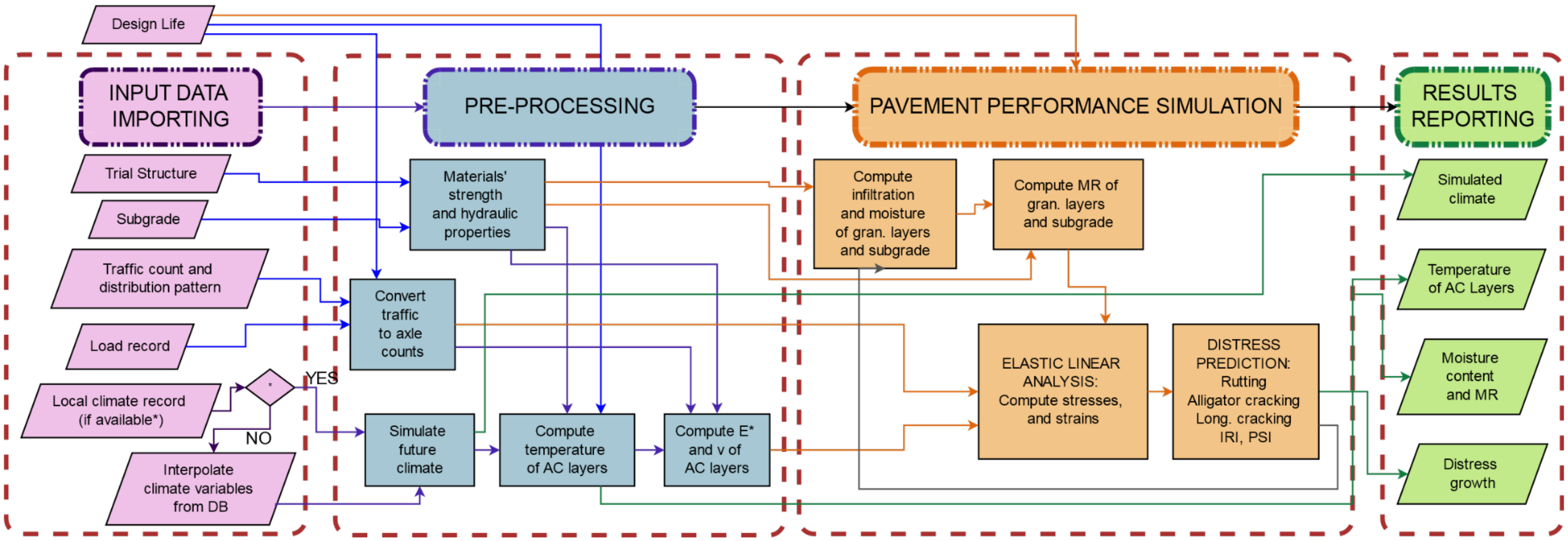

Design Life
INPUT DATA IMPORTING
Trial Structure
Subgrade
Traffic count and distribution pattern
Load record
Local climate record (if available*)
*
YES
NO
Interpolate climate variables from DB
PRE-PROCESSING
Materials' strength and hydraulic properties
Convert traffic to axle counts
Simulate future climate
Compute temperature of AC layers
Compute E* and v of AC layers
PAVEMENT PERFORMANCE SIMULATION
Compute infiltration and moisture of gran. layers and subgrade
Compute MR of gran. layers and subgrade
ELASTIC LINEAR ANALYSIS: Compute stresses, and strains
DISTRESS PREDICTION: Rutting Alligator cracking Long. cracking IRI, PSI
RESULTS REPORTING
Simulated climate
Temperature of AC Layers
Moisture content and MR
Distress growth

Figure 13: Conceptual flow chart of all Product-One's modules

3.3.1 Notation

During the remainder of this chapter, modules of Product-One and variables involved in calculations will be named repeatedly. In an effort to prevent any entanglement among the many names, the following notation convention will be followed:

- *Variables* are expressed in italics
- *"textVariablesValues"* are expressed in italics too, but within brackets
- The names of ***Files, scripts, software modules,*** and the ***components of the data input and output spreadsheets*** appear in bold italics.

3.4 Input data workbook – *dataImport*

This six-spreadsheet workbook contains the level-2 and level-3 data on traffic and materials' properties collected as part of this Thesis – the Product-Two. It has been designed to facilitate the loading of all necessary inputs into Product-One in an intuitive, graphical manner. The input tables in ***dataImport*** match the formats in which the required input data are published by the local road agency The following sections describe the contents of each spreadsheet of ***dataImport***.

3.4.1 Project's general information – the *Input* spreadsheet

DataImport's first spreadsheet allows the user to type in the following project – related information:

- Project's location (UTM Zone 21 coordinates),
- Design life (years),
- Date of pavement's opening to traffic (date format),
- Initial distress rates for select distresses [namely International Roughness Index (IRI), rut depth, and longitudinal, transverse, and alligator cracking],
- Maximum admissible distress rates (same as the above),
- Name of results file: The user must give a name for the file in which Product-One will write all numerical results once the execution ends. (default value: "*trialRun*"),
- Program run configuration options, namely:
    - ○ Verbose execution: If enabled (*verbose* = 1), Product-One will report on the Interpreter's command window every progress that is made during execution.

    - Create figures on screen?: A list of four cells are reserved for the user to decide if plots are to be generated showing the calculation results. The user decides by giving each either a 1 or 0 (*"yes"* or *"no"*, respectively). Product-One can create plots for the following outputs:
        - The predicted climate variables
        - The forecast traffic
        - The climate-affected materials' properties
        - The predicted distress growth
- Use Level-1 climate records (station located at the Project site): if that is the case (*climateLevel1* = 1), then the user must load the site's temperature, humidity, wind speed, solar radiation, and rainfall in the ***climate*** spreadsheet. Otherwise (*climateLevel1 = 0*), Product-One will estimate the project site's weather using the Inverse Distance Interpolation technique (Wei and Mc. Guinness, 1973). The calculations are done by the ***climateInterpolator*** script, which is described in detail in section 3.6.
- Force recalculation of weather variables: if enabled (value = 1), both the climate records interpolation for level-2 and level-3 design and the climate prediction routine are run, regardless of the results of the last execution. It is recommended to maintain a value of 0 to avoid unnecessary time-consuming calculations, thus saving time. Product-One would compare the current run inputs against last run's and only re-do the climate prediction if any design input (like the project site's location, date of opening to traffic, and even the future weather random seed, further explained below) was modified.
- Random seed for future weather: this variable is an integer ranging from 1 to 50, which points to a single randomly-generated sequence of weather record years to construct the project site's future weather. This variable is used by the ***climateScrambler*** routine, which runs automatically upon execution.

3.4.2 Pavement structure data – *pavStructure* spreadsheet

The trial pavement structure must be entered in this spreadsheet. The end-user must define each pavement layer (thickness and material), provide information on the type of soil that composes the site's subgrade, and additional information on the roadway's cross section.

Product-One admits up to five pavement layers plus one subgrade material. The user must define each by typing in their respective target thicknesses (in centimeters), plus the respective *materialID* – a unique numerical identifier for each material. The spreadsheet would solve the material names from the *materialID*, and Product-One will utilize them to retrieve their respective strength and hydraulic properties. The *materialID* for the default Level-3 materials can be queried in the ***materialsCatalog*** sheet, and the end-user may add information about locally available materials (Level-1 or Level-2 inputs) from ***materialsCatalog*** as well.

3.4.3 Project-site's weather records [level 1 input] – *climate* spreadsheet

This spreadsheet is only queried by Product-One when the user had stated that level-1 weather records are available. This rather self-explanatory spreadsheet provides the template the end-user must follow in order to load such weather records into Product-One (the same table template used to store the reduced INIA weather records – table 2 in Chapter 2**).** Upon execution, Product-One would ask the end-user for instructions in order to perform the data query itself (refer to the ***climateLoadLocal*** scripts).

3.4.4 Materials' properties library – *materialsCatalog* spreadsheet

This spreadsheet conveys the outcome of the materials' properties data collection effort. As of this initial Product-One release, the materials' library contains Level-3 properties for hot-mix asphalt (HMA) concrete materials (drawn with correlations from the Uruguayan road construction standards (MTOP, 1990; MTOP, 2003)), and level-2/level-3 strength and hydraulic properties for soils and granular materials, as obtained from Uruguayan laboratory test results and from correlations provided in the literature (NCHRP, 2004; FHWA, 2006). Some blank extra fields have been reserved within the materials' properties' tables for the user to add Level-1 data in the event such information becomes available. However, the following restrictions must be observed:

- Certain ranges for *materialID* must not be modified: HMA materials must be tagged with a *materialID* between 1 and 99; the *materialID* for fine soils and granular materials (Clays or Silts, according to their USCS[10] classification) must range between 101 and 150, whereas Sands or Gravels must be tagged with *materialID* ranging between 151 and 199. Higher *materialID* values have been reserved for other material types to be added in future enhanced releases of Product-One.
- The physical units of each typed input variable must be respected. Each column in the materials properties' tables represent a single material property, its physical unit has been carefully checked

[10] USCS stands for *Unified Soil Classification System*, it is the oldest (launched in 1948 by Casagrande) widely-accepted standard system to classify soils and granular materials (Carter and Bentley, 2016). For details on this soil classification system, refer to FHWA (2006).

and highlighted to ensure proper acknowledgment by the end-user – most variables must be typed in in SI units, with the exception of Level-1 Resilient modulus for soils, which is given in psi (pound per square inch).

The following are Product-One's fundamental requirements in terms of HMA materials' strength-related properties since these are used to compute the mixes' dynamic moduli for different temperatures, load magnitudes, and axle speed – refer to section 3.9.3, later on in this chapter:

- Percentage (by weight) of aggregate passing the #200 sieve ,
- Air void content (in percentage volume),
- Effective bitumen content (in percentage volume),
- Penetration at 25 degrees Celsius (77 F), in cPoise.

This release of Product-One provides level-3 (default values) for the different standard conventional HMA mixes defined in the Uruguayan road materials specifications (MTOP, 1990) [*materialID* = 001 to 012], the user can nonetheless add information about readily-available HMA mixtures for any given project (Level-1) as a new material (with its own *materialID*), provided that testing for all required properties has been completed.

If available, the user may also modify the HMA surface heat absorptivity, emissivity, and heat transfer coefficients [SI units][11], as well as the top-most layer's placement temperature (in degrees Celsius), in order to improve the design process done with Product-One even further. However, this can be done only in Product-One's code itself, by modifying the default values in the ***materialsParameters*** scripts.

Similarly, the materials' properties table for granular materials and soils includes columns for Product-One's minimum requirements in terms of strength and hydraulic properties – listed below. Additionally, extra columns have been provided for other soil's physical properties with which the former may be correlated using formulas available in the literature.

The following soil strength and hydraulic properties must be provided into Product- One for Level-1 design:

- Material's Resilient Modulus at optimum compaction level (PSI),
- Material's Poisson coefficient (dimensionless),
- Optimum moisture content (percentage by weight),

[11] The default values for these HMA mixes properties that were plugged into Product-One are the Level-3 defaults provided in the MEPDG guide (NCHRP, 2004) and by the FHWA (1993)

- Saturation water content (percentage by weight),
- Maximum dry density (kg /m^3), and percentage of maximum density achieved in the field,
- Degree of saturation at optimum moisture content (%),
- Specific gravity (Gs),
- Saturated hydraulic conductivity (m/sec),
- The SWCC curve parameters, namely a, b, c, and h_r (retaining MEPDG's notation (NCHRP, 2004), PSI, dimensionless, dimensionless, and PSI respectively).

The following major simplifications concerning the dynamic behavior of granular materials have been assumed when programming Product-One, these constitute a departure from the MEPDG when attempting a Level-1 design:

- All granular materials are assumed to be linear elastic. As a consequence, these can be characterized by a single resilient modulus (MR) value for any given moisture content. In other words, the granular materials' and soils' MR was assumed not to be stress-dependent – the same assumption the MEPDG sets forth for level-2 and level-3 design (NCHRP, 2004). Hence the requirement of a single MR (at optimum compaction) value for a given soil or granular material.
- Since no freeze-thaw cycles occur on Uruguayan pavement structures, Product-One disregarded the MEPDG's models to quantify such effects (FHWA, 1993; Larson and Dempsey, 1997). Thus, no data concerning the materials' response to freezing and thawing would be required by Product-One.

If the end-user wishes to conduct a level-2 or level-3 design using Product-One, the above variables can be computed within the ***materialsCatalog*** spreadsheet using the MEPDG's formulas (NCHRP, 2004; FHWA, 2006):

Material's resilient modulus from CBR

$$Mr\left[PSI\right] = 2555 \times CBR^{0.64} \tag{1}$$

Material's specific gravity

$$Gs = 0.041\,wPI^{0.29} + 2.65$$

$$wPI = \frac{PI\left[\%\right] \times P_{200}\left[\%\right]}{100} \tag{2}$$

Optimum compaction moisture content by weight vs optimum compaction moisture content by volume

$$\theta_{opt}\left[\%\right]=\frac{\gamma_{dmax}\times\omega_{opt}\left[\%\right]}{\gamma_{water}} \tag{3}$$

Where θ_{opt} is the material's optimum water content (in percent volume), $\gamma_{d\,max}$ is its maximum density, γ_{water} is the density of water, and ω_{opt} is the optimum water content by weight.

Saturation degree at optimum compaction moisture

$$S_{sat}=\frac{\theta_{opt}}{1-\frac{\gamma_{dmax}}{\gamma_{water}Gs}} \tag{4}$$

Saturation level moisture content (by volume)

$$\theta_{sat}\left[\%\right]=\frac{\theta_{opt}\left[\%\right]}{S_{sat}} \tag{5}$$

Soil-Water Characteristic curve (SWCC) parameters

$$\begin{gathered}\text{if wPI>0}\\ SWCCa\left[PSI\right]=\frac{0.00364\,wPI^{3.55}+4\,wPI+11}{6.895}\\ SWCCb=SWCCc\times\left(-2.313\,wPI^{0.14}+5\right)\\ SWCCc=0.0514\,wPI^{0.465}+0.5\\ SWCCh\left[PSI\right]=SWCCa\times\left(32.44\,e^{0.0186wPI}\right)\\ \text{if wPI = 0}\\ SWCCa\left[PSI\right]=\frac{0.8627\,D_{60}{}^{-0.751}}{6.895}\\ SWCCb=7.5\\ SWCCc=0.1772\,Ln\left(D_{60}\right)+0.7734\\ SWCCh\left[PSI\right]=\frac{SWCCa}{\left(D_{60}+9.7\times e^{-4}\right)}\end{gathered} \tag{6}$$

Material's hydraulic conductivity at saturation level

$$\begin{gathered}if\,0<wPI<1\\ k_{sat}\left[ft/hr\right]=118.11\times10^{\left[-1.1275\left(\log_{10}\left(D_{60}+2\right)\right)^{2}+7.2816\left(\log_{10}\left(D_{60}+2\right)\right)-11.2891\right]}\\ ifwPI\geq1\\ k_{sat}\left[ft/hr\right]=118.11\times10^{\left[0.0004wPI^{2}-0.929\,wPI-6.56\right]}\end{gathered} \tag{7}$$

Notes:

- k_{sat} is the material's hydraulic conductivity when saturated
- D_{60} must be provided in inches

- The formula for the 0<wPI<1 case is valid for D_{60} < 0.75 inches, if D_{60} > 0.75 in, compute k_{sat} assuming D_{60} = 0.75 in

The units for wPI are not explicit In the bibliographical sources. The units for wPI are only mentioned in a rather concealed manner in appendices CC and GG of the MEPDG guide (NCHRP, 2004). It is pointed out therein that wPI is the product of the P200 as a ratio (and not as a percentage value) and the PI as percentage value. To prove this point, equation 7 was tested with input values for fine soils with both sets of units, that is both variables as percentage value and P200 as a 0-to-1 ratio value: The first case would lead to unreasonable Gs and k_{sat} values for silty and clayey soils, whereas the other case leads to estimates close to values recorded in the bibliography (Waltham, 2002; FHWA, 2006).

3.4.5 Traffic volume, speed, and load inputs – *trafficCountInput* and *trafficLoadInput* spreadsheets

These two spreadsheets can use all the traffic-related information from the Uruguayan sources directly for design. Minor calculations only would eventually be needed to turn the available raw data into design inputs for Product-One. Two separate spreadsheets have been prepared, intended for traffic-count-related and load-record-related data respectively.

The ***trafficCountInput*** spreadsheet is the end-user interface to load the site's AADT count from a *Reporte 103* (figure 9): both the AADT for each standard Uruguayan category on a base year, the forecast traffic growth rate (as a percentage), and all the necessary traffic distribution factors (among which the direction and lane distribution factors, and the time-related AADT distribution factors) are loaded here. The spreadsheet itself converts the "as-received" AADT report into a "design-lane" AADT vector, using slightly different formulas whether the project consists of a two-lane or a 4+-lane road (the underlying reason being that the *Reporte 103* provides two-way AADT for two-lane highways, whereas only one-way traffic is reported for multi-lane highways, and in that latter case no direction factor is needed).

These formulas have been reproduced below.

Convert reported AADT to design lane AADT

$$\begin{array}{l} \text{if two-lane road} \\ AADT_{designlane} = AADT_{reported} \times DF \\ \text{ otherwise} \\ AADT_{designlane} = AADT_{reported} \times LF \end{array} \tag{8}$$

Where DF is the "direction" factor for two-lane roads, and LF is the lane factor for multi-lane roads.

Product-One would import the design-lane AADT vector and store it under the name *AADTBaseYear*

It must be highlighted that no other calculation is done by this spreadsheet, the ***trafficSimulator.m*** script within Product-One (section 3.8.1) computes the forecast AADT for each vehicle category from the initial AADT records and the corresponding growth rates and following derives the predicted traffic volume at each simulation time step by applying the monthly, daily, and hourly distribution factors in an orderly manner.

However, this spreadsheet is the end-user interface for plugging-in Level-1 monthly, daily, and hourly distribution factors. If these factors become available for a particular design, the following conditions must be met (FHWA, 2016):

- The monthly factors must be given as an increment/decrement value around 1. Thus, for any given traffic category, the average of all monthly factors must equal 1.
- The average of all daily distribution factors for a given category must be 1
- The hourly distribution factors are 0 to 1 ratios, and the sum of all 24 factors for a given vehicle category must equal 1

Level-2 and Level-3 factors compliant with the above restrictions have been made available for the end-user to use, the user must state in the ***trafficCountInput*** spreadsheet (cell M10) which level of input the provided traffic count data corresponds to, and Product-One will retrieve accordingly for calculation purposes. The level-2 input that has been programmed by default with this release of Product-One corresponds to a "uniform traffic distribution", which in other words implies that the design vehicle flow is equally distributed at all times. That is an assumption previous pavement design methodologies (such as the AASHTO (1993) method) rely on, thus this default set of distribution patterns proves suitable for comparisons between the AASHTO (1993) method and Product-One. Meanwhile, the default Level-3 input has been retrieved from the FHWA (2016) and consists of an adaptation of the US-wide default distribution factors for urban and rural roads – for Level-3 only, the user must specify in ***trafficCountInput*** Cell *M11* if either a rural or urban road is being designed.

Finally, the ***trafficCountInput*** spreadsheet performs one slight calculation on the hourly distribution factors (regardless of the data input level): the spreadsheet converts the 24 hourly distribution factors into three cumulative distribution factors in order to match Product-One 12-hour simulation time steps. The actual calculation is intuitive – the hourly distribution factors need just be summed together in order to obtain the cumulative distribution factors. Three cumulative hourly factors are defined for use with Product-One, their equations below:

- The “Early morning” distribution factor: comprising the 0:00 to 6:00 AM period
- The “daytime” distribution factor: which covers the 6:00 AM – 6:00 PM period
- And the “night-time” distribution factor: which spans over the 6:00 PM – 0:00 period

Cumulative hourly distribution factors from hourly factors

$$\begin{aligned} CHF_{earlymorning} &= \sum_{t=0:00}^{6:00\,AM} HF(t) \\ CHF_{daytime} &= \sum_{t=6:00\,AM}^{6:00\,PM} HF(t) \\ CHF_{nighttime} &= \sum_{t=6:00\,PM}^{0:00} HF(t) \end{aligned} \tag{9}$$

Note: Refer to section 3.8.1 – the ***trafficSimulator*** code – for an explanation on how these cumulative factors are used.

Finally, since the vehicles’ speed is also a consideration for the M-E computation of the hot-mix asphalt layers’ strength parameters (section 3.9.3), an interface is provided in the ***trafficCountInput*** spreadsheet for the end-user to load the project site’s speed records. These speed values must be in kilometers per hour [km/h], Product-One would convert units as necessary. Product-One default values are the maximum legal speeds for the different vehicle types in Uruguayan roads.

The ***trafficLoadInput*** spreadsheet is the front-end to input the available load records from a *Reporte 401* to Product-One. The end-user must provide the weight by axle (in metric tonnes, rounded by 0.5 ton) and the tire pressure for each vehicle category (in pounds per square inch [PSI][12]) and each level of load - “unloaded”, “partially loaded”, “fully loaded”, and “overloaded”.

Since the inquiry filed against the Uruguayan authority on the subject of traffic volume and load information for this project turned out fruitless, no Level-1 or Level-2 inputs can be provided with this release of Product-One. Blank tables were left for the end-user to write Level-1 and/or Level-2 data should these become available, and a Level-3 default scenario of all “fully-loaded” traffic in which all heavy vehicles carried their maximum legal load (MTOP, 2016), has been programmed.

In summary, Product-One would import the following into its core calculator. For the sake of interpretation of the following chapters and Product-One Matlab-format outputs, the names of the involved variables are also provided:

[12] Despite the effort to stick to the metric system as much as possible, the decision had been made during Product-One development to keep the tire pressure input in Imperial units [PSI] since it is still the customary unit in Uruguayan vehicle management practice.

- Base year AADT for the design lane – stored as *AADTBaseYear,*
- Vehicle speed for each category – stored as *AADTSpeed,*
- Traffic annual growth rate for each category – stored as *AADTGrowthRate,*
- Ratios of unloaded, partially-loaded, fully loaded, and overloaded vehicles in each category – stored in *trafficLoadPerc,*
- Load by axle by vehicle type and load level – stored in *trafficAxleLoad.*

The tire pressure value by vehicle type must be re-written directly into Product-One's ***MainCode.*** So the user must do if he or she intends to overwrite the default load ranges by axle type – which are defined in the ***axlesWeights.m*** script.

3.5 MainCode – Product-One's front-end script

The ***mainCode*** is one of Product-One's core components. This script is the only one the end-user must interact with in order to run a pavement analysis run with Product-One In order to start Product-One, the user must run the ***mainCode*** as a script from Matlab's or Octave's command window. Upon execution, the ***mainCode*** calls in sequence the scripts and functions listed below and performs the following tasks:

- Call the ***integrityCheck*** script, which checks whether the interpreter is either Matlab or Octave and adds the paths to all Product-One's functions to the interpreter's workspace.
- Ask the user to provide the name of the ***dataImport*** spreadsheet (after this point, Product-One would continue automatically, based on the input data and run configuration options fed from ***dataImport.***
- Import the design input data. This task is distributed over a number of auxiliary scripts, each of which read data of different categories from the ***dataImport*** spreadsheet. These are ***mainDataInput.m, trafficDataInput.m, materialsParameters.m,*** their names provide an intuitive idea of what category of inputs each deals with, thus saving the need for a more thorough explanation. Also if level-1 climate data has been made available, ***climateLoadLocal.m*** is also called. Otherwise, the ***climateSimulator*** function will be called in order to estimate the climate record at the project's site.
- Perform the pre-processing stage calculations (refer to figure 13):
  1. The ***climateScrambler*** function will be called to compute the predicted weather record from either the interpolated or local historical weather record.

2. The temperature profile in the trial structure's HMA layers -which only depends on the HMA layers themselves and the forecast weather – is calculated for the entire life-cycle by ***HMALayersTemperature***.
3. Preliminary calculations to determine the granular layers' moisture at any time) during the simulation (computing the Soil-Water Content Curve [SWCC] and hydraulic conductivity for the unsaturated state) are executed in ***kUnsatSWCC*** (section 3.10).
4. The design-lane AADT record retrieved from ***dataImport*** is firstly extrapolated over the pavement's design life in ***trafficSimulator*** and then is merged with the axle load information and converted to counts of axle passes over each simulation timestep (preset at 12h). This latter part is achieved when calling the ***trafficToAxles*** function. Finally, the MEPDG's formula (NCHRP, 2004) is utilized to obtain load frequency values for each axle pass from the vehicle's assumed speed, firstly by computing an average axle speed (when the axle type is shared by many vehicle types, the ***axleSpeedfromVehicleSpeed*** function is called), then by calling the ***HMALoadFrequency*** function.
5. Obtain the dynamic moduli (E*) and Poisson coefficients for each HMA layer at each moment in time (related to a temperature value) and load level. To this end, the ***HMAModulus*** and ***HMAPoisson*** functions are called separately for each of the many axle types Product-One has been programmed to handle, and so are their outcomes (E* and υ) stored.

- Start the pavement simulation and run the time-dependent calculations contained within. The time-dependent calculation loop starts by solving a water content balance in the granular layers and the subgrade in order to obtain their current-time volumetric moisture content – these calculations are done in the ***climateModule.m*** script. Then, the granular materials' unsaturated-state resilient moduli are computed with the Witczak et al. equation (NCHRP, 2004) in ***calculateMR.*** Once completed, the second core component of Product-One is launched: the ***MLE_FrontEnd*** script is called to run the stress and strain calculations over the entire trial pavement structure for all the load levels stated for each axle type. The ***MLE_frontEnd*** performs successive calls to ***MLE_sigma***, which computes the radial, tangential, and vertical stresses that would develop in the trial structure due to a single load application, and would compose the effects for multiple-wheel axles thereafter. The time-dependent calculation loop closes with the distress prediction functions, namely rut depth , bottom-up and top-down cracking, and IRI and PSI estimations.

- Finally, a series of report-creating functions are called to prepare graphical and numerical outcomes. Five plotting functions, each similar in nature but with different names, are called in sequence to provide intuitive graphical outputs of the forecast weather variables, predicted traffic, temperature in the HMA layers, moisture in the granular layers, and predicted distress level. Additionally, the ***exportOutput*** script will write these outputs' numerical values to a specially-crafted spreadsheet for the end-user to further process if deemed necessary.

3.6 Climate data interpolator (Level of input 2-3)

The ***climateSimulator*** script, which is called directly from the ***mainCode*** upon execution, and runs automatically, generates an hourly estimate of the five weather variables of interest (namely air temperature, humidity, wind speed, solar radiation, and rainfall) at the project site from the available INIA records for the 2010-2018 period. The "interpolator" is an implementation of Inverse Distance Weighted (IDW) interpolation technique (Wei and McGuinness, 1973), which allows to infer the value of any variable P at any location from measured values at point locations [namely P_i]. This interpolation technique was initially proposed for interpolating rainfall records (Wei and McGuinness, 1973; Chow et al., 1993) but its usage has been extended beyond the hydrological research as a standard geospatial analysis tool (Chang, 2019).

An example of an IDW interpolated field is shown in figure 14 below: the 48-hr accumulated rainfall readings from seven out of the eight INIA weather stations (the Rocha station did not have data for that event) for the rain event which spanned between the 6th and the 8th of February of 2018 have been used as point values, and the IDW interpolated rainfall reading for the entire country has been generated in a GIS application [QGis 2.18].

IDW interpolation for variable P at location (x, y) from n known point values at stations located at (x_i, y_i) respectively.

$$P(x,y)=\frac{\sum_{i=1}^{n}\frac{P_i\times 1}{d_i^2}}{\sum_{i=1}^{n}\frac{1}{d_i^2}} \tag{10}$$

where

$$d_i^2=(x-x_i)^2+(y-y_i)^2$$

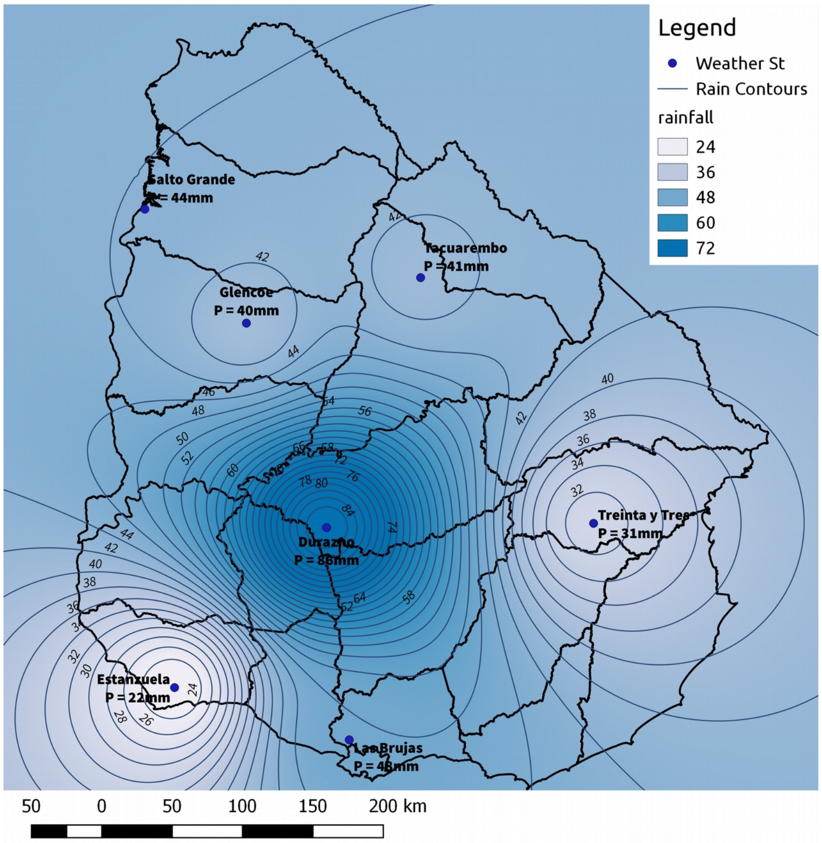


Figure 14: Example of Inverse Distance Weighed interpolation. 48-hour rainfall record for 2018-02-08

An extra enhancement has been added to the IDW formula in its implementation within Product-One: the ***climateSimulator*** script would define a time stamp vector spanning over the entire 2010-2018 period at a 1-hour step, and check how many INIA weather stations have data at such time moments prior the IDW interpolation calculations. This error check has been implemented using an additional binary value bin_i which equals 1 if station *i* has weather records for that moment and 0 if otherwise. Such a construct allows the ***climateSimulator*** to "smartly" interpolate at each moment in time differently depending on how many stations are actually providing data and so preventing foul zero values to being added in.

3.7 Climate records generator – *climateScrambler.m*

This routine converts the source site's meteorological data (either actual record (level 1) or IDW interpolated record (level 2-3). It is executed right after the ***climateSimulator.m*** finishes. It is based on the EICM module that provides the site project's weather variables forecast for the MEPDG, but it does not repeat the measured meteorological variables' values in a cycle (NCHRP, 2004), but creates a pseudo-random weather pattern from the (actual or IDW-interpolated) records. The ***climateScrambler*** would firstly make use of a pseudo-random sequence of years from the 2010-2018 period and as long as the intended pavement's design life, and then construct the forecast weather time series drawing the recorded yearly variables accordingly. The pseudo-random year vectors are tied to a single integer *randomSeed* (whose value can range from 0 to 50 and the user must define in the ***dataImport*** spreadsheet). Upon execution, the ***climateScrambler*** would pick the randomly-generated years vector matching that random seed and construct the forecast weather variables as described previously.

The ***climateScrambler*** would also compare the given project site location, design life, and random seed value against a cached set of values (those of the previous run) to save running time, especially when running Product-One repeatedly for the same project site, design life, and random seed. If all of these variables match their cached counterparts, the ***climateScrambler*** would just retrieve the cached simulated weather records (stored in the ***climateMatrixCached.mat*** data file). Otherwise, the ***climateScrambler*** will recalculate the forecast weather variables and store them in the cache file. However, the recalculation of climate variables can be forced by the user by setting the *forceClimateRecalculation* variable to 1 in the ***dataImport*** spreadsheet.

3.8 Pre-processing routines – Traffic analysis

Two scripts predict the axle traffic volume at each simulation time step from the aforementioned traffic inputs (section 3.4): ***trafficSimulator.m*** and ***trafficToAxles.m***. These scripts run in sequence when Product-One is executed, they are called from the ***MainCode***.

3.8.1 From AADT to predicted traffic flow – trafficSimulator.m

The outcome of the ***trafficSimulator.m*** routine within Product-One is twofold. First, it calculates each year's AADT for each vehicle category and then distributes these AADT values over time to generate the predicted traffic volume at each 12-hour timestep (stored in the *shortTimestamp* vector). Its inputs are the *shortTimestamp,* the provided AADT for each vehicle category for the base year (*AADTBaseYear*)*,* the yearly traffic growth rate (*AADTGrowthRate*), and the monthly, daily, and hourly distribution factors for each vehicle category (*AADTMonthlyDistr, AADTDailyDistr, AADTHourlyDistr*). This script outputs the number of vehicles in each category at each timestamp (*designTraffic,* a matrix whose size is that of *shortTimestamp* by the number of vehicle categories), which is passed on to ***trafficToAxles.m***, and the projected AADT for each year in the pavement's service life (*designAADT,* which is only used in the results reports).

The predicted yearly AADT (which is calculated for all the years covered in the pavement's service life span) is computed using the formula below:

Estimate following year's AADT (for any given category) from current year AADT

$$AADT_i\left(year_{n+1}\right)=AADT_i\left(year_n\right)\times\left(1+g_i\right) \tag{11}$$

Where g is the yearly AADT growth rate for the i-th vehicle category. It must be noted that ***trafficSimulator.m*** assumes that the traffic growth rates remain constant over the entire pavement's service life.

Secondly, the ***trafficSimulator*** script computes *designTraffic,* i.e. the number of passing vehicles, within each category at each 12-hour period by reading each *shortTimestamp* value and applying the corresponding monthly, daily and "hourly" factors – an adaptation from the original MEPDG framework, which estimates traffic on an hourly basis (NCHRP, 2004). It must be remembered that since each *shortTimestamp* moment in time is either at 6:00 AM and 6:00 PM, the hourly factors correspond to "early morning [0:00 to 6:00 AM]", "daytime [6:00 AM – 6:00 PM]", and "night [6:00 PM-0:00]", hence Product-One will use separate equations for accumulating 12-hour traffic depending on the hour of the day within each *shortTimestamp*:

- For a *shortTimestamp* that occurs at 6:00 AM, accumulate that day's "early morning" traffic plus previous day's night traffic (which requires two sets of monthly, daily and hourly factors),
- Otherwise, only accumulate that day's "daytime" traffic (use only the current date's monthly, daily and hourly factors).

The above statements were implemented with the equations that follow:

Compute the traffic volume of category-i vehicles at any given time moment t_n. Case t_n is at 6:00 AM

$$\begin{aligned}Trf_i\left(t_n\right)=AADT_i\left(year\left(t_n\right)\right)\times MF_i\left(month\left(t_n\right)\right)\times DF_i\left(day\left(t_n\right)\right)\times 0.01\times CHF_{earlymorning,i}\left(t_n\right)+\ldots\\ \ldots+AADT_i\left(year\left(t_{n-1}\right)\right)\times MF_i\left(month\left(t_{n-1}\right)\right)\times DF_i\left(day\left(t_{n-1}\right)\right)\times 0.01\times CHF_{nighttime,i}\left(t_{n-1}\right)\end{aligned} \tag{12}$$

Compute the traffic volume of category-i vehicles at any given time moment t_n. Case t_n is at 6:00 PM

$$Trf_i\left(t_n\right)=AADT_i\left(year\left(t_n\right)\right)\times MF_i\left(month\left(t_n\right)\right)\times DF_i\left(day\left(t_n\right)\right)\times 0.01\times CHF_{daytime,i}\left(t_n\right) \tag{13}$$

3.8.2 Predicted design-lane traffic to axle count conversion – trafficToAxles.m

The second traffic-processing script takes the *designTraffic* variable from the ***trafficSimulator*** and solves the number of passes of each axle type, while also classifying them by weight based on the ratios of unloaded, partially loaded, fully-loaded, and overloaded vehicles. Product-One uses the same ranges as the MEPDG (NCHRP, 2004) to sort passes of axles of a given type with varied loads, but appeals to six categories to sort axles or axle groups so as to comply the closest with the normalized axle types recognized by the Uruguayan highway authority (figure 8):

- Light-weight single axle, only appearing in cars (A11) category,
- Heavy single-wheel single axle (front axle of all heavy vehicles),
- Heavy dual-wheel single axle (4-wheel axle, the mandatory configuration for all heavy vehicles with a single rear axle),
- Heavy single-wheel tandem axle (4-wheel axle, the front axle of C22 and O22 vehicles only),
- Heavy Non-homogeneous tandem axle (6-wheel axle, the rear axle of O12 and O22 vehicles only),
- Heavy dual-wheel tandem axle (8-wheel axle, mandatory configuration all trucks and trailers with tandem axle must comply with),
- Heavy dual-wheel tridem axle (12-wheel axle, mandatory for all heavy vehicles with a rear tridem axle).

The inputs to ***trafficToAxles*** are the vehicle counts at every time moment (by vehicle category, the *designTraffic* matrix computed in ***trafficSimulator***), and two load-related pieces of data, the ratio of unloaded, partially-loaded, fully-loaded, and overloaded vehicles in each category (stored in the *trafficLoadPerc* vector variable), and the load by axle (or axle group) record for each vehicle category and load level (the *trafficAxleLoad* matrix). This routine's output consists of a series of matrices whose columns are the axle count at each moment in time (matching the 12-hour timestamps stored in *shortTimestamp*), for each of the six categories of axles mentioned in the above paragraphs, and for each axle load (their respective names being *axesSingleLight, axlesSingle6, axlesSingle105, axlesTandem18, axlesTandem10, axlesTandem14, axlesTridem,* a reference to their configurations and maximum legal loads). A step-wise procedure, which is summarized below, is followed to calculate these outputs:

1. The ancillary ***axleWeights*** script is executed. This code defines the load ranges in which the axles of each category are to be sorted.

2. For each vehicle category, the percentage of ith-category vehicles in each load level, the type of axles the vehicles have, and the corresponding axle loads are retrieved from the input matrices.
3. Once the axle types of the vehicle are recognized, their load is read and compared against the load ranges generated in step 1 Thus, the “load slots” in which to add these vehicles’ axles to the total axle count matrices are determined.
4. Add the axles due to the passing vehicles of the i^{th} category to the corresponding axle count matrix, at the corresponding load level.

For the sake of clarification, let the following succinct example: on a certain design, 10 fully loaded O12 buses are to pass at a given time step (t_n) (6 and 14 ton are their respective axle weights). When ***trafficToAxles*** converts the O12 buses into axle counts, it will recognize non-zero load values in columns 3 and 9 of the *trafficLoadMatrix* (single-wheel single axle and non-homogeneous tandem axle). The routine’s code has been programmed to tie such positions within *trafficLoadMatrix* to the corresponding axle types so that the O12’s axles can be added to the single-wheel single axle and non-homogeneous tandem axle under the matching load level (6-ton and 14-ton respectively). Since each O12 bus has one axle of each type, then 10 axles will be added to the *axlesSingle6* matrix in the nth row and column corresponding to 6 tons and 10 axles will be added to *axlesTandem14,* in row n and the column corresponding to a 14-ton load.

Once the calculations are complete (the axle counts were completed for all unloaded, partially-loaded, fully-loaded, and overloaded vehicles), the function reports the axle counts to the ***mainCode***. These variables are again used in the multi-layer linear elastic stress and strain calculator (section 3.11.4).

3.8.3 Axles’ load area and contact radius – *wheelFootprint* function

The multi-layer linear elastic stress and strain calculator that runs in Product-One’s core would assume the traffic load would be applied to the pavement over a circular contact area – its radius would depend on the actual load the vehicle’s axles are placing on the pavement and the vehicle’s tire pressure. The ***wheelFootprint*** function would calculate such contact area radii for the different axle types handled by Product-One and each one’s load range, using the equation below:

Tire-pavement contact area radius for a given axle type

$$a=\sqrt{\frac{\frac{P}{n_{wh}}}{\pi \times p}} \tag{14}$$

In equation 14 above P is the total axle load [for calculation purposes in metric tonnes], n_{wh} is the axle's number of wheels, and p is the wheels' tire pressure. The ***wheelFootprint*** function receives *P* in metric tonnes, the tire pressure *p* in pounds per square inch, and returns the radii *a* in centimeters, all units conversions are done internally.

3.8.4 Traffic load frequency from vehicle speed – the *HMALoadFrequency* function

The last traffic-related pre-processing calculation is tied to the time-dependent nature of the HMA-layers elastic moduli: the load frequency associated with the axle's speed must be computed in order to properly quantify the HMA-layers' dynamic moduli. To this end, the default MEPDG's formula for the duration of the load cycle – as printed in Appendix CC3 of NCHRP (2004) – and its conversion to the frequency domain (observing the corrections suggested by Al Qadi et al, 2008) have been implemented in the ***HMALoadFrequency*** function:

Duration of load application and load frequency from axle's speed

$$time_{load}\left[sec\right]=\frac{L_{eff}}{17.6\times v}$$

$$freq_{load}\left[Hz\right]=CF\frac{1}{time_{load}} \tag{15}$$

$$CF=0.03\times z\left[cm\right]+0.2333$$

Where *v* is the axle's speed in miles per hour [mph], and L_{eff} [in] is the "effective length" of the load application – which is based on the assumption of the axles' wheels placing a haversine-shaped load on the pavement. *CF* is the correction factor suggested by Al Qadi et al (2008), the formula to compute it at different depths *(z)* in the pavement structure is also given above.

L_{eff} is computed at the mid-depth of each layer with the formula below – a simplified expression for single axle loads, but also valid for multi-axle load applications, according to what stated in Appendix CC3 of the MEPDG guide (NCHRP, 2004):

Effective length of load application

$$Leff_i=2\left(a+Zeff_i\right)$$

$$Zeff_i=0.5\times h_i\left(\frac{E_i}{E_{SG}}\right)^{\frac{1}{3}}+\sum_{j=1}^{i}\left[h_j\left(\frac{E_i}{E_{SG}}\right)^{\frac{1}{3}}\right] \tag{16}$$

Where E_i is the elastic modulus (i.e. dynamic modulus) of HMA layer *i* and E_{SG} is the subgrade elastic modulus; *a* is the radius of the tire-pavement contact area.

A major simplification has been set forth in Product-One to avoid a lengthy iteration loop – that of requiring the HMA layers' *E** values to compute the load frequency which will be then used to compute the layers' *E**: a high-frequency load (10^9 Hz) *E** is assumed as "default value" for the load frequency calculation. Similarly, the subgrade's resilient modulus at saturated state is used as the value for E_{SG}, regardless of the subgrade actually being moist yet not saturated.

The ***HMALoadFrequency*** function receives the thicknesses of the HMA layers and the load area radius in centimeters, and the axle's speed in km/h. This function is called from the ***mainCode*** for the many types of axles handled by Product-One. The time of load for each load level and for each axle type is returned in seconds and the corresponding load frequency in Hertz. All unit conversions are done internally.

3.9 Pre-processing routines – Asphalt concrete layers climate-dependent properties: Temperature, Dynamic Modulus, and Poisson Modulus.

The characterization of HMA materials for mechanistic-empirical design depends both on the load being applied and the material's temperature at the time of the load application. This section reviews the approach utilized to compute the pavement's temperature during its service life is provided below.

The ***mainCode*** calls the ***HMALayersTemperature*** function, which computes the temperature of the pavement's HMA surface and within each of the pavement's asphalt concrete layers based on a start condition and the simulated atmospheric conditions. Different models are used to compute the pavement's surface temperature and the HMA layers' temperatures, a description of each follows.

3.9.1 AC Surface temperature prediction

The ***HMALayersTemperature*** calculates the HMA surface temperature from the simulated weather variables. The methodology proposed by Soulamanian and Kennedy (1993) has been adopted in this software implementation, because it is based on the original heat exchange models embedded in the MEPDG's Integrated Climate Model (developed by Dempsey et al., 1986 and FHWA, 1993) but provides formulas to quantify the heat exchange between the asphalt surface and the atmosphere on an hourly basis.

The heat flow between the atmosphere and the pavement surface is governed by the Scott and Berg's formula (equation 17). This formula is utilized to solve the pavement's surface temperature by replacing each heat flux component – described in detail in the following lines – by its numeric value and solving for the unknown variable.

Heat flux between the atmosphere and the pavement surface

$$q_s + q_a - q_c - q_k - q_r = 0 \tag{17}$$

The ICM (FHWA, 1993) and Soulamanian and Kennedy (1993) framework neglect the heat exchange due to transpiration, condensation, evaporation, and sublimation from the pavement's surface**.** The remaining heat flow components can be computed with the following equations:

Heat due to solar radiation q_s

$$q_s = \alpha \times SR_{\text{in}} \tag{18}$$

where α is the asphalt surface absorptivity, and SR_{in} is the net incident solar radiation. A single constant value of 0.93 (level-3 default according to the MEPDG guide (NCHRP, 2004) and Soulamanian and Kennedy (1993)) has been adopted for α.

Additional equations are provided by Soulamanian and Kennedy (1993) to estimate the incident solar radiation at any given time from cloud cover records and a number of additional inputs, such as the project site's latitude. However, there is no need for implementing these in Product-One as actual solar radiation values are available (either from the INIA database or from a compatible Level-1 source). Thus, the forecast solar radiation record computed in the ***climateScrambler*** is utilized in this formula.

Equation 19, which is derived from the Stephan-Boltzmann black body radiation model, is used to compute the amount of heat the atmosphere radiates:

Heat due to absorbed atmospheric radiation q_a

$$\begin{gathered} q_a = \varepsilon \times \sigma T_{air}^4 \\ \varepsilon = 0.77 - 0.28\left(10^{-0.074 \times \rho}\right) \\ \rho = Hum \times \rho_{sat}\left(T_{air}\right) \end{gathered} \tag{19}$$

where T_{air} is the air absolute temperature (in degrees Kelvin), σ is the Stephan-Boltzmann constant [5.68×10^{-8} Watt/(m^2K^4)], and ρ is the vapor pressure [in mm of mercury] at temperature T_{air} and air humidity content level *Hum* [as ratio] $\rho_{\mathbf{sat}}$, which is the saturation vapor pressure at temperature T_{air} is a commonly known value – Product-One retrieves $\rho_{\mathbf{sat}}$, from a lookup table stored in ***pVapTable.mat.***

The pavement similarly radiates heat to the atmosphere. The black body model can be applied to the HMA surface to quantify the amount of heat exchange (equation 20):

Heat loss due to emission from the asphalt surface q_r

$$q_r = \varepsilon_{HMA} \times \sigma T_s^4 \tag{20}$$

where ε_{HMA} is the HMA emissivity, and T_s is the pavement's surface absolute temperature, in degrees Kelvin. The MEPDG's default value for ε_{HMA} (ε_{HMA} = 0.93) has been adopted in Product-One's implementation.

The heat exchanged by convection between the pavement and the atmosphere is computed as follows:

Heat exchange by convection q_c

$$\begin{gathered} q_c = h_c\left(T_s - T_{air}\right) \\ h_c = 698{,}24\left[0.00144\, T_m^{0.3} \times Wspd^{0.7} + 0.00097\left(T_s - T_{air}\right)^{0.3}\right] \end{gathered} \tag{21}$$

where T_s and T_{air} are the temperatures of the asphalt surface and the surrounding atmosphere respectively and h_c is the surface's coefficient of heat transfer. Vehrencamp's formula (Vehrencamp, 1953; Dempsey et al., 1986; Soulamanian and Kennedy, 1993) is used to calculate h_c from meteorological variables, namely the average between the surface and the air temperature ($T_{m,}$ in degrees Kelvin) and the wind speed $W_{spd,}$ in meters per second.

Heat conducted to and from the pavement q_k

$$q_k = -k\frac{\left(T_{depth} - T_s\right)}{d} \tag{22}$$

In which k is the HMA thermal conductivity. T_{depth} is the temperature of the HMA material (assumed as either the placement temperature of the uppermost layer or the top-most layer temperature in the previous time step), and T_s is the HMA surface temperature. Meanwhile, d is the depth from the pavement surface [in meters] at which T_{depth} is taken. The MEPDG default value for k is used also in Product-One: k = 1.0811 Watt/(m deg C).

Finally, when the heat transfer components in equation 17 are replaced by their respective expressions and the equation is rearranged to solve for T_s it yields:

Equation to solve the current time pavement surface temperature from the heat transfer balance

$$\alpha \times SR_{\mathrm{in}} + \varepsilon \times \sigma T_{air}{}^4 - h_c\left(T_s - T_{air}\right) - \frac{k}{d}\left(T_s - T_{depth}\right) - \varepsilon_{HMA} \times \sigma T_s{}^4 = 0 \tag{23}$$

Which is solved numerically with a Newton-Raphson method (Quarteroni et al., 2000), utilizing T_s from the previous time step as start value and with a tolerance on the result of 3%.

3.9.2 AC Layers temperature prediction

Once the surface temperature is solved, the Bells2 equation (FHWA, 2000; reproduced as Equation 24) is applied to compute the temperature profile throughout the entire depth of HMA material. This calculation is done in Product-One by the ***HMALayersTemperature*** function**.**

Bells2 equation to obtain the current HMA temperature at time t and a depth z from the surface

$$\begin{aligned} T(z,t) &= 2.78 + 0.912\, x T_s(t) + \left[\log_{10}(z[mm]) - 1.25\right] \times \ldots \\ \ldots & \left[-0.428 \times T_s(t) + 0.553 T_s(t-1) + 2.63 \times \sin_{18}(hr - 15.5)\right] + \ldots \\ \ldots & \, 0.027 \times T_s(t) \times \sin_{18}(Hr - 13.5) \end{aligned} \tag{24}$$

Where *z* is depth at which to calculate the material's temperature (in millimeters), $T_s(t)$ and $T_s(t\text{-}1)$ is the surface temperature at current time *t* and at the day before (*t-1*), and sin_{18} is a purposefully-defined sinusoidal function whose period is 18 hours. It has been implemented as the ancillary ***sin18*** function.

The result of this module (temperature of the pavement's surface and at half the depth of each HMA layer, in degrees C) is both sent to the plotting tools (Software's output *figure 31*) and to the HMA dynamic modulus calculator (section 3.9.3). Figures 15 to 17 present an example test run of the ***HMALayersTemperature.*** The temperature profiles over a one-week period for a pavement structure with two HMA layers is reproduced in figure 15. The predicted temperature trends for each layer follow a daily pattern consistent with that period's weather, with greater daily temperature increase on clear days and a more assuaged cycle on overcast, rainy days; thus proving the soundness of the coded formulas. The air temperature profile is the dashed line in figure 15, and figures 16 and 17 portray the solar radiation and rainfall time series respectively.

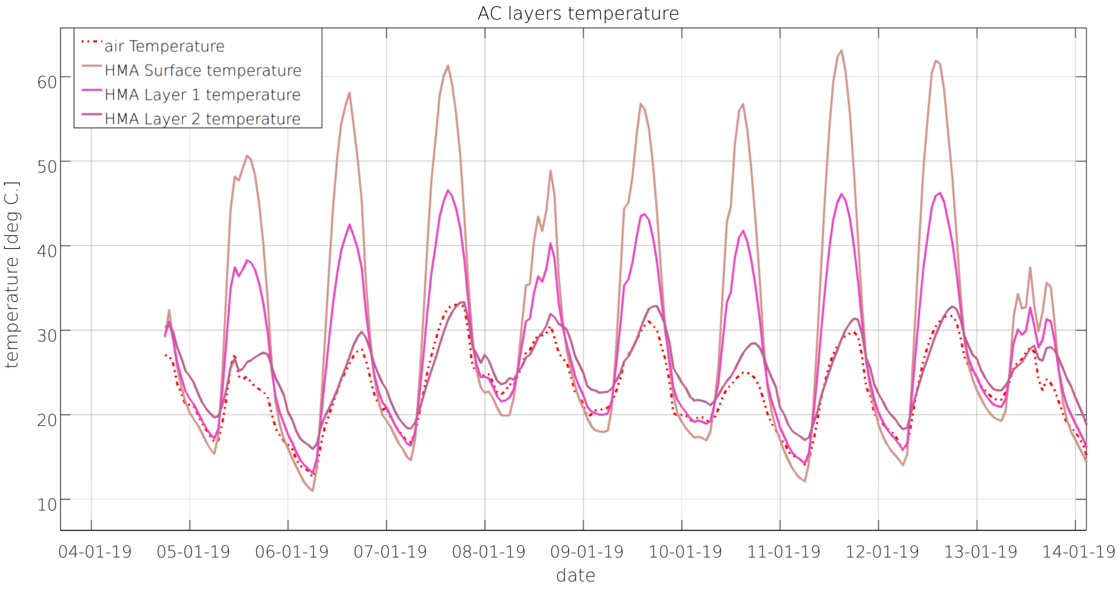


Figure 15: Simulated HMA temperature profile for a trial structure with two HMA layers. One-week period

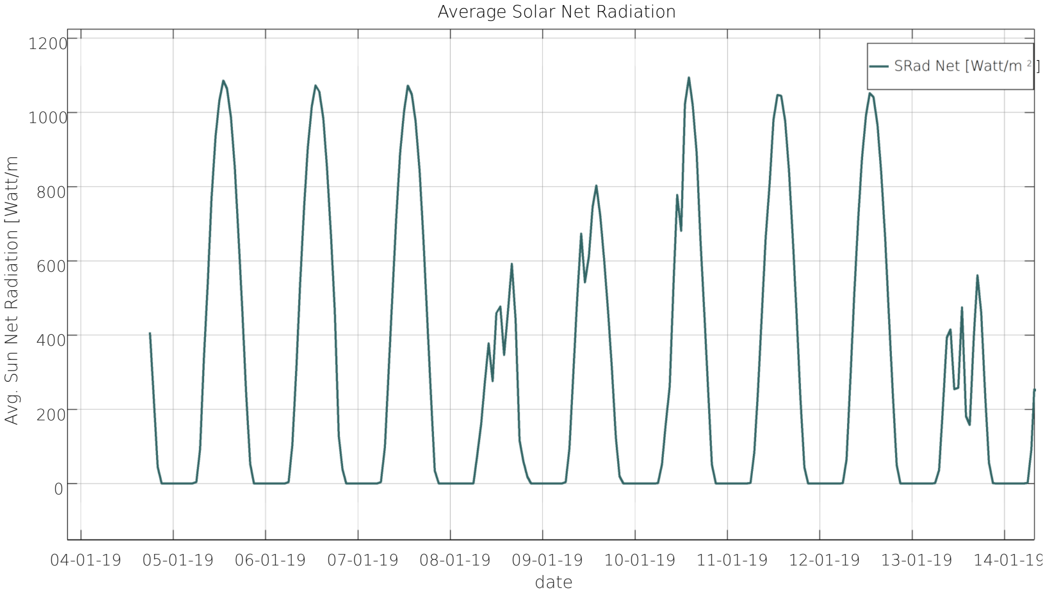


Figure 16: Solar radiation series for the one-week analysis period

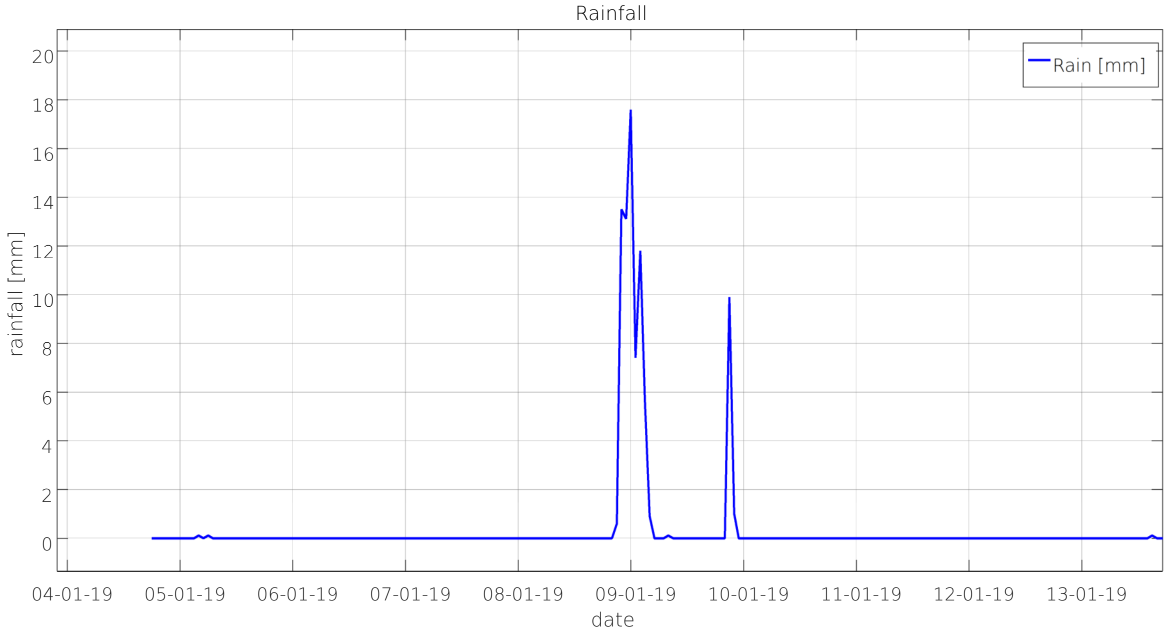


Figure 17: Rainfall time series for the one-week analysis period

3.9.3 AC Dynamic modulus and Poisson ratio prediction

Finally, as all input data needs have been fulfilled, the HMA layers' strength properties – the dynamic modulus E* and the Poisson coefficient υ – can be computed along the pavement's service life and in correspondence with each axle load level. For this purpose, the Asphalt Institute equations for the determination of E* – as printed by Huang (2004) – have been implemented in Product-One instead of the Master Curve methodology that runs within the MEPDG, the underlying reason is twofold:

- The Asphalt Institute formula would require fewer mix properties as inputs, and testing infrastructure for these properties is readily available.
- The Uruguayan materials and construction standard specification documents (MTOP, 1990; MTOP, 2003) utilize the same mix properties that appear in the Asphalt Institute equations to characterize standard HMA mixes (thus indirectly providing a set of level-3 default values to compute E*), whereas no reference is made therein to the Master Curve.

Asphalt Institute equation for the calculation of the E for HMA materials*

$$\begin{gathered} E^{*}\left[PSI\right]=100000\times 10^{\beta_1} \\ \beta_1=\beta_3+0.000005\times\beta_2-0.00189\times\beta_2\times freq^{-1.1} \\ \beta_2=\beta_4^{0.5}\times T\left[F\right]^{\beta_5} \\ \beta_3=0{,}553833+0{,}028829\left(P_{200}\times freq^{-0.1703}\right)-\ldots \\ \ldots\text{-}0.03476\times V_a+0.070377\lambda+0.931757\times freq^{-0.02774} \\ \beta_4=0.483\times V_b \\ \beta_5=1.3+0.49825\log_{10}\left(freq\right) \\ \lambda=29508.2\left(P_{77F}\right)^{-2.1939} \end{gathered} \tag{25}$$

In the above equation, *freq* is the load frequency in hertz, T[F] is the HMA material temperature in degrees Fahrenheit, P_{200} is its percentage passing the #200 mesh, V_a is the mix air void content, V_b is the bitumen volume content [in percentage] and P_{77F} is the mix's penetration at 77 degrees Fahrenheit [25 degrees C].

Meanwhile, the Poisson ratio for the HMA materials (which is also temperature- and load-dependent) is computed with the MEPDG's formula for level-2 input data:

Poisson coefficient for HMA materials

$$\upsilon=0.15+\frac{0.35}{1+e^{\left(-1.63+3.84\times 10^{-6}E^{*}\right)}} \tag{26}$$

Two separate functions have been implemented as part of Product-One to perform the above-mentioned calculations, namely ***HMAModulus*** and ***HMAPoisson.*** These run separately for all the axle types handled by Product-One, and their outputs are 3-D arrays (stacks of matrices), each stack's size is the number of HMA layers in the trial structure by the number of load values in the load range for each axle type, and the number of stacks is given by the pavement's service life (a stack is generated for each 12-hours *shortTimestamp*).

The ***HMAModulus*** contains the implementation of the Asphalt Institute equations for all levels of input, the mix properties are received as imported in the ***materialsInput*** scripts, the frequency value for different load levels and the different axle types is as computed in ***HMALoadFrequency,*** and the temperature values are the output of the ***HMALayersTemperature*** function. When called from the ***mainCode***, the *HMAParameters* matrix defined in the ***materialsInput*** scripts (which contains the values of the four variables for all the HMA layers in the trial pavement structure) is passed as input. Meanwhile, the ***HMAPoisson*** function only requires the previously calculated *E** values for its output to be delivered.

No asphalt materials aging model (such as that programmed into the MEPDG) has been implemented in this release of Product-One, meaning that no hardening of the HMA materials as a consequence of their aging is being modeled herein, but the current version of ***HMALPoisson*** and ***HMAModulus's*** source code should enable such an upgrade without a sizable additional coding effort.

3.10 Pre-processing routines – Granular materials SWCC curves and permeability for the unsaturated state – the *kUnsatSWCC* function

This is an ancillary calculation to the infiltration module. It computes the soil-water characteristic curves (SWCC, relationships between the soil's volumetric water content and matric suction) for all granular materials and the subgrade soil according to the Fredlund and Xi (1994) equations, which are also featured in the MEPDG These are used afterward to compute the materials' hydraulic conductivity in the unsaturated state (Fredlund et al., 1994), for different degrees of matric suction.

The SWCC curve for a given granular material is given by the following equations:

SWCC curve for a granular unbound material: Relationship between matric suction and volumetric water content

$$\theta(\psi)=C(\psi)\left[\frac{\theta_{sat}}{\left(\ln\left[e+\left(\frac{\psi}{a}\right)^{b}\right]\right)^{c}}\right]$$

$$C(\psi)=\frac{1-\ln\left(\frac{\psi}{h_r}\right)}{\ln\left(1+\frac{1.45\times10^{5}\mathrm{PSI}}{h_r}\right)} \tag{27}$$

Where θ is the material's volumetric water content, ψ is the suction [in PSI], and θ_{sat} is the material's saturated volumetric water content. The four SWCC parameters (namely a, b, c, and h_r) have been defined in equation 6, which is implemented in the ***dataImport*** spreadsheet**.**

With the above results, the hydraulic conductivity value for the unbound materials can be computed with the Fredlund et al. (1994) equation below, which is derived after a numerical integration of the θ-ψ relationship:

Hydraulic conductivity in the unsaturated state as a function of matric suction

$$k(\theta_i) = k_{Sat} \times \left[\frac{\sum_{j=i}^{m} \frac{2(j-i)+1}{\psi_j^2}}{\sum_{j=i}^{m} \frac{2j-1}{\psi_j^2}} \right] \tag{28}$$

Equation 28 above is intended to be applied onto numerical arrays of related θ-ψ values, the formula will return the material's hydraulic conductivity (in the same units as k_{sat}, the saturated-state conductivity) for the θ_i-ψ_i pair. The *i = 1* case corresponds to θ_i close to θ_{sat} and so ψ_i almost equal to zero, while the *i = m* case corresponds to a suction pressure of 1000 MPa (or 145.05 kPSI maximum value suggested by Fredlund et al. (1994) for calculation purposes).

The output of ***kUnsatSWCC*** is a 3-D array, each of its stacks containing a three-column lookup table for each granular material in the trial pavement structure. The first column is the matric suction domain (from 0 to 1000 MPa), the second column is the corresponding volumetric water content (computed with equation 27) and the last column is the corresponding hydraulic conductivity value in such unsaturated conditions (after equation 28). All these three variables are utilized by the time-dependent moisture prediction module to provide estimates of the water runoff seeping out of the pavement structure at any time.

3.11 Pavement Performance Simulation routines

Product-One's core modules feature mathematical models for those time-dependent processes that accrue during the pavement's service life and are intertwined amongst each other. Conceptually speaking, these constitute Product-One's "Pavement performance simulation" component. A description of these modules follows, it covers the module's theoretical background, their mathematical formulation, and their implementation into Product-One.

3.11.1 Rainfall infiltration and runoff model – the *climateEffectsModule* script.

This script – the first time-dependent component to be called upon execution – computes the amount of rainfall that infiltrates through the cracks on the pavement surface and performs a water content balance to determine the granular layers' and subgrade's moisture content. Its output is threefold:

1. how much rainfall infiltrates through the pavement's surface cracks,
2. how much escapes the pavement as surface runoff,

3. calculates each granular layer's and the subgrade's volumetric water content.

This module's "time-dependent" nature is due to the fact that the rainfall infiltration estimation is tied to the degree of alligator, longitudinal, and transverse cracking, the extent of which is calculated later by the distress prediction modules.

The infiltration prediction methodology adopted for Product-One differs slightly from the MEPDG's proposed framework: the pavement's potential (maximum) infiltration rate is computed at each simulation *shortTimestamp* based on the exact extent of surface cracking instead of relying solely on an ordinal percentage-based rule. Additionally, the guidelines to the MEPDG's rule to state what percentage of rainfall would infiltrate the pavement take into account concerns not directly related with the cracking rate increase but with the existence of drainage systems and the type of shoulder to be built at the pavement's edges. Meanwhile, Product-One's methodology would actively relate the pavement's maximum rate of infiltration with its degree of surface distress, thus penalizing poorly-designed, prone to premature cracking structures. However, since no shoulder-pavement cracking prediction model has been launched with the MEPDG and hence not been programmed within Product-One, this software would not be able to account for the infiltration process that may take place at the pavement-shoulder joint (or cracked joint), reported to be a major source of infiltration (NCHRP, 2004).

The infiltration calculations are done over a 1.00 m^2 of pavement area. The maximum amount of infiltration is computed by simply multiplying the maximum infiltration rate per meter of cracks by the total length of the cracks in the pavement. A value of 0.10 ft^3/h (recommended for bituminous pavements according to (FHWA, 1993)) has been adopted as the maximum infiltration rate per foot of crack [0.0093 m^3/h per meter of crack], while the length of cracks is estimated from the reported cracked values from the previous *shortTimestamp* as follows:

- Alligator cracks are assumed as square blocks 0.10m by 0.10m in size
- Longitudinal (top-down) cracks are reported by Product-One in meters/km (the unit conversion from feet per mile, as estimated by the MEPDG formulas, is done in the distress prediction functions). Thus, the total length of such cracks in a 1-meter-long stretch of road is 0.001 times the reported value.
- Although not implemented in this release of Product-One, the total length calculator also assumes values for reflective cracks, these are regarded as block cracks 3.6m by 4.5m in size, the typical size of concrete pavement slabs in Uruguayan Pavements.

In conclusion, the infiltration rate at any given *shortTimestamp t* is the minimum of the actual rainfall that took place at that moment in time and the maximum infiltration rate. Consequently, the difference between the two values is the amount of surface runoff that took place during that period.

Only the computed actual infiltration rate is passed on to the granular layers' moisture content calculator, whereas the two variables are retrieved by the plotting tools and graphed along with the rainfall time series – figure 18 is an example (from this function's bug-checking tests) for a pavement structure in which alligator cracking was forced to increase at a constant rate at each time step and was subject to a simulated rainfall pattern stretching over a two-year period.

3.11.2 Moisture prediction in granular layers and subgrade

This is the infiltration module's second component, it computes how much water is retained in each pavement layer and the top-most subgrade soil, by accounting both for the inflow that infiltrated through the pavement surface, and the runoff that may leave the road embankment sideways.

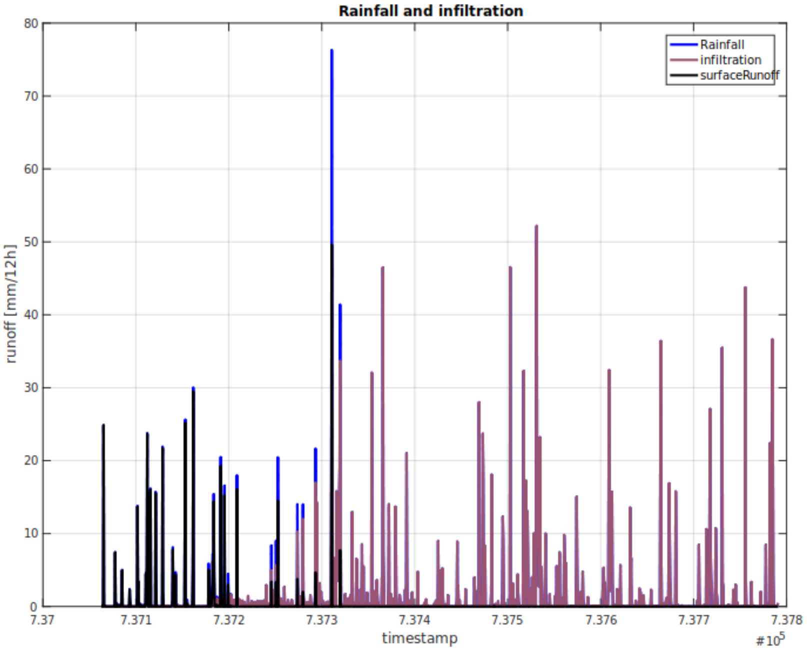


Figure 18: Infiltration and runoff for a pavement with constantly-increasing cracking

Equation 29 below is the mathematical expression of the water content balance that is formulated at the i-th granular layer. It must be highlighted that the assumption of water entering the structure only as infiltration from the roadway surface is underlying in equation 29**.** In other words, this formulation disregards any inflow from the underground water table – which hardly ever is a concern in Uruguayan pavement practice, as highways are mostly built over an embankment or are fringed with longitudinal ditches to collect surface runoff and lower the water table height.

Water balance equations at layer i and time t_k (unit volume computation)

$$\theta_i(t_k) \times h_i = \theta_i(t_{k-1}) \times h_i + Qi_i(t_k) - Qs_i(t_k) - Q\,d_i(t_k) \tag{29}$$

In equation 29, h_i is the thickness of the i-th layer (in meters for the sake of calculations) and θ_i is the volumetric moisture content. The remaining terms of the water content balance are defined following:

Water in-flow for layer i at time t_k

$$\begin{array}{l} \text{if top-most layer} \\ \quad Qi_i(t_k) = Inf(t_k) \\ \text{otherwise} \\ Qi_i(t_k) = Qd_{i-1}(t_k) \end{array} \tag{30}$$

Sideways runoff for layer i at time t_k

$$Qs_i(t_k) = dt \times k_i(t_k) \times h_i[m] \times 1.00[m] \times S_x \tag{31}$$

The sideways runoff component is a straightforward application of Darcy's Law (Chow et al., 1994), the flow's hydraulic grade is assumed to be equal to the pavement's cross slope (S_x), and the flow's cross-section is 1.00m wide and as tall as the thickness of the i-th layer (h_i). The formula in equation 31 utilizes the unsaturated-state material's hydraulic conductivity for the current moisture content, as computed by the ***kUnsatSWCC*** function (section 3.10).

Downward runoff for layer i at time t_k

$$Qd_i(t_k) = dt \times k_{i+1}(t_k) \times 1.00[m] \times 1.00[m] \tag{32}$$

In a similar manner, the downward runoff estimation presented as equation 32 is based in Darcy's Law, but also applying the simplification proposed by Davidson et al. (1969)[13]**.** Additionally, since the downward flow must seep through the i-th layer and the one underneath it – which is often a less permeable, finer material thus constraining the flow the most – the lower layer k value has been put in the above formula.

Equations 29 to 32 have been programmed to be solved in sequence, starting from the top-most granular layer and moving downward until the subgrade is reached. A 1m x 1m cell is assumed for all volume

[13] Davidson et al. (1969) state that the soil moisture content throughout the layer remains constant as the flow seeps down, with the exception of the uppermost level. Thus, in the flow zone the soil's matric suction is not expected to change and so the Darcy Law can be simplified to the expression depicted herein.

computations. For the first layer, the water inflow at time t_k is the amount of infiltration (in m^3 / 12h / m^2) whereas the water inflow for the remaining layers is the downward runoff from the layer standing directly above. Additionally, the outbound water flows for a given layer are also subtracted in sequence: the "sideways" runoff is computed in the first place and so deducted from the initial water content plus the flow in; the downward flow is deducted secondly from the remaining water content. The remaining water content is the reported moisture content for layer *i.*

3.11.3 Resilient modulus of unsaturated unbound layers and subgrade soil – *calculateMR.m*

The ***calculateMR*** function utilizes the water content balance results to compute the resilient moduli of the granular layers and the subgrade at the calculated moisture levels for *shortTimestamp* t_k. This function makes use of the equations developed by Witczak et al. (in the formulation adopted by the MEPDG (NCHRP, 2004), reproduced here as equation 33), in which a sigmoid relationship is assumed between the resilient modulus (MR) of a given granular material or the subgrade soil and its volumetric moisture content θ.

Relationship with the saturated-state MR versus the MR for any moisture content

$$\log_{10}\frac{MR_i(\theta)}{MR_{i,opt}} = a + \frac{b-a}{1+e^{\beta+k_s\times\left(S(\theta)-S_{opt}\right)}} \qquad \text{where} \qquad \beta = \ln\left(\frac{-b}{a}\right) \tag{33}$$

The MEPDG guide (NCHRP, 2004) provides default values for a, b, and k_s for fine and coarse soils, these have been also adopted for Product-One.

Table 3: Default Values for the Witczak et Al. Formula for Determining an Unbound Material MR

Variable	Fine Soils	Coarse Soils
a	-0.5934	-0.3123
b	0.4	0.3
ks	6.1324	6.8157

The inputs for ***calculateMR*** are the pavement materials' saturated state resilient moduli and their degree of saturation at the optimum moisture content. The function's output, however, is the corresponding estimated resilient moduli at the "actual predicted" moisture content.

3.11.4 Multi-layered Elastic Pavement Deformation Model – Stress and Strain calculator

Product-One relies on a multi-layered linear elastic (MLE) stress and strain sub-system to calculate the stresses and strains that would develop throughout the pavement structure as a consequence of the traffic loads. This subsystem features an implementation of the MLE equations developed by Huang (2004), whose core assumption is that the granular materials can be characterized by a single resilient modulus value. Such assumption is also made by the MEPDG implementations at input levels 2 and 3, the non-linear nature of the unbound granular materials and subgrade soils is only acknowledged and modeled at input level 1 (NCHRP, 2004).

The equations governing the MLE stress and strain calculation problem are rather convoluted in nature, and reproducing them here would add little value to this Thesis while also rendering this Chapter's reading process unnecessarily cumbersome. Nonetheless, if the reader intends to review the mathematical formulation and solution of the MLE stress and strain problem, they are encouraged to proceed to Chapter 3 and Appendix B of Huang (2004), or Trejos Castillo et al. (2017), or Caicedo (2019).

The computation of stresses and strains caused by the load of a single-wheel on a multi-layered pavement is implemented in ***MLE_sigma*** and ***MLE_BC**,* which calculate the vertical, radial and tangential stresses (in cylindrical coordinates) at any set of distances [r] and depths [z] from the center of the load application. ***MLE_sigma*** is the front-end to the Huang's formulas, whereas the ***MLE_BC*** function is called from it to compute a series of ancillary parameters.

Product-One's MLE core's third component is the ***MLE_frontEnd.m***, which is called from the ***mainCode*** for every timestamp t_k during the pavement performance simulation. The ***MLE_FrontEnd*** script centralizes all the calls to ***MLE_sigma*** necessary to solve the stress and strain problem for the many axles handled by Product-One – by "axle" the reference is made not only to axle type but also to the different load ranges within each of these. Each load value within each axle type will be tied to the MLE-problem parameters listed below:

- Amount of load [ton] and radius of the loaded area [cm],
- Asphalt layers' dynamic moduli [E*, PSI] and Poisson coefficient [υ, dimensionless].

Besides, despite not being dependent on the load level and axle type, the resilient moduli of the structure's granular layers and subgrade would also vary over each time step t_k. The ***MLE_frontEnd*** script so retrieves all the axle-type- and load-level-specific MLE-problem parameters and calls the ***MLE_sigma*** accordingly.

Furthermore, the front-end script must account for the compound stress state produced by superimposing the stresses due to each wheel of a multiple-wheel or compound axle. For those cases (all axle types but for the light-weight single axle and single-wheel single heavy axle), the ***MLE_frontEnd*** computes the stresses due to each wheel of the axle at select planar locations and superimpose them afterward.

Figures 19 to 24 provide the planar locations at which the stresses due to each wheel are computed over a Cartesian grid; whereas Appendix C presents the equations to their respective radial distances and angles from the coordinate axes. Meanwhile, figure 25 portrays the vertical positions at which the stresses and strains are computed – pavement surface, mid-depth of each layer, contact area between each layer and that underneath (or the subgrade), and 6 inches below the subgrade surface. The example in figure 25 shows a graphical representation of the "under the wheel" planar position, but the same array of vertical positions apply for each planar location described for all axle types in figures 19 to 24.

The superposition of the vertical stresses due to each wheel is straightforward, the only calculation consists of summing the stresses of each wheel directly (the underlying reason is that the vehicle load application is supposed to consist only of a vertical load). But, as pointed out by Huang (2004), that is not the case for the horizontal stress components: since each reported stress is oriented in a different radial/tangential coordinate system (which depends on each wheel of the axle), these must first be projected to a single Cartesian grid and them summed together. To this end, the ***MLE_frontEnd*** script implements the coordinate-system-change formulas below (equation 34), which were brought from Huang (2004):

Rotation of radial and tangential stresses to X-Y components

$$\begin{aligned} \sigma_x(r_i) &= \sigma_r \times \cos^2(\alpha_i) + \sigma_t \times \sin^2(\alpha_i) \\ \sigma_y(r_i) &= \sigma_r \times \sin^2(\alpha_i) + \sigma_t \times \cos^2(\alpha_i) \\ \tau_{xy}(r_i) &= (\sigma_r - \sigma_t) \times \cos(\alpha_i) \sin(\alpha_i) \end{aligned} \tag{34}$$

Note that since the reference Cartesian grid does not necessarily correspond to a pair of horizontal axes that match the direction of each wheel's principal stresses (that is, axes orientation for which no shear stress component occurs), the most likely scenario is that non-null shear stresses (τ_{XY}) will develop, hence these must be computed with equation 34.

The stress-rotation formulas in equation 34 are applied to all the computed stress states at different depths and different planar positions as shown in figures 19 through 24. The corresponding α_i values for each planar location are given in the Appendix C**.** Once the stresses due to every wheel in the axle have been rotated to the common Cartesian grid, these can be superimposed and the amount of strain in the pavement structure inferred.

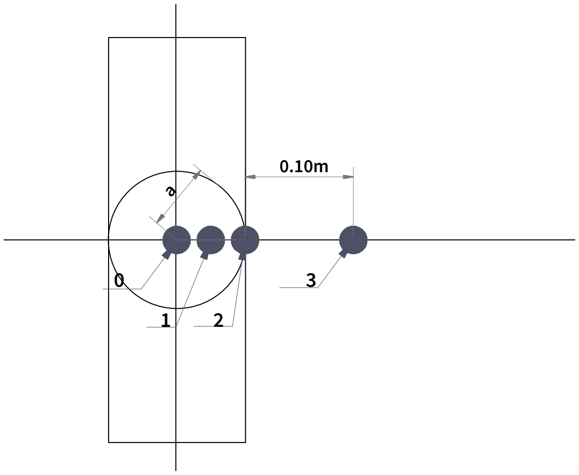


Figure 19: Stress and strain calculation positions for the single-wheel single axle.

Note: The four highlighted points are the locations at which stresses and strains are computed. Refer to Appendix C for further details.

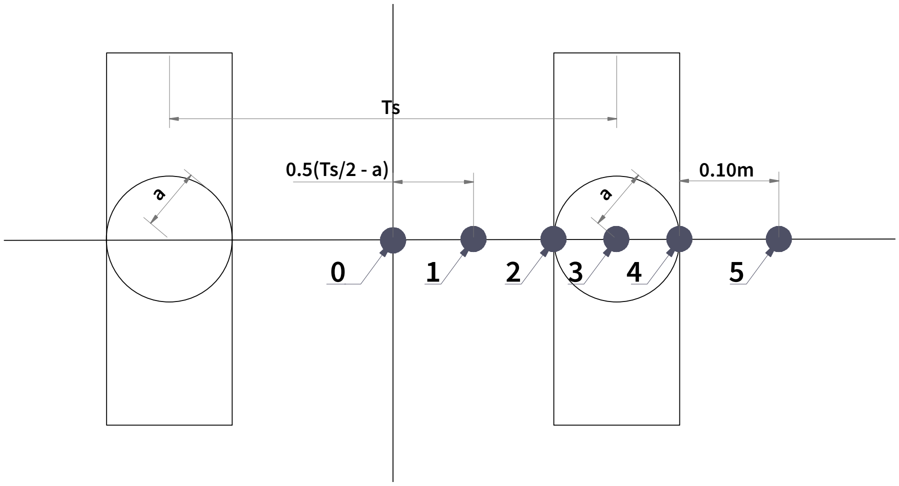


Figure 20: Stress and strain calculation positions for the dual-wheel single axle.

Note: The four highlighted points are the locations at which stresses and strains are computed. Refer to Appendix C for further details.

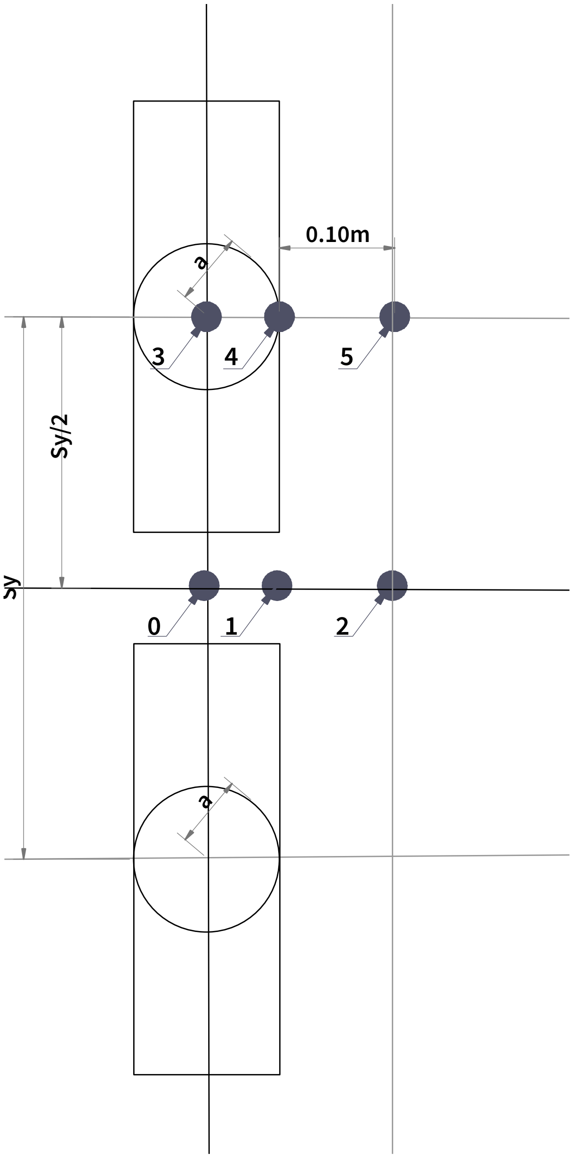


Figure 21: Stress and strain calculation positions for the single-wheel tandem axle.

Note: The six highlighted points are the locations at which stresses and strains are computed. Refer to Appendix C for further details.

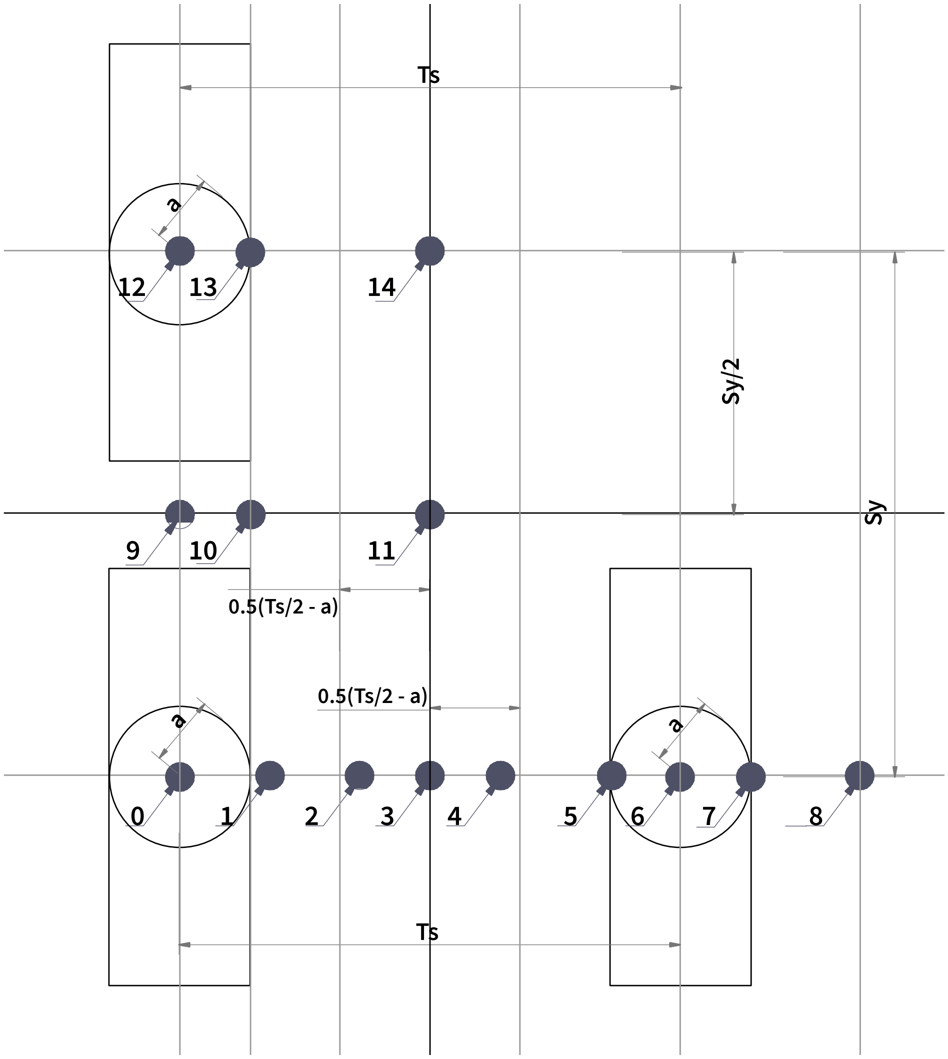


Figure 22: Stress and strain calculation positions for the non-homogeneous tandem axle.

Note: Given the non-symmetrical nature of this tandem axle, fifteen points needed to be defined as locations at which to calculate stresses and strains. Refer to Appendix C for further details.

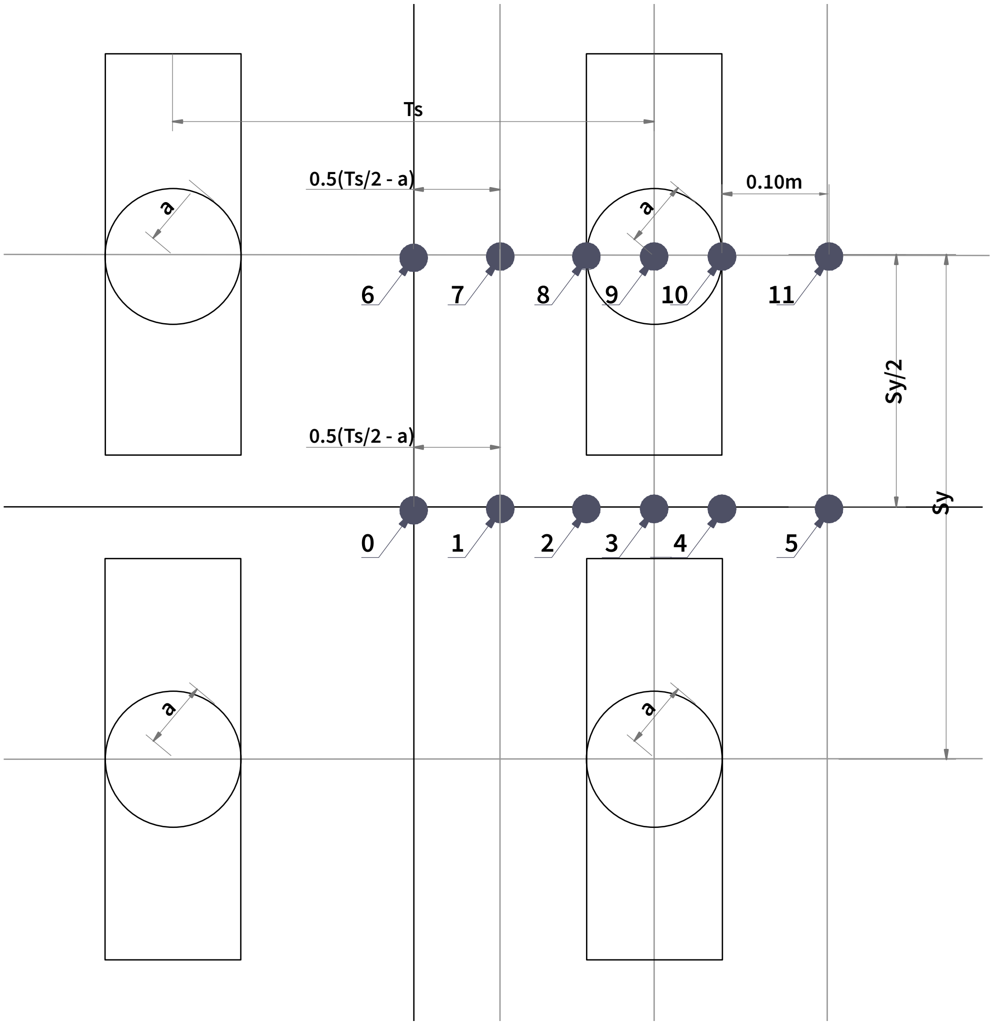


Figure 23: Stress and strain calculation positions for the dual-wheel tandem axle.

Note: The twelve highlighted points are the locations at which stresses and strains are computed. Refer to Appendix C for further details.

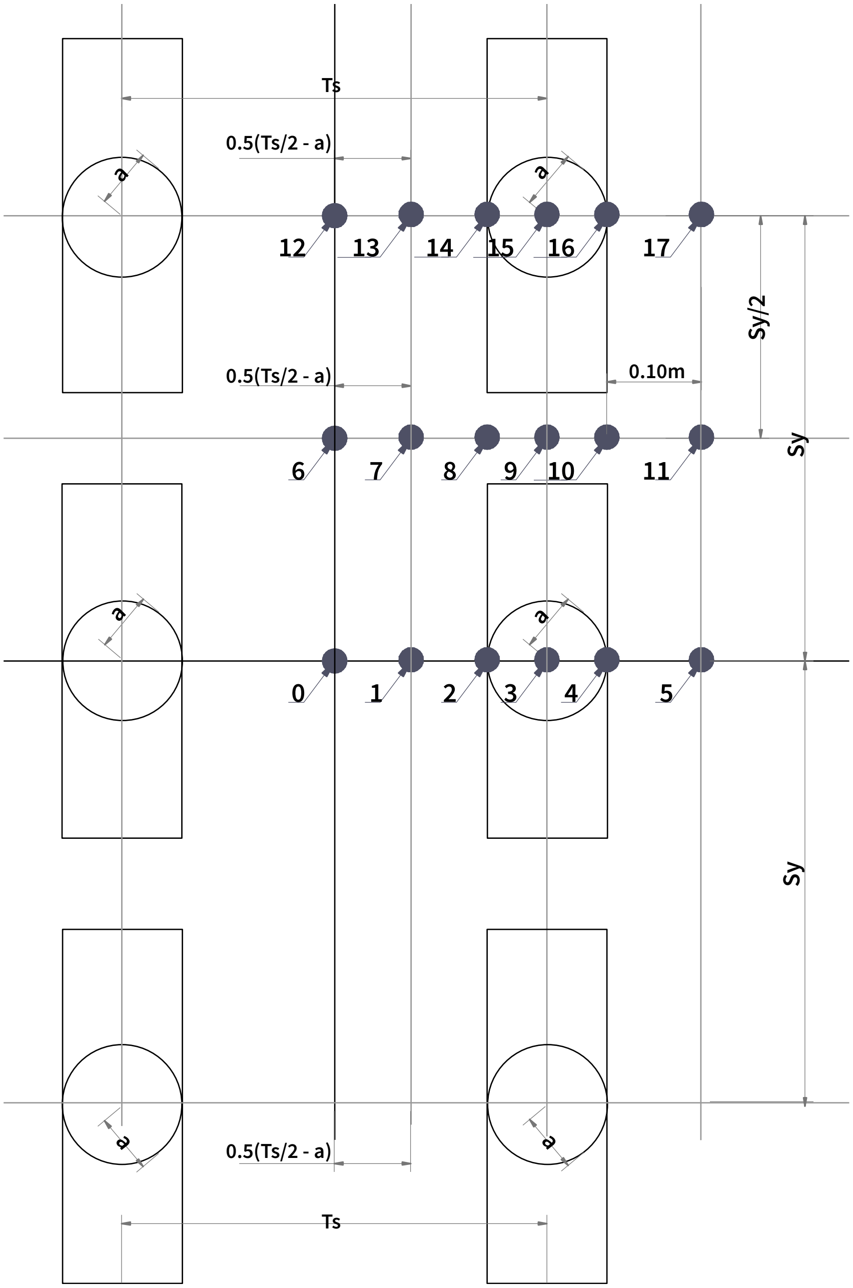


Figure 24: Stress and strain calculation positions for the dual-wheel tridem axle

Note: The eighteen highlighted points are the locations at which stresses and strains are computed. Refer to Appendix C for further details.

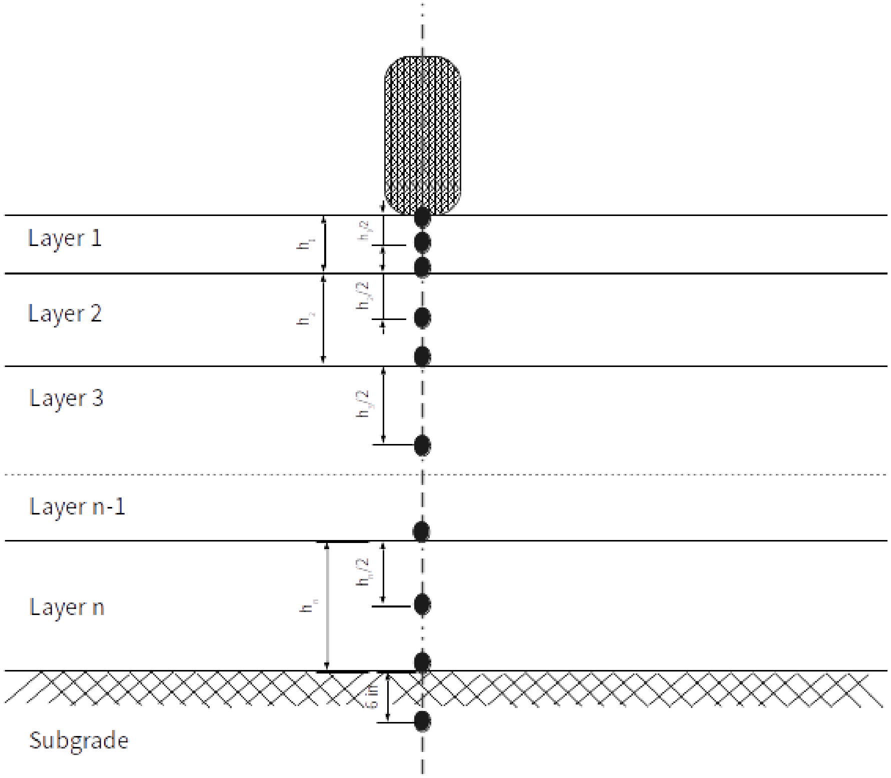


Figure 25: Vertical positions at which stresses and strains are computed.

The last computation done by the ***MLE_frontEnd*** script consists of solving the strains triggered by the superimposed stresses due to the axle loads, which are eventually used by the distress prediction routines. Equation 35 (NCHRP, 2004; Huang, 2004) has been implemented in the ***MLE_frontEnd*** to compute the total vertical strain ε_z at a planar position r and depth z, which is an input variable to the rut depth calculator (the ***rutDepthFrontEnd*** script, described in section 3.11.5.1).

Prediction of the vertical strain (ε_z) at a given (r, z) location from the principal stresses

$$\varepsilon_z(r,z) = \frac{1}{E(z)}\left[\sigma_z - v(z)\left(\sigma_x(r,z) + \sigma_y(r,z)\right)\right] \tag{35}$$

Where E(z) is either the dynamic modulus E* (if the material at a depth of z from the surface is an asphalt layer) or the resilient modulus MR (if otherwise is an unbound material), ν is the material's Poisson coefficient, σ_z is the total vertical stress, and σ_x and σ_y are the total horizontal stress components (in the common Cartesian grid) at the planar position r and depth z (as solved with equation 34).

Meanwhile, the horizontal principal strain ε$_h$, which is the input to the fatigue cracking model implemented in Product-One (the ***alligatorCrackFrontEnd*** script, section 3.11.5.2), is computed with equation 36, reproduced from Huang (2004).

Prediction of the horizontal principal strain (ε$_h$) at a given (r, z) location from the principal stresses

$$\varepsilon_h(r,z)=\frac{\varepsilon_x+\varepsilon_y}{2}-\sqrt{\left(\frac{\varepsilon_x-\varepsilon_y}{2}\right)^2+\gamma_{xy}^2}$$

where:

$$\varepsilon_x(r,z)=\frac{1}{E(z)}\left[\sigma_x-v(z)(\sigma_z+\sigma_y)\right]$$

$$\varepsilon_y(r,z)=\frac{1}{E(z)}\left[\sigma_y-v(z)(\sigma_x+\sigma_z)\right] \tag{36}$$

$$\gamma_{xy}(r,z)=\frac{2(1+v(z))}{E(z)}\tau_{xy}(r,z)$$

Once again, E(z) is either the material's dynamic modulus E* or resilient modulus MR – depending or whether the material at depth z is an unbound material or an asphalt mix, ν is the material's Poisson coefficient, ε$_x$ and ε$_y$ are the horizontal unit elongations (in the X and Y axes of the aforementioned Common Cartesian grid), γ$_{xy}$ is the horizontal shear strain, σ$_x$, σ$_y$, and σ$_z$ are the projected horizontal and vertical stresses, and τ$_{xy}$ is the planar shear stress at the (r, z) location according to the X-Y Cartesian coordinates.

3.11.5 Distress prediction models

The MEPDG features transfer functions to compute the rutting depth, the extent of bottom-up alligator cracking and top-down longitudinal cracking, the degree of reflective cracking (when the asphalt layers are placed on top of a jointed or cracked concrete slab), and thermal transverse cracking (due to the asphalt materials shrinking under cold weather conditions). Additionally, a framework to predict the IRI of the trial pavement's surface at any moment during the pavement's service life based on the amount of distress is provided. These models (except for the transverse cracking, which was reported as pointless by Caliendo (2012) for pavements that are not subject to harsh winter conditions, and reflective cracking) have also been implemented into Product-One, using the US-wide calibration parameters featured in the MEPDG. **Thus, this release of Product-One will compute the distresses in the trial parameter structure with "uncalibrated" formulas. Since this project is solely aimed at developing the M-E tool itself along with its input data library, the distress model calibration and verification would remain for future research activities.**

The following sections discuss how the MEPDG's models for the selected flexible pavement distresses have been implemented in Product-One.

3.11.5.1 Rut depth prediction model

Rut depth calculation in the MEPDG consists of separately computing the extent of such distress over the different materials within that structure. To that end, different equations are proposed for asphalt materials, unbound base materials, and the subgrade soil. For the asphalt materials, the amount of rutting is dependent on the vertical plastic (non-recoverable) deformation at each layer's mid-depth points whereas the equations for granular materials and the subgrade soil compute the amount of vertical plastic deformation directly from the resilient vertical strain – Tseng and Lytton's equation (1989).

Equations 37 and 38 are the MEPDG formulas – with the US-wide default calibration parameters values – to calculate the amount of rutting at an HMA layer, the counterparts for unbound base layers and the foundation soil are given in equations 39 and 40 respectively. These three equations correspond to the amount of rutting to occur at a given moment of time t_k, the total permanent deformation at t_k results from summing the computed rut depths over the entire pavement's life.

Vertical permanent deformation for the asphalt layers

$$RD_{HMA} = \sum_{i=1}^{\text{AC layers}} \varepsilon_{pl,i} \times h_i \tag{37}$$

Where RD_{HMA} is the total depth of rutting in the asphalt layers, h_i is the thickness of the i-th asphalt layer (both RD and hi will have the same units, either meters or inches), $\varepsilon_{pl,i}$ is the amount of plastic deformation at the i-th asphalt layer, which is computed according to equation 38 (same expression that is proposed in the MEPDG guide (NCHRP, 2004) but with updated regression coefficients (AASHTO, 2008; AASHTO 2015):

Plastic deformation at an asphalt layer

$$\begin{gathered} \varepsilon_{pl,i} = \varepsilon_{z,i}(z) \times \left[k_1(z) \times 10^{-3.35412} Temp_i^{1.5606} \times N^{0.4791} \right] \\ \text{where} \\ k_1(z) = (C_1 + C_2 \times z) \times 0.328196^z \\ C_1 = -0.1039\, h_{ACT}^2 + 2.4868\, h_{ACT} - 17.342 \\ C_2 = 0.0172\, h_{ACT}^2 - 1.7331\, h_{ACT} + 27.428 \end{gathered} \tag{38}$$

Where $\varepsilon_{z,i}$ (z) is the total vertical strain at depth *z* (mid-point of the i-th asphalt layer), as computed when solving the MLE-problem, *Temp_i* is the temperature of the i-th asphalt layer at time t_k [in degrees F], N is the total number of load applications (of a given axle type and load level) up to the time period t_k, *z* (depth of the midpoint of the i-th asphalt layer) must be given in inches for k_1, and h_{ACT} is the total depth of all the asphalt layers, in inches.

It must be highlighted though that the plastic deformation model in equation 38 has been constructed from laboratory tests in which the test subjects were subjected to repeated load applications while keeping their strength properties constant. However, an actual asphalt material within a pavement structure may alter its strength properties during the elapsed period because of temperature changes, and as a consequence, such plastic deformation model cannot be applied directly. Such conditions force the adoption of the *strain hardening* procedure, which accounts for the load applied historically before a certain moment of time by estimating the equivalent number of loads that would match the actual amount of plastic strain had these all occurred at the current conditions[14]. The amount of plastic strain at the current time can then be computed as resulting from the current traffic level plus the equivalent traffic from previous periods.

The rut-depth equations for unbound base materials and the foundation soil are an implementation of the Tseng and Lytton (1989) formulas. Equation 39 below provides the formula to determine the amount of rutting at each granular layer and equation 40 summarizes the MEPDG procedure (developed by Ayres and printed in NCHRP, 2004) to compute the total vertical deformation of the assumed semi-infinite subgrade given the amount of deflection at the interface with the pavement's structure and a second location within the subgrade soil itself:

MEPDG's equation to compute the depth of rutting at an unbound base material

$$RD_{Base}[\text{in}] = 1.673\left(\frac{\varepsilon_0}{\varepsilon_r}\right)e^{-\left(\frac{\rho}{N}\right)^{\beta}} \times \varepsilon_z \times h \tag{39}$$

MEPDG's equation to compute the vertical deformation of a semi-infinite subgrade soil

$$RD_{SubGrade}[\text{in}] = \left(\frac{1-e^{-kh_0}}{k}\right) \times 1.35\left(\frac{\varepsilon_0}{\varepsilon_r}\right)e^{-\left(\frac{\rho}{N}\right)^{\beta}} \times \varepsilon_z(z_0)$$

$$k = \frac{1}{6\,\text{in}} \ln\left(\frac{1.35\left(\frac{\varepsilon_0}{\varepsilon_r}\right)e^{-\left(\frac{\rho}{N}\right)^{\beta}} \times \varepsilon_z(z_0)}{1.35\left(\frac{\varepsilon_0}{\varepsilon_r}\right)e^{-\left(\frac{\rho}{N}\right)^{\beta}} \times \varepsilon_z(z_0+6\,\text{in})}\right) \tag{40}$$

[14] Refer to the original MEPDG guide (NCHRP, 2004) for an in-depth explanation on how to apply the *strain hardening* procedure to accumulate the number of loads to estimate the amount of plastic strain in asphalt layers.

Where ε_z is the resilient vertical strain at a depth z from the pavement surface (For base/sub-base layers, z is taken at half the thickness of that layer; whereas the Ayres expression (equation 40) uses z at the interface between the subgrade and the deepest sub-base layer (the z_0 point), and 6 inches underneath that interface), *h* is the thickness of the granular layer in inches (equation 39 only), while in equation 40 h_0 is the depth to the bedrock measured from the subgrade/sub-base interface, and *N* is the total number of loaded axle passes (i.e. total number of axle passes of a given type and carrying a certain load).

The parameters ε_0 ε_r, ρ, and β are computed with the equations below – from Appendix GG of NCHRP (2004):

Parameters to the rut-depth equation for unbound base materials and subgrade soils

$$
\begin{aligned}
\log_{10}(\beta) &= -0.61119 - 0.017638(\omega[\%]) \\
\rho &= 10^9 \times \left[\frac{C_0}{1-(10^9)^\beta}\right]^{\frac{1}{\beta}} \\
C_0 &= \ln\left[\frac{0.15}{20}\right] \\
\frac{\varepsilon_0}{\varepsilon_r} &= \frac{1}{2}\times\left[0.15 \times e^{(\rho^\beta)} + 20 \times e^{\left(\left(\frac{\rho}{10^9}\right)^\beta\right)}\right]
\end{aligned}
\tag{41}
$$

where β is dependent on the unbound material's water content (ω) by weight. The MEPDG guide provides a formula to estimate ω from the material's CBR ratio (or resilient modulus) and the depth of the water table. However, such estimation has been disregarded from Product-One and in lieu of such a model, a straightforward prediction of ω from the estimated volumetric water content θ (which Product-One computes during the Performance Simulation) has been adopted. By multiplying θ by the specific gravity of the granular material, which is the reported compacted unit weight, it is possible to obtain ω from θ. Two underlying reasons ground such a decision: In the first place, the MEPDG's equation would force the user to have knowledge of the depth water table at the pavement's site and how it fluctuates after rain events in order to properly model the water table rises and drops. Additionally, the MEPDG's formula does not account for rainfall infiltrating the pavement structure and thus soaking the pavement's unbound layers, which has however been modeled in Product-One.

Concerning how these equations have been coded in Product-One, two separate computer routines were generated: the ***computeRutDepth*** function, which computes the vertical deformation of all the pavement's layers and the subgrade after the application of axle loads of a given type and weight at a single strain evaluation planar position, and a front end script (named ***rutDepthCalcFrontEnd)***, which organizes all calls to the former for all planar positions and load level for a given axle type and then superimposes these deformation results for all the axle configurations. The ***FrontEnd*** script is called by the ***mainCode*** at each time step t_k right after the MLE solver ends, it repeats the following step-wise procedure for each axle configuration:

1. The amount of vertical deformation for a given axle configuration and load level is computed at all the planar locations at which stresses and strains were calculated (figures 19 to 24). To this end, a number of calls to ***computeRutDepth*** are made. On each call, the ***FrontEnd*** passes the vertical deformation vector at that planar locations, number of axle passes, asphalt layers temperature, unbound materials' moisture, and an auxiliary variable for ***computeRutDepth*** to accurately relate each strain-vector location with proper temperature and moisture values.
2. The planar location at which that axle configuration and load level trigger the highest total permanent deformation (i.e. the sum of all vertical deformation of all layers plus the subgrade) is detected. The corresponding vertical deformation profile is stored in an auxiliary variable as representative of that load level for that axle type.
3. Steps 1 and 2 are repeated for all the load levels within each axle configuration, and the maximum rut depth provoked on each layer by each superimposed over one another. These are stored in the *maxRutDepthHMA, maxRutDepthGran,* and *maxRutDepthSubGrade* variables.
4. Finally, the vertical deformation vectors for all the axle configurations are summed up together. The *rutDepth* variable would contain the total vertical deformation at each layer plus the overall rut depth (the sum of the former) at time $t_{k.}$

One final remark must be made concerning the rut depth estimation procedure described above: it is implied that the traffic axles run over the pavement in such a manner that their respective planar positions would be coincident over the pavement cross section.

3.11.5.2 Bottom-up (alligator) and top-down (longitudinal) cracking prediction model

The MEPDG uses the Asphalt Institute Equation (NCHRP, 2004; Huang, 2004) with specially calibrated parameters to compute the admissible number of passes for fatigue cracking to develop on 40% of the roadway area (the area where vehicles most often tread on when circulating). It has been reproduced here as equation 42.

Asphalt Institute equation to the admissible number of passes for fatigue cracking to occur

$$Nf=0.00432\times\left[10^{M(z)}\times k_1\times\varepsilon_h^{-3.9492}\times E(z)^{-1.281}\right] \tag{42}$$

Where ε_h is the largest horizontal strain at position z (bottom of each layer), computed with equation 36; *E* is the material's elastic modulus at depth *z* in PSI *(*either the dynamic modulus E* or resilient modulus, depending on whether the material at depth *z* is an asphalt material or an unbound base), K_1 is a parameter that depends on the total thickness of the asphalt layers and is calculated differently for bottom-up alligator cracking and top-down longitudinal cracking (equation 43)[15], and *M* is a parameter that depends on the asphalt layer air void content and bitumen content percentage volume (equation 44).

Auxiliary parameter k_1 to the MEPDG's fatigue cracking equation

For bottom-up cracking:

$$k_1 = \frac{1}{0.000398 + \frac{0.003602}{1+e^{11.02-3.49 h_{ACT}}}}$$

(43)

For top-down cracking:

$$k_1 = \frac{1}{0.0001 + \frac{29.844}{1+e^{30.544-5.7357 h_{ACT}}}}$$

where h_{ACT} is the total depth of the asphalt layers in inches.

Auxiliary parameter M to the MEPDG's fatigue cracking equation

$$M = 4.84\left(\frac{Vb}{(Va+Vb)} - 0.69\right) \tag{44}$$

Where Va and Vb are the asphalt mix air void content and bitumen content, in percentage volume. In Product-One's implementation of these formulas, the M factor is computed for each asphalt layer

It must be highlighted that despite being calibrated for North-American conditions, equation 42 keeps resemblance with Shell's fatigue equation for asphalt materials, known to be suitable to predict rutting in Uruguayan pavements (Flintsch et al, 1998).

Equation 42 returns the estimate of the admissible number of loaded axle passes (with a certain wheel configuration and load level) for fatigue cracking to happen. The actual distress prediction results by applying the Miner's Law (NCHRP, 2004) over the period elapsed since the beginning of the pavement's design life. To this end, a *Damage Ratio* (which can be accumulated over time and superimposed over all sorts of axle configurations and load levels) is defined (reproduced in equation 45).

[15] It must be highlighted that the formula for the top-down k_1 parameter depicted in equation 43 corresponds to the LTPP-calibrated parameter; as presented in Appendix II of the MEPDG guide (NCHRP, 2004).

Damage Ratio for fatigue cracking

$$D=\sum_{time=0}^{t_k}\frac{n_i}{N_i} \tag{45}$$

Where n_i is the actual axle count of a given axle type and load level at time i (i ranges from 0 to time t_k), and N_i is the number of admissible axle passes for such a wheel configuration and load level, as per equation 42.

Finally, the amount of cracked area (for alligator cracking) or the length of cracks per mile of road (top-down cracking) at time t_k is computed with the equations below:

Amount of bottom-up and top-down cracking at a given time

$$\begin{gathered}\text{for bottom-up cracking}\\ FC_{bu}=\frac{1}{60}\times\frac{6000}{\left(1+e^{1\times C1+1\times C2\log_{10}(100D)}\right)}\\ \text{for top-down cracking}\\ FC_{TD}=10.56\times\frac{1000}{\left(1+e^{7.0-3.5\log_{10}(100D)}\right)}\end{gathered} \tag{46}$$

Where FC_{bu} and FC_{TD} are the amount of alligator cracking (in % roadway area) and top-down cracking (in feet per mile of roadway) at time t_k respectively, D is the damage ratio at time t_k (as a 0-to-1 ratio value, equation 45), and C1 and C2 are parameters that depend on the total thickness of the asphalt layers, their definition is as follows:

Auxiliary parameters to the fatigue cracking equations

$$\begin{gathered}C1=-2C2\\ C2=-2.40874-39.748\left(1+h_{ACT}\right)^{-2.856}\end{gathered} \tag{47}$$

Where h_{ACT} is the total depth of the asphalt layers in inches.

The above equations were implemented in Product-One in a manner similar to the rut depth computations: three separate pieces of code have been prepared, namely the ***alligatorCracking*** function (which computes the admissible number of passes Ni and the incremental damage ratio at time t_k for a given axle configuration type and load level), ***topDownCracking*** (which is the exact same code as the ***alligatorCracking*** function but crafted to solve the top-down cracking increment caused by a single axle configuration and load level) and the ***alligatorCalcFrontEnd*** script, which manages the calls to ***alligatorCracking*** and ***topDownCracking*** for the different axle configurations and load levels, and superimposes the effects of each together with the Miner's Law. The outcome of the ***FrontEnd*** script is twofold: the *alligatorCrack* and *topDownCrack* vectors, which contain the resulting fatigue-cracked area and top-down crack lengths at every moment in time $t_{k.}$

3.11.5.3 IRI and PSI computation based on predicted distresses

Since its first release, the MEPDG's IRI prediction equation has been revised. Product-One features an implementation of the IRI equation that was reported in the 2008 M-E Design Manual of Practice (AASHTO, 2008). This equation has been preferred over both the original formulation (NCHRP, 2004) and the latest revised formula (AASHTO, 2015) because of its notorious simplicity and for the other equations lacking proper specification of their input variables – the reference manuals cited above explain partially some of the equation's inputs, particularly those to the "Site Factor".

The MEPDG supplies distinct equations to compute the pavement surface's IRI depending on its base material, a different equation must be utilized whether the pavement structure features an unbound base, a hot-mix asphalt base, or a cement-treated base. Only the first case has been included in this release of Product-One (equation 48).

Predict the IRI of the pavement surface (asphalt surface layer plus unbound base)

$$IRI = IRI_0 + 0.0150 \times SF + 0.400 \times FC + 0.0080 \times TC + 40.00 \times RD \tag{48}$$

In this equation, IRI_0 is the pavement's roughness at the beginning of its service life [in inches/mile], SF is the *site factor*, a function of the area's temperature and rainfall, and the pavement's age; FC is the fatigue-cracking-affected area (sum of top-down and bottom-up cracks) [percentage of lane area], TC is the length of the transverse cracks, and RD is the total rut depth [inches]. The area of fatigue-related cracks and the depth of ruts are as calculated with the formulas given by equations 39 to 47; the transverse cracking term (sum of cold-weather-shrinkage cracks and reflective cracks) will be neglected in this release of Product-One[16]. The *Site Factor* term is computed with the following formula – retrieved from AASHTO (2008):

Site Factor for the IRI equation

$$SF = Age \times \left[0.02003\left(PI + 1\right) + 0.007947\left(AnnP + 1\right) + 0.000636\left(FI + 1\right)\right] \tag{49}$$

Where *Age* is the pavement age [years], *PI* is the subgrade's plasticity index [in %], *AnnP* is the average annual precipitation [inch], and *FI* is the average annual freezing index [in deg. F days]. Huang (2004) provides the methodology to compute FI from daily temperature averages.

[16] The reason behind such a decision is twofold: this release of the M-E design software would not focus on concrete pavement overlays nor asphalt structures over cement-treated bases, and on top of that transverse cracking cold-weather-shrinking is a negligible phenomenon in areas with mild winter weather (non-freezing temperatures predominantly). Such a point has been highlighted by Caliendo (2012).

Product-One also calculates the Pavement Serviceability Index (PSI, a performance measure still widespread in Uruguayan pavement practice, partly due to the AASHTO ’93 method still being used) from the estimated IRI values – the formula by Al-Omari and Darter (1994) is applied (equation 50).

PSI from IRI (after Al-Omari and Darter, 1994)

$$PSI = 5.0\, e^{-0.26\, IRI} \tag{50}$$

Where the IRI is expressed in m/km (conversion from the estimated value in inches per mile is done internally by Product-One).

In Product-One’s implementation of the above formulas, the *Site Factor* is computed from the simulated climate variables (more precisely hourly temperature and rainfall) by the ***IRIclimateSiteFactor*** function, which is called by ***IRIPSICalcFrontEnd***, the IRI and PSI calculation routine. As with the previous distress, the ***FrontEnd*** script is called automatically from the ***mainCode,*** and calculates the pavement surface’s roughness index and PSI at time t_k. These values are stored in the *IRI* and *PSI* variables.

3.12 Results Reporting module

The last components within Product-One are a set of scripts and functions to provide the end-user with the pavement simulation output in an easy-to-read manner. These are executed automatically once the pavement performance simulation ends. This release features a series of functions for graphical outputs plus export-to-spreadsheet capabilities.

3.12.1 Graphical output functions

These functions plot relevant reduced input data and calculation results in an intuitive way. Product-One may generate automatically plots for the climate and traffic input data, the results of the AC temperature calculation module, the predicted surface infiltration, runoff, and moisture content of the unbound materials, and the predicted extent of distress (rut depth for each layer, degree of fatigue cracking, IRI, and PSI). The functions that create these plots are called automatically from the ***mainCode,*** depending on whether the user specified in the ***dataImport*** spreadsheet that such plots are to be generated. Figures 15 to 17 in this chapter, and figures 27 to 30 in Chapter 4 have been obtained with the graphical output functions.

3.12.2 Numerical output

Finally, Product-One will export the pavement design results in a spreadsheet so as to enhance the ease of reading and manipulation of the pavement design end results. The output sheet is composed out of a template stored internally and is named after the saveFilename provided by the user in the dataImport spreadsheet. Product-One will export the following to the output spreadsheet:

- Future climate variables
- Future yearly AADT by vehicle category
- HMA temperature prediction (surface temperature and mid-depth within all the asphalt layers)
- Volumetric moisture content and resilient modulus of the unbound layers and the subgrade
- Rut depth, alligator cracking, and top-down cracking
- IRI and PSI

CHAPTER 4 — SOFTWARE VALIDATION AND BETA TESTING

4.1 Introduction

In an effort to both test Product-One's performance as a pavement design tool and compare its output with those of traditional methods (to the extent possible) and also familiarize with other existing M-E design software for flexible pavements, a design example that emulates an actual design process for an Uruguayan road has been devised. A secondary highway belonging to the Uruguayan National road network has been chosen, a new pavement has been calculated for it for a 10-year design life using the traditional AASHTO (1993) method, and the resulting structure has been loaded into Product-One and two other M-E pavement design applications (MeDiNa and CR-ME), in order to quantify the amount of distress predicted by each software and the PSI decay after the design period, which would serve as a means to compare against the AASHTO (1993) method.

The following sections portray a review of this trial design exercise, the input parameters that have been retrieved and utilized have will be presented and the core design considerations that have been observed will be highlighted. Furthermore, it will be assumed that the reader is familiar with the AASHTO (1993) methodology, its data requirements, its outcomes, and the many caveats that must be observed when utilizing it in engineering practice, for no review of this methodology itself will be provided.

Afterward, the results of evaluating the suitability of the trial structure with Product-One, CR-ME, and MeDiNa will be introduced. A brief comment on how the design verification procedure has been achieved with each software will also be included, but this chapter will not dive into how these third-party software packages work as such is not the core object of discussion – it will revolve around the comparison of each software's outcome instead.

4.2 Trial design problem statement

For this trial design exercise, which keeps resemblance to an actual pavement structure draft project, a new pavement for two-lane two-way Highway 48, in the outskirts of Montevideo and near the town of Las Brujas, Uruguay (figure 26), has been calculated using the traditional AASHTO (1993) methodology for a design life of 10 years. The method's design parameters (namely overall standard deviation, reliability level, and initial and final serviceability indices) were set to the defaults suggested in the design method's guide (AASHTO, 1993) for a flexible pavement design for a secondary road:

- Overall standard deviation: 0.42
- Reliability level: 85%
- Initial serviceability index $PSI_{initial} = 4.2$
- Final serviceability index $PSI_{final} = 2.0$

By appealing to the correlation formula by Al-Omari and Darter (1994), the initial and final PSI values relate to initial and final IRI values of 0.7 and 3.5 m/km respectively.

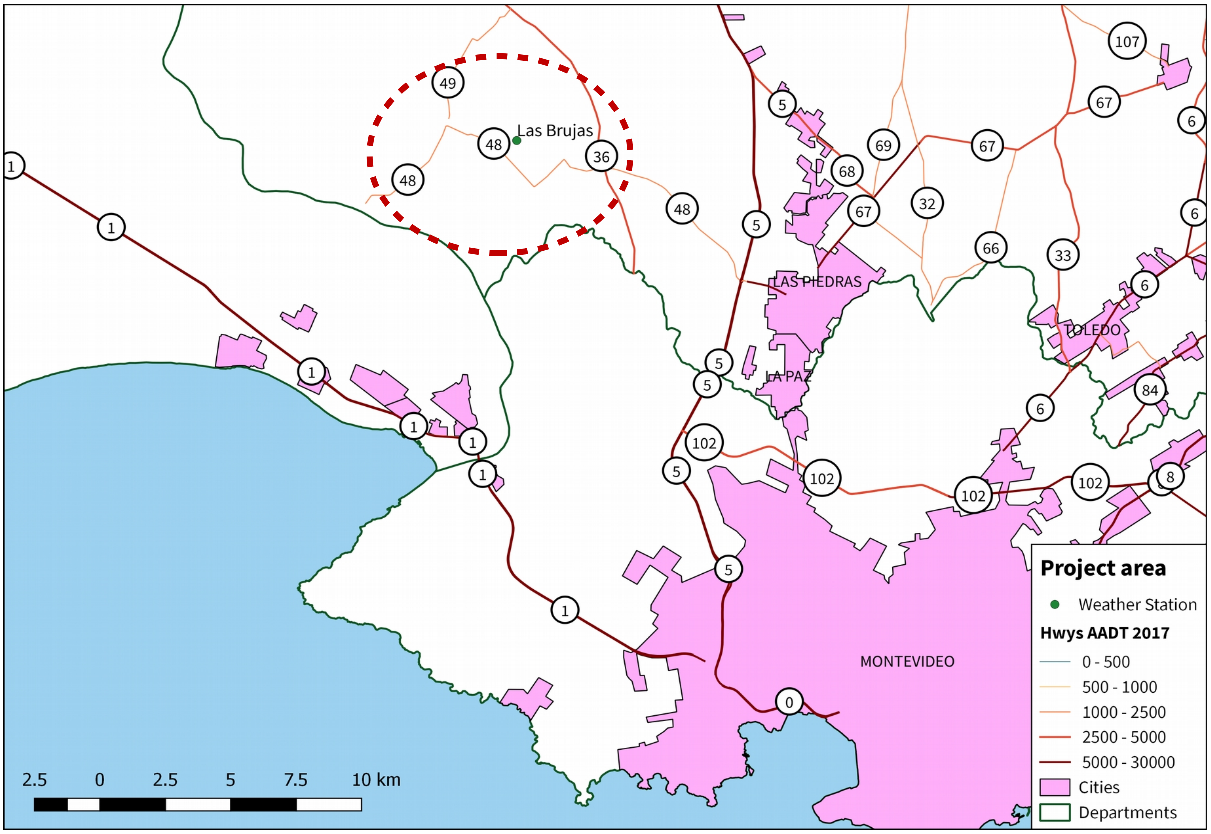


Figure 26: Location of the study area – Highway 48, Uruguay.

Cartography sources: IDE_UY [Agesic], Geoportal MTOP

4.2.1.1 Climate data

Highway 48 has been selected on purpose because of its closeness to the *INIA Las Brujas* weather station (highlighted in figure 26). Thus, its weather records for the 2010-2018 period have been regarded as Level-1 input for this exercise. It must be highlighted though that the AASHTO (1993) method does not make use of these climate records, but Product-One and the other third-party applications will.

4.2.1.2 Traffic volume and load data

The last publicly available traffic volume record for Highway 48 corresponds to the year 2017 and only features 5 vehicle categories (cars, buses, and light, medium, and heavy trucks). The two-way daily vehicle traffic for Highway 48 and its lane distribution factor have been downloaded from the MTOP online GIS data service[17] and reproduced in table 4 below. The traffic lane distribution factor equals 0.50. Meanwhile, since the MTOP online service does not provide load records for heavy vehicles, the hypothesis of all vehicles running at their legal load has been presumed true for this trial design.

Table 4: Average Annual Daily Traffic for Highway 48. Year 2017

Category	**AADT (2017)**
Cars	1023
Buses	37
Light Trucks	232
Medium Trucks	11
Heavy-haul trucks	29

Source: Geoportal MTOP

A constant traffic growth rate of 3.0% for all vehicle categories was used for this pavement design exercise. Such value is widely used for long-term traffic growth predictions in Uruguay, although no scholarly paper justifying such a choice has been found. Additionally, It has been assumed that Highway 48 would operate as a multi-purpose road with uniformly-distributed freight traffic plus commuter car traffic since it is close to Montevideo. Thus, although not necessary for the AASHTO (1993) method, the monthly, daily, and hourly factors presented in this report for such a functional type have been applied in the M-E verification stage of this design.

Using a spreadsheet-based calculator, the design traffic for the 10-year design period is 1.5 million 18-kip ESALs,

4.2.1.3 Subgrade and unbound materials

According to the *Carta Geológica del Uruguay* (Preciozzi et al., 1985)[18], the local subgrade soils originate from silty sedimentary rocks. Although core data for this material is not available, a CBR value of 6 can be assigned to this material without falling into over-estimation of its strength properties.

[17] The MTOP GIS data, used recurrently throughout this thesis project, can be accessed over the internet at http://geoportal.mtop.gub.uy/

[18] The *Carta Geológica* is available at: https://www.miem.gub.uy/mineria-y-geologia/carta-geologica-y-memoria-de-recursos-minerales-no-metalicos-del-uruguay

Meanwhile, the unbound base and sub-base materials that have been selected for the trial structure are among those of quarries around Montevideo for which sieving test results, Atterberg limits, and compaction and strength parameters were made available. The chosen sub-base material is a quarried gravelly aggregate whose CBR index reaches 35 when compacted at 95% of its maximum dry weight, and the base material is a stronger, specially blended crushed stone aggregate with a CBR of 110 when compacted to its maximum dry unit weight (as obtained from the modified Proctor test).

4.2.2 Trial structure – AASHTO (1993) design results.

The following pavement structure has resulted for Highway 48 and as such it will be tried with the M-E design suites:

- Hot Mix Asphalt surface layer: Level-3 HMA surface course AC 20 binder 1/2" maximum aggregate size – compliant to the Uruguayan standard (MTOP, 2003) [ID #001]. **Thickness 4cm. Initial IRI: 0.7 m/km**
- Hot mix Asphalt binder layer: Level-3 HMA binder course AC 20 binder 3/4" maximum aggregate size– compliant to the Uruguayan standard (MTOP, 2003) [ID #005]. **Thickness 6cm**
- Granular base: Level-2 aggregate #227 quarried from the Montevideo-metro area, compacted at 100% of its maximum dry unit weight (Modified Proctor test) (ID #186). **Thickness 15cm**
- Sub-base: Level-2 aggregate #138 quarried from the Montevideo-metro area, compacted at 95% of its maximum dry unit weight (Modified Proctor test) (ID #182). **Thickness 25cm**
- Subgrade: Fine soil from weathered silty sedimentary rocks. CBR value of 6 when compacted.

As it was stated previously, this structure has been calculated using the AASHTO (1993) model for a 10-year service life. The initial and final serviceability index values were set to 4.2 and 2.0 respectively (as recommended by AASHTO (1993)). The overall reliability index was 85% The resilient moduli of the subgrade soil and the granular materials have been calculated with the TRL formula (Powell et al., 1984), which is used for level-2 inputs in the MEPDG (NCHRP, 2004). The

4.3 Trial design results with Product-One

The trial structure described above, as computed by the AASHTO (1993) method, has been loaded into Product-One for a 10-year-service-life distress prediction analysis. The initial cracking was set to 0, so was the depth of rutting as well, while the initial IRI was set to 0.7 m/km, which correlates to a PSI of 4.2. Each of the constituent materials has been selected from the *Materials Library* (their *materialIDs* have already been highlighted), so either a level-2 or level-3 estimate of all their relevant physical and strength properties is readily available for the software to utilize. Level-1 climate data is available for this location (the INIA Las Brujas weather station, for which 9 years of hourly meteorological data are available, is located nearby), and it has been imported to Product-One as such. Finally, as it has been mentioned in previous sections, the traffic distribution factors that correspond to a road with mostly uniformly-distributed traffic with a commuter peak have been selected to be used in Product-One.

All the software run preparation steps described in the previous paragraph have been conducted in the ***dataImport*** spreadsheet. Once ready, the ***mainCode*** was executed in the Matlab console.

4.3.1 Predicted distresses

The ten-year performance simulation with Product-One held the predicted levels of distress shown in figures 27 to 30. These rather notoriously evince the consequences of the asphalt material's strength properties changing with the increases in air temperature: during the austral summer months (December to March), the asphalt layers heat more and become weaker and supple, hence the distress extent increase more rapidly.

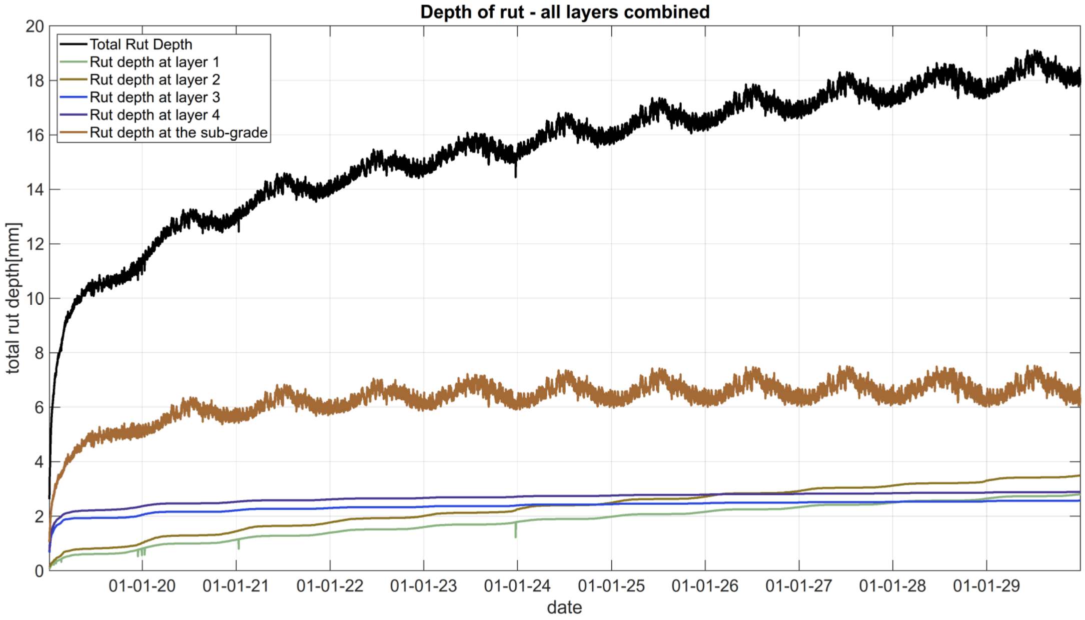


Figure 27: Product-One output: Predicted rut depth for all pavement layers.

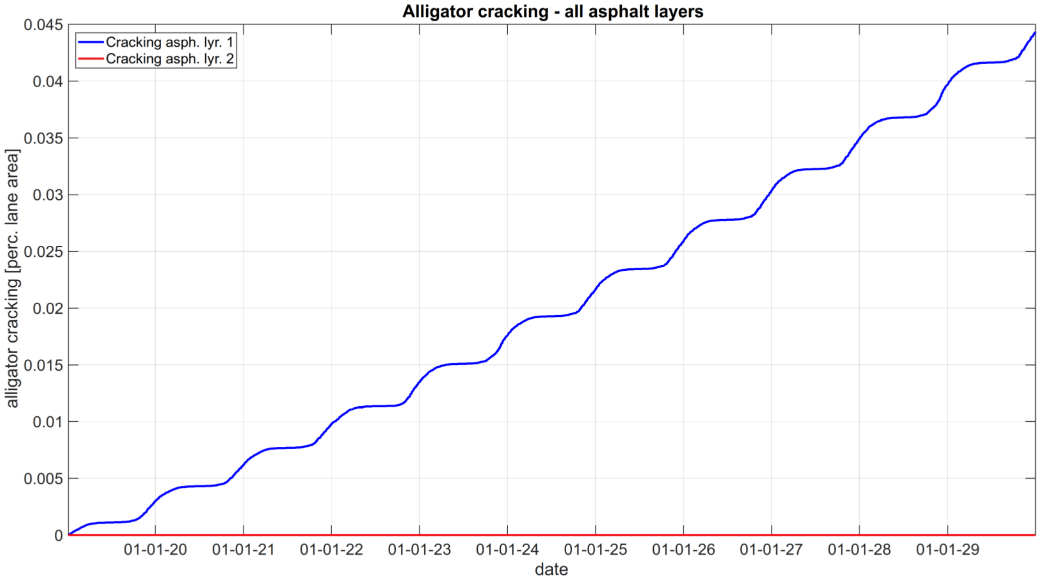


Figure 28: Product-One output: Predicted alligator cracking development in the asphalt layers.

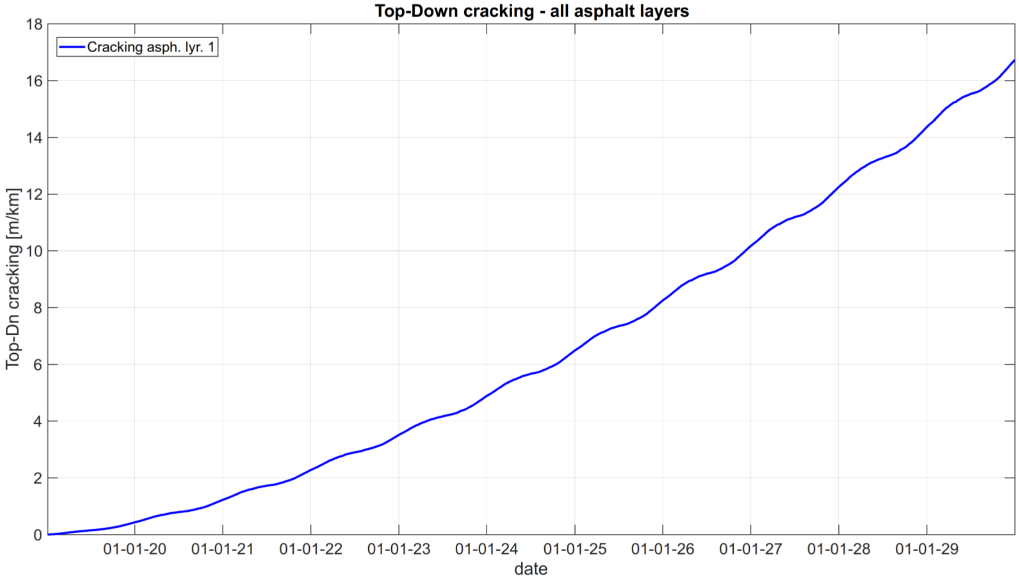


Figure 29: Product-One output: Predicted longitudinal cracking at the pavement surface.

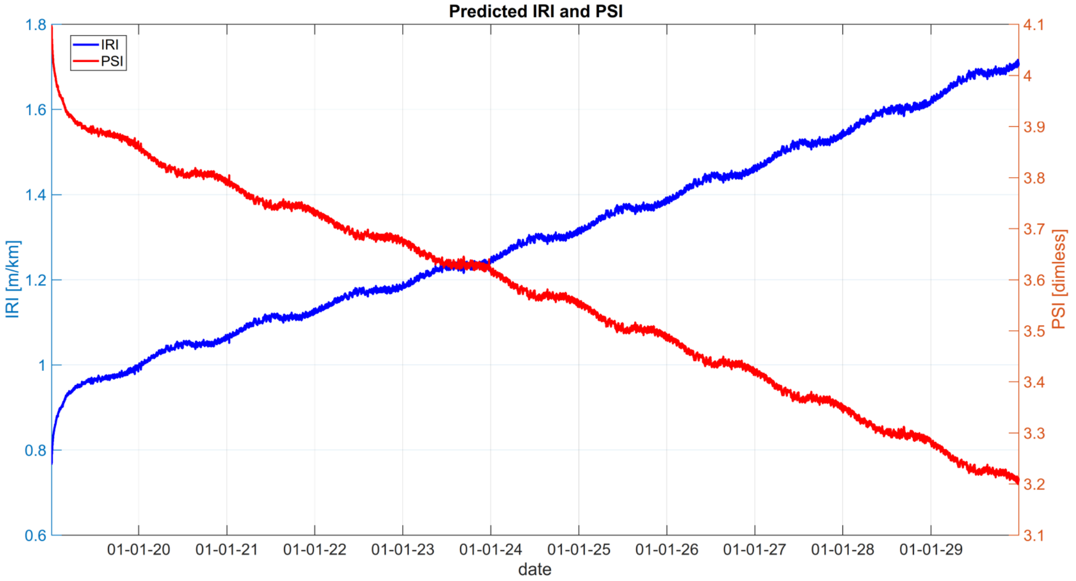


Figure 30: Product-One output: Predicted IRI increase and PSI decrease.

The predicted total permanent deformation after the 10-year cycle is 18.3mm [0.72 inch], most of it due to deformation in the subgrade [6.5 mm, or 0.26 inch]. Yet some numeric instability occurs when predicting the amount of rutting in the subgrade, possibly due to the estimation of the equivalent number of passes from previous periods at each moment in time. Further investigation of the causes of such behavior is warranted.

The fatigue-related cracking grew to only cover a very slight percentage of the pavement area, less than 0.1% of the roadway area. However, after ten years of traffic, longitudinal cracking would extend to a value of 16.4 m/km [65.4 ft/mile]. Further review of the prediction equations may be needed to address the causes of such a rather awkwardly low alligator cracking outcome.

Finally, the IRI and PSI predictions reflect both previously described pathologies: the shape of the IRI curve mimics the rut depth plots and its unstable behavior. On top of that, the pavement's IRI would hit 1.7 m/km [107.8 in/mi] after ten years in service – the final serviceability index would only decrease to 3.21, about 45% of the decrease that would be expected if the design with the AASHTO '93 method is taken for granted.

4.4 Trial results with other M-E design software

This section collects the results of evaluating the trial pavement structure for Highway 48 with two third-party M-E design software suites: MeDiNa and PitraPAVE. A brief digest of how the design process was carried with each software suite follows, depicting both a log of how the many inputs to the design process have been loaded and what special assumptions were regarded in each case.

4.4.1 Results from MeDiNa

As it has been presented in Chapter 1, MeDiNa is a user-friendly updated implementation of the SisPavBR, a mechanistic-empirical pavement design tool created by Franco (2007) to be used in Brazilian highway engineering practice. As such, it relies strongly on Brazilian technical standards and manuals. This software can be utilized both for the design of a new structure from scratch (provided the design traffic, materials, and admissible distress thresholds are fed to the program) or evaluate the suitability of a trial pavement structure for a predefined service period and admissible distress levels.

The core stress and strain calculator within MeDiNa is the software AEMC, originally developed by Franco (2007), which is an application of the multi-layer elastic linear theory equations. Concerning distress generation, MeDiNa computes the extent of alligator cracking on the pavement surface and the depth of rutting every semester over the pavement's service life.

Additionally, the design traffic is fed into MeDiNa in a customary Brazilian manner: the total AADT, growth rate, and design lane factor represent all vehicle traffic; but then the actual traffic is converted to a number of equivalent axle loads by applying the *Vehicle Factor*, which is a function of the percentage of all traffic corresponding to different axle types (such as singles, tandems, or tridem) of varying weights (DNIT, 2006). MeDiNa features an interface for the user to define the types of axles that would roll over the designed pavement and the percentage of the total traffic represented by each (Appendix D).

Besides, MeDiNa includes a basic library of pavement materials' loaded with all relevant properties that the AEMC core and the transfer functions would need to compute the distress generation accurately. Such default parameter values would match a Level-3 design input in the MEPDG jargon but the user can nonetheless add project-specific material-related data if available. It must be observed that on the topic of subgrade soils properties, MeDiNa does not accept sieving test results nor Atterberg limits, but the results of the MCT test[19] are warranted; the main reason being that both the SUCS and AASHTO soil

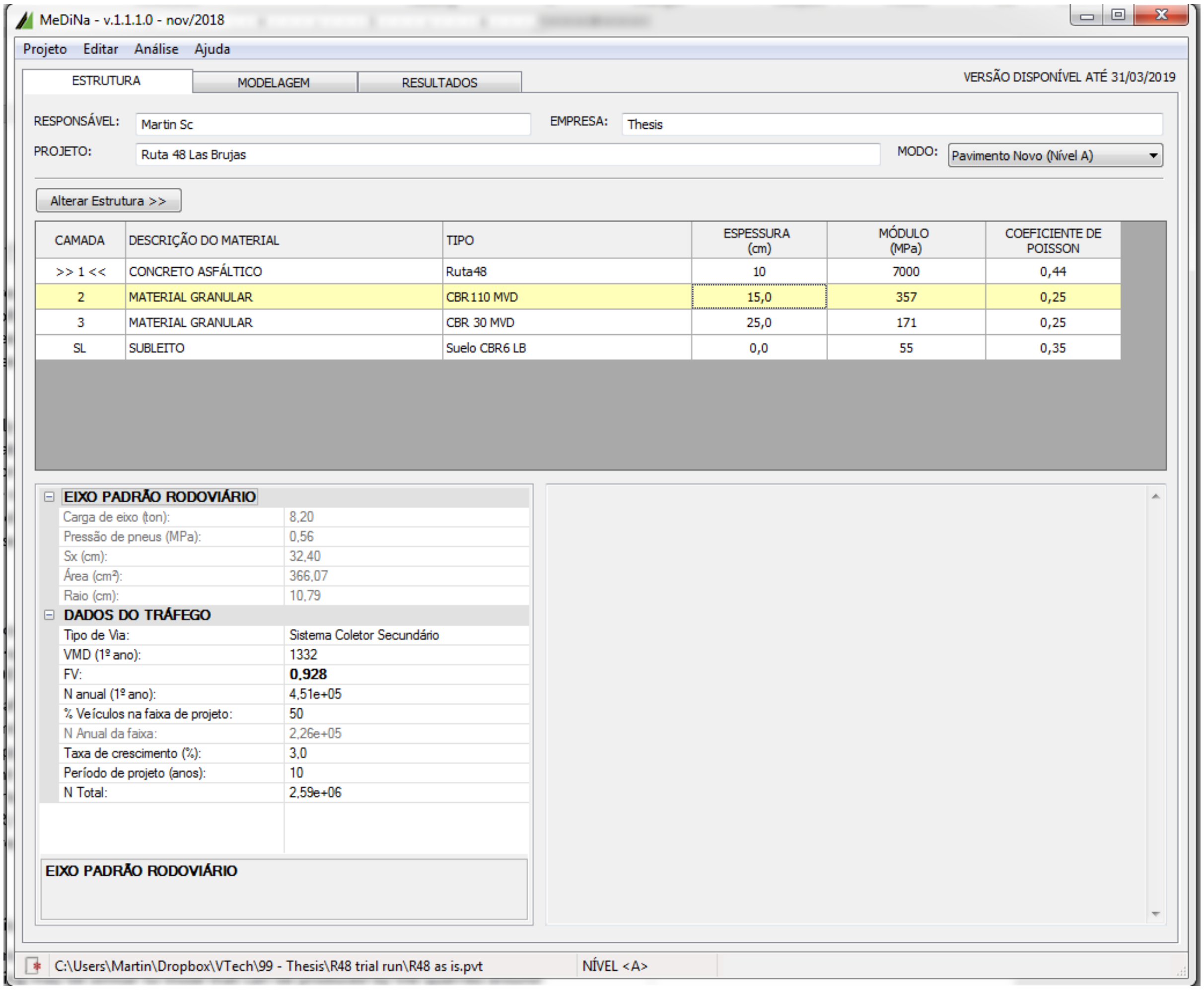
[19] MCT is the acronym for *Miniature, Compacted, Tropical* (Moura Fortes and Merighi, 2013), a reference to the test apparatus used in the characterization tests. For further reference about this soil classification system, please refer to Nogami et al. (1989), and the DNER-ME 254 (1997); DNER-ME 256 (1994); and DNER-ME 258 (1994) standards.

classification systems may fail to properly represent Brazilian tropical subgrades, whilst the MCT-based classification has been crafted purposefully to classify tropical soils (Moura Fortes and Merighi, 2013). However, the creators of the MCT-based classification system also furnish a comparison table between the MCT soil categories and the AASHTO and SUCS classifications (Nogami et al, 1989).

Finally, it must be highlighted though that, contrary to SisPavBR, MeDiNa does not take into consideration the site climate into the design process – its predecessor required temperature records be loaded along with the traffic and materials' inputs.

Concerning this particular design example, the design inputs were loaded into MeDiNa as follows:

- The pavement structure has been defined with the thicknesses given by the AASHTO '93 method and appealing to the "default" materials that resemble the Uruguayan counterparts the most in terms of strength and physical properties.
- According to the chart by Nogami et al. (1989), the site's silty soil may resemble an NS' [sitly saprolitic (non-lateritic)] soil. MeDiNa includes default strength parameters for such soils and those were utilized herein. Figure 31 depicts the trial structure in MeDiNa's main menu.
- The design traffic has been loaded as AADT plus vehicle factor *("Fator de Veiculo – FV"* in Portuguese), and MeDiNa's interface has been utilized to add the relevant axle types and the percentage of total traffic corresponding to each (Appendix D).



VERSÃO DISPONÍVEL ATÉ 31/03/2019

RESPONSÁVEL: Martin Sc

EMPRESA: Thesis

PROJETO: Ruta 48 Las Brujas

MODO: Pavimento Novo (Nível A)

CAMADA	DESCRIÇÃO DO MATERIAL	TIPO	ESPESSURA (cm)	MÓDULO (MPa)	COEFICIENTE DE POISSON
>> 1 <<	CONCRETO ASFÁLTICO	Ruta48	10	7000	0,44
2	MATERIAL GRANULAR	CBR 110 MVD	15,0	357	0,25
3	MATERIAL GRANULAR	CBR 30 MVD	25,0	171	0,25
SL	SUBLEITO	Suelo CBR6 LB	0,0	55	0,35

EIXO PADRÃO RODOVIÁRIO	
Carga de eixo (ton):	8,20
Pressão de pneus (MPa):	0,56
Sx (cm):	32,40
Área (cm²):	366,07
Raio (cm):	10,79
DADOS DO TRÁFEGO	
Tipo de Via:	Sistema Coletor Secundário
VMD (1º ano):	1332
FV:	**0,928**
N anual (1º ano):	4,51e+05
% Veículos na faixa de projeto:	50
N Anual da faixa:	2,26e+05
Taxa de crescimento (%):	3,0
Período de projeto (anos):	10
N Total:	2,59e+06

EIXO PADRÃO RODOVIÁRIO

C:\Users\Martin\Dropbox\VTech\99 - Thesis\R48 trial run\R48 as is.pvt   NÍVEL <A>

Figure 31: Screen capture of MeDiNa's main menu

After execution, MeDiNa would return the final fatigue-cracked area and rut depth. The predicted evolution of such distresses is given in table 5, while table 6 elaborates on remarking which layers of the trial structure suffered the most rutting. It stands out that according to MeDiNa the predicted structure will develop fatigue cracking over 13.5% of the pavement surface area and **that the overall rut depth hits 2.7 mm (0.12 inch), the subgrade layer taking the lion's share (1.44 mm, 53.3%).**

Table 5: MeDiNa Distress Results

Month	ESALS	Cracked Area	Rut depth [mm]
1	1.855e+04	0.66%	1.8
6	1.120e+05	1.33%	2.1
12	2.256e+05	1.83%	2.2
18	3.409e+05	2.28%	2.3
24	4.579e+05	2.71%	2.3
30	5.767e+05	3.14%	2.4
36	6.973e+05	3.58%	2.4
42	8.196e+05	4.04%	2.4
48	9.438e+05	4.53%	2.5
54	1.070e+06	5.04%	2.5
60	1.198e+06	5.58%	2.5
66	1.327e+06	6.16%	2.5
72	1.459e+06	6.78%	2.5
78	1.593e+06	7.44%	2.6
84	1.729e+06	8.14%	2.6
90	1.866e+06	8.90%	2.6
96	2.006e+06	9.70%	2.6
102	2.148e+06	10.56%	2.6
108	2.292e+06	11.47%	2.6
114	2.438e+06	12.45%	2.6
120	2.586e+06	13.49%	2.7

Table 6: MeDiNa Rut Depth Detailed Report

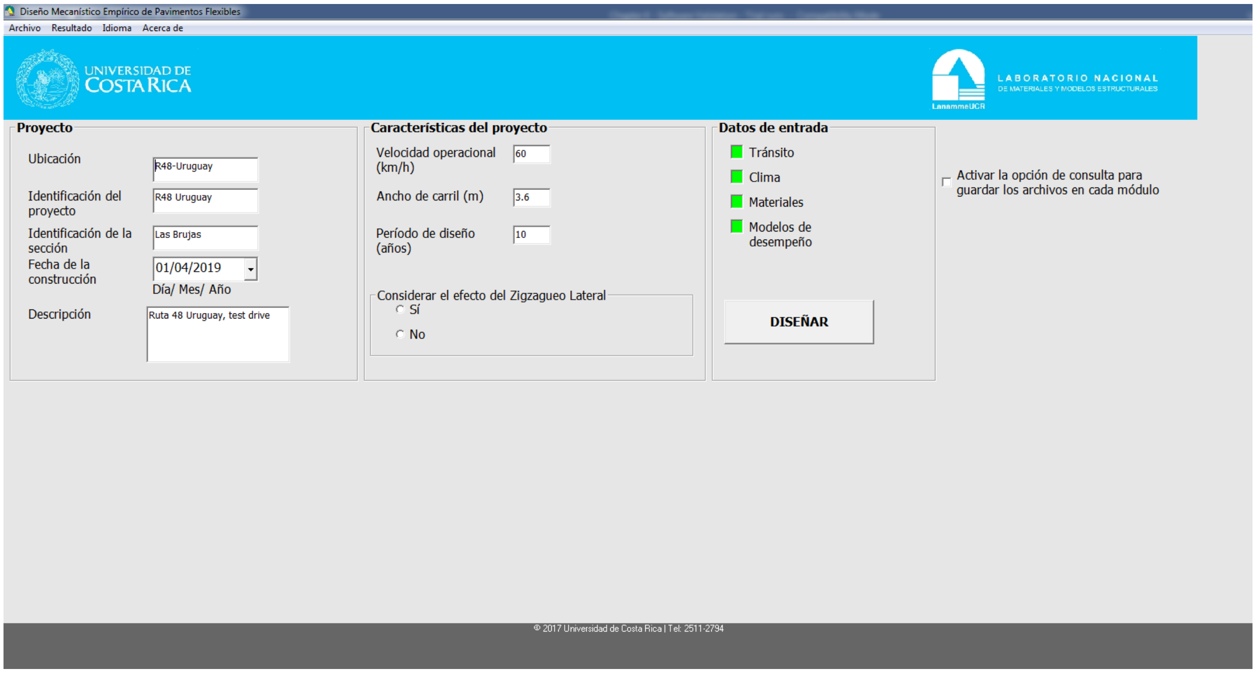
Layer	Material	Rut depth (mm)
1	Asphalt concrete	0.00
2	Granular material	0.62
3	Granular material	1.44
4	Subgrade	0.60
	Total rut depth (mm)	**2.70**

4.4.2 Results from CR-ME

The trial structure for this case study was also analyzed with CR-ME. This software follows a step-wise approach to the data input process. Data about expected traffic, climate, and materials are loaded in sequence, the user may only proceed to the following input data category only if the minimum data requirements for the current category are fully met. Furthermore, CR-ME supports three levels of input in a similar way as the MEPDG does, level-1 allowing for the highest degree of detail while level-3 on the other hand allows for designs to be done with little data available.

Figure 32 is CR-ME's main menu window, the green check marks next to each data input category signal that all required data needs have been fulfilled.

Firstly, the predicted design traffic is loaded onto the software. Levels 2 and 3 inputs for traffic variables are based on vehicle counts and truck factors, which are then used to compute equivalent axle loads (Loría Salazar, 2013), whereas level-1 traffic data consists of the predicted axle count by configuration and load. CR-ME accepts single-wheel and dual-wheel single axles, dual-wheel tandem axles, and dual-wheel tridem axles. For this project, the total axle counts computed with the ***trafficToAxles*** function in Product-One for the entire design period were loaded into CR-ME, as shown in Appendix D.


Diseño Mecanístico Empírico de Pavimentos Flexibles
Archivo
Resultado
Idioma
Acerca de
UNIVERSIDAD DE COSTA RICA
LABORATORIO NACIONAL DE MATERIALES Y MODELOS ESTRUCTURALES
LanammeUCR
DISEÑAR
© 2017 Universidad de Costa Rica | Tel: 2511-2794

Proyecto

Ubicación	R48-Uruguay
Identificación del proyecto	R48 Uruguay
Identificación de la sección	Las Brujas
Fecha de la construcción	01/04/2019 (Día/ Mes/ Año)
Descripción	Ruta 48 Uruguay, test drive

Características del proyecto

Velocidad operacional (km/h)	60
Ancho de carril (m)	3.6
Período de diseño (años)	10

Considerar el efecto del Zigzagueo Lateral
○ Sí
○ No

Datos de entrada

🟩 Tránsito
🟩 Clima
🟩 Materiales
🟩 Modelos de desempeño

□ Activar la opción de consulta para guardar los archivos en cada módulo

Figure 32: Screen capture of CR-ME's main menu

Afterward, the site's climate variables must be loaded. It must be highlighted that although the climate data requirement is the same across all input levels, CR-ME requires only average monthly temperatures and the values for the Thornwaite Moisture Index, which is a measurement of the site's aridity and is computed from the total amount of rainfall and evapotranspiration (Thornwaite, 1948), reproduced in equation 51.

Thornwaite's Moisture Index

$$TMI = \frac{100s - 60d}{ETP} \tag{51}$$

In order to compute a monthly TMI with equation 51, ETP is the site's annual evapotranspiration, and *s* and *d* are the amounts of water surplus (monthly precipitation minus ETP, when there is excess rainfall) and water deficit (ETP minus rainfall on "dry" months) respectively.

The average monthly temperature for the design period (its weather assumed to be that that has been predicted with Product-One) has been computed in a spreadsheet from the forecast hourly values and copied into CR-ME as is. The calculation of the moisture index, however, required both the rainfall averages for each month (also computed directly from the predicted hourly records) and an estimate of the potential ETP. To this end, Thornwaite's ETP model has been utilized (equation 52).

Thornwaite's Evapotranspiration formula

$$ETP\left[mm\right]=k \times ET\,P_{\left[12,30\right]}$$

where

$$\begin{gathered}
k=\frac{hours}{12}\times\frac{days}{30}\\
ETP_{12,30}\left[\frac{mm}{month}\right]=16\left(10\frac{temp}{I}\right)^{a}\\
a=6.75\times10^{-7}\times I^{3}-7.71\times10^{-5}I^{2}+1.792\times10^{-2}I+0.49239\\
I=\sum_{i+1}^{12} i_{j}\\
i=\left(\frac{temp}{5}\right)^{1.514}
\end{gathered} \tag{52}$$

In the above formula "*I*" is called the "annual heat index", *"i"* is similarly named the "monthly heat index". In all cases, the temperature values [*temp*] are input in degrees Celsius. The "hours" variable refers to the average number of daylight hours during the month (which depends on the site's latitude), and "days" is the number of days in the month.

It can be argued that Thornwaite's is not the most comprehensive ETP estimation model (Chow et al. 1993) since it only accounts for ETP due to temperature increase (and solar direction indirectly) and disregards the effect of wind. But such is also the model's virtue, as it allows for a fast ETP calculation while only requiring one single input variable. The resulting values of temperature, precipitation, ETP, and TMI are summarized in table 7, whereas Refer to Appendix D for CR-ME's climate variables input dialog.

The trial pavement structure (layer thickness and material properties) must be loaded onto the software. The software allows different input level data for each material category, namely asphalt mixes, unbound materials, cement-treated materials, and subgrade soils. The data requirements for each level are the same as those of the MEPDG, the materials properties data available for this project (retrieved from Product-One's default values library) matches the minimum needs for a level-3 design. The different CR-ME's data input screens for asphalt and unbound materials are shown in Appendix D with the properties of the trial structure's layers already loaded.

Table 7: Site's Climate Inputs for CR-ME's

Month	Avg. Temp. [C]	Rainfall [mm]	ETP [mm]	TMI month
1	23.2	58.0	131.3	-5.3
2	23.0	91.0	108.9	-1.3
3	19.8	78.3	87.2	-0.6
4	16.6	72.0	56.3	1.9
5	15.0	79.1	44.0	4.2
6	10.6	48.2	22.5	3.1
7	8.6	50.4	16.6	4.1
8	12.7	205.6	35.2	20.6
9	13.5	81.2	41.8	4.8
10	16.3	158.0	64.8	11.3
11	19.3	75.1	90.3	-1.1
12	22.7	167.0	128.4	4.7

As a side note, the site's annual TMI is 46.2, which according to Thornwaite (1948) corresponds to a "B2-Humid" climate.

An interesting feature in CR-ME's is its ability to construct a default master curve for asphalt mixes using volumetric parameters, basic binder properties (like its conventional AC grade, still in use in Uruguay (MTOP, 1990; MTOP, 2003)) and the Asphalt Institute E* model. These are shown in Appendix D for the two asphalt layers defined for Highway 48, along with the level-3 inputs for Highway 48's trial structure's granular layers and the subgrade, CR-ME's minimum data requirements match the information available in Product-One's material data library.

Finally, the user must specify what transfer functions and distress generation models are to be applied by CR-ME. For this exercise, the MEPDG models with their default calibration parameters (and not a user-specified parameter set) have been selected – these are the same formulas that were implemented in Product-One.

The execution results returned by CR-ME are distributed in three categories, namely material's properties, predicted rut depth, and predicted alligator cracking development. The following figures reproduce the most noteworthy outputs from the pavement structural evaluation done by CR-ME:

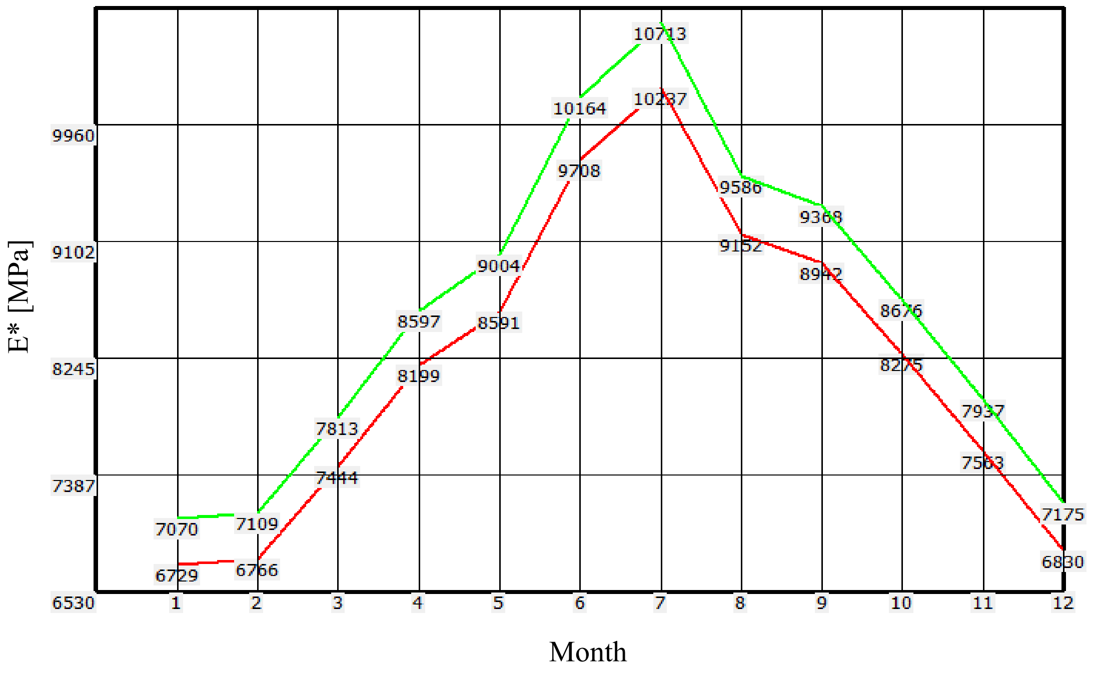


Figure 33: CR-ME's prediction of asphalt layers' dynamic moduli

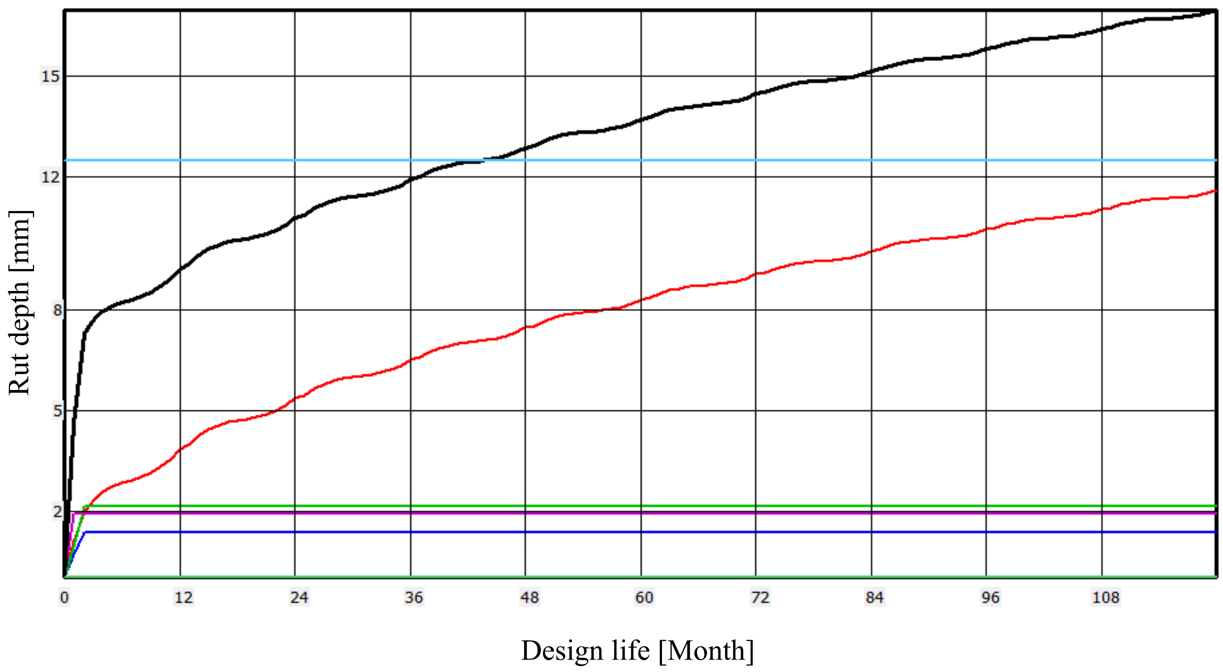


Figure 34: CR-ME's predicted rut depth increase

Note: Plots' color code: black = total rut depth; red = asphalt layers rut depth; blue = base layer rut depth; green = sub-base layer rut depth; pink = subgrade rutting; light blue = rut depth threshold (preset at 12.5mm).

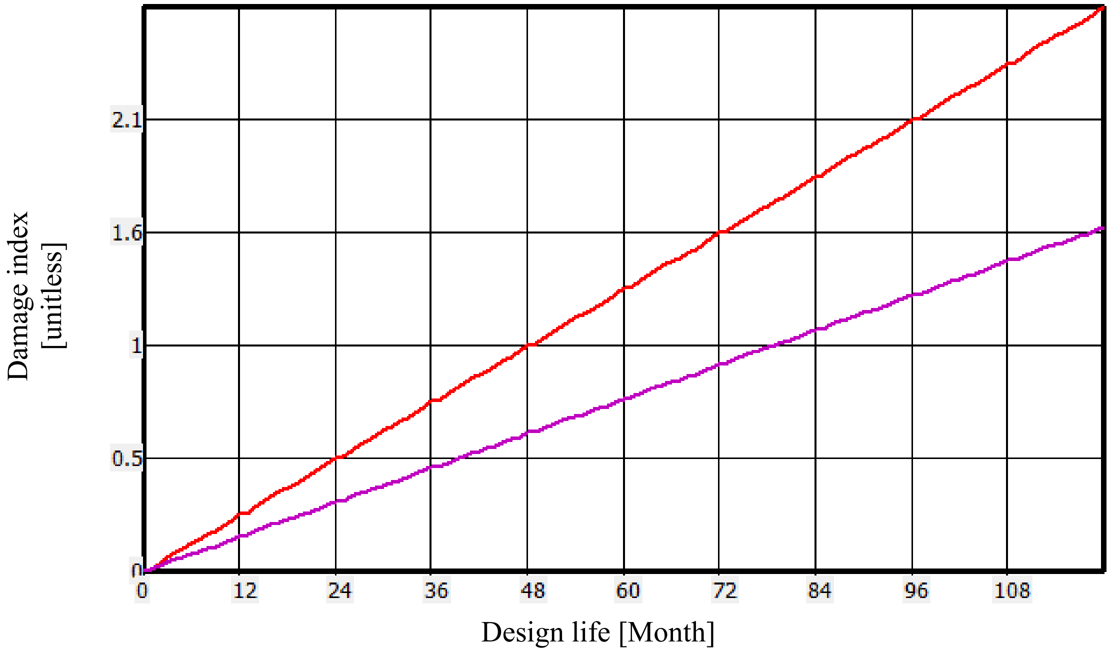


Figure 35: CR-ME's predicted fatigue damage

Plots' color code: red = bottom-up damage; purple = top-down damage.

CR-ME also produced graphical outputs for cracked area but these were not added for this report since the predicted values would not depart significantly from zero and since the plots cannot be scaled up no trend can be seen.

CR-ME also provides numerical outputs for the end-of-service period relevant distress rates. It holds that after the 10-year design period:

- The final alligator cracking damage factor is 2.6%; 0.3% of the roadway area will experience bottom-up cracking.
- The final longitudinal cracking damage factor is 1.6%; top-down cracking would extend to 0.08 m/km.
- The total vertical permanent deformation will reach 17mm [0.67 inches], the most of it taking place in the asphalt layers (11.4mm).

4.5 Trial structure evaluation results comparison

Figures 36 to 41 provide a direct comparison on the distress prediction given by the three M-E software tools, these are a more intuitive means to compare Product-One's calculation results against the two established software tools.

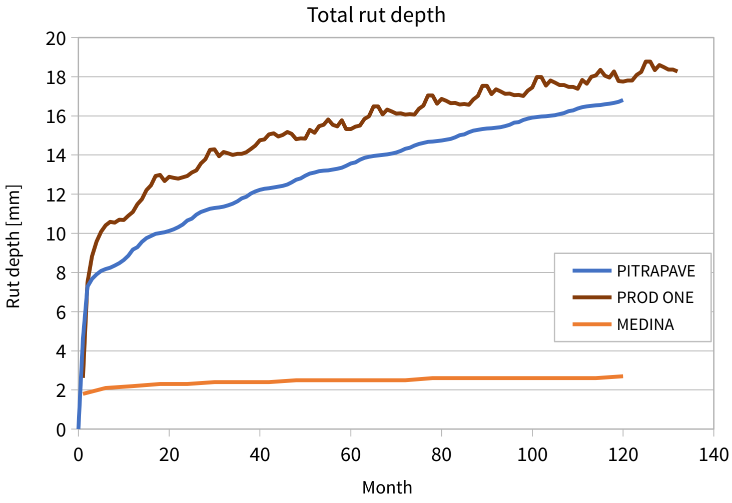


Figure 36: Predicted total rut depth comparison plot

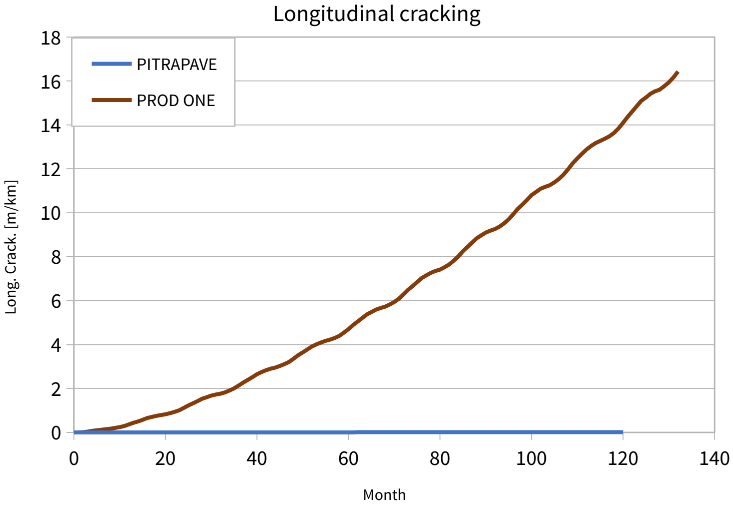


Figure 37: Predicted longitudinal cracking comparison plot.

Note: MeDiNa does not furnish predictions of longitudinal cracking.

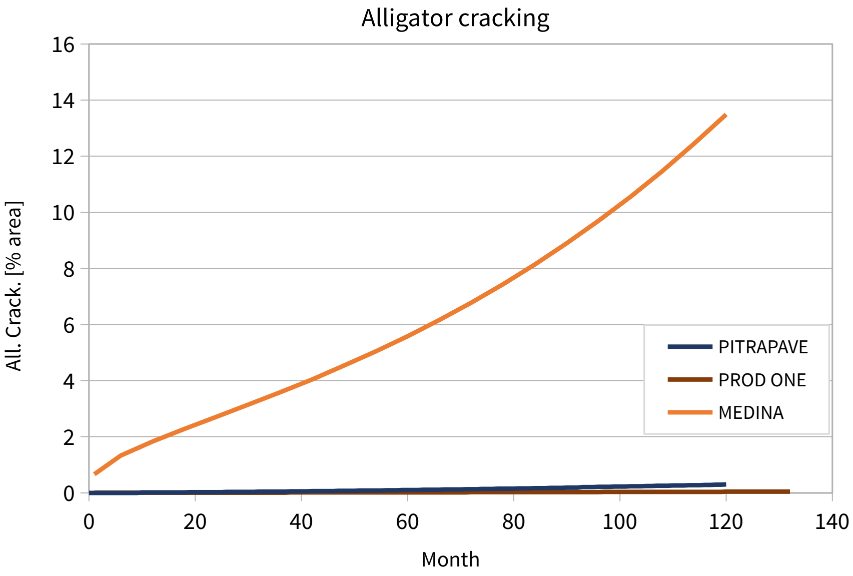


Figure 38: Predicted alligator cracking comparison plot

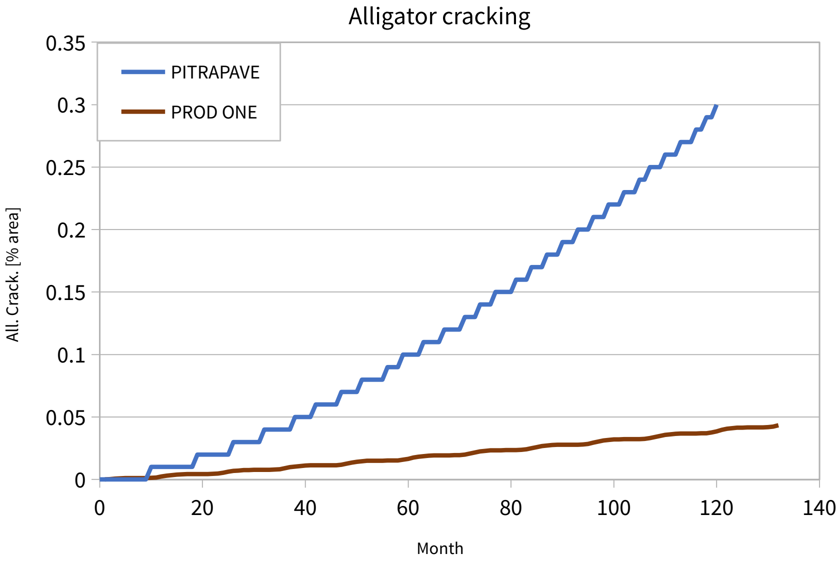


Figure 39: Predicted alligator cracking comparison plot. Pairwise comparison PitraPAVE vs Product-One

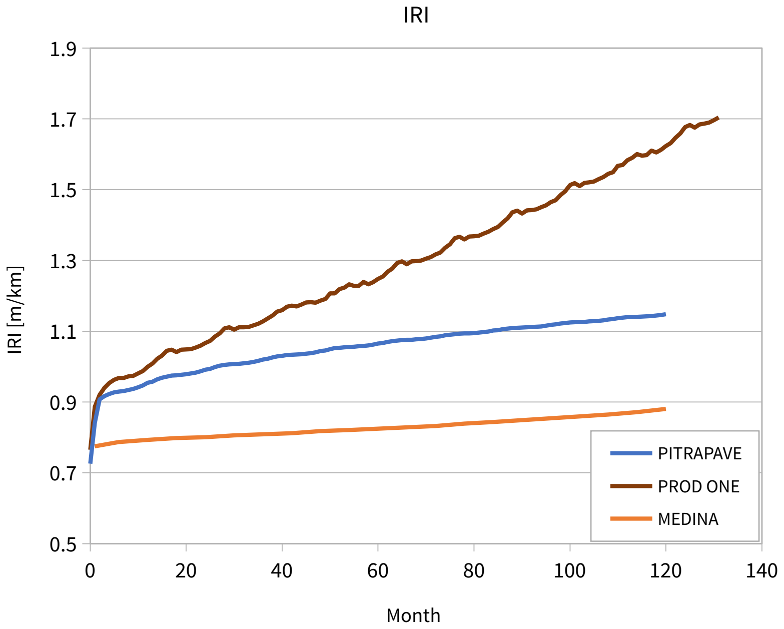


Figure 40: Predicted IRI comparison plot

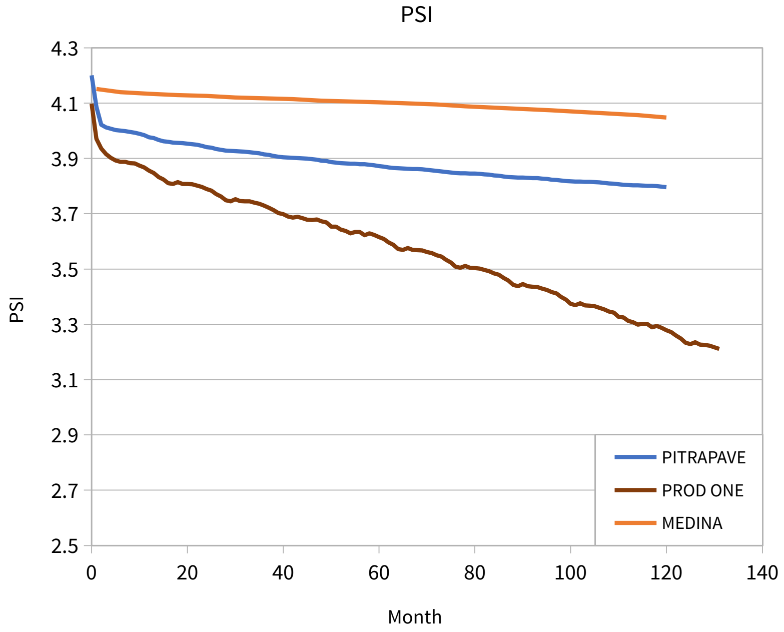


Figure 41: Predicted PSI comparison plot

Note: Neither Pitra-Pave nor MeDiNa natively produce IRI or PSI estimates. These have been computed with theMEPDG formula and the Al-Omari and Darter (1994) formula in the same manner Product-One does.

4.5.1 Conclusions of the Comparison

The main conclusions that can be inferred from the above plots are the following:

- MeDiNa predicts that the trial pavement structure would develop distress to a lesser degree compared both to PitraPAVE and Product-One – both based on the MEPDG. Such a trend aligns with the findings by Dambros Fernandes (2016), who highlighted that the AASHTOWare (a commercial software implementation of the MEPDG) turns out more severe distress predictions (and therefore thicker structures being needed) than the Brazilian M-E design tools.
- PitraPAVE's[20] and Product-One's rut depth prediction roughly align with each other (if the numerical instability issue with Product-One's rut depth model is disregarded). Such result proves that Product-One's models have been correctly programmed into code. However, both softwares' alligator cracking and longitudinal cracking predictions are divergent. The possibility exists that PitraPAVE utilizes the uncalibrated MEPDG formula for the Top-Down cracking k_1 value (equation 43), which yields values lower than Product-One's outcomes. Further analysis on the source code of both software tools may be warranted.
- The IRI predictions from PitraPAVE's and Product-One distress predictions held somewhat similar values. This is owed to the fact that rutting is accounted in a much more relevant way than cracking in the MEPDG's IRI equation. However, the end-of-period IRI values (slightly below 1.2 m/km) are notoriously smaller than what would be expected from the AASHTO (1993) model (the preset terminal PSI of 2.0 would match an IRI of roughly 3.5 m/km if the Al-Omari and Darter's formula is inverted). Given such results, it can be inferred that, compared to the MEPDG-based M-E models, the AASHTO (1993) model would underestimate the pavement performance by punishing its structural capacity and turning it more damage-prone (which leads to AASHTO (1993) returning a thicker structure. Such finding had also been reported by Ayman Aguib (2013), although it is the complete opposite to the conclusion drawn by El-Badawi (2011) and Boone (2013) after conducting similar comparative analysis for their respective local conditions.

From the points above it derives that the current version of Product-One, once the numerical issues with the rutting depth prediction model and the divergent cracking predictions are solved, may be able to aid a pavement engineer in designing a flexible structure with an M-E method and yield similar results to those that would be obtained with other MEPDG-based design tools at level of input 2/3 and with the same data

[20] PitraPAVE is running the MEPDG transfer functions and distress models with default calibration parameters for this exercise.

inputs. Nevertheless, since Product-One's distress prediction models are using the MEPDG default calibration parameters, the actual extent of distress to take place in the pavement structure after the service period may differ to the predicted values, thus stressing the need for calibration. Yet if such a calibration effort is eventually carried out on Uruguayan roads, its outcome may be applicable to Product-One and (either directly or after slight adjustment) to other MEPDG M-E design suites.

CHAPTER 5 – CONCLUSION

5.1 Final Remarks: Lessons Learned and Recommendations for Future Research

A functional version of a mechanistic-empirical (M-E) design tool for flexible pavements for Uruguayan roads (Product-One) has been built. It has been subjected to preliminary testing with other M-E design tools (MeDiNa and CR-ME, the latter also featuring an MEPDG-based core) and the outcome is auspicious: the extent of distress predicted by Product-One resembles CR-ME's predictions yet both software packages differ with the predictions by MeDiNa, which is not surprising given that similar trends were reported by Dambros Fernandes (2016). Moreover, Product-One's open source nature would also encourage further development towards its improvement in two ways, both of which may be grounds for further research:

- Inclusion of extra features that may improve the accuracy of the structure's behavior (like an asphalt aging model resembling that in the MEPDG, or actual visco-elastic behavior for asphalt materials in the stress and strain calculator), or enhanced design capabilities (like a framework for the design of rigid structures, not included in this version),
- Update of the existing code to reflect newer development in material characterization, such as performance grade characterization for asphalt binders, a policy not yet adopted in Uruguay.

Nonetheless, the foremost task to follow is to verify the reliability of Product-One' distress prediction models against field measurements in Uruguayan roads before adopting the program as a standardized design tool. In other words, the embedded distress models must be calibrated to Uruguayan conditions in the same way both American and foreign researchers have been proceeding prior to adopting the MEPDG in engineering practice. Yet the main advantage of Product-One using the same distress prediction equations as the MEPDG is that a single calibration effort may yield the adjustment parameters for both software suites without a remarkable amount of additional work.

Besides, on the availability of background information and project data for a most accurate design, this project concludes that Uruguayan laboratories and agencies' archives possess and still collect most of the data Product-One would require for level-2 design. However, these data is either partially available or unreachable for the designer. An effort must be undertaken to effectively broaden a library of input data for design, similar to that included with Product-One.

Finally, the draft-project design example presented in Chapter 4 allowed both to compare Product-One in performance terms against other M-E software design tools and also to assess the extent at which the pavement thickness design process can be undertaken by relying mostly in publicly and readily available information as input data. The data library (Product-Two) may fulfill the data needs whenever the publicly-available repositories are exhausted.

Level-3 design can be achieved with information available in the online repositories and of the local highway agencies and technical literature. Country-wide level-2 climate data is now available in Product-Two, so are the traffic distribution factors (on the assumption that any Uruguayan highway could be classified as belonging to any of the four functional categories described in Chapter 2), and materials information can be retrieved from local laboratories' archives – a task that could not be fully accomplished in this project.

Concerning Product-One's performance, this early release is lagging compared to the other available software (slower execution time and dependency on a Matlab interpreter), and its results, the predicted distress levels, are at a discrepancy with the other tested M-E software (for which special adaptations of the Uruguayan inputs needed nonetheless to be made). But all in all such differences in the predicted calculation outcomes would be negligible once the field calibration and validation of Product-One is completed.

It has been intended that the outcome of this project would bring the Uruguayan highway engineering practice closer to the state-of-the-art techniques in pavement design. Such has been a rather ambitious objective, and hence the limited capabilities of this project's main outcome [Product-One]. However, the author remains confident that after all this project will spearhead research into M-E pavement design for Uruguayan highways.

REFERENCES

- Abu Sufian, A. (2016): “Local Calibration of the Mechanistic-Empirical Pavement Design Guide for Kansas”. M.Sc. Thesis, Kansas State University.
- Al-Omari, B.; Darter, M. I. (1994): “Roughness Index and Present Serviceability Rating”. Transportation Research Record 1435. Pp. 130-136.
- Al-Qadi, I. L.; Elsefi, M. A.; Yoo, P. J.; Dessouky, S. H.; Gibson, N.; Harman, T.; D’Angelo, J.; Petros, K. (2008): “Accuracy of Current Complex Modulus Selection Procedure from Vehicular Load Pulse. NCHRP Project 1-37A Mechanistic-Empirical Pavement Design Guide”. Transportation Research Record 2087, Pp. 81-90.
- AASHTO (1993): “AASHTO Guide for the Design of Pavement Structures”. American Association of State Highway and Transportation Officials. ISBN: 1-56051-055-2
- AASHTO (2008): “Mechanistic-Empirical Pavement Design Guide. A Manual of Practice”. First Edition. American Association of State Highway and Transportation Officials . ISBN: 978-1-56051-423-7
- AASHTO (2010): “Guide for the Local Calibration of the Mechanistic-Empirical Pavement Design Guide”. First Edition. American Association of State Highway and Transportation Officials. ISBN: 978-1-56051-449-7
- AASHTO (2015): “Mechanistic-Empirical Pavement Design Guide. A Manual of Practice”. Second Edition. American Association of State Highway and Transportation Officials. ISBN: 978-1-56051-597-5
- Ayman Aguib, A. (2013): “Flexible Pavement Design. AASHTO 1993 versus Mechanistic-Empirical Pavement Design”. M. Sc. Thesis. American University at El Cairo. Egypt.
- Boone, J. N. (2013): “Comparison of Ontario Pavement Designs Using the AASHTO 1993 Empirical Method and the Mechanistic-Empirical Pavement Design Guide Method.” M. Sc. Thesis. University of Waterloo, Ontario, Canada. Available at: http://hdl.handle.net/10012/8047
- Bustos , M.; Cordo, O; Girardi, P.; Pereyra, M. (2011): “Calibration of MEPDG Distress models for Rigid Pavement Design in Argentina” Transportation Research Record No. 2226. Pp 3-12.
- Caicedo, B. (2019): “Geothecnics of Roads: Fundamentals.” first Edition. CRC Press. ISBN 978-1-315-22639-2
- Caliendo, C. (2012): “Local Calibration and Implementation of the Mechanistic-Empirical Pavement Design Guide for Flexible Pavement Design”. ASCE Journal of Transportation Engineering. Vol. 138, No. 3. Pp. 348-360.

- Carter, M; Bentley, S. P. (2016): “Soil Properties and their Correlations”. Second Edition. Wiley. ISBN 978-1-5231-1489-4.
- Chang Albitres, C.M.; Vidal, Valencia, J.; Loria Salazar, L.G.; Bustos, M.; Delgadillo, R. (2013): “Aplicabilidad del Método Mecanístico Empírico de Diseño de Pavimentos (MEPDG) AASHTO 2008 en Lationamérica”. Report IAG 239-03-2013.
- Chang, K-T. (2019): “Introduction to Geographic Information Systems”. 9th edition. McGraw Hill. ISBN 978-1-259-92964-9.
- Chow, V. T.; Maidment, D. R.; Mays, L. W. (1994): “Hidrología Aplicada”. McGraw Hill Interamericana S.A. ISBN 958-600-171-7.
- Cordo, O.; Bustos, M.; Girardi, P.; Pereyra, M (2008): “Calibración a Condiciones Locales en Argentina de la Guía Empírico-Mecanicista para el Diseño de Pavimentos Rígidos”. Final report, Project 21/I535. Escuela de Ingeniería de Caminos de Montaña, Facultad de Ingeniería U.N.S.J. Argentina, March 2008.
- Dambros Fernandes, W. (2016): “Análise Comparativa Entre os Métodos de Dimensionamento de Pavimentos Flexíveis do Brasil e o Método da AASHTO”. M. Sc. Thesis, Universidade de Santa Maria. July 2016.
- Davidson, j. M.; Stone, L. R.; Nielsen, D. R.; Larue, M. E. (1969): “Field Measurement and Use of Soil-Water Properties”. Water Resources Research. Vol 5, No. 6. Pp. 1312-1321.
- Delgadillo, R.; Wahr, C.; Alarcón, J.P. (2011): “Toward Implementation of the Mechanistic-Empirical Pavement Design Guide in Latin America. Preliminary Work in Chile”.Transportation Research Record No. 2226, pp 142-148.
- Dempsey, B. J.; Herlache, W. A.; Patel, A. J. (1986): “Climatic-Materials-Structural Pavement Analysis Program”. Transportation Research Record 1095, Pp. 111-123.
- DNER (1994): “Solos Compactados em equipamento miniatura – determinação da perda de massa por imersão. Método de Ensaio”. Ministerio dos Transportes, Departamento nacional de Estradas de Rodagem. Norma Rodoviária DNER-ME 256/94.
- DNER (1994): “Solos Compactados em equipamento miniatura – Mini-MCV. Método de Ensaio”. Ministerio dos Transportes, Departamento nacional de Estradas de Rodagem. Norma Rodoviária DNER-ME 258/94.
- DNER (1997): “Solos Compactados em equipamento miniatura – Mini-CBR e expansão. Método de Ensaio”. Ministerio dos Transportes, Departamento nacional de Estradas de Rodagem. Norma Rodoviária DNER-ME 254/94.

- DNIT (2006): “Manual de Pavimentação”. Departamento Nacional de Infraestrutura dos Transportes. Publication ID-719. Third Edition.
- El-Badawy, S. M.; Bayomy, F. M.; Santi, M.; Clawson, C. W. (2011): “Comparison of Idaho Pavement Design Procedure with AASHTO 1993 and MEPDG Methods”. First Congress of Transportation and Development Institute. Pp. 586-595.
- FHWA (1993): “An Integrated Model Of The Climatic Effects On Pavements”. United States Department of Transportation, Federal Highway Administration. Publication ID FHWA-RD-90-033. November 1993.
- FHWA (2000): “Temperature Predictions and Adjustment Factors for Asphalt Pavement”. United States Department of Transportation, Federal Highway Administration. Publication ID FHWA-RD-98-085. June 2000
- FHWA (2006): “Geotechnical Aspects of Pavements Reference Manual”. United States Department of Transportation, Federal Highway Administration. Publication ID FHWA-NHI-05-037. May 2006.
- FHWA (2010): “Local calibration of the MEPDG Using Pavement Management Systems”. United States Department of Transportation, Federal Highway Administration. Publication ID HIF-11-026. July 2010
- FHWA (2016): “Traffic Monitoring Guide”. United States Department of Transportation, Federal Highway Administration . Publication ID: FHWA-PL-17-003. October 2016.
- Flintsch, G. W.; Medina, A.; Luongo, G; Ayerza, M.; Barchesi, A.; Krugman, M; Enrich, P (1998): “Análisis de Deformaciones Permanentes en Ruta 2 Rodó – Palmitas”. Presented at the 2o Congreso de la Vialidad Uruguaya. In Asociación Uruguaya de Caminos (1998): “Memorias del Congreso”. Pp. 273-288
- Franco, F. A. C. P (2007): “Método de Dimensionamento Mecanístico-Empírico de Pavimentos Asfálticos – SISPAV”. Ph. D. Thesis, Universidade Federal do Rio de Janeiro, COPPE. September, 2007
- Fredlund, D.G.; Xing, A. (1994): “Equations for the Soil-Water Characteristic Curve”. Canadian Geotechnical Journal 31. Pp. 521-532.
- Fredlund, D.G.; Xing, A.; Huang, S. (1994): “Predicting the Permeability Function for Unsaturated Soils Using the Soil-Water Characteristic Curve”. Canadian Geotechnical Journal 31. Pp. 533-546.

- Harvey, J. and Basheer, I. (2011): “California’s Transition to Mechanistic-Empirical Pavement Design”. Pavement Technology Update Vol 3. No. 1. May 2011. Pp 1-12.
- Huang, Y. (2004): “Pavement Analysis and Design”. Second Edition. Pearson. ISBN 978-81-317-2124-7.
- Johanneck, L.; Khazanovich, L. (2010): “Comprehensive Evaluation of Effect of Climate in Mechanistic-Empirical Pavement Design Guide Predictions”. Transportation Research Record 2170, pp. 45-55.
- Kim, S.; Ceylan, H.; Ma, D.; Gopalakrishnan, K. (2014): “Calibration of Pavement ME Design and Mechanistic-Empirical Pavement Design Guide Performance Prediction Models for Iowa Pavement Systems”. ASCE Journal of Transportation Engineering. Vol. 140.
- Kim, Y. R.; Jadoun, F. M.; Hou, T.; Muhtadi, N. (2011): “Local Calibration of the MEPDG for Flexible Pavement Design” Final Report. Research Project HWY-2007-007. Department of Civil, Construction, and Environmental Engineering, North Carolina State University.
- Larson, G. Dempsey, L (1997): “Enhanced Integrated Climatic Model. Version 2.0”. Final Report ID DTFA MN/DOT 721 14. October 1997.
- Li, Q.; Xiao, D. X.; Wang, K. C. P; Hall, K. D.; Qiu, Y. (2011): “Mechanistic-empirical pavement design guide (MEPDG): a bird’s-eye view”. Journal of Modern Transportation, volume 19, pp. 114-133.
- Liu, S. J. Jeyapalan, J. K.; Lytton, R.L. (1983): “Characteristics of Base and Subgrade Drainage of Pavements”. Transportation Research Record No 945, pp. 1-10
- Loría Salazar, L. G. (2013): “Desarrollo de la Guía de Diseño de Pavimentos de Costa Rica: CR_ME”. Technical document. Available at: https://www.lanamme.ucr.ac.cr/repositorio/handle/50625112500/533
- Moura Fortes, R.; Merighi, J. V. (2013): “The Use of MCT Methodology for Rapid Classification of Tropical Soils in Brazil”. International Journal of Pavements. Vol 2. No. 3. ISSN 1676-2797. Pp. 1-13.
- Muftah, A.; Bayomy, F.; Kassem, E. (2019): “Calibration Of The AASHTOWare Pavement ME Design Performance Models For Flexible Pavements In Idaho”. Presented at the 98th TRB Annual Meeting. Jan 13th-17th, 2019.
- NCHRP (2004): “Guide for Mechanistic-Empirical Design of New and Rehabilitated Pavement Structures”. National Cooperative Highway Research Program. NCHRP Project 1-37A Report.

- Nogami, J. s.; Cozzolino, V. M. N; Villibor, D. F. (1989): Meaning of Coefficients and Index of MCT Soil Classification for Tropical Soils". Proceedings. International Conference on Soil Mechanics and Foundation Engineering. Pp. 547-550.
- Pierce, L. and McGovern, G. (2014): "Implementation of the AASHTO Mechanistic-Empirical Pavement Design Guide and Software". NCHRP Synthesis #457. ISBN: 978-0-309-27121-9
- Powell, W. D; Potter, J. F.; Nunn, M. E. (1984): "The Design of Bituminous Roads". Transportation and Road Research Laboratory. TRRL Report 1132. ISSN 0305-1293.
- Preciozzi Porta, F; Spoturno Pioppo, J.; Heinzen Marziotto, W.; Rossi Kempa, P. (1985): "Carta Geológica del Uruguay a Escala 1:500000. Texto Explicativo". Dirección Nacional de Minería y Geología, Ministerio de Industria y Energía.
- Quarteroni, A; Sacco, R.; Saleri, F. (2000): "Numerical Mathematics". Texts in Applied Mathematics 37. Springer. ISBN: 0-387-98959-5.
- Severova, V. (1997): "Clima del Uruguay". Online report, available at https://www.rau.edu.uy/uruguay/geografia/Uy_c-info.htm
- Smith, B.; Nair, H. (2015): "Development of local Calibration Factors and Design Criteria Values for Mechanistic-Empirical Pavement Design". Report 16-R1. Virginia Center For Transportation Innovation and Research.
- Soulamainian, M.; Kennedy, T.W. (1993): "Predicting Maximum Pavement Surface Temperature Using Maximum Air Temperature and Hourly Solar Radiation". Transportation Research Record No 1417, pp 1- 11.
- Tarefder, R.; Rodriguez-Ruiz, J. (2013): "Local Calibration of MEPDG for Flexible Pavements in New Mexico" ASCE Journal of Transportation Engineering. Vol 139, no 10. Pp. 981-991
- Thornwaite, C. W. (1948): "An Approach Toward a Rational Classification of Climate". Geographical Review, Vol. 38, No 1. Pp 55-94.
- Trejos Castillo, C.; Rojas Pérez, F.; Loría Salazar, L. G.; Aguiar Moya, J. P. (2017): "Solución a la Teoría Multicapa Elástica y Software de Cálculo de las Respuestas del Pavimento PITRA PAVE". Presented at the XIX Congreso Ibero-Latinoamericano del Asfalto, Medellín 2017.
- Trejos-Castillo, C. ; Aguiar-Moya, J. P. ; Loría-Salazar, L. G. (2016): "Desarrollo de Software de Análisis y Diseño de Pavimentos para Costa Rica".Technical report LM-PI-UMP-057-R2. Programa de Infraestructura del Transporte, Lanamme, Universidad de Costa Rica. December 2016. Available at: https://www.lanamme.ucr.ac.cr/repositorio/handle/50625112500/294

- Tseng, K. H.; Lytton, R. L. (1989): “Prediction of Permanent Deformation in Flexible Pavement Materials”. In “Implication of Aggregates in the Design, Construction, and Performance of Flexible Pavements”. ASTM STP 1016. ISBN 0-8031-1193-2.
- Ullditz, P.; Harvey, J.; Basheer, I.; Jones, D.; Wu, R. (2010): “CalME, a Mechanistic-Empirical Program to Analyze and Design Flexible Pavement Rehabilitation”. Transportation Research Record No. 2153. Pp 143-152.
- Uruguay (1984): “Reglamento Nacional de Circulación Vial”. Decree 118/1984. Available at: http://www.mtop.gub.uy/documents/20182/21195/Decreto+N%C2%BA+118-984+Reglamento+Nacional+de+circulaci%C3%B3n+vial/97a12580-5594-4d7c-9274-4bad01c2321e?version=1.1
- MTOP (1990): “Texto Ordenado del Pliego de Condiciones de la Dirección Nacional de Vialidad para la Construcción de Puentes y Carreteras” Decree 9/1990. Ministerio de Transporte y Obras Públicas. Available at: http://www.mtop.gub.uy/pliegos-generales
- MTOP (2003): “Especificaciones técnicas complementarias y/o modificativas del Pliego de Condiciones para la Construcción de Puentes y Carreteras de la Dirección Nacional de Vialidad”. Dirección Nacional de Vialidad, Uruguay, Ministerio de Transporte y Obras Públicas. August 2003
- MTOP (2017): “Anuario Estadístico de Transporte 2016”. Ministerio de Transporte y Obras Públicas. Available online at: http://www.mtop.gub.uy/anuario-estadistico-de-transporte-2016
- Vehrencamp, J. E. (1953): “Experimental Investigation of Heat Transfer at an Air-Earth Interface”. Transactions, American Geophysical Union. Vol. 34, No. 1. Pp. 22-30.
- Waltham, T. (2002): “Foundations of Engineering Geology”. Second Edition. Taylor and Francis. ISBN: 0-203-78043-4.
- Wei, T. C. and McGuiness, J. L. (1973): “Reciprocal Distance Squared Method. A Computer Technique for Estimating Areal Precipitation”. United States Department of Agriculture report ARS-NC-8, August 1973.
- World Meteorological Organization (2011): “Guide to Climatological Practices”. 2011 Edition. ISBN: 978-92-63-10100-6.
- Wu, Z.; Xiao, D.; Zhang, Z.; Temple, W. (2014): “Evaluation of AASHTO Mechanistic-Empirical Pavement Design Guide for Designing Rigid Pavements in Louisiana”. International Journal of Pavement Research and Technology, Vol 7, No 6. Pp 405-416.

APPENDIX A – WEATHER STATION INFORMATION

Table 1 below portrays a graphical depiction of the Uruguayan meteorological raw data that has been made available for this Project. These hourly data has been drawn from the INIA weather stations and span over the last decade – hourly data records go as far back as 2010 and the INIA network is still recording. The actual numerical data for the five weather variables of interest (viz. Air temperature, air moisture, rainfall, wind speed, and solar radiation) have been stored within Product-One in the ***INIAClimate.mat*** data file and are available there for visualization.

Table 1: Meteorological Data Availability From INIA Weather Stations

Date		La Estanzuela					Las Brujas					Durazno					Salto Grande					Glencoe					Tacuarembo					Treinta y Tres					Rocha				
Year	**Mo**	**T**	**H**	**W**	**S**	**R**	**T**	**H**	**W**	**S**	**R**	**T**	**H**	**W**	**S**	**R**	**T**	**H**	**W**	**S**	**R**	**T**	**H**	**W**	**S**	**R**	**T**	**H**	**W**	**S**	**R**	**T**	**H**	**W**	**S**	**R**	**T**	**H**	**W**	**S**	**R**
2010	1	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓										
2010	2	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	░	░	░	░	░										
2010	3	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓										
2010	4	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓										
2010	5	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓										
2010	6	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓										
2010	7	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓										
2010	8	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓										
2010	9	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓										
2010	10	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	░	░	░	░	░					
2010	11	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2010	12	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2011	1	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓										
2011	2	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	░	░	░	░	░					
2011	3	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	░	░	░	░	░					
2011	4	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2011	5	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2011	6	▓	▓	▓	▓		░	░	░	░	░											░	░	░	░	░	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2011	7	▓	▓	▓	▓		░	░	░	░	░											░	░	░	░	░	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2011	8	▓	▓	▓	▓		░	░	░	░	░											░	░	░	░	░	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2011	9	▓	▓	▓	▓		░	░	░	░	░											░	░	░	░	░	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2011	10	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2011	11	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2011	12	▓	▓	▓	▓		░	░	░	░	░											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2012	1	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2012	2	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2012	3	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2012	4	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2012	5	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2012	6	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2012	7	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2012	8	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2012	9	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					
2012	10	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓											▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓	▓					

Date		La Estanzuela					Las Brujas					Durazno					Salto Grande					Glencoe					Tacuarembo					Treinta y Tres					Rocha				
Year	Mo	T	H	W	S	R	T	H	W	S	R	T	H	W	S	R	T	H	W	S	R	T	H	W	S	R	T	H	W	S	R	T	H	W	S	R	T	H	W	S	R
2012	11	■	■	■	■	■	■	■	■	■	■											■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2012	12	■	■	■	■	■	■	■	■	■	■											■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2013	1	■	■	■	■	■	■	■	■	■	■											■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2013	2	■	■	■	■	■	■	■	■	■	■											■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2013	3	■	■	■	■	■	■	■	■	■	■											■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2013	4	■	■	■	■	■	■	■	■	■	■											■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2013	5	■	■	■	■	■	■	■	■	■	■											■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2013	6	■	■	■	■	■	■	■	■	■	■											■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2013	7	■	■	■	■	■	■	■	■	■	■											■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2013	8	■	■	■	■	■	■	■	■	■	■											■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2013	9	■	■	■	■	■	■	■	■	■	■											■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2013	10	■	■	■	■	■	■	■	■	■	■											▒	▒	▒	▒	▒	■	■	■	■	■	■	■	■	■	■					
2013	11	■	■	■	■	■	■	■	■	■	■																■	■	■	■	■	■	■	■	■	■					
2013	12	■	■	■	■	■	■	■	■	■	■	▒	▒	▒	▒	▒											■	■	■	■	■	■	■	■	■	■					
2014	1	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■											■	■	■	■	■	■	■	■	■	■					
2014	2	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■											■	■	■	■	■	■	■	■	■	■					
2014	3	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■						▒	▒	▒	▒	▒	■	■	■	■	■	■	■	■	■	■					
2014	4	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■						■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2014	5	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■						■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2014	6	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■						■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2014	7	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■						■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2014	8	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■						■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2014	9	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	▒	▒	▒	▒	▒	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2014	10	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2014	11	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2014	12	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2015	1	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2015	2	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2015	3	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2015	4	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2015	5	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2015	6	▒	▒	▒	▒		■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2015	7	■	■	■	■		■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2015	8	■	■	■	■		■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2015	9	▒	▒	▒	▒		■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2015	10	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					

Date		La Estanzuela					Las Brujas					Durazno					Salto Grande					Glencoe					Tacuarembo					Treinta y Tres					Rocha				
Year	Mo	T	H	W	S	R	T	H	W	S	R	T	H	W	S	R	T	H	W	S	R	T	H	W	S	R	T	H	W	S	R	T	H	W	S	R	T	H	W	S	R
2015	11	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2015	12	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2016	1	■	■	■	■	■	▒	▒	▒	▒	▒	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2016	2	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2016	3	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2016	4	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2016	5	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	▒	▒	▒	▒	▒	■	■	■	■	■	■	■	■	■	■					
2016	6	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	▒	▒	▒	▒	▒	■	■	■	■	■	■	■	■	■	■					
2016	7	■	■	■	■	■	▒	▒	▒	▒	▒	■	■	■	■	■	▒	▒	▒	▒	▒	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2016	8	■	■	■	■	■	▒	▒	▒	▒	▒	■	■	■	■	■						■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2016	9	■	■	■	■	■	▒	▒	▒	▒	▒	■	■	■	■	■	▒	▒	▒	▒	▒	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2016	10	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2016	11	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	▒	▒	▒	▒	▒	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2016	12	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■						■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2017	1	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■						■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2017	2	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■						■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2017	3	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	▒	▒	▒	▒	▒	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2017	4	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	▒	▒	▒	▒	▒	■	■	■	■	■					
2017	5	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2017	6	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
2017	7	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	▒	▒	▒	▒	▒
2017	8	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
2017	9	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
2017	10	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
2017	11	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
2017	12	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
2018	1	■	■	■	■	■	■	■	■	■	■	▒	▒	▒	▒	▒	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
2018	2	■	■	■	■	■	■	■	■	■	■	▒	▒	▒	▒	▒	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
2018	3	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
2018	4	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
2018	5	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
2018	6	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
2018	7	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	▒	▒	▒	▒	▒	▒	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
2018	8	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	▒	▒	▒	▒	▒	▒	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
2018	9	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	▒	▒	▒	▒	▒	▒	■	■	■	■	■	■	■	■	■	■	▒	▒	▒	▒	▒
2018	10	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					

Date		La Estanzuela					Las Brujas					Durazno					Salto Grande					Glencoe					Tacuarembo					Treinta y Tres					Rocha				
Year	**Mo**	**T**	**H**	**W**	**S**	**R**	**T**	**H**	**W**	**S**	**R**	**T**	**H**	**W**	**S**	**R**	**T**	**H**	**W**	**S**	**R**	**T**	**H**	**W**	**S**	**R**	**T**	**H**	**W**	**S**	**R**	**T**	**H**	**W**	**S**	**R**	**T**	**H**	**W**	**S**	**R**
2018	11	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	□	□	□	□	□
2018	12	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	□	□	□	□	□

Legend:

- T: Temperature
- H: Humidity
- W: wind speed
- S: Solar radiation
- R: Rainfall

■	Full Month
▨	Partial data
□	No Data

Last data retrieval: 2019-02-12

APPENDIX B – TRAFFIC DISTRIBUTION FACTORS

This Appendix contains the final average monthly, daily, and hourly traffic distribution factors for the four functional road categories discussed in Chapter 2; plus the results of the analyses of variance (ANOVA) on the daily and hourly factors for the different days of the week and months of the year.

Monthly traffic distribution factors

Table B-1: Monthly Traffic Distribution Factors for a Suburban Arterial Highway

Month	Vehicle Type				
	Cars	**Buses**	**Light trucks**	**Mid-size trucks**	**Heavy haul trucks**
January	0.91	0.89	0.84	0.89	0.74
February	0.98	1.03	0.95	0.89	0.73
March	1.03	1.02	0.96	0.92	0.79
April	0.91	0.92	0.87	0.90	0.82
May	0.97	1.00	0.98	1.04	1.13
June	1.01	1.11	0.96	1.17	1.05
July	1.00	1.02	0.92	1.02	0.96
August	0.98	0.94	1.05	1.07	1.17
September	1.01	1.01	1.15	1.11	1.30
October	1.04	1.03	1.13	1.01	1.08
November	1.06	1.05	1.15	1.00	1.22
December	1.09	0.98	1.06	0.98	1.00

Table B-2: Monthly Traffic Distribution Factors for a Heavy Haul Corridor With Uniform Traffic

Month	Vehicle Type				
	Cars	**Buses**	**Light trucks**	**Mid-size trucks**	**Heavy haul trucks**
January	1.29	0.87	1.12	0.80	0.64
February	1.22	0.91	1.32	0.95	0.69
March	0.94	0.89	0.93	0.94	0.75
April	1.13	1.08	1.12	1.11	1.12
May	0.76	0.86	0.86	0.79	0.98
June	0.79	0.92	0.87	0.97	1.05
July	0.93	0.93	0.83	0.90	1.17
August	0.84	0.95	0.84	1.00	1.10
September	0.99	1.09	0.90	1.08	1.17
October	1.03	1.15	0.95	1.17	1.03
November	1.03	1.18	1.14	1.18	1.12
December	1.05	1.19	1.13	1.10	1.16

Table B-3: Monthly Traffic Distribution Factors for a Heavy Haul Corridor With Seasonal Peak Traffic

Month	Vehicle Type				
	Cars	**Buses**	**Light trucks**	**Mid-size trucks**	**Heavy haul trucks**
January	1.26	1.10	1.02	0.89	0.82
February	1.16	1.26	1.14	1.01	0.84
March	1.01	0.98	1.10	1.02	0.96
April	1.18	1.08	0.90	0.93	0.93
May	0.83	0.92	0.82	0.94	1.04
June	0.82	0.94	0.84	1.21	1.26
July	0.91	0.88	0.82	0.89	1.04
August	0.85	0.87	0.84	0.85	0.93
September	0.94	0.95	0.86	0.86	0.90
October	0.92	1.02	1.04	1.05	1.13
November	0.97	1.05	1.30	1.26	1.25
December	1.13	0.96	1.32	1.10	0.92

Table B-4: Monthly Traffic Distribution Factors for a Mixed Traffic Arterial Road With Passenger Car Peak

Month	**Vehicle Type**				
	Cars	**Buses**	**Light trucks**	**Mid-size trucks**	**Heavy haul trucks**
January	1.06	0.93	1.04	0.88	0.95
February	1.18	1.04	1.25	0.88	0.91
March	0.99	0.99	1.17	0.97	0.98
April	1.46	1.11	1.10	0.97	1.06
May	0.86	0.97	0.95	1.19	1.24
June	0.85	0.97	0.90	1.14	1.20
July	0.96	0.95	0.82	0.95	0.96
August	0.85	1.01	0.90	1.02	0.98
September	0.90	1.04	0.96	0.98	0.94
October	0.96	1.08	0.96	0.96	0.96
November	0.86	0.91	0.89	0.92	0.85
December	1.06	0.99	1.06	1.14	0.96

Daily traffic distribution factors

Table B-5: Daily Traffic Distribution Factors for a Suburban Arterial Highway

Day	Vehicle Type				
	Cars	**Buses**	**Light trucks**	**Mid-size trucks**	**Heavy haul trucks**
Sunday	0.72	0.69	0.63	0.56	0.64
Monday	1.04	1.06	1.08	1.06	1.04
Tuesday	1.06	1.05	1.10	1.11	1.09
Wednesday	1.08	1.07	1.12	1.15	1.16
Thursday	1.08	1.08	1.11	1.14	1.10
Friday	1.13	1.17	1.14	1.15	1.10
Saturday	0.89	0.88	0.82	0.83	0.87

Table B-6: Daily Traffic Distribution Factors for a Heavy Haul corridor With Uniform Traffic

Day	Vehicle Type				
	Cars	**Buses**	**Light trucks**	**Mid-size trucks**	**Heavy haul trucks**
Sunday	0.96	0.93	0.91	0.88	0.90
Monday	1.05	1.07	1.05	1.05	1.06
Tuesday	0.91	0.94	0.94	0.97	0.96
Wednesday	0.94	0.95	0.96	0.99	0.98
Thursday	1.01	1.00	1.02	1.02	1.03
Friday	1.17	1.17	1.19	1.18	1.17
Saturday	0.93	0.93	0.91	0.91	0.90

Table B-7: Daily Traffic Distribution Factors for a Heavy Haul Corridor With Seasonal Peak Traffic

Day	**Vehicle Type**				
	Cars	**Buses**	**Light trucks**	**Mid-size trucks**	**Heavy haul trucks**
Sunday	1.10	0.77	0.62	0.40	0.51
Monday	1.04	1.08	1.01	1.00	1.06
Tuesday	0.86	1.04	1.08	1.25	1.23
Wednesday	0.83	0.98	1.11	1.30	1.25
Thursday	0.89	0.95	1.19	1.22	1.21
Friday	1.21	1.15	1.24	1.18	1.16
Saturday	1.06	1.06	0.83	0.78	0.70

Table B-8: Daily Traffic Distribution Factors for a Mixed Traffic Arterial Road With Passenger Car Peak

Day	**Vehicle Type**				
	Cars	**Buses**	**Light trucks**	**Mid-size trucks**	**Heavy haul trucks**
Sunday	1.14	0.90	0.63	0.37	0.43
Monday	0.98	1.14	1.01	1.05	1.11
Tuesday	0.85	1.00	1.08	1.32	1.29
Wednesday	0.86	0.94	1.13	1.38	1.35
Thursday	0.91	0.94	1.21	1.30	1.28
Friday	1.19	1.11	1.23	1.11	1.09
Saturday	1.09	1.03	0.80	0.60	0.59

Hourly traffic distribution factors

Table B-9: Hourly Traffic Distribution Factors for a Suburban Arterial Highway

Hour	**Vehicle Type**				
	Cars	**Buses**	**Light trucks**	**Mid-size trucks**	**Heavy haul trucks**
00:00	0.018	0.033	0.014	0.009	0.023
01:00	0.012	0.011	0.010	0.008	0.020
02:00	0.009	0.008	0.009	0.007	0.017
03:00	0.007	0.008	0.010	0.004	0.015
04:00	0.007	0.014	0.010	0.008	0.017
05:00	0.018	0.044	0.021	0.014	0.028
06:00	0.034	0.062	0.041	0.030	0.048
07:00	0.050	0.060	0.061	0.066	0.069
08:00	0.058	0.060	0.064	0.074	0.069
09:00	0.051	0.045	0.063	0.081	0.063
10:00	0.052	0.039	0.064	0.087	0.061
11:00	0.055	0.041	0.064	0.074	0.056
12:00	0.060	0.044	0.061	0.078	0.057
13:00	0.059	0.051	0.060	0.072	0.057
14:00	0.058	0.041	0.061	0.068	0.057
15:00	0.058	0.041	0.061	0.060	0.050
16:00	0.061	0.048	0.061	0.064	0.049
17:00	0.069	0.061	0.060	0.057	0.050
18:00	0.066	0.065	0.055	0.044	0.046
19:00	0.057	0.059	0.045	0.027	0.038
20:00	0.047	0.046	0.036	0.024	0.033
21:00	0.040	0.039	0.029	0.019	0.030
22:00	0.033	0.041	0.023	0.014	0.028
23:00	0.022	0.036	0.017	0.008	0.023

Table B-10: Hourly Traffic Distribution Factors for a Heavy Haul Corridor With Uniform Traffic

Hour	Vehicle Type				
	Cars	**Buses**	**Light trucks**	**Mid-size trucks**	**Heavy haul trucks**
00:00	0.010	0.005	0.015	0.026	0.026
01:00	0.007	0.037	0.012	0.018	0.021
02:00	0.006	0.139	0.012	0.013	0.015
03:00	0.006	0.130	0.014	0.012	0.013
04:00	0.009	0.014	0.020	0.025	0.018
05:00	0.015	0.010	0.023	0.032	0.023
06:00	0.025	0.052	0.030	0.033	0.026
07:00	0.037	0.006	0.038	0.039	0.030
08:00	0.050	0.007	0.043	0.036	0.030
09:00	0.060	0.039	0.049	0.044	0.033
10:00	0.065	0.035	0.053	0.047	0.040
11:00	0.068	0.005	0.057	0.049	0.044
12:00	0.069	0.036	0.059	0.059	0.043
13:00	0.064	0.004	0.060	0.054	0.041
14:00	0.065	0.033	0.064	0.054	0.045
15:00	0.069	0.052	0.065	0.051	0.049
16:00	0.071	0.094	0.070	0.053	0.054
17:00	0.070	0.016	0.068	0.058	0.059
18:00	0.066	0.070	0.066	0.056	0.064
19:00	0.055	0.019	0.057	0.053	0.069
20:00	0.044	0.066	0.046	0.049	0.069
21:00	0.033	0.046	0.034	0.052	0.078
22:00	0.022	0.057	0.026	0.049	0.068
23:00	0.015	0.029	0.019	0.034	0.042

Table B-11: Hourly Traffic Distribution Factors for a Heavy Haul Corridor With Seasonal Peak Traffic

Hour	Vehicle Type				
	Cars	**Buses**	**Light trucks**	**Mid-size trucks**	**Heavy haul trucks**
00:00	0.012	0.005	0.015	0.028	0.033
01:00	0.009	0.042	0.013	0.017	0.022
02:00	0.007	0.104	0.011	0.010	0.016
03:00	0.006	0.059	0.012	0.008	0.011
04:00	0.008	0.074	0.012	0.007	0.011
05:00	0.012	0.011	0.016	0.015	0.020
06:00	0.021	0.017	0.025	0.029	0.027
07:00	0.033	0.013	0.032	0.031	0.029
08:00	0.040	0.029	0.038	0.031	0.030
09:00	0.048	0.031	0.042	0.033	0.031
10:00	0.051	0.025	0.043	0.032	0.032
11:00	0.050	0.017	0.044	0.034	0.034
12:00	0.045	0.015	0.043	0.047	0.035
13:00	0.044	0.030	0.046	0.054	0.037
14:00	0.048	0.019	0.053	0.059	0.042
15:00	0.054	0.050	0.054	0.050	0.039
16:00	0.058	0.059	0.056	0.050	0.040
17:00	0.061	0.039	0.056	0.050	0.041
18:00	0.060	0.034	0.053	0.043	0.043
19:00	0.055	0.027	0.050	0.046	0.055
20:00	0.044	0.023	0.044	0.044	0.060
21:00	0.032	0.045	0.032	0.047	0.057
22:00	0.021	0.058	0.025	0.039	0.053
23:00	0.015	0.007	0.017	0.029	0.036

Table B-12: Hourly Traffic Distribution Factors for a Mixed Traffic Arterial Road With Passenger Car Peak

Hour	**Vehicle Type**				
	Cars	**Buses**	**Light trucks**	**Mid-size trucks**	**Heavy haul trucks**
00:00	0.012	0.030	0.015	0.030	0.030
01:00	0.008	0.032	0.011	0.016	0.019
02:00	0.007	0.055	0.011	0.010	0.012
03:00	0.006	0.088	0.014	0.009	0.009
04:00	0.007	0.026	0.015	0.010	0.013
05:00	0.012	0.017	0.022	0.018	0.025
06:00	0.024	0.040	0.031	0.026	0.037
07:00	0.038	0.010	0.043	0.040	0.043
08:00	0.050	0.051	0.051	0.043	0.042
09:00	0.055	0.035	0.049	0.042	0.044
10:00	0.059	0.038	0.053	0.055	0.047
11:00	0.060	0.019	0.056	0.061	0.048
12:00	0.056	0.039	0.059	0.068	0.051
13:00	0.053	0.031	0.058	0.072	0.055
14:00	0.058	0.022	0.059	0.066	0.048
15:00	0.067	0.041	0.064	0.060	0.046
16:00	0.073	0.056	0.069	0.054	0.047
17:00	0.079	0.093	0.068	0.055	0.050
18:00	0.075	0.033	0.064	0.055	0.051
19:00	0.067	0.055	0.058	0.047	0.057
20:00	0.053	0.069	0.048	0.044	0.062
21:00	0.040	0.072	0.036	0.049	0.064
22:00	0.025	0.024	0.026	0.041	0.058
23:00	0.017	0.025	0.019	0.032	0.042

ANOVA Results on the Traffic Daily Distribution Factors

Table B-13: ANOVA Summary for the Suburban Arterial Highway

	(Month/day) Interaction		Month effect		Day effect	
Vehicle Cat	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**
Cars	0.5651	Not sign.	0.989	Not sign.	**0.0011**	**Significant**
Buses	0.5565	Not sign.	0.9340	Not sign.	**0.0011**	**Significant**
Light trucks	0.8338	Not sign.	0.9662	Not sign.	**0.0162**	**Significant**
Mid-size trucks	0.6191	Not sign.	0.9694	Not sign.	**0.0053**	**Significant**
Heavy haul trucks	0.7927	Not sign.	0.9915	Not sign.	**0.0063**	**Significant**

Table B-14: ANOVA Summary for the Heavy Haul Corridor With Uniform Traffic

	(Month/day) Interaction		Month effect		Day effect	
Vehicle Cat	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**
Cars	0.4726	Not sign.	0.8178	Not sign.	0.1683	Not sign.
Buses	0.3803	Not sign.	0.8504	Not sign.	0.1935	Not sign.
Light trucks	0.5328	Not sign.	0.8743	Not sign.	0.0663	Not sign.
Mid-size trucks	0.5195	Not sign.	0.9570	Not sign.	0.0879	Not sign.
Heavy haul trucks	0.5401	Not sign.	0.9469	Not sign.	0.2033	Not sign.

Table B-15: ANOVA Summary for the Heavy Haul Corridor With Seasonal Peak Traffic

	(Month/day) Interaction		Month effect		Day effect	
Vehicle Cat	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**
Cars	0.1009	Not sign.	0.9660	Not sign.	0.3191	Not sign.
Buses	**0.0393**	**Significant**	0.8625	Not sign.	**0.0001**	**Significant**
Light trucks	0.2502	Not sign.	0.6967	Not sign.	0.0013	Not sign.
Mid-size trucks	0.5277	Not sign.	0.7619	Not sign.	0.0054	Not sign.
Heavy haul trucks	0.6892	Not sign.	0.7656	Not sign.	0.1295	Not sign.

Table B-16: ANOVA Summary for the Mixed Traffic Arterial Road With Passenger Car Peak

	(Month/day) Interaction		Month effect		Day effect	
Vehicle Cat	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**
Cars	0.8500	Not sign.	0.3893	Not sign.	0.1828	Not sign.
Buses	0.1669	Not sign.	0.2674	Not sign.	0.1488	Not sign.
Light trucks	0.7357	Not sign.	0.4912	Not sign.	**0.0011**	**Significant**
Mid-size trucks	0.7073	Not sign.	0.6996	Not sign.	0.1755	Not sign.
Heavy haul trucks	0.7956	Not sign.	0.6725	Not sign.	0.4680	Not sign.

ANOVA Results on the Traffic Hourly Distribution Factors

Tables B-17 to B-20 below summarize the results of the ANOVA tests conducted on the hourly distribution factors for the four functional categories presented and discussed in Chapter 2. These are provided herein for the four road functional classes that have been analyzed.

Table B-17: ANOVA Summary for the Suburban Arterial Highway Hourly Distribution Factors

Vehicle Cat	**D/H/M interact.**		**D/M interact**		**H/M interact**		**D/H interact**		**hour**		**day**		**month**	
	P (F > fstat)	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**
Cars	0.9823	Not S.	0.9913	Not S.	0.7101	Not S.	0.5022	Not S.	**0.000**	**Sign.**	0.9945	Not S.	0.9829	Not S.
Buses	0.3873	Not S.	0.9961	Not S.	0.6091	Not S.	0.7544	Not S.	**0.000**	**Sign.**	0.9895	Not S.	0.9858	Not S.
Light trucks	0.7811	Not S.	0.9868	Not S.	0.9123	Not S.	0.8782	Not S.	**0.000**	**Sign.**	0.9892	Not S.	0.9855	Not S.
Mid-size trucks	0.2372	Not S.	0.9952	Not S.	0.3261	Not S.	0.1480	Not S.	**0.000**	**Sign.**	0.9966	Not S.	0.9975	Not S.
Heavy haul trucks	0.2455	Not S.	0.9940	Not S.	0.8429	Not S.	**0.0292**	**Sign.**	**0.000**	**Sign.**	0.9982	Not S.	0.9799	Not S.

Table B-18: ANOVA Summary for the Heavy Haul Corridor With Uniform Traffic Hourly Distribution Factors

Vehicle Cat	**D/H/M interact.**		**D/M interact**		**H/M interact**		**D/H interact**		**hour**		**day**		**month**	
	P (F > fstat)	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**
Cars	0.9927	Not S.	0.9902	Not S.	0.2974	Not S.	0.9912	Not S.	**0.000**	**Sign.**	0.9835	Not S.	0.9980	Not S.
Buses	0.9661	Not S.	0.9970	Not S.	0.6640	Not S.	0.8871	Not S.	0.447	Not S.	0.9937	Not S.	0.9967	Not S.
Light trucks	0.9649	Not S.	0.9839	Not S.	0.6165	Not S.	0.9200	Not S.	**0.000**	**Sign.**	0.9923	Not S.	0.9989	Not S.
Mid-size trucks	0.9674	Not S.	0.9688	Not S.	0.3508	Not S.	0.9107	Not S.	**0.000**	**Sign.**	0.9753	Not S.	0.9554	Not S.
Heavy haul trucks	0.8973	Not S.	0.9876	Not S.	**0.0007**	**Sign.**	0.7995	Not S.	**0.000**	**Sign.**	0.9928	Not S.	0.9896	Not S.

Table B-19: ANOVA Summary for the Heavy Haul Corridor With Seasonal Peak Traffic Hourly Distribution Factors

Vehicle Cat	**D/H/M interact.**		**D/M interact**		**H/M interact**		**D/H interact**		**hour**		**day**		**month**	
	P (F > fstat)	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**
Cars	0.4511	Not S.	0.9866	Not S.	0.7086	Not S.	0.3898	Not S.	**0.000**	**Sign.**	0.9890	Not S.	0.9911	Not S.
Buses	0.9130	Not S.	0.9998	Not S.	0.7092	Not S.	0.8049	Not S.	**0.0006**	**Sign.**	0.9925	Not S.	0.9902	Not S.
Light trucks	0.3557	Not S.	0.9837	Not S.	0.4583	Not S.	0.9394	Not S.	**0.000**	**Sign.**	0.9780	Not S.	0.9973	Not S.
Mid-size trucks	**0.008**	**Sign.**	0.9873	Not S.	**0.0111**	**Sign.**	**0.0001**	**Sign.**	**0.000**	**Sign.**	0.9900	Not S.	0.9952	Not S.
Heavy haul trucks	0.0537	Not S.	0.9760	Not S.	**0.0043**	**Sign.**	**0.0043**	**Sign.**	**0.000**	**Sign.**	0.9643	Not S.	0.9988	Not S.

Table B-20: ANOVA Summary for the Mixed Traffic Arterial Road With Passenger Car Peak Hourly Distribution Factors

Vehicle Cat	**D/H/M interact.**		**D/M interact**		**H/M interact**		**D/H interact**		**hour**		**day**		**month**	
	P (F > fstat)	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**	**P (F > fstat)**	**Result**
Cars	0.8817	Not S.	0.9872	Not S.	0.9453	Not S.	0.1589	Not S.	**0.000**	**Sign.**	0.9855	Not S.	0.9970	Not S.
Buses	0.6757	Not S.	0.9996	Not S.	0.6831	Not S.	0.4483	Not S.	**0.0005**	**Sign.**	0.9971	Not S.	0.9970	Not S.
Light trucks	0.9633	Not S.	0.9957	Not S.	0.6629	Not S.	**0.0199**	**Sign.**	**0.000**	**Sign.**	0.9816	Not S.	0.9894	Not S.
Mid-size trucks	0.9241	Not S.	0.9899	Not S.	0.2074	Not S.	**0.000**	**Sign.**	**0.000**	**Sign.**	0.9779	Not S.	0.9824	Not S.
Heavy haul trucks	0.4012	Not S.	0.9793	Not S.	0.5053	Not S.	**0.000**	**Sign.**	**0.000**	**Sign.**	0.9787	Not S.	0.9908	Not S.

APPENDIX C – STRESS AND STRAIN COMPUTATION POINTS

This Appendix is a supplement to the description of Product-One's stress and strain calculation routine. The tables that follow complement the diagrams with the planar positions of the stress and strain calculation points for each axle type (figures 19 to 24, Chapter 3), and provide the formulas to the radial distance and angle between the center of each application of load and each stress and strain evaluation point. Separate tables are provided for each axle type handled by Product-One.

For all of the tables below, *a* refers to the radius of the contact area between the pavement surface and the axle's wheels; Ts is the spacing between wheels in dual-wheel arrangements, and Sy is the center-to-center axle distance for tandem and tridem axles. The values of *a* vary for each axle type (since these depend on the load level and the wheel's tire pressure), Product-One's default value for Ts is 0.68m (as retrieved from commercial brochures of heavy-duty vehicles manufacturers), and the default value for Sy is 1.20m – maximum legal separation for Uruguayan heavy vehicles, as stated by MTOP (2017).

Meanwhile, the angle values provided hereafter correspond to the angle of the vector starting at each evaluation point toward the center of the loaded wheels. The zero-radian angle corresponds to such a vector being co-linear with the Cartesian grid x axis – the same convention that Huang (2004) adopted.

MLE stress and strain calculation points for single axles

Table C-1: Radial Distance and Angle for the Stress Calculation Locations. Single-Wheel Single Axles.

Point	Radius	Angle [rad]
0	0	0
1	$\frac{a}{2}$	0
2	a	0
3	$a+0.10\,m$	0

Note: Refer to figure 19 for details on the location of the stress and strain evaluation points.

Table C-2: Radial Distance and Angle for the Stress Calculation Locations. Dual-Wheel Single Axles.

	Wheel 1		Wheel 2	
Point	**Radius**	**Angle [rad]**	**Radius**	**Angle [rad]**
0	$\frac{Ts}{2}$	0	$\frac{Ts}{2}$	π
1	$\frac{3Ts}{4}-\frac{a}{2}$	0	$\frac{Ts}{4}+\frac{a}{2}$	π
2	$Ts-a$	0	a	π
3	Ts	0	0	0
4	$Ts+a$	0	a	0
5	$Ts+a+0.10m$	0	$a+0.10m$	0

Note: Refer to figure 20 for details on the location of the stress and strain evaluation points.

MLE stress and strain calculation points for tandem and tridem axles

Table C-33: Radial Distance and Angle for the Stress Calculation Locations. Single-Wheel Tandem Axles

	Wheel 1		Wheel 2	
Point	**Radius**	**Angle**	**Radius**	**Angle**
0	$\frac{Sy}{2}$	π/2	$\frac{Sy}{2}$	-π/2
1	$\sqrt{a^2+\left(\frac{Sy}{2}\right)^2}$	$\tan^{-1}\left(\frac{Sy}{2a}\right)$	$\sqrt{a^2+\left(\frac{Sy}{2}\right)^2}$	$\tan^{-1}\left(\frac{-Sy}{2a}\right)$
2	$\sqrt{(a+0.10m)^2+\left(\frac{Sy}{2}\right)^2}$	$\tan^{-1}\left(\frac{Sy}{2(a+0.10m)}\right)$	$\sqrt{(a+0.10m)^2+\left(\frac{Sy}{2}\right)^2}$	$\tan^{-1}\left(\frac{-Sy}{2(a+0.10m)}\right)$
3	Sy	π/2	0	0
4	$\sqrt{a^2+Sy^2}$	$\tan^{-1}\left(\frac{Sy}{a}\right)$	a	0
5	$\sqrt{(a+0.10m)^2+Sy^2}$	$\tan^{-1}\left(\frac{Sy}{(a+0.10m)}\right)$	$a+0.10m$	0

Note: Refer to figure 21 for details on the location of the stress and strain evaluation points.

Table C-4: Radial Distance and Angle for the Stress Calculation Locations. Non-Homogeneous Tandem Axles

	Wheel 1		**Wheel 2**		**Wheel 3**	
Point	**Radius**	**Angle**	**Radius**	**Angle**	**Radius**	**Angle**
0	0	0	Ts	π	Sy	-π/2
1	a	0	$Ts-a$	π	$\sqrt{a^2+Sy^2}$	$\tan^{-1}\left(\frac{-Sy}{a}\right)$
2	$\frac{Ts}{4}+\frac{a}{2}$	0	$\frac{3Ts}{4}-\frac{a}{2}$	π	$\sqrt{Sy^2+\left(\frac{Ts}{4}+\frac{a}{2}\right)^2}$	$\tan^{-1}\left(\frac{-2Sy}{0.5Ts+a}\right)$
3	$\frac{Ts}{2}$	0	$\frac{Ts}{2}$	π	$\sqrt{Sy^2+\left(\frac{Ts}{2}\right)^2}$	$\tan^{-1}\left(\frac{-2Sy}{Ts}\right)$
4	$\frac{3Ts}{4}-\frac{a}{2}$	0	$\frac{Ts}{4}+\frac{a}{2}$	π	$\sqrt{Sy^2+\left(\frac{3Ts}{4}-\frac{a}{2}\right)^2}$	$\tan^{-1}\left(\frac{-2Sy}{1.5Ts-a}\right)$
5	$Ts-a$	0	a	π	$\sqrt{Sy^2+(Ts-a)^2}$	$\tan^{-1}\left(\frac{-Sy}{Ts-a}\right)$
6	Ts	0	0	0	$\sqrt{Sy^2+Ts^2}$	$\tan^{-1}\left(\frac{-Sy}{Ts}\right)$
7	$Ts+a$	0	a	0	$\sqrt{Sy^2+(Ts+a)^2}$	$\tan^{-1}\left(\frac{-Sy}{Ts+a}\right)$
8	$Ts+a+0.10\,m$	0	$a+0.10\,m$	0	$\sqrt{Sy^2+(Ts+a+0.10)^2}$	$\tan^{-1}\left(\frac{-Sy}{Ts+a+0.10m}\right)$
9	$\frac{Sy}{2}$	π/2	$\sqrt{Ts^2+\left(\frac{Sy}{2}\right)^2}$	$\pi-\tan^{-1}\left(\frac{Sy}{2\,Ts}\right)$	$\frac{Sy}{2}$	- π/2
10	$\sqrt{a^2+\left(\frac{Sy}{2}\right)^2}$	$\tan^{-1}\left(\frac{Sy}{2a}\right)$	$\sqrt{(Ts-a)^2+\left(\frac{Sy}{2}\right)^2}$	$\pi-\tan^{-1}\left(\frac{Sy}{2(Ts-a)}\right)$	$\sqrt{a^2+\left(\frac{Sy}{2}\right)^2}$	$\tan^{-1}\left(\frac{-Sy}{2a}\right)$
11	$\sqrt{\left(\frac{Ts}{2}\right)^2+\left(\frac{Sy}{2}\right)^2}$	$\tan^{-1}\left(\frac{Sy}{Ts}\right)$	$\sqrt{\left(\frac{Ts}{2}\right)^2+\left(\frac{Sy}{2}\right)^2}$	$\pi-\tan^{-1}\left(\frac{Sy}{Ts}\right)$	$\sqrt{\left(\frac{Ts}{2}\right)^2+\left(\frac{Sy}{2}\right)^2}$	$\tan^{-1}\left(\frac{-Sy}{Ts}\right)$
12	Sy	π/2	$\sqrt{Ts^2+Sy^2}$	$\tan^{-1}\left(\frac{-Sy}{Ts}\right)$	0	0
13	$\sqrt{a^2+Sy^2}$	$\tan^{-1}\left(\frac{Sy}{a}\right)$	$\sqrt{(Ts-a)^2+Sy^2}$	$\tan^{-1}\left(\frac{-Sy}{Ts-a}\right)$	a	0
14	$\sqrt{(a+0.10\,m)^2+Sy^2}$	$\tan^{-1}\left(\frac{Sy}{a+0.10\,m}\right)$	$\sqrt{(Ts-a-0.10)^2+Sy^2}$	$\tan^{-1}\left(\frac{-Sy}{Ts-a-0.10}\right)$	$a+0.10\,m$	0

Note: Refer to figure 22 for details on the location of the stress and strain evaluation points.

Table C-5:Radial Distance and Angle for the Stress Calculation Locations. Dual-Wheel Tandem Axles

	Wheel 1		**Wheel 2**	
Point	**Radius**	**Angle**	**Radius**	**Angle**
0	$\sqrt{\left(\frac{Ts}{2}\right)^2+\left(\frac{Sy}{2}\right)^2}$	$\tan^{-1}\left(\frac{Sy}{Ts}\right)$	$\sqrt{\left(\frac{Ts}{2}\right)^2+\left(\frac{Sy}{2}\right)^2}$	$\pi-\tan^{-1}\left(\frac{Sy}{Ts}\right)$
1	$\sqrt{\left(\frac{Sy}{2}\right)^2+\left(\frac{3\,Ts}{4}-\frac{a}{2}\right)^2}$	$\tan^{-1}\left(\frac{Sy}{1.5\,Ts-a}\right)$	$\sqrt{\left(\frac{Sy}{2}\right)^2+\left(\frac{Ts}{4}+\frac{a}{2}\right)^2}$	$\pi-\tan^{-1}\left(\frac{Sy}{0.5\,Ts+a}\right)$
2	$\sqrt{\left(\frac{Sy}{2}\right)^2+(Ts-a)^2}$	$\tan^{-1}\left(\frac{Sy}{2(Ts-a)}\right)$	$\sqrt{\left(\frac{Sy}{2}\right)^2+a^2}$	$\pi-\tan^{-1}\left(\frac{Sy}{2\,a}\right)$
3	$\sqrt{\left(\frac{Sy}{2}\right)^2+(Ts)^2}$	$\tan^{-1}\left(\frac{Sy}{2\,Ts}\right)$	$\frac{Sy}{2}$	π/2
4	$\sqrt{\left(\frac{Sy}{2}\right)^2+(Ts+a)^2}$	$\tan^{-1}\left(\frac{Sy}{2(Ts+a)}\right)$	$\sqrt{\left(\frac{Sy}{2}\right)^2+a^2}$	$\tan^{-1}\left(\frac{Sy}{2\,a}\right)$
5	$\sqrt{\left(\frac{Sy}{2}\right)^2+(Ts+a+0.10\,m)^2}$	$\tan^{-1}\left(\frac{Sy}{2(Ts+a+0.10\,m)}\right)$	$\sqrt{\left(\frac{Sy}{2}\right)^2+(a+0.10\,m)^2}$	$\tan^{-1}\left(\frac{Sy}{2(a+0.10\,m)}\right)$
6	$\sqrt{Sy^2+\left(\frac{Ts}{2}\right)^2}$	$\tan^{-1}\left(\frac{2\,Sy}{Ts}\right)$	$\sqrt{Sy^2+\left(\frac{Ts}{2}\right)^2}$	$\pi-\tan^{-1}\left(\frac{2\,Sy}{Ts}\right)$
7	$\sqrt{Sy^2+\left(\frac{3\,Ts}{4}-\frac{a}{2}\right)^2}$	$\tan^{-1}\left(\frac{Sy}{0.75\,Ts-0.5\,a}\right)$	$\sqrt{Sy^2+\left(\frac{Ts}{4}+\frac{a}{2}\right)^2}$	$\pi-\tan^{-1}\left(\frac{-Sy}{0.25\,Ts+0.5\,a}\right)$
8	$\sqrt{Sy^2+(Ts-a)^2}$	$\tan^{-1}\left(\frac{Sy}{Ts-a}\right)$	$\sqrt{Sy^2+a^2}$	$\pi-\tan^{-1}\left(\frac{Sy}{a}\right)$
9	$\sqrt{Sy^2+Ts^2}$	$\tan^{-1}\left(\frac{Sy}{Ts}\right)$	Sy	π/2
10	$\sqrt{Sy^2+(Ts+a)^2}$	$\tan^{-1}\left(\frac{Sy}{(Ts+a)}\right)$	$\sqrt{(Sy)^2+a^2}$	$\tan^{-1}\left(\frac{Sy}{a}\right)$
11	$\sqrt{Sy^2+(Ts+a+0.10\,m)^2}$	$\tan^{-1}\left(\frac{Sy}{(Ts+a+0.10\,m)}\right)$	$\sqrt{Sy^2+(a+0.10\,m)^2}$	$\tan^{-1}\left(\frac{Sy}{(a+0.10\,m)}\right)$

Table C-6:Radial Distance and Angle for the Stress Calculation Locations. Dual-Wheel Tandem Axles [continued]

	Wheel 3		**Wheel 4**	
Point	**Radius**	**Angle**	**Radius**	**Angle**
0	$\sqrt{\left(\frac{Ts}{2}\right)^2+\left(\frac{Sy}{2}\right)^2}$	$\tan^{-1}\left(\frac{-Sy}{Ts}\right)$	$\sqrt{\left(\frac{Ts}{2}\right)^2+\left(\frac{Sy}{2}\right)^2}$	$\pi+\tan^{-1}\left(\frac{Sy}{Ts}\right)$
1	$\sqrt{\left(\frac{Sy}{2}\right)^2+\left(\frac{3Ts}{4}-\frac{a}{2}\right)^2}$	$\tan^{-1}\left(\frac{-Sy}{1.5Ts-a}\right)$	$\sqrt{\left(\frac{Sy}{2}\right)^2+\left(\frac{Ts}{4}+\frac{a}{2}\right)^2}$	$\pi+\tan^{-1}\left(\frac{Sy}{0.5Ts+a}\right)$
2	$\sqrt{\left(\frac{Sy}{2}\right)^2+(Ts-a)^2}$	$\tan^{-1}\left(\frac{-Sy}{2(Ts-a)}\right)$	$\sqrt{\left(\frac{Sy}{2}\right)^2+a^2}$	$\pi+\tan^{-1}\left(\frac{Sy}{2a}\right)$
3	$\sqrt{\left(\frac{Sy}{2}\right)^2+(Ts)^2}$	$\tan^{-1}\left(\frac{-Sy}{2Ts}\right)$	$\frac{Sy}{2}$	-π/2
4	$\sqrt{\left(\frac{Sy}{2}\right)^2+(Ts+a)^2}$	$\tan^{-1}\left(\frac{-Sy}{2(Ts+a)}\right)$	$\sqrt{\left(\frac{Sy}{2}\right)^2+a^2}$	$\tan^{-1}\left(\frac{-Sy}{2a}\right)$
5	$\sqrt{\left(\frac{Sy}{2}\right)^2+(Ts+a+0.10\,m)^2}$	$\tan^{-1}\left(\frac{-Sy}{2(Ts+a+0.10m)}\right)$	$\sqrt{\left(\frac{Sy}{2}\right)^2+(a+0.10m)^2}$	$\tan^{-1}\left(\frac{-Sy}{2(a+0.10m)}\right)$
6	$\frac{Ts}{2}$	0	$\frac{Ts}{2}$	π
7	$\frac{3Ts}{4}-\frac{a}{2}$	0	$\frac{Ts}{4}+\frac{a}{2}$	π
8	$Ts-a$	0	a	π
9	Ts	0	0	0
10	$Ts+a$	0	a	0
11	$Ts+a+0.10m$	0	$a+0.10m$	0

Note: Refer to figure 23 for details on the location of the stress and strain evaluation points.

Table C-7: Radial Distance and Angle for the Stress Calculation Locations. Tridem Axles

	Wheel 1		Wheel 2	
Point	**Radius**	**Angle**	**Radius**	**Angle**
0	$\sqrt{\left(\frac{Ts}{2}\right)^2+(Sy)^2}$	$\tan^{-1}\left(\frac{2\,Sy}{Ts}\right)$	$\sqrt{\left(\frac{Ts}{2}\right)^2+(Sy)^2}$	$\pi-\tan^{-1}\left(\frac{2\,Sy}{Ts}\right)$
1	$\sqrt{(Sy)^2+\left(\frac{3\,Ts}{4}-\frac{a}{2}\right)^2}$	$\tan^{-1}\left(\frac{2\,Sy}{1.5\,Ts-a}\right)$	$\sqrt{(Sy)^2+\left(\frac{Ts}{4}+\frac{a}{2}\right)^2}$	$\pi-\tan^{-1}\left(\frac{2\,Sy}{0.5\,Ts+a}\right)$
2	$\sqrt{(Sy)^2+(Ts-a)^2}$	$\tan^{-1}\left(\frac{Sy}{(Ts-a)}\right)$	$\sqrt{(Sy)^2+a^2}$	$\pi-\tan^{-1}\left(\frac{Sy}{a}\right)$
3	$\sqrt{(Sy)^2+(Ts)^2}$	$\tan^{-1}\left(\frac{Sy}{Ts}\right)$	Sy	π/2
4	$\sqrt{(Sy)^2+(Ts+a)^2}$	$\tan^{-1}\left(\frac{Sy}{(Ts+a)}\right)$	$\sqrt{(Sy)^2+a^2}$	$\tan^{-1}\left(\frac{Sy}{a}\right)$
5	$\sqrt{(Sy)^2+(Ts+a+0.10\,m)^2}$	$\tan^{-1}\left(\frac{Sy}{(Ts+a+0.10\,m)}\right)$	$\sqrt{(Sy)^2+(a+0.10\,m)^2}$	$\tan^{-1}\left(\frac{Sy}{(a+0.10\,m)}\right)$
6	$\sqrt{\left(\frac{Ts}{2}\right)^2+\left(\frac{3\,Sy}{2}\right)^2}$	$\tan^{-1}\left(\frac{3\,Sy}{Ts}\right)$	$\sqrt{\left(\frac{Ts}{2}\right)^2+\left(\frac{3\,Sy}{2}\right)^2}$	$\pi-\tan^{-1}\left(\frac{3\,Sy}{Ts}\right)$
7	$\sqrt{\left(\frac{3\,Sy}{2}\right)^2+\left(\frac{3\,Ts}{4}-\frac{a}{2}\right)^2}$	$\tan^{-1}\left(\frac{3\,Sy}{1.5\,Ts-a}\right)$	$\sqrt{\left(\frac{3\,Sy}{2}\right)^2+\left(\frac{Ts}{4}+\frac{a}{2}\right)^2}$	$\pi-\tan^{-1}\left(\frac{3\,Sy}{0.5\,Ts+a}\right)$
8	$\sqrt{\left(\frac{3\,Sy}{2}\right)^2+(Ts-a)^2}$	$\tan^{-1}\left(\frac{3\,Sy}{(2(Ts-a))}\right)$	$\sqrt{\left(\frac{3\,Sy}{2}\right)^2+a^2}$	$\pi-\tan^{-1}\left(\frac{3\,Sy}{2\,a}\right)$
9	$\sqrt{\left(\frac{3\,Sy}{2}\right)^2+(Ts)^2}$	$\tan^{-1}\left(\frac{3\,Sy}{2\,Ts}\right)$	$\frac{3\,Sy}{2}$	π/2
10	$\sqrt{\left(\frac{3\,Sy}{2}\right)^2+(Ts+a)^2}$	$\tan^{-1}\left(\frac{3\,Sy}{(2(Ts+a))}\right)$	$\sqrt{\left(\frac{3\,Sy}{2}\right)^2+a^2}$	$\tan^{-1}\left(\frac{3\,Sy}{2\,a}\right)$
11	$\sqrt{\left(\frac{3\,Sy}{2}\right)^2+(Ts+a+0.10\,m)^2}$	$\tan^{-1}\left(\frac{3\,Sy}{2\,(Ts+a+0.10\,m)}\right)$	$\sqrt{\left(\frac{3\,Sy}{2}\right)^2+(a+0.10\,m)^2}$	$\tan^{-1}\left(\frac{3\,Sy}{2\,(a+0.10\,m)}\right)$
12	$\sqrt{\left(\frac{Ts}{2}\right)^2+(2\,Sy)^2}$	$\tan^{-1}\left(\frac{4\,Sy}{Ts}\right)$	$\sqrt{\left(\frac{Ts}{2}\right)^2+(2\,Sy)^2}$	$\pi-\tan^{-1}\left(\frac{4\,Sy}{Ts}\right)$
13	$\sqrt{(2\,Sy)^2+\left(\frac{3\,Ts}{4}-\frac{a}{2}\right)^2}$	$\tan^{-1}\left(\frac{2\,Sy}{0.75\,Ts-0.5\,a}\right)$	$\sqrt{(2\,Sy)^2+\left(\frac{Ts}{4}+\frac{a}{2}\right)^2}$	$\pi-\tan^{-1}\left(\frac{4\,Sy}{0.5\,Ts+a}\right)$
14	$\sqrt{(2\,Sy)^2+(Ts-a)^2}$	$\tan^{-1}\left(\frac{2\,Sy}{(Ts-a)}\right)$	$\sqrt{(2\,Sy)^2+a^2}$	$\pi-\tan^{-1}\left(\frac{2\,Sy}{a}\right)$
15	$\sqrt{(2\,Sy)^2+(Ts)^2}$	$\tan^{-1}\left(\frac{2\,Sy}{Ts}\right)$	$2\,Sy$	π/2
16	$\sqrt{(2\,Sy)^2+(Ts+a)^2}$	$\tan^{-1}\left(\frac{2\,Sy}{(Ts+a)}\right)$	$\sqrt{(2\,Sy)^2+a^2}$	$\tan^{-1}\left(\frac{2\,Sy}{a}\right)$
17	$\sqrt{(2\,Sy)^2+(Ts+a+0.10\,m)^2}$	$\tan^{-1}\left(\frac{2\,Sy}{(Ts+a+0.10\,m)}\right)$	$\sqrt{(2\,Sy)^2+(a+0.10\,m)^2}$	$\tan^{-1}\left(\frac{2\,Sy}{(a+0.10\,m)}\right)$

Table C-8: Radial Distance and Angle for the Stress Calculation Locations. Tridem Axles [continued]

	Wheel 3		**Wheel 4**	
Point	**Radius**	**Angle**	**Radius**	**Angle**
0	$\frac{Ts}{2}$	0	$\frac{Ts}{2}$	π
1	$\frac{3Ts}{4}-\frac{a}{2}$	0	$\frac{Ts}{4}+\frac{a}{2}$	π
2	$Ts-a$	0	a	π
3	Ts	0	0	0
4	$Ts+a$	0	a	0
5	$Ts+a+0.10\,m$	0	$a+0.10\,m$	0
6	$\sqrt{\left(\frac{Ts}{2}\right)^2+\left(\frac{Sy}{2}\right)^2}$	$\tan^{-1}\left(\frac{Sy}{Ts}\right)$	$\sqrt{\left(\frac{Ts}{2}\right)^2+\left(\frac{Sy}{2}\right)^2}$	$\pi-\tan^{-1}\left(\frac{Sy}{Ts}\right)$
7	$\sqrt{\left(\frac{Sy}{2}\right)^2+\left(\frac{3Ts}{4}-\frac{a}{2}\right)^2}$	$\tan^{-1}\left(\frac{Sy}{1.5\,Ts-a}\right)$	$\sqrt{\left(\frac{Sy}{2}\right)^2+\left(\frac{Ts}{4}+\frac{a}{2}\right)^2}$	$\pi-\tan^{-1}\left(\frac{Sy}{0.5\,Ts+a}\right)$
8	$\sqrt{\left(\frac{Sy}{2}\right)^2+(Ts-a)^2}$	$\tan^{-1}\left(\frac{Sy}{2(Ts-a)}\right)$	$\sqrt{\left(\frac{Sy}{2}\right)^2+a^2}$	$\pi-\tan^{-1}\left(\frac{Sy}{2\,a}\right)$
9	$\sqrt{\left(\frac{Sy}{2}\right)^2+(Ts)^2}$	$\tan^{-1}\left(\frac{Sy}{2\,Ts}\right)$	$\frac{Sy}{2}$	$\pi/2$
10	$\sqrt{\left(\frac{Sy}{2}\right)^2+(Ts+a)^2}$	$\tan^{-1}\left(\frac{Sy}{2(Ts+a)}\right)$	$\sqrt{\left(\frac{Sy}{2}\right)^2+a^2}$	$\tan^{-1}\left(\frac{Sy}{2\,a}\right)$
11	$\sqrt{\left(\frac{Sy}{2}\right)^2+(Ts+a+0.10\,m)^2}$	$\tan^{-1}\left(\frac{Sy}{2(Ts+a+0.10\,m)}\right)$	$\sqrt{\left(\frac{Sy}{2}\right)^2+(a+0.10\,m)^2}$	$\tan^{1}\left(\frac{Sy}{2(a+0.10\,m)}\right)$
12	$\sqrt{\left(\frac{Ts}{2}\right)^2+(Sy)^2}$	$\tan^{-1}\left(\frac{2\,Sy}{Ts}\right)$	$\sqrt{\left(\frac{Ts}{2}\right)^2+(Sy)^2}$	$\pi-\tan^{-1}\left(\frac{2\,Sy}{Ts}\right)$
13	$\sqrt{(Sy)^2+\left(\frac{3\,Ts}{4}-\frac{a}{2}\right)^2}$	$\tan^{-1}\left(\frac{Sy}{0.75\,Ts-0.5\,a}\right)$	$\sqrt{(Sy)^2+\left(\frac{Ts}{4}+\frac{a}{2}\right)^2}$	$\pi-\tan^{-1}\left(\frac{Sy}{0.5\,Ts+a}\right)$
14	$\sqrt{(Sy)^2+(Ts-a)^2}$	$\tan^{-1}\left(\frac{Sy}{(Ts-a)}\right)$	$\sqrt{(Sy)^2+a^2}$	$\pi-\tan^{-1}\left(\frac{-Sy}{a}\right)$
15	$\sqrt{(Sy)^2+(Ts)^2}$	$\tan^{-1}\left(\frac{Sy}{Ts}\right)$	Sy	$\pi/2$
16	$\sqrt{(Sy)^2+(Ts+a)^2}$	$\tan^{-1}\left(\frac{Sy}{(Ts+a)}\right)$	$\sqrt{(Sy)^2+a^2}$	$\tan^{-1}\left(\frac{Sy}{a}\right)$
17	$\sqrt{(Sy)^2+(Ts+a+0.10\,m)^2}$	$\tan^{-1}\left(\frac{Sy}{(Ts+a+0.10\,m)}\right)$	$\sqrt{(Sy)^2+(a+0.10\,m)^2}$	$\tan^{1}\left(\frac{Sy}{(a+0.10\,m)}\right)$

Table 9: Radial Distance and Angle for the Stress Calculation Locations. Tridem Axles [continued]

	Wheel 5		Wheel 6	
Point	**Radius**	**Angle**	**Radius**	**Angle**
0	$\sqrt{\left(\frac{Ts}{2}\right)^2+(Sy)^2}$	$\tan^{-1}\left(\frac{-2Sy}{Ts}\right)$	$\sqrt{\left(\frac{Ts}{2}\right)^2+(Sy)^2}$	$\pi+\tan^{-1}\left(\frac{2Sy}{Ts}\right)$
1	$\sqrt{(Sy)^2+\left(\frac{3Ts}{4}-\frac{a}{2}\right)^2}$	$\tan^{-1}\left(\frac{-2Sy}{1.5Ts-a}\right)$	$\sqrt{(Sy)^2+\left(\frac{Ts}{4}+\frac{a}{2}\right)^2}$	$\pi+\tan^{-1}\left(\frac{2Sy}{0.5Ts+a}\right)$
2	$\sqrt{(Sy)^2+(Ts-a)^2}$	$\tan^{-1}\left(\frac{-Sy}{(Ts-a)}\right)$	$\sqrt{(Sy)^2+a^2}$	$\pi+\tan^{-1}\left(\frac{Sy}{a}\right)$
3	$\sqrt{(Sy)^2+(Ts)^2}$	$\tan^{-1}\left(\frac{-Sy}{Ts}\right)$	Sy	-π/2
4	$\sqrt{(Sy)^2+(Ts+a)^2}$	$\tan^{-1}\left(\frac{-Sy}{(Ts+a)}\right)$	$\sqrt{(Sy)^2+a^2}$	$\tan^{-1}\left(\frac{-Sy}{a}\right)$
5	$\sqrt{(Sy)^2+(Ts+a+0.10\,m)^2}$	$\tan^{-1}\left(\frac{-Sy}{(Ts+a+0.10\,m)}\right)$	$\sqrt{(Sy)^2+(a+0.10m)^2}$	$\tan^{-1}\left(\frac{-Sy}{(a+0.10m)}\right)$
6	$\sqrt{\left(\frac{Ts}{2}\right)^2+\left(\frac{Sy}{2}\right)^2}$	$\tan^{-1}\left(\frac{-Sy}{Ts}\right)$	$\sqrt{\left(\frac{Ts}{2}\right)^2+\left(\frac{Sy}{2}\right)^2}$	$\pi+\tan^{-1}\left(\frac{Sy}{Ts}\right)$
7	$\sqrt{\left(\frac{Sy}{2}\right)^2+\left(\frac{3Ts}{4}-\frac{a}{2}\right)^2}$	$\tan^{-1}\left(\frac{-Sy}{1.5Ts-a}\right)$	$\sqrt{\left(\frac{Sy}{2}\right)^2+\left(\frac{Ts}{4}+\frac{a}{2}\right)^2}$	$\pi+\tan^{-1}\left(\frac{Sy}{0.5Ts+a}\right)$
8	$\sqrt{\left(\frac{Sy}{2}\right)^2+(Ts-a)^2}$	$\tan^{-1}\left(\frac{-Sy}{2(Ts-a)}\right)$	$\sqrt{\left(\frac{Sy}{2}\right)^2+a^2}$	$\pi+\tan^{-1}\left(\frac{Sy}{2a}\right)$
9	$\sqrt{\left(\frac{Sy}{2}\right)^2+(Ts)^2}$	$\tan^{-1}\left(\frac{-Sy}{2Ts}\right)$	$\frac{Sy}{2}$	-π/2
10	$\sqrt{\left(\frac{Sy}{2}\right)^2+(Ts+a)^2}$	$\tan^{-1}\left(\frac{-Sy}{2(Ts+a)}\right)$	$\sqrt{\left(\frac{Sy}{2}\right)^2+a^2}$	$\tan^{-1}\left(\frac{-Sy}{2a}\right)$
11	$\sqrt{\left(\frac{Sy}{2}\right)^2+(Ts+a+0.10\,m)^2}$	$\tan^{-1}\left(\frac{-Sy}{2(Ts+a+0.10m)}\right)$	$\sqrt{\left(\frac{Sy}{2}\right)^2+(a+0.10m)^2}$	$\tan^{-1}\left(\frac{-Sy}{2(a+0.10m)}\right)$
12	$\frac{Ts}{2}$	0	$\frac{Ts}{2}$	π
13	$\frac{3Ts}{4}-\frac{a}{2}$	0	$\frac{Ts}{4}+\frac{a}{2}$	π
14	$Ts-a$	0	a	π
15	Ts	0	0	0
16	$Ts+a$	0	a	0
17	$Ts+a+0.10\,m$	0	$a+0.10\,m$	0

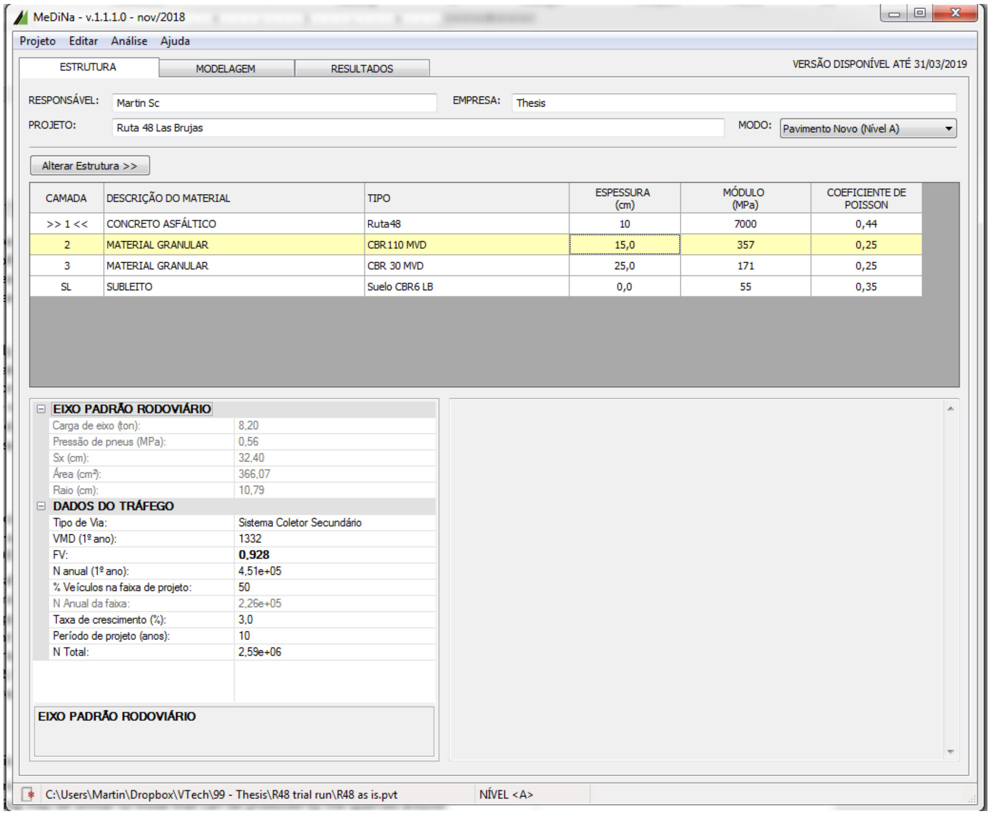
Note: Refer to figure 24 for details on the location of the stress and strain evaluation points.

APPENDIX D – HIGHWAY 48 DESIGN EXAMPLE: SCREENSHOTS OF DESIGN INPUTS IN THIRD-PARTY PAVEMENT M-E TOOLS

This Appendix is a supplement to the information provided in Chapter 4 – the draft-project-level design of a pavement structure with Product-One and two other M-E design suites. It features screenshots of the two third-party M-E design tools tried in this project which depict how the many inputs to the design process were loaded unto each. The author's intention behind including these screenshots is to allow for the most calculations in the case study presented in Chapter 4 to be reproduced should the reader desires to do so.

Two sections compose this Appendix, firstly a set of screenshots of MeDiNa's many data input windows will be shown and afterward captures from CR-ME's step-wise procedure follow.

Highway 48 example in MeDiNa



VERSÃO DISPONÍVEL ATÉ 31/03/2019

RESPONSÁVEL: Martin Sc

EMPRESA: Thesis

PROJETO: Ruta 48 Las Brujas

MODO: Pavimento Novo (Nível A)

CAMADA	DESCRIÇÃO DO MATERIAL	TIPO	ESPESSURA (cm)	MÓDULO (MPa)	COEFICIENTE DE POISSON
>> 1 <<	CONCRETO ASFÁLTICO	Ruta48	10	7000	0,44
2	MATERIAL GRANULAR	CBR 110 MVD	15,0	357	0,25
3	MATERIAL GRANULAR	CBR 30 MVD	25,0	171	0,25
SL	SUBLEITO	Suelo CBR6 LB	0,0	55	0,35

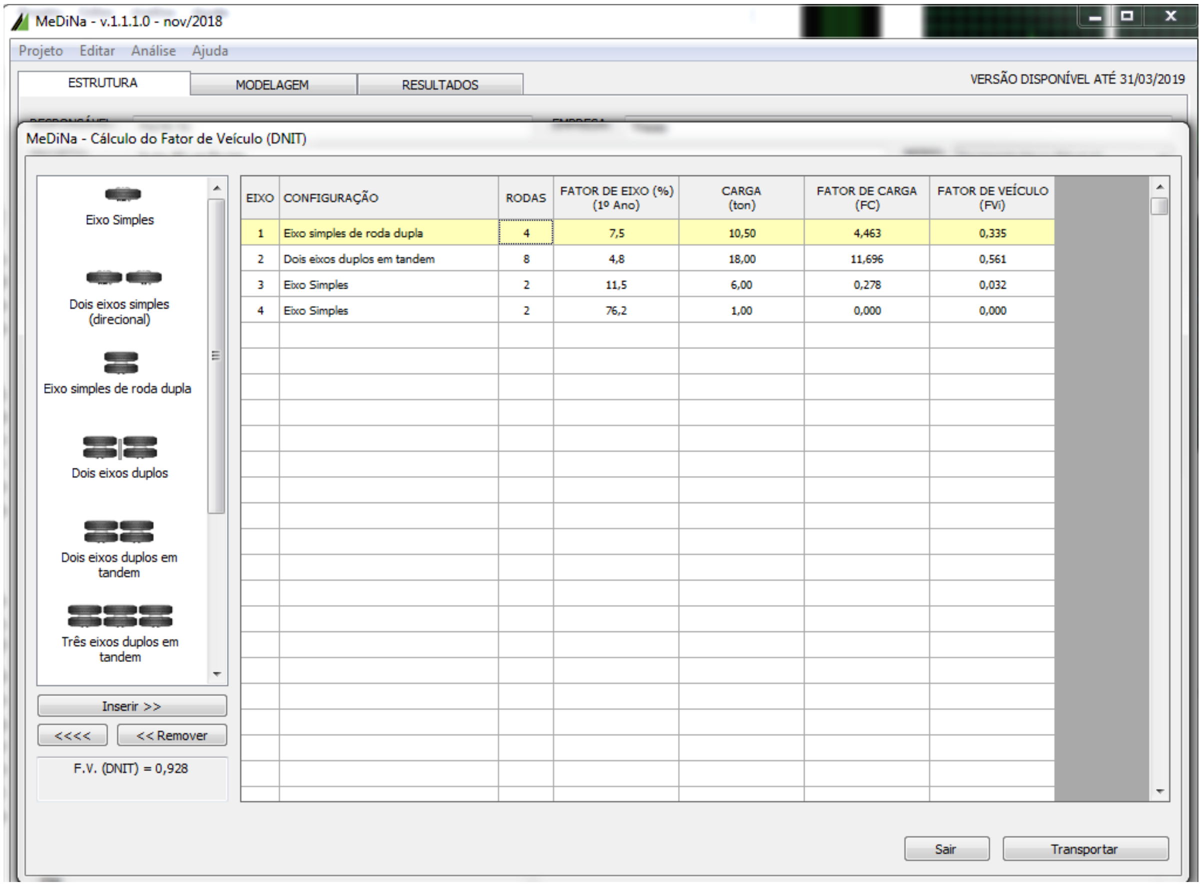
EIXO PADRÃO RODOVIÁRIO	
Carga de eixo (ton):	8,20
Pressão de pneus (MPa):	0,56
Sx (cm):	32,40
Área (cm²):	366,07
Raio (cm):	10,79
DADOS DO TRÁFEGO	
Tipo de Via:	Sistema Coletor Secundário
VMD (1º ano):	1332
FV:	**0,928**
N anual (1º ano):	4,51e+05
% Veículos na faixa de projeto:	50
N Anual da faixa:	2,26e+05
Taxa de crescimento (%):	3,0
Período de projeto (anos):	10
N Total:	2,59e+06

EIXO PADRÃO RODOVIÁRIO

C:\Users\Martin\Dropbox\VTech\99 - Thesis\R48 trial run\R48 as is.pvt

NÍVEL <A>

Figure D-1: Screen capture of MeDiNa's main menu


MeDiNa - v.1.1.1.0 - nov/2018
Projeto
Editar
Análise
Ajuda
ESTRUTURA
MODELAGEM
RESULTADOS
Eixo Simples
Dois eixos simples (direcional)
Eixo simples de roda dupla
Dois eixos duplos
Dois eixos duplos em tandem
Três eixos duplos em tandem
Inserir >>
<<<<
<< Remover
Sair
Transportar

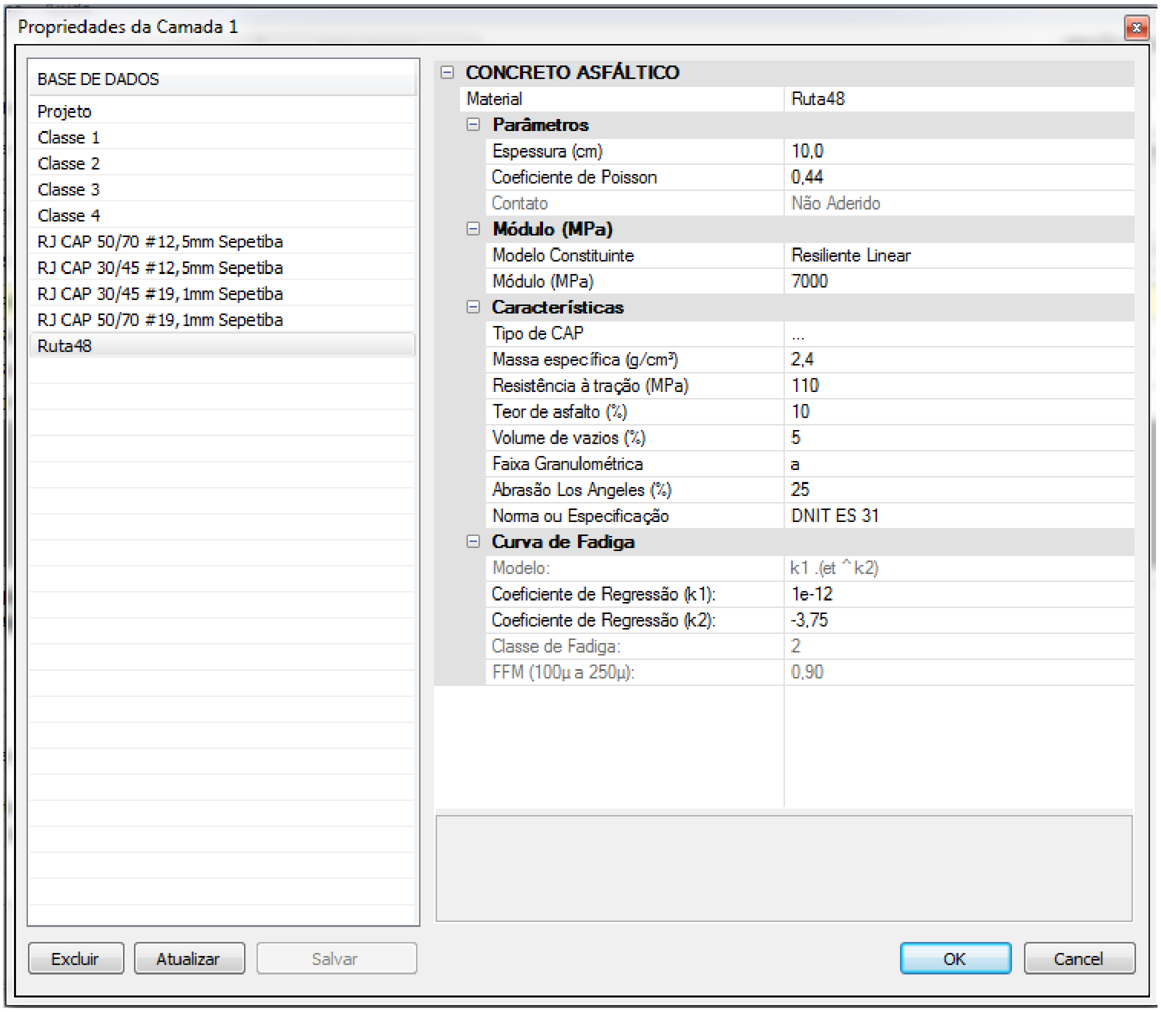
VERSÃO DISPONÍVEL ATÉ 31/03/2019

MeDiNa - Cálculo do Fator de Veículo (DNIT)

EIXO	CONFIGURAÇÃO	RODAS	FATOR DE EIXO (%) (1º Ano)	CARGA (ton)	FATOR DE CARGA (FC)	FATOR DE VEÍCULO (FVi)
1	Eixo simples de roda dupla	4	7,5	10,50	4,463	0,335
2	Dois eixos duplos em tandem	8	4,8	18,00	11,696	0,561
3	Eixo Simples	2	11,5	6,00	0,278	0,032
4	Eixo Simples	2	76,2	1,00	0,000	0,000

F.V. (DNIT) = 0,928

Figure D-2: MeDiNa traffic load and axle configuration data form

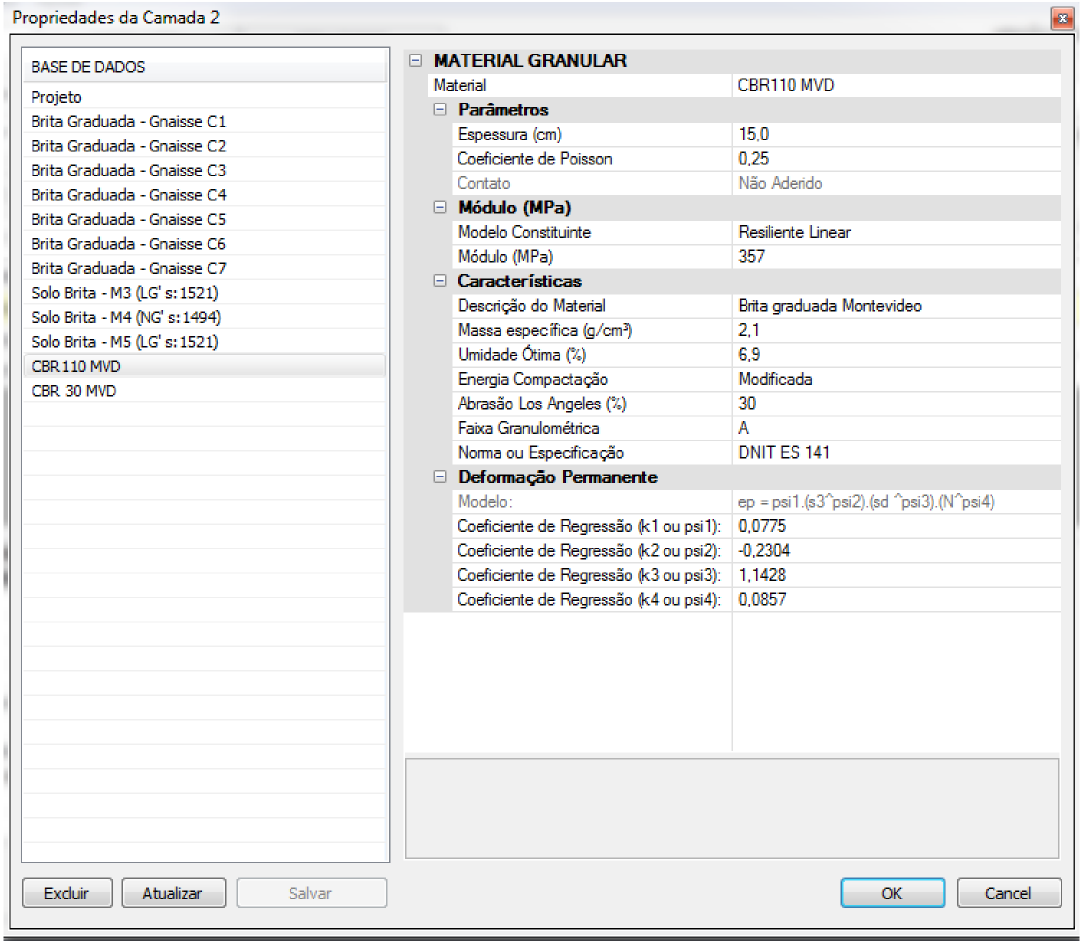


BASE DE DADOS

- Projeto
- Classe 1
- Classe 2
- Classe 3
- Classe 4
- RJ CAP 50/70 #12,5mm Sepetiba
- RJ CAP 30/45 #12,5mm Sepetiba
- RJ CAP 30/45 #19,1mm Sepetiba
- RJ CAP 50/70 #19,1mm Sepetiba
- Ruta48

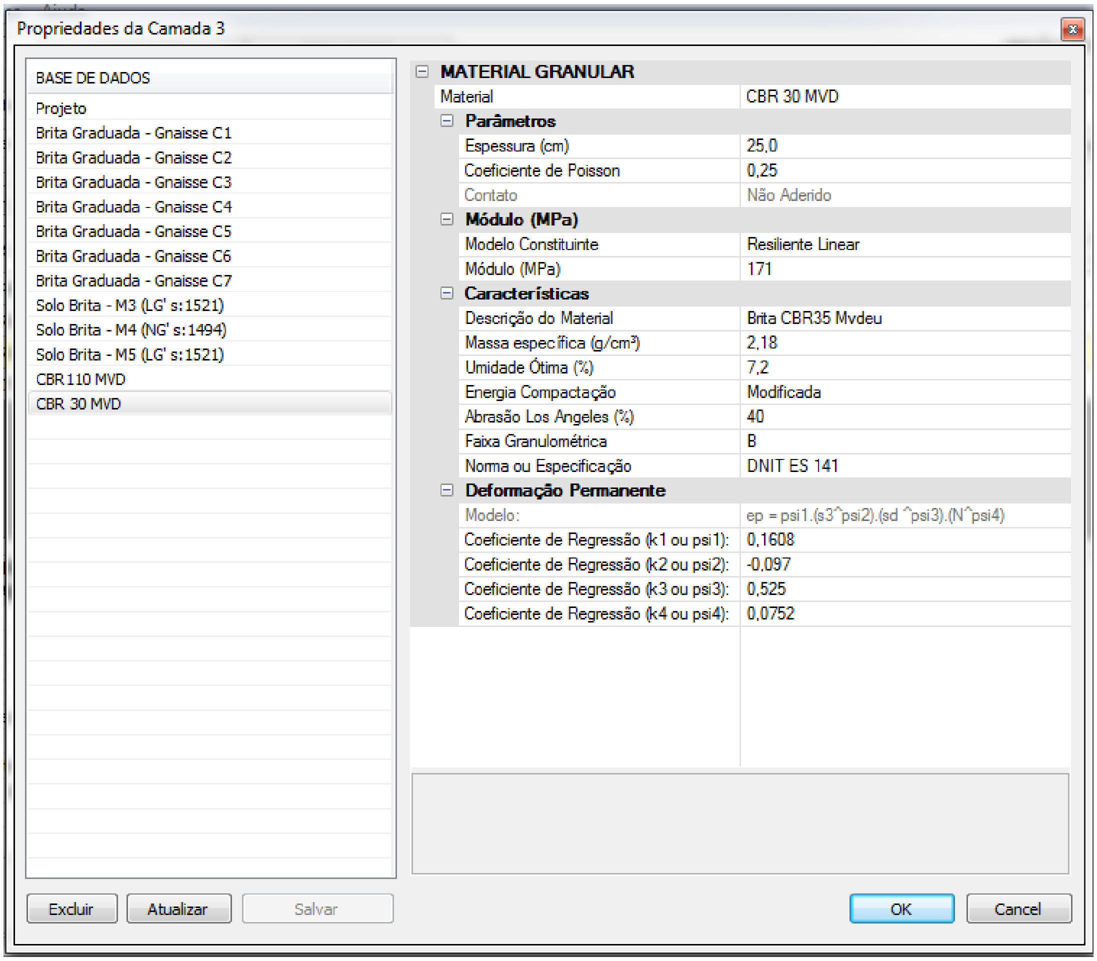
CONCRETO ASFÁLTICO	
Material	Ruta48
Parâmetros	
Espessura (cm)	10,0
Coeficiente de Poisson	0,44
Contato	Não Aderido
Módulo (MPa)	
Modelo Constituinte	Resiliente Linear
Módulo (MPa)	7000
Características	
Tipo de CAP	...
Massa específica (g/cm³)	2,4
Resistência à tração (MPa)	110
Teor de asfalto (%)	10
Volume de vazios (%)	5
Faixa Granulométrica	a
Abrasão Los Angeles (%)	25
Norma ou Especificação	DNIT ES 31
Curva de Fadiga	
Modelo:	k1 .(et ^ k2)
Coeficiente de Regressão (k1):	1e-12
Coeficiente de Regressão (k2):	-3,75
Classe de Fadiga:	2
FFM (100µ a 250µ):	0,90

Figure D-3: MeDiNa's material properties dialog – asphalt mixes


Propriedades da Camada 2
BASE DE DADOS
Projeto
Brita Graduada - Gnaisse C1
Brita Graduada - Gnaisse C2
Brita Graduada - Gnaisse C3
Brita Graduada - Gnaisse C4
Brita Graduada - Gnaisse C5
Brita Graduada - Gnaisse C6
Brita Graduada - Gnaisse C7
Solo Brita - M3 (LG' s: 1521)
Solo Brita - M4 (NG' s: 1494)
Solo Brita - M5 (LG' s: 1521)
CBR 110 MVD
CBR 30 MVD
Excluir
Atualizar
Salvar
OK
Cancel

MATERIAL GRANULAR	
Material	CBR110 MVD
Parâmetros	
Espessura (cm)	15,0
Coeficiente de Poisson	0,25
Contato	Não Aderido
Módulo (MPa)	
Modelo Constituinte	Resiliente Linear
Módulo (MPa)	357
Características	
Descrição do Material	Brita graduada Montevideo
Massa específica (g/cm^3)	2,1
Umidade Ótima (%)	6,9
Energia Compactação	Modificada
Abrasão Los Angeles (%)	30
Faixa Granulométrica	A
Norma ou Especificação	DNIT ES 141
Deformação Permanente	
Modelo:	ep = psi1.(s3^psi2).(sd ^psi3).(N^psi4)
Coeficiente de Regressão (k1 ou psi1):	0,0775
Coeficiente de Regressão (k2 ou psi2):	-0,2304
Coeficiente de Regressão (k3 ou psi3):	1,1428
Coeficiente de Regressão (k4 ou psi4):	0,0857

Figure D-4: MeDiNa's material properties dialog – Crushed stone granular material



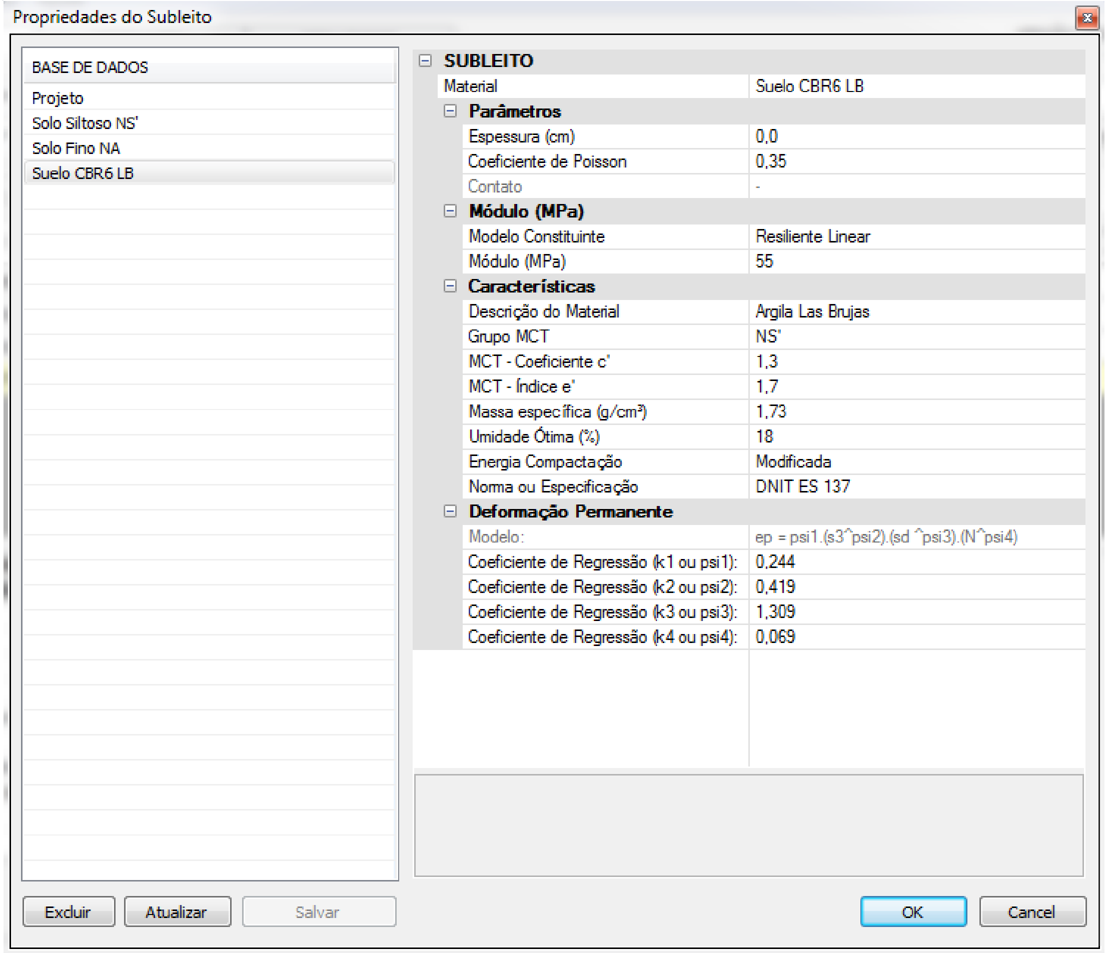
Propriedades da Camada 3

BASE DE DADOS

- Projeto
- Brita Graduada - Gnaisse C1
- Brita Graduada - Gnaisse C2
- Brita Graduada - Gnaisse C3
- Brita Graduada - Gnaisse C4
- Brita Graduada - Gnaisse C5
- Brita Graduada - Gnaisse C6
- Brita Graduada - Gnaisse C7
- Solo Brita - M3 (LG' s:1521)
- Solo Brita - M4 (NG' s:1494)
- Solo Brita - M5 (LG' s:1521)
- CBR 110 MVD
- CBR 30 MVD

MATERIAL GRANULAR	
Material	CBR 30 MVD
Parâmetros	
Espessura (cm)	25,0
Coeficiente de Poisson	0,25
Contato	Não Aderido
Módulo (MPa)	
Modelo Constituinte	Resiliente Linear
Módulo (MPa)	171
Características	
Descrição do Material	Brita CBR35 Mvdeu
Massa específica (g/cm³)	2,18
Umidade Ótima (%)	7,2
Energia Compactação	Modificada
Abrasão Los Angeles (%)	40
Faixa Granulométrica	B
Norma ou Especificação	DNIT ES 141
Deformação Permanente	
Modelo:	ep = psi1.(s3^psi2).(sd ^psi3).(N^psi4)
Coeficiente de Regressão (k1 ou psi1):	0,1608
Coeficiente de Regressão (k2 ou psi2):	-0,097
Coeficiente de Regressão (k3 ou psi3):	0,525
Coeficiente de Regressão (k4 ou psi4):	0,0752

Figure D-5: MeDiNa's material properties dialog –granular sub-base material



Propriedades do Subleito

BASE DE DADOS

Projeto

Solo Siltoso NS'

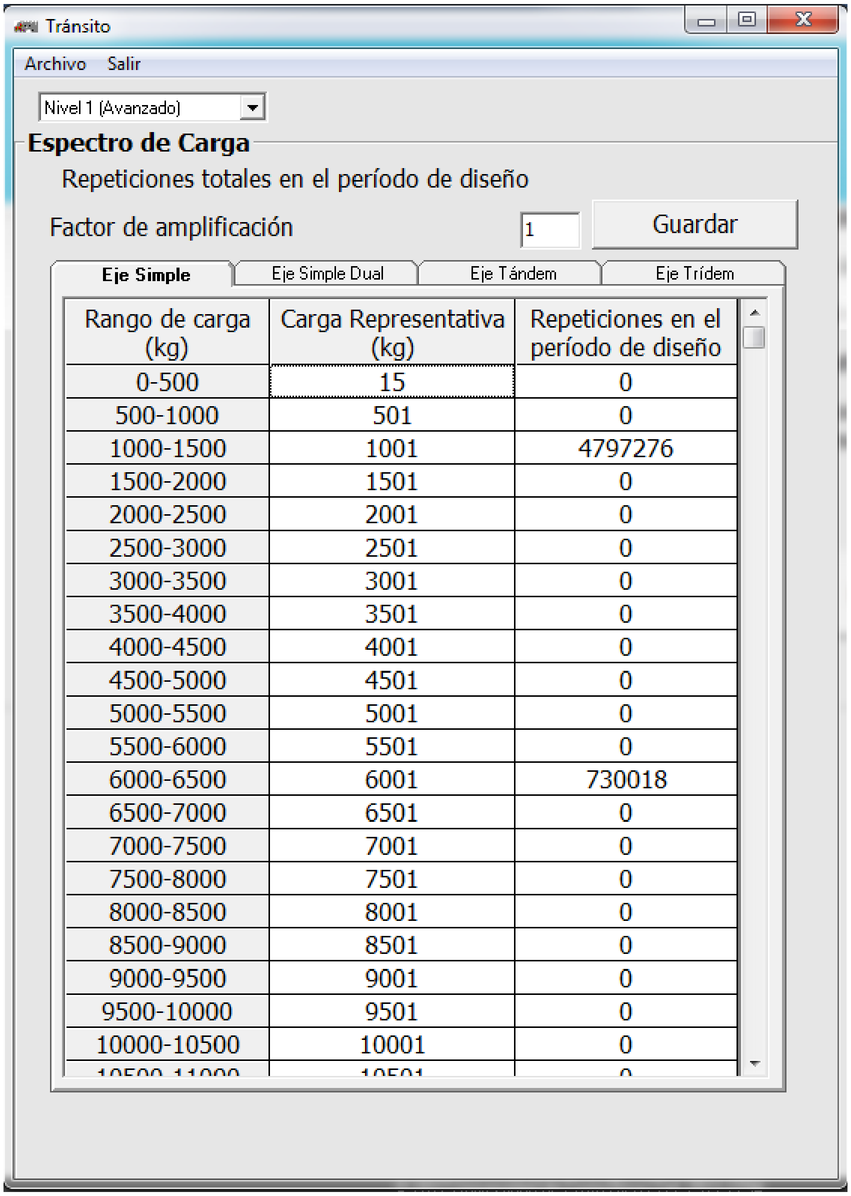
Solo Fino NA

Suelo CBR6 LB

SUBLEITO	
Material	Suelo CBR6 LB
Parâmetros	
Espessura (cm)	0,0
Coeficiente de Poisson	0,35
Contato	-
Módulo (MPa)	
Modelo Constituinte	Resiliente Linear
Módulo (MPa)	55
Características	
Descrição do Material	Argila Las Brujas
Grupo MCT	NS'
MCT - Coeficiente c'	1,3
MCT - Índice e'	1,7
Massa específica (g/cm³)	1,73
Umidade Ótima (%)	18
Energia Compactação	Modificada
Norma ou Especificação	DNIT ES 137
Deformação Permanente	
Modelo:	ep = psi1.(s3^psi2).(sd ^psi3).(N^psi4)
Coeficiente de Regressão (k1 ou psi1):	0,244
Coeficiente de Regressão (k2 ou psi2):	0,419
Coeficiente de Regressão (k3 ou psi3):	1,309
Coeficiente de Regressão (k4 ou psi4):	0,069

Figure D-6: MeDiNa's material properties dialog –subgrade

Highway 48 example in CR-ME



Nivel 1 (Avanzado)

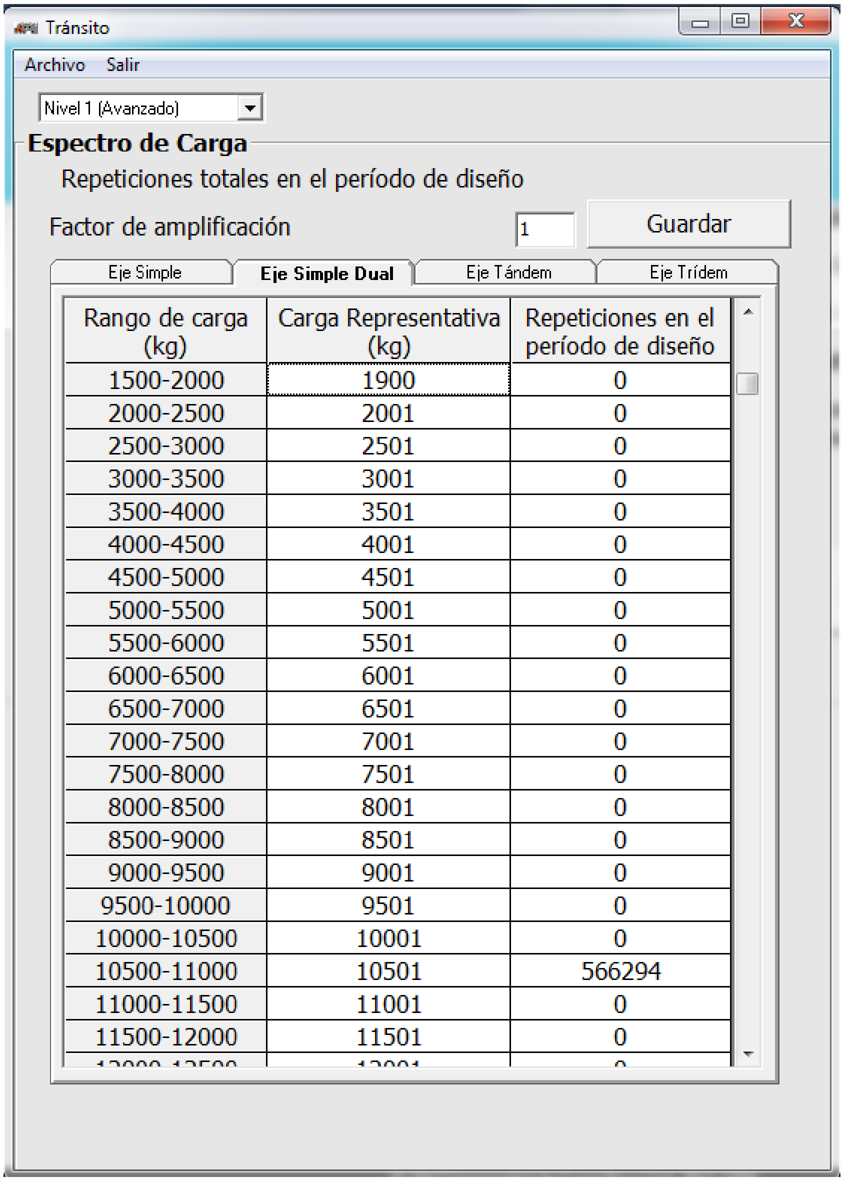
Espectro de Carga

Repeticiones totales en el período de diseño

Factor de amplificación: 1

Rango de carga (kg)	Carga Representativa (kg)	Repeticiones en el período de diseño
0-500	15	0
500-1000	501	0
1000-1500	1001	4797276
1500-2000	1501	0
2000-2500	2001	0
2500-3000	2501	0
3000-3500	3001	0
3500-4000	3501	0
4000-4500	4001	0
4500-5000	4501	0
5000-5500	5001	0
5500-6000	5501	0
6000-6500	6001	730018
6500-7000	6501	0
7000-7500	7001	0
7500-8000	7501	0
8000-8500	8001	0
8500-9000	8501	0
9000-9500	9001	0
9500-10000	9501	0
10000-10500	10001	0
10500-11000	10501	0

Figure D-7: Traffic inputs for CR-ME: Single-wheel single axles



Nivel 1 (Avanzado)

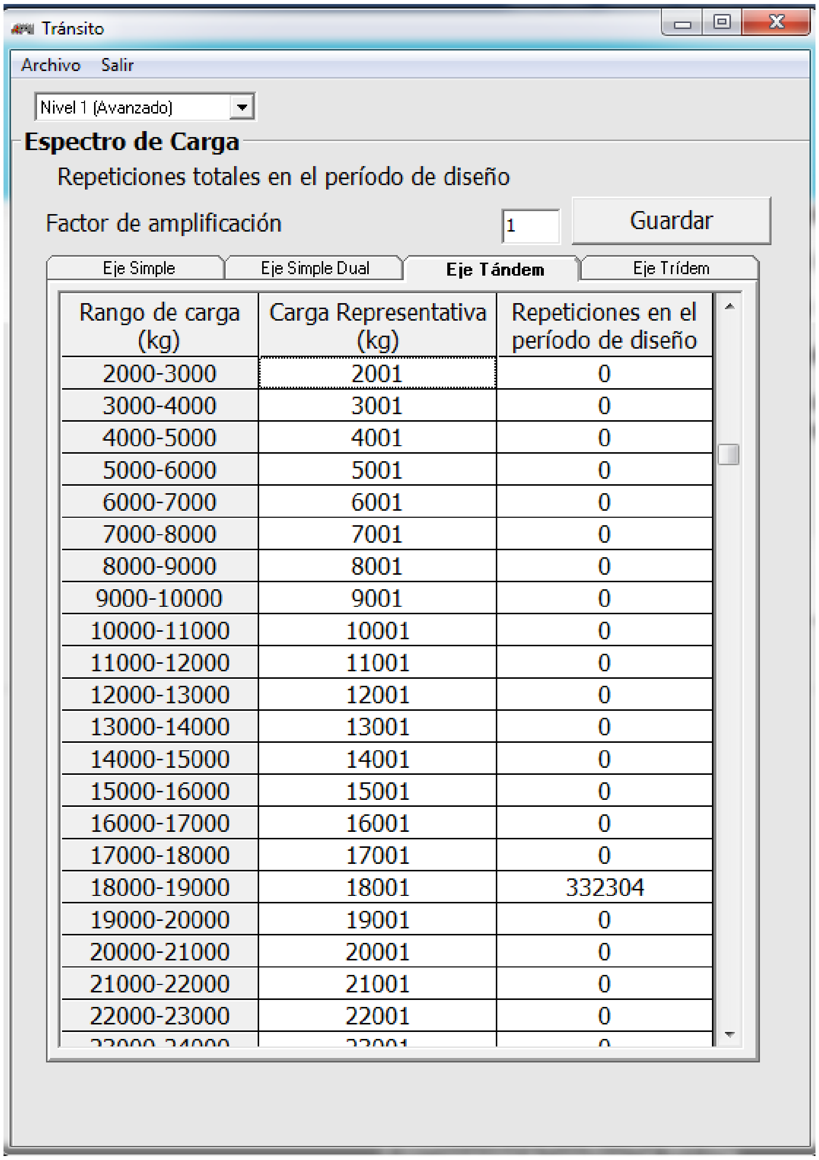
Espectro de Carga

Repeticiones totales en el período de diseño

Factor de amplificación 1

Rango de carga (kg)	Carga Representativa (kg)	Repeticiones en el período de diseño
1500-2000	1900	0
2000-2500	2001	0
2500-3000	2501	0
3000-3500	3001	0
3500-4000	3501	0
4000-4500	4001	0
4500-5000	4501	0
5000-5500	5001	0
5500-6000	5501	0
6000-6500	6001	0
6500-7000	6501	0
7000-7500	7001	0
7500-8000	7501	0
8000-8500	8001	0
8500-9000	8501	0
9000-9500	9001	0
9500-10000	9501	0
10000-10500	10001	0
10500-11000	10501	566294
11000-11500	11001	0
11500-12000	11501	0
12000-12500	12001	0

Figure D-8: Traffic inputs for CR-ME: Dual-wheel single axles



Nivel 1 (Avanzado)

Espectro de Carga

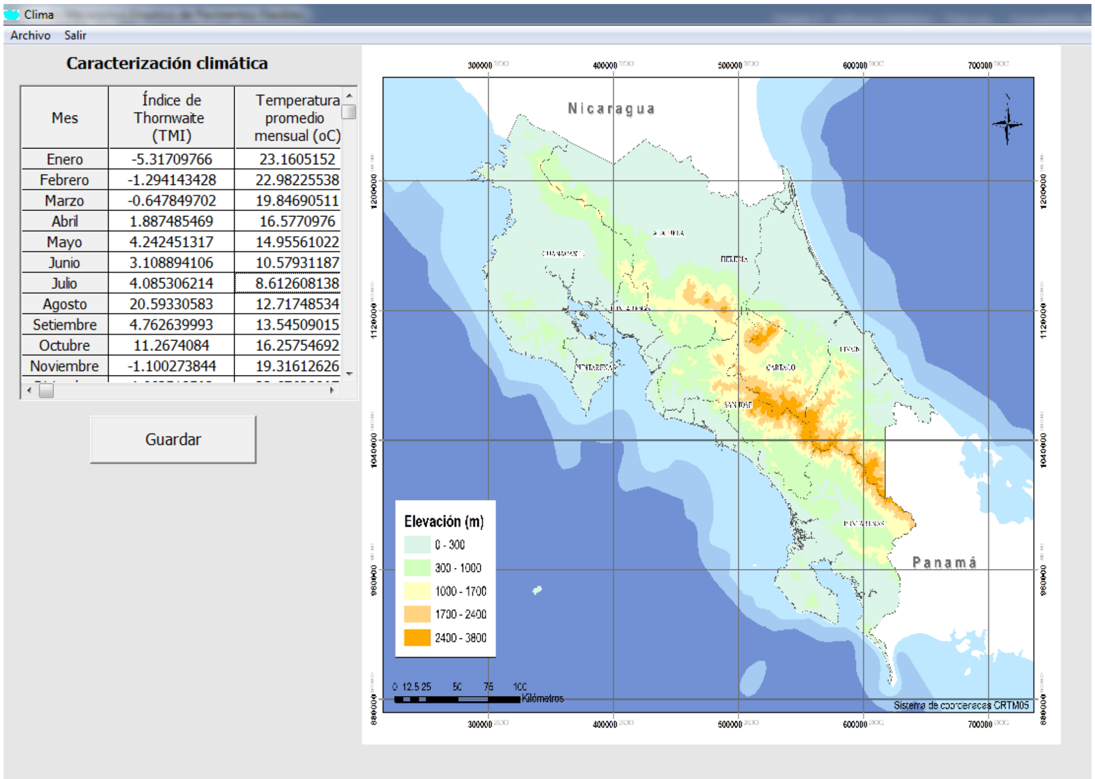
Repeticiones totales en el período de diseño

Factor de amplificación 1

Rango de carga (kg)	Carga Representativa (kg)	Repeticiones en el período de diseño
2000-3000	2001	0
3000-4000	3001	0
4000-5000	4001	0
5000-6000	5001	0
6000-7000	6001	0
7000-8000	7001	0
8000-9000	8001	0
9000-10000	9001	0
10000-11000	10001	0
11000-12000	11001	0
12000-13000	12001	0
13000-14000	13001	0
14000-15000	14001	0
15000-16000	15001	0
16000-17000	16001	0
17000-18000	17001	0
18000-19000	18001	332304
19000-20000	19001	0
20000-21000	20001	0
21000-22000	21001	0
22000-23000	22001	0
23000-24000	23001	0

Figure D-9: Traffic inputs for CR-ME: Tandem axles

Note: No figure was created depicting the input dialog for the tridem axles since it does not differ from the dialogs for the other axle types and also because no tridem axle traffic has been predicted for the Highway 48 case study.

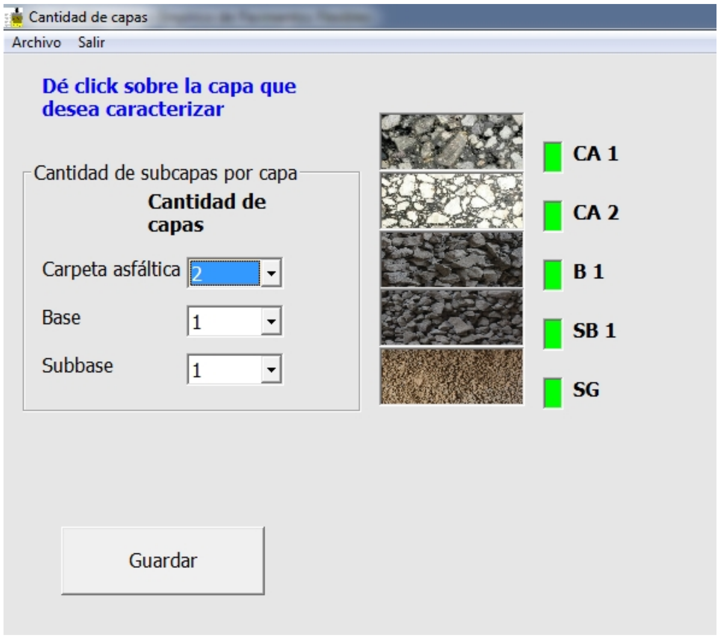


Clima

Caracterización climática

Mes	Índice de Thornwaite (TMI)	Temperatura promedio mensual (oC)
Enero	-5.31709766	23.1605152
Febrero	-1.294143428	22.98225538
Marzo	-0.647849702	19.84690511
Abril	1.887485469	16.5770976
Mayo	4.242451317	14.95561022
Junio	3.108894106	10.57931187
Julio	4.085306214	8.612608138
Agosto	20.59330583	12.71748534
Setiembre	4.762639993	13.54509015
Octubre	11.2674084	16.25754692
Noviembre	-1.100273844	19.31612626

Figure D-10: Climate input dialog in CR-ME.



Cantidad de capas

Dé click sobre la capa que desea caracterizar

Cantidad de subcapas por capa

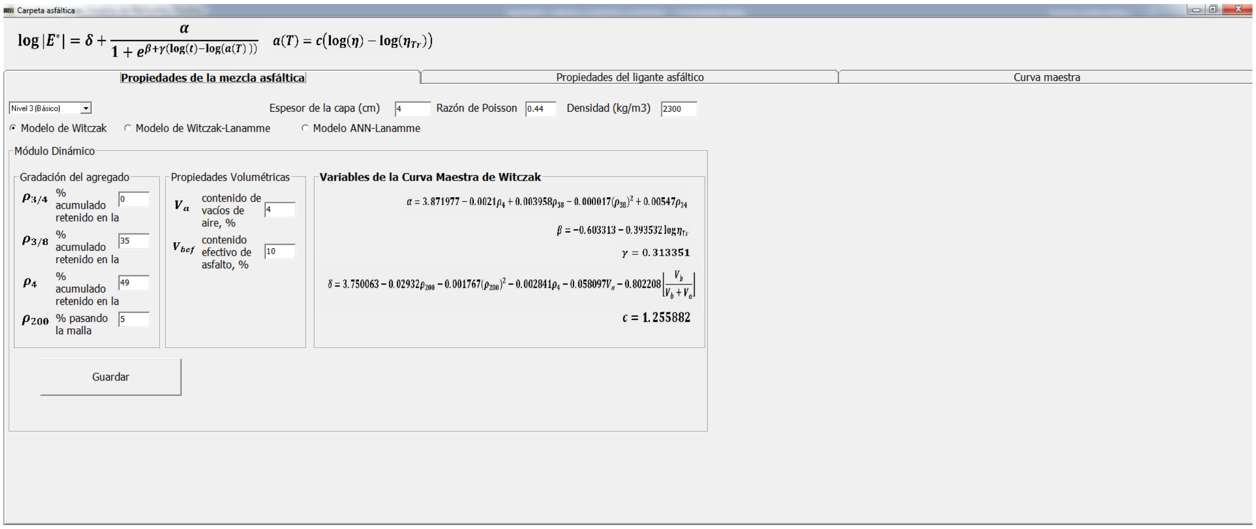
Cantidad de capas

Carpeta asfáltica: 2

Base: 1

Subbase: 1

Figure D-11: CR-ME's pavement structure definition screen.


Carpeta asfáltica
Propiedades de la mezcla asfáltica
Propiedades del ligante asfáltico
Curva maestra
Guardar

$$\log|E^*| = \delta + \frac{\alpha}{1 + e^{\beta+\gamma(\log(t)-\log(a(T)))}} \quad a(T) = c(\log(\eta) - \log(\eta_{Tr}))$$

Nivel 3 (Básico)

Espesor de la capa (cm): 4 — Razón de Poisson: 0.44 — Densidad (kg/m3): 2300

● Modelo de Witczak ○ Modelo de Witczak-Lanamme ○ Modelo ANN-Lanamme

Módulo Dinámico

Gradación del agregado

- $\rho_{3/4}$ % acumulado retenido en la: 0
- $\rho_{3/8}$ % acumulado retenido en la: 35
- ρ_4 % acumulado retenido en la: 49
- ρ_{200} % pasando la malla: 5

Propiedades Volumétricas

- V_a contenido de vacíos de aire, %: 4
- V_{bef} contenido efectivo de asfalto, %: 10

Variables de la Curva Maestra de Witczak

$$\alpha = 3.871977 - 0.0021\rho_4 + 0.003958\rho_{38} - 0.000017(\rho_{38})^2 + 0.00547\rho_{34}$$

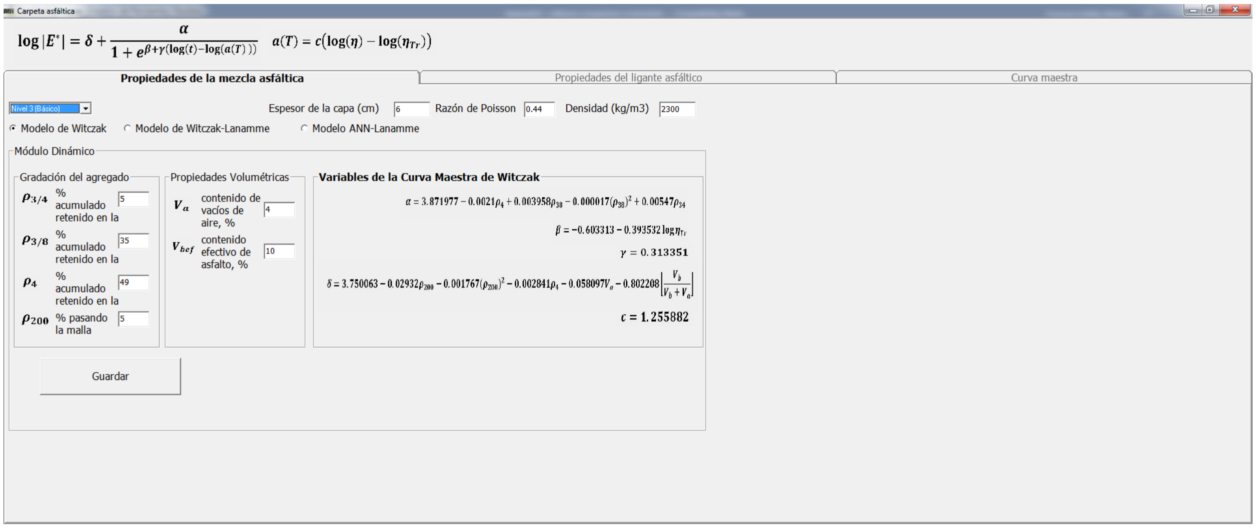
$$\beta = -0.603313 - 0.393532\log\eta_{Tr}$$

$$\gamma = 0.313351$$

$$\delta = 3.750063 - 0.02932\rho_{200} - 0.001767(\rho_{200})^2 - 0.002841\rho_4 - 0.058097V_a - 0.802208\left[\frac{V_b}{V_b + V_a}\right]$$

$$c = 1.255882$$

Figure D-12: CR-ME's asphalt materials input dialog. Mix properties for the lower asphalt layer


Carpeta asfáltica
Propiedades de la mezcla asfáltica
Propiedades del ligante asfáltico
Curva maestra
Guardar

$$\log|E^*| = \delta + \frac{\alpha}{1 + e^{\beta+\gamma(\log(t)-\log(a(T)))}} \quad a(T) = c(\log(\eta) - \log(\eta_{Tr}))$$

Nivel 3 (Básico)

Espesor de la capa (cm): 6 — Razón de Poisson: 0.44 — Densidad (kg/m3): 2300

● Modelo de Witczak ○ Modelo de Witczak-Lanamme ○ Modelo ANN-Lanamme

Módulo Dinámico

Gradación del agregado

- $\rho_{3/4}$ % acumulado retenido en la: 5
- $\rho_{3/8}$ % acumulado retenido en la: 35
- ρ_4 % acumulado retenido en la: 49
- ρ_{200} % pasando la malla: 5

Propiedades Volumétricas

- V_a contenido de vacíos de aire, %: 4
- V_{bef} contenido efectivo de asfalto, %: 10

Variables de la Curva Maestra de Witczak

$$\alpha = 3.871977 - 0.0021\rho_4 + 0.003958\rho_{38} - 0.000017(\rho_{38})^2 + 0.00547\rho_{34}$$

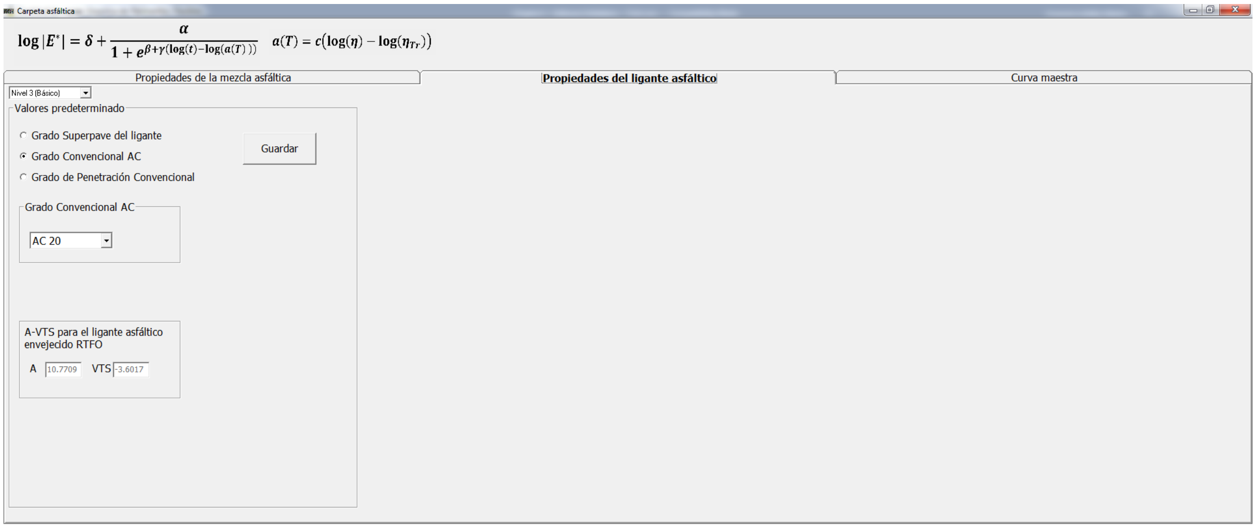
$$\beta = -0.603313 - 0.393532\log\eta_{Tr}$$

$$\gamma = 0.313351$$

$$\delta = 3.750063 - 0.02932\rho_{200} - 0.001767(\rho_{200})^2 - 0.002841\rho_4 - 0.058097V_a - 0.802208\left[\frac{V_b}{V_b + V_a}\right]$$

$$c = 1.255882$$

Figure D-13: CR-ME's asphalt materials input dialog. Mix properties for the lower asphalt layer



Carpeta asfáltica

$$\log|E^*| = \delta + \frac{\alpha}{1 + e^{\beta+\gamma(\log(t)-\log(a(T)\,))}} \quad a(T) = c\big(\log(\eta) - \log(\eta_{Tr})\big)$$

Propiedades de la mezcla asfáltica | **Propiedades del ligante asfáltico** | Curva maestra

Nivel 3 (Básico)

Valores predeterminado

○ Grado Superpave del ligante

● Grado Convencional AC

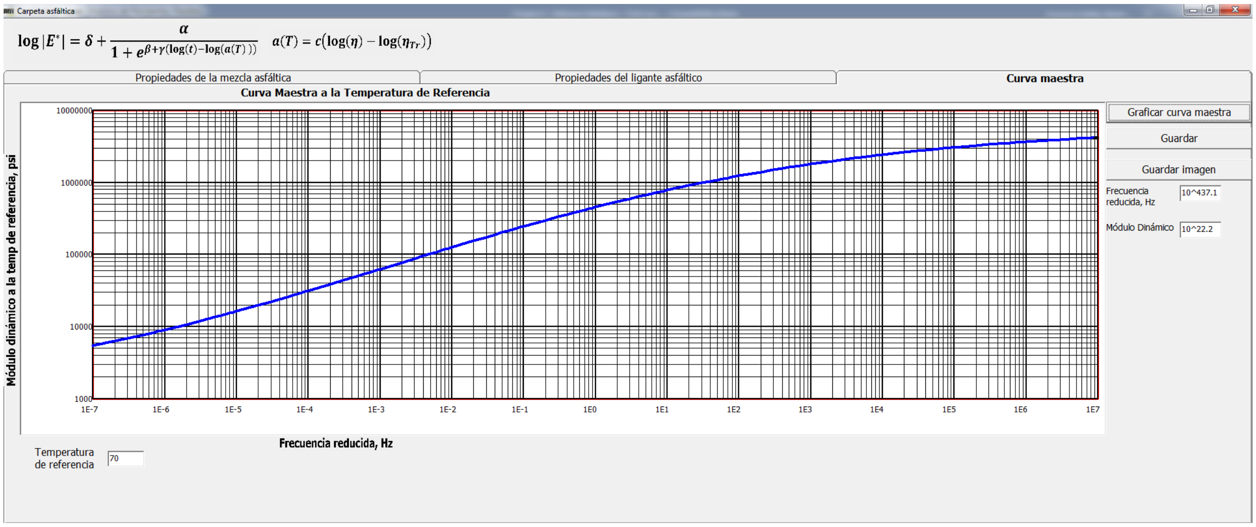
○ Grado de Penetración Convencional

Grado Convencional AC: AC 20

A-VTS para el ligante asfáltico envejecido RTFO

A 10.7709 VTS -3.6017

Figure D-14: CR-ME's asphalt materials input dialog. Binder properties for both asphalt layers



Carpeta asfáltica

$$\log|E^*| = \delta + \frac{\alpha}{1 + e^{\beta+\gamma(\log(t)-\log(a(T)\,))}} \quad a(T) = c\big(\log(\eta) - \log(\eta_{Tr})\big)$$

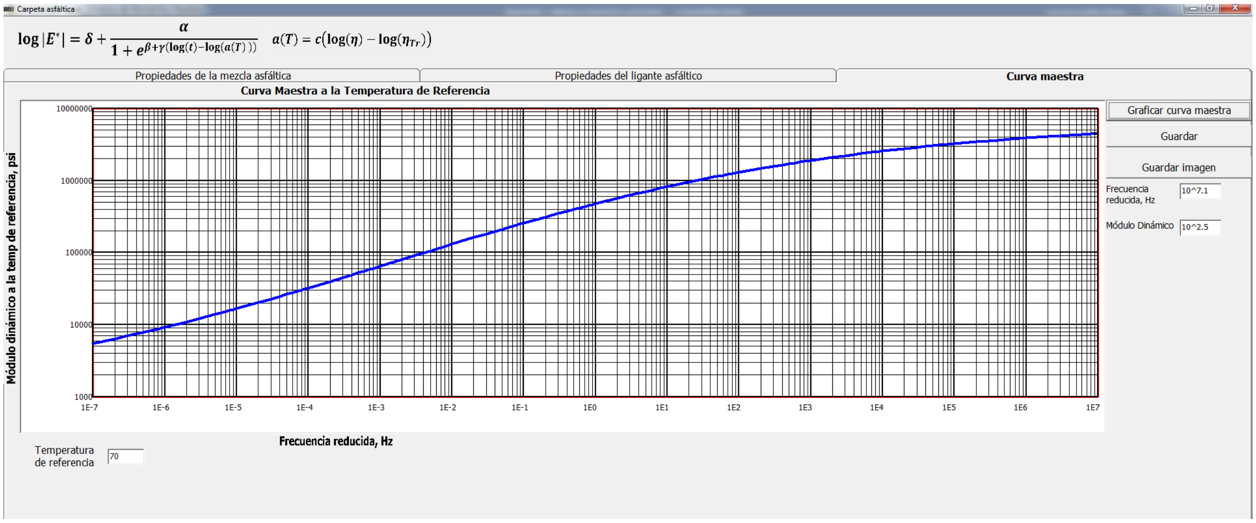
Propiedades de la mezcla asfáltica | Propiedades del ligante asfáltico | **Curva maestra**

Frecuencia reducida, Hz 10^437.1

Módulo Dinámico 10^22.2

Temperatura de referencia 70

Figure D-15: CR-ME's estimated master curve for the surface asphalt layer



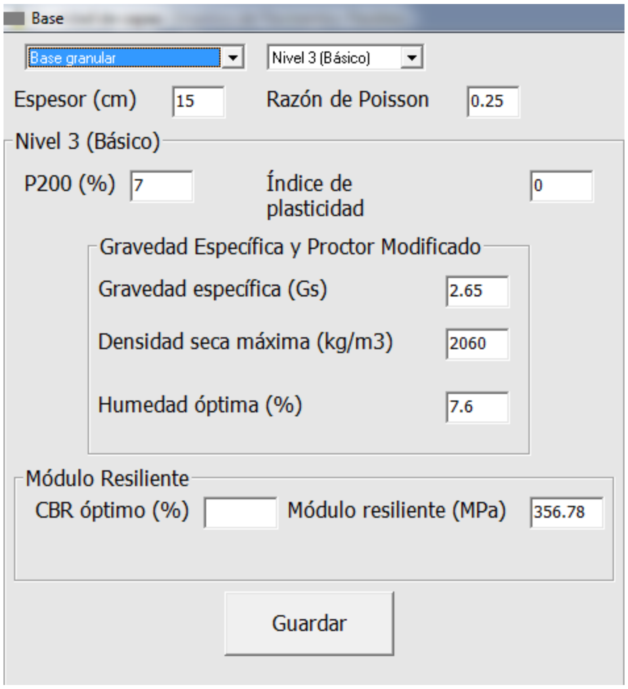
$$\log|E^*| = \delta + \frac{\alpha}{1+e^{\beta+\gamma(\log(t)-\log(a(T)))}} \quad a(T) = c\left(\log(\eta) - \log(\eta_{Tr})\right)$$

Frecuencia reducida, Hz: 10^7.1

Módulo Dinámico: 10^2.5

Temperatura de referencia: 70

Figure D-16: CR-ME's estimated master curve for the asphalt binder layer



Base granular | Nivel 3 (Básico)

Espesor (cm): 15

Razón de Poisson: 0.25

Nivel 3 (Básico)

P200 (%): 7

Índice de plasticidad: 0

Gravedad Específica y Proctor Modificado

Gravedad específica (Gs): 2.65

Densidad seca máxima (kg/m3): 2060

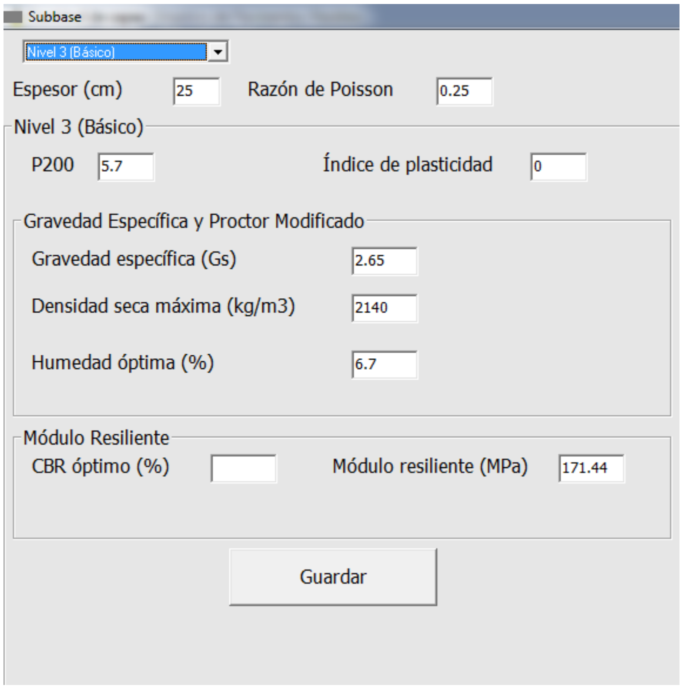
Humedad óptima (%): 7.6

Módulo Resiliente

CBR óptimo (%):

Módulo resiliente (MPa): 356.78

Figure D-17: CR-ME's material properties dialog for the base layer



Subbase

Nivel 3 (Básico)

Espesor (cm): 25

Razón de Poisson: 0.25

Nivel 3 (Básico)

P200: 5.7

Índice de plasticidad: 0

Gravedad Específica y Proctor Modificado

Gravedad específica (Gs): 2.65

Densidad seca máxima (kg/m3): 2140

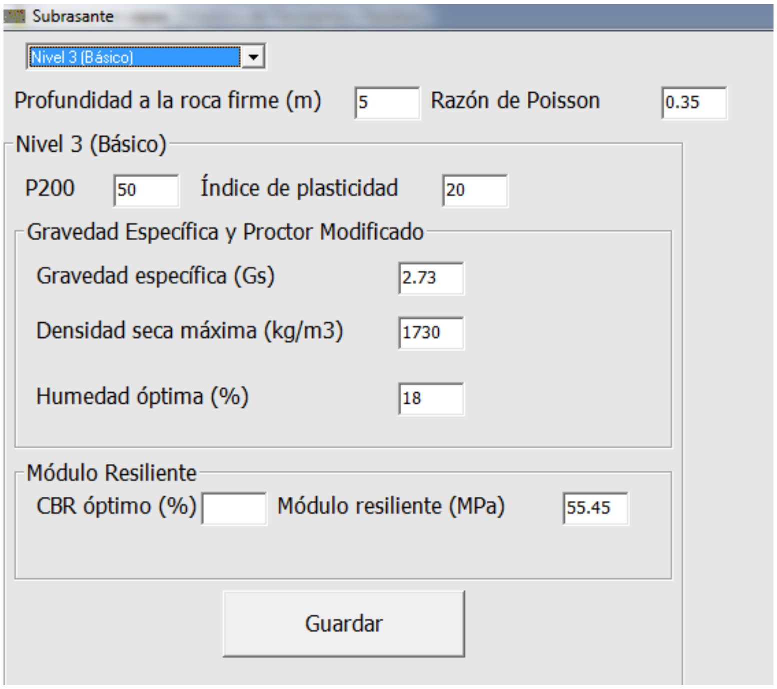
Humedad óptima (%): 6.7

Módulo Resiliente

CBR óptimo (%):

Módulo resiliente (MPa): 171.44

Figure D-18: CR-ME's material properties dialog for the sub-base layer



Subrasante

Nivel 3 (Básico)

Profundidad a la roca firme (m): 5

Razón de Poisson: 0.35

Nivel 3 (Básico)

P200: 50

Índice de plasticidad: 20

Gravedad Específica y Proctor Modificado

Gravedad específica (Gs): 2.73

Densidad seca máxima (kg/m3): 1730

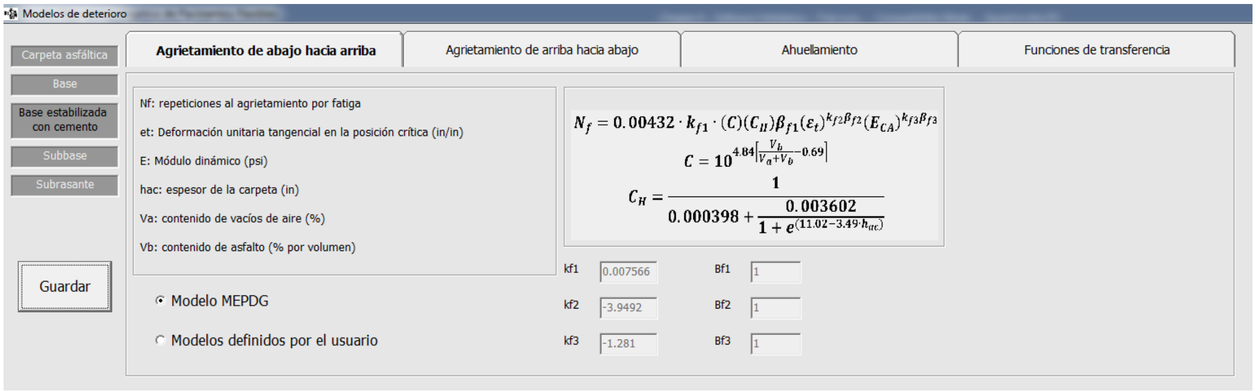
Humedad óptima (%): 18

Módulo Resiliente

CBR óptimo (%):

Módulo resiliente (MPa): 55.45

Figure D-19: CR-ME's material properties dialog for the subgrade


Modelos de deterioro
Carpeta asfáltica
Base
Base estabilizada con cemento
Subbase
Subrasante
Guardar
Agrietamiento de abajo hacia arriba
Agrietamiento de arriba hacia abajo
Ahuellamiento
Funciones de transferencia

Nf: repeticiones al agrietamiento por fatiga

et: Deformación unitaria tangencial en la posición crítica (in/in)

E: Módulo dinámico (psi)

hac: espesor de la carpeta (in)

Va: contenido de vacíos de aire (%)

Vb: contenido de asfalto (% por volumen)

$$N_f = 0.00432 \cdot k_{f1} \cdot (C)(C_H)\beta_{f1}(\varepsilon_t)^{k_{f2}\beta_{f2}}(E_{CA})^{k_{f3}\beta_{f3}}$$

$$C = 10^{4.84\left[\frac{V_b}{V_a+V_b}-0.69\right]}$$

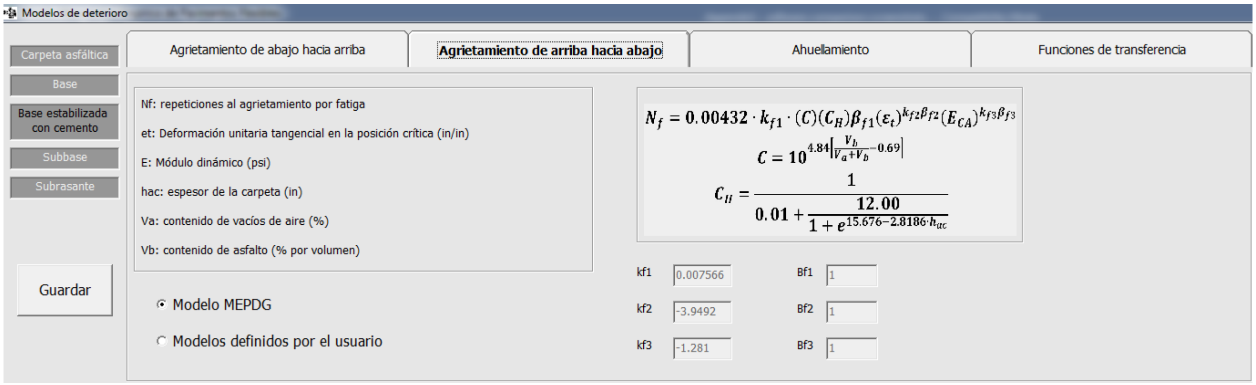
$$C_H = \frac{1}{0.000398 + \frac{0.003602}{1+e^{(11.02-3.49\cdot h_{ac})}}}$$

● Modelo MEPDG

○ Modelos definidos por el usuario

kf1	0.007566	Bf1	1
kf2	-3.9492	Bf2	1
kf3	-1.281	Bf3	1

Figure D-20: CR-ME's asphalt materials model selection - bottom-up cracking


Modelos de deterioro
Carpeta asfáltica
Base
Base estabilizada con cemento
Subbase
Subrasante
Guardar
Agrietamiento de abajo hacia arriba
Agrietamiento de arriba hacia abajo
Ahuellamiento
Funciones de transferencia

Nf: repeticiones al agrietamiento por fatiga

et: Deformación unitaria tangencial en la posición crítica (in/in)

E: Módulo dinámico (psi)

hac: espesor de la carpeta (in)

Va: contenido de vacíos de aire (%)

Vb: contenido de asfalto (% por volumen)

$$N_f = 0.00432 \cdot k_{f1} \cdot (C)(C_H)\beta_{f1}(\varepsilon_t)^{k_{f2}\beta_{f2}}(E_{CA})^{k_{f3}\beta_{f3}}$$

$$C = 10^{4.84\left[\frac{V_b}{V_a+V_b}-0.69\right]}$$

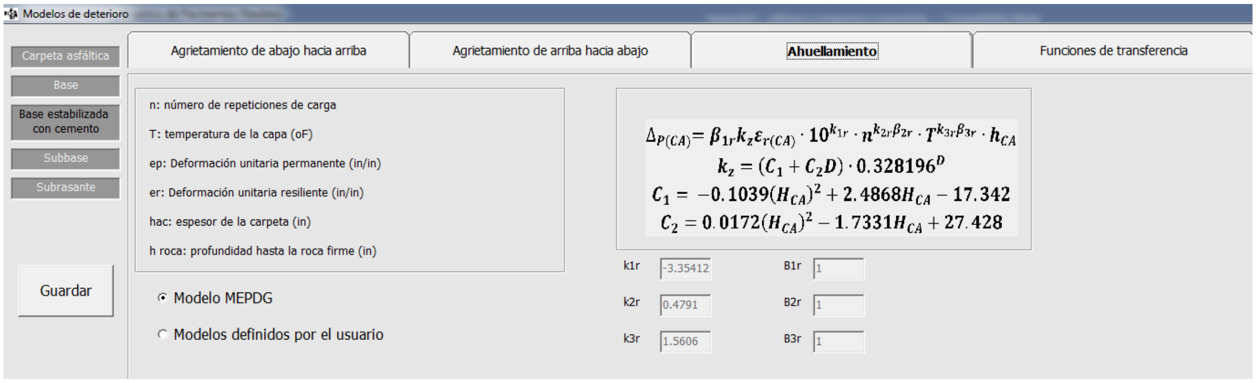
$$C_H = \frac{1}{0.01 + \frac{12.00}{1+e^{15.676-2.8186\cdot h_{ac}}}}$$

● Modelo MEPDG

○ Modelos definidos por el usuario

kf1	0.007566	Bf1	1
kf2	-3.9492	Bf2	1
kf3	-1.281	Bf3	1

Figure D-21: CR-ME's asphalt materials model selection - top-down cracking


Modelos de deterioro
Carpeta asfáltica
Base
Base estabilizada con cemento
Subbase
Subrasante
Guardar
Agrietamiento de abajo hacia arriba
Agrietamiento de arriba hacia abajo
Ahuellamiento
Funciones de transferencia

n: número de repeticiones de carga

T: temperatura de la capa (oF)

ep: Deformación unitaria permanente (in/in)

er: Deformación unitaria resiliente (in/in)

hac: espesor de la carpeta (in)

h roca: profundidad hasta la roca firme (in)

$$\Delta_{P(CA)} = \beta_{1r}k_z\varepsilon_{r(CA)} \cdot 10^{k_{1r}} \cdot n^{k_{2r}\beta_{2r}} \cdot T^{k_{3r}\beta_{3r}} \cdot h_{CA}$$

$$k_z = (C_1 + C_2D) \cdot 0.328196^D$$

$$C_1 = -0.1039(H_{CA})^2 + 2.4868H_{CA} - 17.342$$

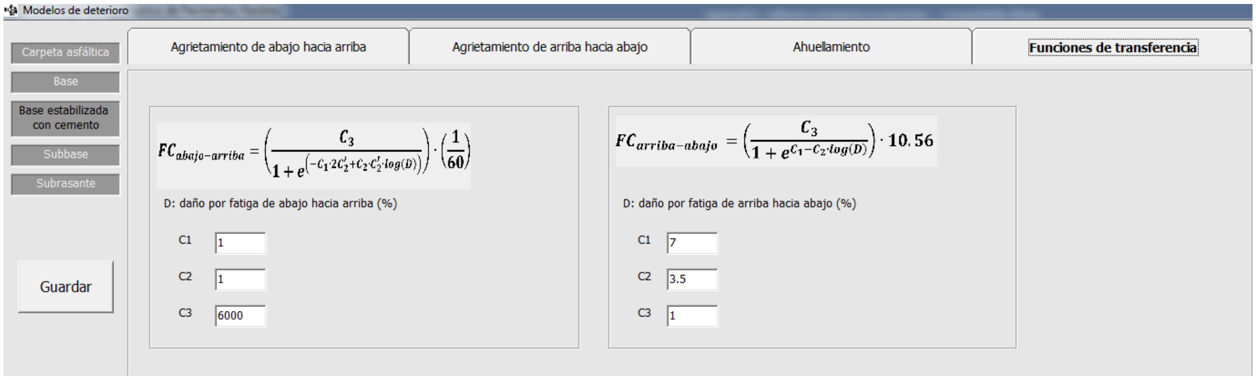
$$C_2 = 0.0172(H_{CA})^2 - 1.7331H_{CA} + 27.428$$

● Modelo MEPDG

○ Modelos definidos por el usuario

k1r	-3.35412	B1r	1
k2r	0.4791	B2r	1
k3r	1.5606	B3r	1

Figure D-22: CR-ME's asphalt materials model selection – rut depth prediction


Modelos de deterioro
Carpeta asfáltica
Base
Base estabilizada con cemento
Subbase
Subrasante
Guardar
Agrietamiento de abajo hacia arriba
Agrietamiento de arriba hacia abajo
Ahuellamiento
Funciones de transferencia

$$FC_{abajo-arriba} = \left(\frac{C_3}{1+e^{\left(-C_1 \cdot 2C_2' + C_2 \cdot C_2' \cdot log(D)\right)}}\right) \cdot \left(\frac{1}{60}\right)$$

D: daño por fatiga de abajo hacia arriba (%)

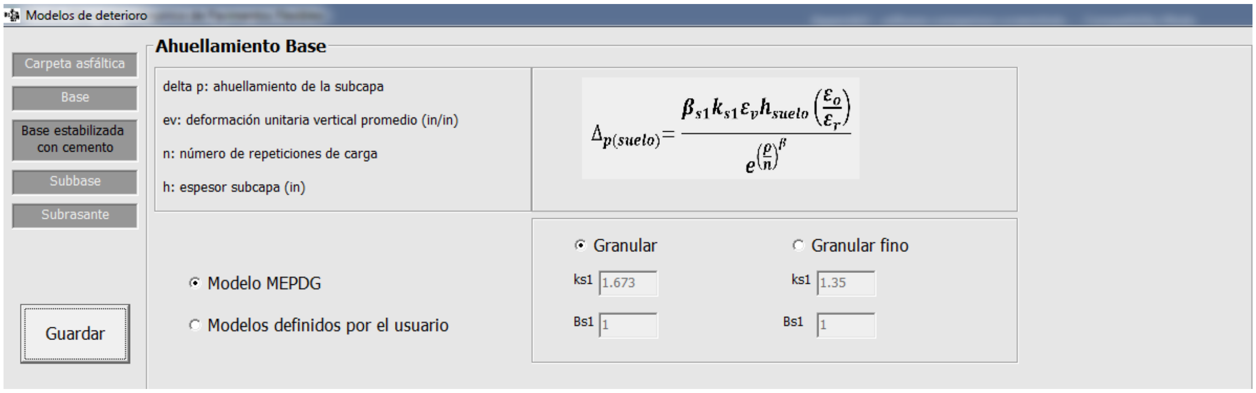
C1	1
C2	1
C3	6000

$$FC_{arriba-abajo} = \left(\frac{C_3}{1+e^{C_1 - C_2 \cdot log(D)}}\right) \cdot 10.56$$

D: daño por fatiga de arriba hacia abajo (%)

C1	7
C2	3.5
C3	1

Figure D-23: CR-ME's granular materials (base and sub-base) model selection – rut depth prediction



Ahuellamiento Base

delta p: ahuellamiento de la subcapa

ev: deformación unitaria vertical promedio (in/in)

n: número de repeticiones de carga

h: espesor subcapa (in)

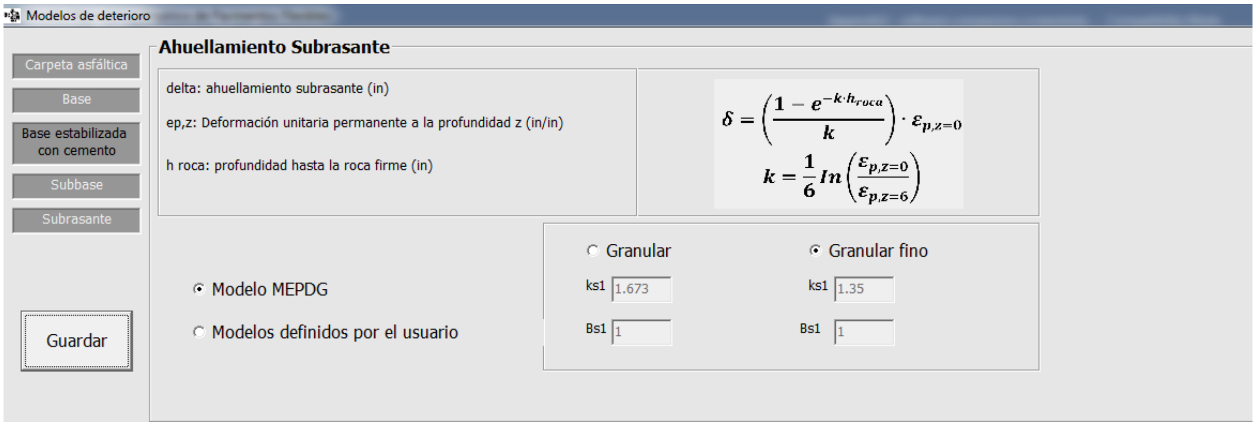
$$\Delta_{p(suelo)} = \frac{\beta_{s1} k_{s1} \varepsilon_v h_{suelo} \left(\frac{\varepsilon_o}{\varepsilon_r}\right)}{e^{\left(\frac{\rho}{n}\right)^{\beta}}}$$

● Modelo MEPDG

○ Modelos definidos por el usuario

● Granular		○ Granular fino	
ks1	1.673	ks1	1.35
Bs1	1	Bs1	1

Figure D-24: CR-ME's asphalt materials model selection – transfer functions for fatigue cracking (top-down and bottom-up)



Ahuellamiento Subrasante

delta: ahuellamiento subrasante (in)

ep,z: Deformación unitaria permanente a la profundidad z (in/in)

h roca: profundidad hasta la roca firme (in)

$$\delta = \left(\frac{1 - e^{-k \cdot h_{roca}}}{k}\right) \cdot \varepsilon_{p,z=0}$$

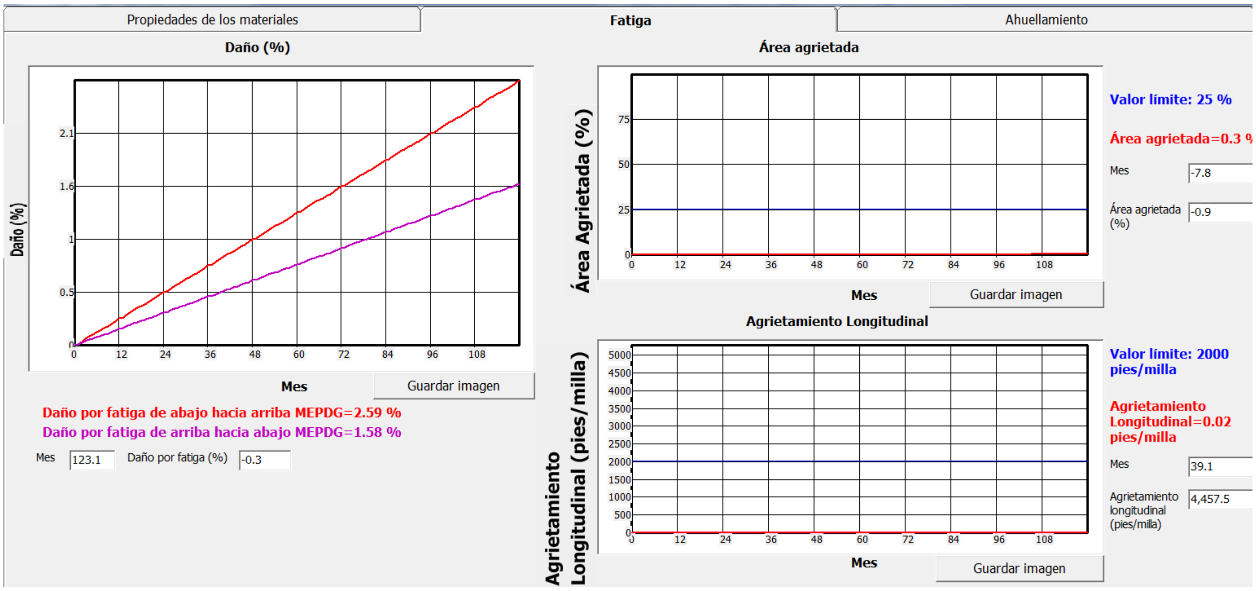
$$k = \frac{1}{6} In\left(\frac{\varepsilon_{p,z=0}}{\varepsilon_{p,z=6}}\right)$$

● Modelo MEPDG

○ Modelos definidos por el usuario

○ Granular		● Granular fino	
ks1	1.673	ks1	1.35
Bs1	1	Bs1	1

Figure D-25: CR-ME's subgrade model selection – rut depth prediction



Daño por fatiga de abajo hacia arriba MEPDG=2.59 %

Daño por fatiga de arriba hacia abajo MEPDG=1.58 %

Mes 123.1 Daño por fatiga (%) -0.3

Valor límite: 25 %

Área agrietada=0.3 %

Mes -7.8

Área agrietada (%) -0.9

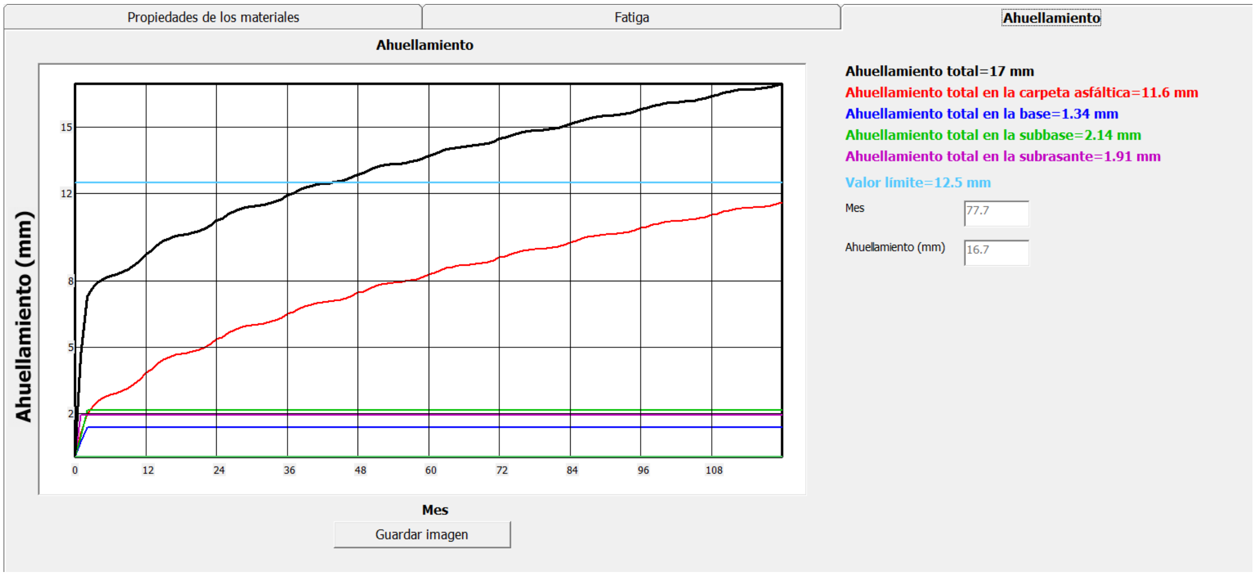
Valor límite: 2000 pies/milla

Agrietamiento Longitudinal=0.02 pies/milla

Mes 39.1

Agrietamiento longitudinal (pies/milla) 4,457.5

Figure D-26: CR-ME's output screens. Alligator and top-down cracking predictions



Ahuellamiento total=17 mm

Ahuellamiento total en la carpeta asfáltica=11.6 mm

Ahuellamiento total en la base=1.34 mm

Ahuellamiento total en la subbase=2.14 mm

Ahuellamiento total en la subrasante=1.91 mm

Valor límite=12.5 mm

Mes 77.7

Ahuellamiento (mm) 16.7

Figure D-27: CR-ME's output screens. Rut depth predictions

APPENDIX E – PRODUCT-ONE’S SOURCE CODE

This Appendix is a compendium of all the Matlab code that has been written when developing Product-One. Table E-1 serves as a summary of all the code files contained within this Appendix, highlighting the location of each within Product-One’s work folder, their inputs and outputs. The scripts are marked as “scriptName.m”, whereas the functions are tagged by their name only. The code to each script and function file has been copied under separate headings.

Table E-1:List of Product-One’s scripts and functions

Name	Location	Category	Description	Input	Output
MainCode.m	/	front-end	Front-end script. Interaction with user	-	-
Integrity Check.m	/	front-end	Add paths to all scripts and functions. Check whether Product-One is running on Matlab or GNU-Octave	-	-
Workspace Cleanup.m	/	front-end	Remove unnecessary variables used in calculations	-	-
shortTimestamp	/	pre-processing	Convert 1-hour time vector to 12-hour time vector	1-hour time stamp vector	12-hour time stamp vector
avgDown	/	pre-processing	Reduce a 1-hour time series to 12-hour time series	1-hr timestamp vector, 12-hr timestamp vector, 1-hr variable	12-hr variable
Climate Simulator	/climate	pre-processing	Create level-2 weather data for project site from IDW interpolation	Project location (X, Y coordinates); INIA weather records	Interpolated weather variables for 2010-2018 period
Climate Scrambler	/climate	pre-processing	Create predicted weather record from 2010-2018 record	2010-2018 weather variables (local or interpolated), weather random seed	Future weather variables for design-life period
climateModule	/climate	Pav. Perf. Sim.	Time-dependent climate effects module for the pav. Perf. Simulation. Compute infiltration and moisture content in unbound layers	Materials’ thicknesses, moisture, k_{unsat}, alligator cracking, rainfall	12-hr Infiltration and runoff; moisture content at each layer
Climate Load Local	/dataImporting	Input data import	Load Level-1 climate data from **dataImport** spreadsheet	-	Level-1 climate records for 2010-2018 period
Main Data import	/dataImporting	Input data import	Load project data from **dataImport** spreadsheet	-	Pav. Structure and materials, design life
Traffic Data importer	/dataImporting	Input data import	Import traffic and load data from **dataImport**	-	AADT for base year, traffic distribution factors, load record
Data Export	/dataImporting	Output data export	Send calculation results to spreadsheet	Predicted climate variables, predicted traffic, calc. HMA temperature, calc unbound mat. MR, distresses , spreadsheet template and name	Spreadsheet output
AlligatorCrack frontEnd.m	/distress Calculator	Pav. Perf. Sim.	Front-end to alligator cracking and top-down cracking calculations. Get results from ***alligatorCracking*** and	Horizontal strain for each axle type, HMA layers properties and thicknesses	Alligator-cracking and top-down cracking prediction at each 12-hr interval

Name	Location	Category	Description	Input	Output
			topDownCracking for different axle types		
Alligator Cracking	/distress Calculator	Pav. Perf. Sim.	Compute alligator cracking damage for given axle type	Horizontal strain for given axle type, HMA layers properties and thicknesses	Alligator cracking damage for given axle type
IRIclimate Sitefactor	/distress Calculator	Pav. Perf. Sim.	Get the site factor for the MEPDG IRI formula	Rainfall, temperature, subgrade’s P_{200}	IRI site factor, used in ***IRIPSICalcFrontEnd***
IRIPSI Calc FrontEnd.m	/distress Calculator	Pav. Perf. Sim.	Compute IRI and PSI for given distress level	Amount of distress at each 12-hr timestep	IRI & PSI values at each 12-hr timestep
RutDepthCalc FrontEnd.m	/distress Calculator	Pav. Perf. Sim.	Front-end to rut-depth calculations. Get results from **rutDepthCompute** for different axle types	Vertical strain for each axle type, pav. layers and subgrade’s properties and thicknesses	Rut depth at each layer and subgrade
RutDepth Compute	/distress Calculator	Pav. Perf. Sim.	Calculate rut of depth for a given axle type and load level	Horizontal strain for given axle and load, HMA layers properties and thicknesses	Rut depth at each time-stamp for given axle type and load level.
TopDown Cracking	/distress Calculator	Pav. Perf. Sim.	Compute top-down cracking damage for given axle type	Horizontal strain for given axle type, HMA layers properties and thicknesses	Alligator cracking damage for given axle type
getAuxEfromE	/elasticLinear analysis	Pav. Perf. Sim.	Generate E, ν series for all the MLE tress and strain calculation points	Pavement thicknesses, E and ν values	E and ν vector for all MLE points
Get Zfrom pavedepth	/elasticLinear analysis	Pav. Perf. Sim.	Create vector with all the vertical locations of the MLE stress and strain calculation points.	Pavement thicknesses	“z” positions
MLE_bc	/elasticLinear analysis	Pav. Perf. Sim.	MLE calculation core, solve the boundary conditions at each pavement layer interface for the stress and strain calculation. Ref Huang (2004)	MLE problem	Boundary conditions parameter
MLE_sigma	/elasticLinear analysis	Pav. Perf. Sim.	MLE calculation core: solve vertical, radial, and tangential stresses at a pavement structure for a single-wheel load	Pavement thicknesses, E and ν, load magnitude, vector of planar and vertical positions of calculation points	Values of σz σr σt at each vertical and planar point
MLEFrontEnd.m	/elasticLinear analysis	Pav. Perf. Sim.	MLE calculation core: solve the stresses and strains over the pavement structure for the different axle types and load levels	-	Stresses and strains by each axle type and load magnitude
sin18	/materials	pre-processing	Auxilary function to Bells2 equations for HMA temperature	time	Value of sin18(t), as in FHWA (2000)
vaporPressure	/materials	pre-processing	Compute vapor pressure for given air temperature	Air temperature series	Vapor pressure series
CalculateMR	/materials	Pav. Perf. Sim.	Compute the resilient modulus of the unbound materials at any 12-hr time step	MR of unbound materials at optimum compaction, material’s moisture content at 12-hr timestep	MR at 12-hr timestep
HMALayers Temperature	/materials	pre-processing	Compute the temperature of the HMA surface and HMA layers from atmospheric variables	Predicted weather record	Predicted temperature of HMA surface and mid-point of each HMA layer

Name	Location	Category	Description	Input	Output
HMA Modulus	/materials	pre-processing	Calculate the dynamic modulus (E*) of the HMA layers for each temperature and load level [Asph. Institute equation]	HMA layers temperature, P200, binder content, air voids, penetration at 77F. Load frequency for each axle type	E* for each 12-hr timestep for each axle type
HMA Poisson	/materials	pre-processing	Calculate the Poisson coefficient ν for the HMA layers for each temperature and load frequency value [MEPDG equation]	E* for the HMA layers for each temperature value and load frequency	Poisson coefficient ν for the asphalt layers
kUnsatSWCC	/materials	pre-processing	Compute the SWCC curve (Fredlund et al., 1994) parameters for unbound materials and the subgrade, and k_{unsat} vs suction curve	SWCC parameters from ***dataImport***	K_{unsat} vs matric suction and moisture content curves for unbound materials and subgrade
Materials Parameters Lvl2.m	/materials	Input data import	Import all material's properties from ***dataImport*** for the pavement layers	-	Inputs for E* for asphalt materials, MR, ν and SWCC parameters for unbound materials and subgrade
Temperature PreviousDay	/materials	pre-processing	Auxiliary function to the Bells 2 equation: get the air temperature for the day before the calculation time	-	-
plotAlligator	/plottingTools	Output data export	Plotting tool: plot predicted alligator cracking for all asphalt layers	Asphalt layers alligator cracking prediction [% area]	-
PlotClimate Variables	/plottingTools	Output data export	Plotting tool: plot all predicted weather variables	Predicted air temperature, humidity, rainfall, wind speed, and solar radiation	-
PlotHMA temperature	/plottingTools	Output data export	Plotting tool: plot the predicted temperature at the HMA layers	Air temperature and HMA layers temperature recodrds	-
plotIRIPSI	/plottingTools	Output data export	Plotting tool: plot the predicted IRI and PSI trends	Predicted IRI and PSI	-
plotMR	/plottingTools	Output data export	Plotting tool: plot the resilient modulus of the unbound materials	Predicted MR for the unbound materials and the subgrade	-
plotRutDepth	/plottingTools	Output data export	Plotting tool: plot the depth of rutting at all pavement layers	Rut depth prediction for all layers and the subgrade	-
plotSoilHydraulics	/plottingTools	Output data export	Plotting tool: plot the moisture content (volumetric) of all unbound materials and the infiltration and runoff at the surface	Predicted rainfall, infiltration and runoff at the surface, predicted moisture content at all layers	-
plotTrafficData	/plottingTools	Output data export	Plotting tool: plot the predicted AADT for all MTOP vehicle categories	Predicted AADT for all MTOP vehicle categories	-
AxleSpeedFrom VehicleSpeed	/trafficProcessing	pre-processing	Compute the (avg) speed of each axle type from the speed of all vehicle types (MTOP categories)	Vehicle speed record for all MTOP categories	Axle speed for all axle types
AxlesWeights.m	/trafficProcessing	pre-processing	Script containing the load ranges for the different axle configurations	-	Load ranges for all axle types
trafficSimulator	/trafficProcessing	pre-processing	Compute the number of	Base year AADT, growth	12-hr traffic prediction for

Name	Location	Category	Description	Input	Output
			vehicles by category for every 12-hr timestep	rate, traffic distribution factors for every category	all vehicle categories.
trafficToAxles	/trafficProcessing	pre-processing	Convert the number of vehicles to axle passes for every 12-hour time step. Distribute over load range.	Predicted 12-hour vehicle count by MTOP category	Predicted number of passes of each axle type, by weight, every 12 hours
wheelFootprint	/trafficProcessing	pre-processing	Determine the radius of the circular footprint of each axle type and load level	Load range for each axle type	Radius of wheel footprint for each axle type and load value

Source Code: mainCode.m

```
%%%% - - - - - PRODUCT ONE - - - - %%%%
%%%% - M.E. Pavement Design tool - %%%%
%%%% - - - MAIN ROUTINE CODE - - - %%%%
%
% Version 0.5 Many changes everywhere!     - 2019-05-20
% Version 0.3 MR Routine added             - 2019-01-22
% Version 0.2 Win-Friendly/Octave-Friendly - 2018-10-19
%%
%% ------INITIALIZATION------------
%
clear variables
clc
%%
disp('Starting PRODUCT-ONE M.E PAVEMENT DESIGN TOOL')
%%CODE INTEGRITY CHECK
%%Run a subscript to check if we have the necessary input files, addpaths
%%and (octave only) install I/O package should it not be there
%%<<integrityCheck.m is located in the program's main folder!>>
run integrityCheck.m;
%%%<< the isThisOctave boolean variable is to be used to check if you are
%%%on Matlab or GNU-Octave>>
%
%Ask the user the name of the XLS/XLSX file to read the inputs from.
dataImport = input('Specify the name of the XLS/XLSX file to read the inputs
from (between single quot. marks)...  ');
%
%% ------SETTINGS READ-------------
%--Read the config. settings---
%%(should the code run silently/verbose; should the code provide output--

disp('starting design problem data import')
if isThisOctave
    run mainDataImportOCT.m
else
    run mainDataImportXLS.m
end

%% --------DATA IMPORT-------------
%--Bring all the needed imports from the spreadsheets--
%
%1 - Project location
%2 - Trial pav. structure and subgrade
%3 - Site climate
%%>>MOVED TO mainDataImport script
%4 - Traffic
%Bring the matrix of all axes passes.
%A) import the traffic count from dataImport
if runVerbose
    disp('importing traffic and weight data')
end
```

```
if isThisOctave  %running Octave - use the odsRead importer for traffic
    run trafficDataImporterOCT.m
else  %running Matlab - use the xlsRead importer for traffic
    run trafficDataImporterXLS.m
end
disp('Input data imported successfully')

run 'axlesWeights.m';  %script with the vectors having the load ranges for
each type of axle  << located in /trafficProcessing folder>>

%% --Initialize Materials' Properties
if paveLevelInput == 1
  %read from an auxiliary file the many inputs   --> auxiliary script to do
the import.

  %%<<the materialsParametersLvlX.m scritps are in /materials folder>>
  run 'materialsParametersLvl1.m'
elseif paveLevelInput ==2
  run 'materialsParametersLvl2.m'
elseif paveLevelInput ==3
  run 'materialsParametersLvl3.m'
else
  error('Illegal level of input for pavement materials. Stopping Execution!')
end

%% ------
disp('Program ready to run')
disp('simulation timestep = 12h')   %%UPDATE JUN 18/2018 - - PAVEMENT PERFOR-
MANCE SIMULATION WILL RUN ON A 12-h cycle (FIXED AT 6AM-6PM)
fprintf('Project"s design life is %g years\n', designLife)

%% -----INPUT PRE-PROCESSING 01 - GENERATION OF SIMULATED CLIMATE
TIMESERIES-----
%- RUN THE climateSimulator.m FILE TO GENERATE THE SITE'S VIRTUAL WEATHER STA-
TION

if climateLevel1
  %This is the case I have climate records from the very site of the pavement
project
  %Create the script (use the simulator and remove the interpolation code)
  if runVerbose
      disp('Simulation will use local climate input [level 1]')
  end
  %%<climateLoadLocalXXX arelocated in /dataImporting/ folder
  if isThisOctave
      climateData = climateLoadLocalOCT(dataImport,runVerbose);
  else
      climateData = climateLoadLocalXLS(dataImport,runVerbose);
  end
else
  %This is the case I don't have local records, I rely on INIA DB.
  if runVerbose
      disp('Simulation will use INIA Data for climate inputs [level 2]')
  end
  %%<climateSimulator is located in /climate/ folder
  climateData = climateSimulator(locX,locY,runVerbose,climateLvl2ForceRecalc);
end

%UPDATE 2018-09-05 - - now that I have the raw climate data, use the climate scrambler to mix ehter Lvl1 and LVl2 site's climate.
%%<<the ClimateScrambler script is in /climate folder>
climateData = climateScrambler(climateData,climateRandomSeed,designLife,startTimeStamp,runVerbose);

%Break climateData by columns (all the needed variables):: climateMatrix = [timestamp temp hum wspd rain srad];
longTimestamp = climateData(:,1); %temporal calculation step = 1 hour (calculated in the climateSimulator)
shortTimestamp = shortTimestamp(longTimestamp);  %auxiliary function that will create a "short" timestamp vector (giving a value every day at 6AM and 6PM - start of daytime and nighttime)
airTemp = climateData(:,2);
humidity = climateData(:,3);
windSpd = climateData(:,4);
rainFall = climateData(:,5);
sunRad = climateData(:,6);
clear climateData;
%% -----INPUT PRE-PROCESSING 02 - CALCULATE HMA TEMPERATURE FROM CLIMATE DATA
%Update 2018-06-20 - move the HMA temperature calculations here, adapt the
%prev. generated code to give output in hourly and 1/2-day wide time-series

if runVerbose
    disp('Climate/Materials data processing: Calculating temperature in HMA Layers')
end
%%<<HMALayersTemperature are in the /materials folder
asphLyrTemp = HMALayersTemperature(longTimestamp,airTemp,humidity,windSpd,sunRad,ACPlacementTemp,ACPaveDepth,asphAbsorvipity,asphEmissivity,asphThermalConductivity,runVerbose);   %Calculate hourly temperature in AC layers
%%<<avgDown is in the program's root folder
shortAsphLyrTemp = avgDown(asphLyrTemp,longTimestamp,shortTimestamp);

%% -----INPUT PRE-PROCESSING 02 - CALCULATE GRANULAR BASES SWCC AND DARCY PERMEABILITY FUNCTIONS
%%Update 2018-10-05 - Adopt the SWCC equation in the MEPDG (Xi et al, 1994)
%%and use it to get the material's hydraulic conductivity at any degree of
%%moisture (as in Fredlund et al, 1994).

if runVerbose
    disp('Climate/Materials data processing: Calculating SWCC and k_unSat for granular materials')
end
%%<< the kUnsatSWCC script is in the /materials folder
%%note: granSWCCParameters is defined in the "materialsParametersLvlXX.m" script
granularKSWCC = kUnsatSWCC(granSWCCParameters,runVerbose);
```

```
%% -----INPUT PRE-PREOCESSING 03 - IMPORT TRAFFIC DATA AND CONSTRUCT THE 1/2-
DAY-LONG TRAFFIC SIMULATION.
% UPDATE 2018-06-20::  Use the shortTimetamp vector - as defined at the pre-
processing 02 routines
%%<<these scripts are in the /trafficProcessing folder.

[designTraffic,designAADT] =
trafficSimulator(shortTimestamp,AADTbaseYear,AADTGrowthRate,AADTMonthlyDistr,A
ADTDailyDistr,AADTHourlyDistr);
%output of this sub-routine: vectors having hourly passes of each type of each
vehicle category.

[axlesSingleLight, axlesSingle6, axlesSingle105, axlesTandem18, axlesTandem10,
axlesTandem14, axlesTridem] =
trafficToAxes(designTraffic,trafficLoadPerc,trafficAxleLoad);
%output of this second sub-routine: axles by weight category (0.5-ton sensi-
tivensess) per hour
%column vectors, the number of rows is the length of the timeStamp vector


%% -----PREPARE THE VARIABLES FOR THE SIMULATION-----
%APPROACH: ALL THE CLIMATE CALCULATIONS WILL BE "DETERMINISTIC" USING THE SIM-
ULATED climateData
%This will deviate from original EICM framework (in which rainfall and infil-
tration to subbase was probability-based
%And I will also use actual sunRadiation instead of the equations in RD-90-033

termination = length(shortTimestamp);

%INITIALIZE VARIABLES - Climate and granulars' hydraulic proprieties
surfaceRunoff = zeros(termination,1);           %record how much rainfall may
not enter the pavement and keep running.
surfaceInfiltration = zeros(termination,1);     %record how much rainfall may
enter the pavement and infiltrate downward (these 3 variables will evolve over
time as cracking increases, can't calculate them now)
layersMoisture = zeros(termination,paveLayersNumber-ACLayersNumber+1);  %
%moisture content on each layer PLUS THE SUBGRADE! (percent)
lateralRunoff = layersMoisture;                 %record how much water escapes
through the layers as sideways flow (m3/m2)
layersMoisture(1,:) = granHumidity(:,1)';       %initialize values in lay-
ersMoisture
shortRainfall = zeros(termination,1);           %I will store the accum. rain-
fall in short-timestamp format. Used at the end for plotting

%INITIALIZE VARIABLES - Stress-strain modeling // HMA Layers dynamic modulus
E*
%%Note: E* for each axle category is a 3-d array, each layer represents a
timestamp
if runVerbose
   disp('Pavement materials preprocessing: Calculating HMA Dynamic Modulus')
end
%:0 Get the speed of each axle type
```

```
axlesSpeed = axleSpeedFromVehicleSpeed(AADTSpeed); %this little function will convert the speed input by vehicle cat to the speed of each axle type (averaging the speeds of all vehicles that have each axle type)

%define this auxiliary point in granMR to prevent matlab from crashing when running
auxLGMR = length(granMR);

%1: Load freq, radius of load, and HMA E* for light single axles
aSingleLight = wheelFootprint(axlesSingleLWeights,2,30);
EdynAux = HMAModulus(shortAsphLyrTemp(6,:),HMAparameters,10^3.*ones(ACLayersNumber,length(aSingleLight)),paveLevelInput,0);   %%auxiliary E* calcuation, needed to get a reference E* value to compute the frequency of load with the Odemark's approach.
[timeOfLoadSL, freqOfLoadSL] = HMALoadFrequency(ACPaveDepth,AADTSpeed(1),aSingleLight,EdynAux,granMR(auxLGMR));
   %%Update 2019-02-17. Correct bug in HMALoadFrequency -> needs a preliminary value of HMA's E* and subgrade MR to compute the effective length of load
EDynSingleLight=
HMAModulus(shortAsphLyrTemp,HMAparameters,freqOfLoadSL,paveLevelInput,runVerbose);   %Create additional function to calculate E* for the light axles
HMAPoissonSingleLight = HMAPoisson(EDynSingleLight,paveLevelInput,runVerbose);

%2: Load freq, radius of load, and HMA E* for 6ton single axles
aSingle6 = wheelFootprint(axlesSingle6Weights,2,80);
EdynAux = HMAModulus(shortAsphLyrTemp(6,:),HMAparameters,10^3.*ones(ACLayersNumber,length(aSingle6)),paveLevelInput,0);   %%auxiliary E* calcuation, needed to get a reference E* value to compute the frequency of load with the Odemark's approach.
[timeOfLoadS6, freqOfLoadS6] = HMALoadFrequency(ACPaveDepth,AADTSpeed(2),aSingle6,EdynAux,granMR(auxLGMR));
EDynSingle6=
HMAModulus(shortAsphLyrTemp,HMAparameters,freqOfLoadS6,paveLevelInput,runVerbose);  %Create additional function to calculate E* for the 6-ton single axles
HMAPoissonSingle6 = HMAPoisson(EDynSingle6,paveLevelInput,runVerbose);

%3: Load freq, radius of load, and HMA E* for 10.5ton single axles
aSingle105 = wheelFootprint(axlesSingle10Weights,4,90);
EdynAux = HMAModulus(shortAsphLyrTemp(6,:),HMAparameters,10^3.*ones(ACLayersNumber,length(aSingle105)),paveLevelInput,0);   %%auxiliary E* calcuation, needed to get a reference E* value to compute the frequency of load with the Odemark's approach.
[timeOfLoadS105, freqOfLoadS105] =
HMALoadFrequency(ACPaveDepth,AADTSpeed(3),aSingle105,EdynAux,granMR(auxLGMR));
EDynSingle105=
HMAModulus(shortAsphLyrTemp,HMAparameters,freqOfLoadS105,paveLevelInput,runVerbose);   %Create additional function to calculate E* for the 10.5-ton single axles
HMAPoissonSingle105 = HMAPoisson(EDynSingle105,paveLevelInput,runVerbose);

%4: Load freq, radius of load, and HMA E* for 18ton tandem axles
aTandem18 = wheelFootprint(axlesTandemWeights,8,90);
EdynAux = HMAModulus(shortAsphLyrTemp(6,:),HMAparameters,10^3.*ones(ACLayersNumber,length(aTandem18)),paveLevelInput,0);   %%auxiliary E* calcuation,
needed to get a reference E* value to compute the frequency of load with the
Odemark's approach.
[timeOfLoadT18, freqOfLoadT18] =
HMALoadFrequency(ACPaveDepth,AADTSpeed(4),aTandem18,EdynAux,granMR(auxLGMR));
EDynTandem18=
HMAModulus(shortAsphLyrTemp,HMAparameters,freqOfLoadT18,paveLevelInput,runVerbose);   %Create additional function to calculate E* for the 18-ton tandem
axles
HMAPoissonTandem18= HMAPoisson(EDynTandem18,paveLevelInput,runVerbose);

%5: Load freq, radius of load, and HMA E* for 14ton NH tandem axles
aTandem14 = wheelFootprint(axlesTandem14Weights,6,90);
EdynAux = HMAModulus(shortAsphLyrTemp(6,:),HMAparameters,10^3.*ones(ACLayersNumber,length(aTandem14)),paveLevelInput,0);   %%auxiliary E* calcuation,
needed to get a reference E* value to compute the frequency of load with the
Odemark's approach.
[timeOfLoadT14, freqOfLoadT14] =
HMALoadFrequency(ACPaveDepth,AADTSpeed(6),aTandem14,EdynAux,granMR(auxLGMR));
EDynTandem14=
HMAModulus(shortAsphLyrTemp,HMAparameters,freqOfLoadT14,paveLevelInput,runVerbose); %Create additional function to calculate E* for the 14-ton tandem NH
axles
HMAPoissonTandem14= HMAPoisson(EDynTandem14,paveLevelInput,runVerbose);

%6: Load freq, radius of load, and HMA E* for 10ton SW tandem axles
aTandem10 = wheelFootprint(axlesTandem10Weights,4,80);
EdynAux = HMAModulus(shortAsphLyrTemp(6,:),HMAparameters,10^3.*ones(ACLayersNumber,length(aTandem10)),paveLevelInput,0);   %%auxiliary E* calcuation,
needed to get a reference E* value to compute the frequency of load with the
Odemark's approach.
[timeOfLoadT10, freqOfLoadT10] =
HMALoadFrequency(ACPaveDepth,AADTSpeed(5),aTandem10,EdynAux,granMR(auxLGMR));
EDynTandem10=
HMAModulus(shortAsphLyrTemp,HMAparameters,freqOfLoadT10,paveLevelInput,runVerbose); %Create additional function to calculate E* for the 10-ton tandem axles
HMAPoissonTandem10= HMAPoisson(EDynTandem10,paveLevelInput,runVerbose);

%7: Load freq, radius of load, and HMA E* for 25ton tridem axles
aTridem = wheelFootprint(axlesTridemWeights,12,90);
EdynAux = HMAModulus(shortAsphLyrTemp(6,:),HMAparameters,10^3.*ones(ACLayersNumber,length(aTridem)),paveLevelInput,0);   %%auxiliary E* calcuation,
needed to get a reference E* value to compute the frequency of load with the
Odemark's approach.
[timeOfLoadTri, freqOfLoadTri] =
HMALoadFrequency(ACPaveDepth,AADTSpeed(7),aTridem,EdynAux,granMR(auxLGMR));
EDynTridem   =
HMAModulus(shortAsphLyrTemp,HMAparameters,freqOfLoadTri,paveLevelInput,runVerbose);  %Create additional function to calculate E* for the 22-ton tridem
axles
HMAPoissonTridem= HMAPoisson(EDynTridem,paveLevelInput,runVerbose);

%INITIALIZE VARIABLES - MR FOR THE GRANULAR LAYERS
```

```
MR = zeros(termination,1+paveLayersNumber-ACLayersNumber);  %prepare matrix
for the actual MR values. Each col. is the MR(t) for each granular layer PLUS
the sub-grade.

%INITIALIZE VARIABLES - distress
rutDepth=zeros(termination,1+length(paveDepths));           % for each layer,
and all summed up! metric Units [m]
alligatorCrack = zeros(termination,ACLayersNumber);   % units: percentage of
lane area
% topDownCrack = zeros(termination,ACLayersNumber);     % units: m/km
[MEPDG's are ft/mile]  update v2019-05-19, only 1 record of topDownCracking
(at the surface)
topDownCrack = zeros(termination,1);     % units: m/km  [MEPDG's are ft/mile]
% reflectiveCrack = zeros(termination,1);  % units: percentage of lane area
% transvCrack = zeros(termination,1);      % units: m/km - DISREGARDED AS
THESE ARE DUE TO COLD WEATHER AND WOULD'NT OCCUR IN A WARM CLIMATE (SHOWN BY
CALIENDO, 2012))

IRI = zeros(termination,1);                               % Metric Units [m/
km]
PSI = zeros(termination,1);              % units: adim. - - TBUsed for comparison
against AASHTO '93

%% -----RUN THE PAVEMENT SIMULATION-------------
for k = 1:termination
  currentTime = datevec(shortTimestamp(k));
  currentYear = currentTime(1);
  currentMonth = currentTime(2);
  currentDay = currentTime(3);
  currentHour = currentTime(4);
  if runVerbose
  %Report date being analyzed. Octave's DateNum and DateVec functions jump 1
unit each day!
    fprintf('Analyzing pavement condition on date %g - %g - %g at %g :00  --
%g percent completed. \n', currentYear, currentMonth, currentDay, currentHour,
k*100/termination)
  end

  %%firstly - run climateModule at time i
  %<<climateModule located in /climate folder
  run 'climateModule.m';

  %CALCULATE THE MATERIALS MR AT TIME k
  %use auxiliary "calculateMR" function  - - Stored in /materials folder
  MR(k,:) =
calculateMR(gran_ID,layersMoisture(k,:)',granMR,granHumidity(:,2),granHumidity(:,1));
  %function MRactual = calculateMR(materialID,moistureContent,MRSat,OptMistureSaturationLevel,OptMoisture in % vol)
  %note: layersMoisture is defined here, updated in the "climateModule.m"
%script
  %      and granMR, gran_ID, and granHumidity (opt. moisture content and
matching saturation) are defined in "materialsParametersLvlX.m script
```

```
  % I also need to transpose the actual humidity values to prevent the func-
tion from mismatching dimensions

  %% Compute stress and strains for each type of load application.
  %%Program a MLE front-end that will run the routines to calculate stresses
and strains for
  %%each type of axle (single/tandem/tridem) at time k and store the results
to be passed on to the distress module

  % MLEFrontEnd scrtipt is in elasticLinearAnalysis folder
  run 'MLEFrontEnd.m'

  if k==2
      disp('stop here!')
  end

   %% Calculate distress increase
  if runVerbose
  %Report date being analyzed. Octave's DateNum and DateVec functions jump 1
unit each day!
    fprintf(' \t Computing distress increase on date %g - %g - %g at %g :00 \
n', currentYear, currentMonth, currentDay, currentHour)
  end

  %Call the front-end calculator. It will compute all the distresses of inter-
est, IRI, and PSI at shortTimestamp t(k)
  run 'rutDepthCalcFrontEnd.m';
  run 'alligatorCalcFrontEnd.m';

  %%update v2019-04-04: Some rut-depth values hit the CMPLX plane at times.
  %force a Re(rutDepth)
  rutDepth(k,:) = real(rutDepth(k,:));

  run 'IRIPSICalcFrontEnd.m';%% IRI and PSI calculator need the k-th value of
all the distresses done, thus it must be called last!

end   %%end to time-dependent pavement performance Simulation!

%% -------CREATE FIGURES--------
%recall the "createFigures" config. variable
%value 1 = figures for climate
%value 2 = figures for traffic
%value 3 = figures for materials' properties
%value 4 = distress generation
%value 5 = reserved for future use
%%< the figure-creation tools are in the /plottingTools folder>

if createFigures(1) >0
    plotted15 =
plotClimateVariables(longTimestamp,airTemp,humidity,windSpd,sunRad,rainFall);
end
    %figures 21
if createFigures(2)>0
```

```
    [plotted21,simpleTrafficReport] = plotTrafficData(shortTimestamp,designAADT(2:end,:));  %%added correection cause 1st row of designAADT is the year
timestamp!
end
if createFigures(3)>0
    %figure 31, 32--
    plotted31 = plotHMATemperature(longTimestamp,airTemp,asphLyrTemp);
    plotted3233 = plotSoilHydraulics(shortTimestamp, shortRainfall,layersMoisture,surfaceInfiltration, surfaceRunoff);
    plotted34 = plotMR(shortTimestamp,MR);
end
if createFigures(4)>0
     plotted4142 = plotRutDepth(shortTimestamp,rutDepth);
     plotted4344 = plotAlligator(shortTimestamp,alligatorCrack,topDownCrack);
     plotted45 = plotIRIPSI(shortTimestamp,IRI,PSI);
end
% if createFigures(5)>0
%     %plotted = plot whatever else
% end

%check if any plot didn't take place
plotCheck =
plotted15*plotted31*plotted3233*plotted34*plotted4142*plotted4344*plotted45;
if runVerbose
    if plotCheck == 0
        disp('Uh-oh:: at least one of the plots did not generate')
        failyPlot = [plotted15 plotted31 plotted3233 plotted34 plotted4142
plotted4344 plotted45];
        auxPlotList = [string('figure 1-5'), string('Figure 31'), string('Figure 32-33'), string('Figure 34') string('Figure 4142') string('Figure4344')
string('Figure45')];
        fprintf('The plots in %s did not generate properly \
n',auxPlotList(failyPlot==0));

    else
        disp('All plots generated OK')
    end
end


%% ----NUMERICAL DATA EXPORT TO SPREADSHEETS----
%first bring the spreadsheet template to the working folder and rename it
%with the name given by the user

finalName = strcat(saveDataName,'.xlsx');
copyfile('./dataImporting/dataExportTemplate.xlsx',char(finalName))   %%<< debugged, 2019-02-05

%Program these modules
if isThisOctave
    run dataExportOCT.m
else
    run dataExportXLS.m
end
```

```
%% -------DATA EXPORT TO PDF OUTPUTS------------

%%----------AS OF FEB 22ND 2019, NOT INCLUDING PDF OUTPUT IN THIS
RELEASE!--------
%check the variable exportToPDF
%use the saveDataName name for the names of the output files

%syntax is >> export_fig FILENAME -pdf -append

%% ---CODE CLOSE-UP - CLEAN THE VARIABLES THAT AREN'T WORTHWHILE FROM THE
WORKSPACE---
doTheCleanup = input('Perform Workspace cleanup (remove unnecessary vari-
ables)? - - 1 means "Yes"........');
if doTheCleanup
    run workspaceCleanup.m
end

%update V2019-05-20: move save numerical output to the very end (after the
workspace cleanup)
disp('Pavement Simulation completed!')
fprintf('Saving numerical outputs to %s .mat \n',saveDataName)
finalName = sprintf('%s.mat',saveDataName);
save(finalName);

disp('----------All completed!-----------')
```

Source Code: integrityCheck.m

```
%%integrity check script to run upon start of the Product-one MainCode

%% - 1 check if I have the I/O functions -octave only- and install if needed
isThisOctave = exist('OCTAVE_VERSION') ~=0;
if isThisOctave
   disp('PRODUCT-ONE DETECTED YOU ARE RUNNING GNU-OCTAVE')
   more on         %%update 2019-02-11. See if this line forces Octave to
print the verbose output to screen in real-time...
   checkforIOPkg = exist('xlsread')==0;    %%if this hold true (the IO package
has not yet been installed/loaded), download and install pkg io
   if checkforIOPkg
       pkg load io   %%try this sentence in octave, see what's the outcome
when io is not installed (and will need to download)
       %Nevertheless, Octave version will use odsread instead for importing
files.
   end
else
     disp('PRODUCT-ONE DETECTED YOU ARE RUNNING MATLAB')
end

%% - 2 check if all the files to run are AVAILABLE
%%% in addpath

%% - 3 addpaths
addpath './plottingTools';  %path to codes for plotting
addpath './climate';  %path to the climate Module scripts
addpath './materials';%path to the materials props. scripts
addpath './trafficProcessing';
addpath './dataImporting';
addpath './elasticLinearAnalysis';
addpath './distressCalculator';
%addpath './export_fig';     %path to export_fig (a 3rd. party code that al-
lows to export graphs to pdf - uses ghostScript, it will perform integrity
checks itself
addpath './dataFiles';
```

Source Code: workspaceCleanup.m

```
%% PRODUCT-ONE PAVEMENT M-E DESIGN TOOL %%%
%Workspace clean-up script
%
%This script will clear out the not necessary varaiables and leave only those that have some sort of meaning.%
%V0.0 2019-03-05

%% code begins
%clear stuff from the export scripts
clear plotted15 plotted21 plotted31 plotted3233 plotted34 plotCheck auxPlotList failyPlot
clear rangeToExport
clear finalRowToExport
clear createFigures

%clear other main-code stuff
clear termination
clear currentDay currentHour currentMonth churrentYear
clear deltaTime
clear auxDate

%clear stuff from the MLE and Distress Calculator
clear r i j k
clear whereInTime
clear nax nrs10 nrs6 nrsL nrTa10 nrTa14 nrTa18 nrTr
clear c1p c2p
clear cummDamage
clear auxDamage
clear auxv_HMA
clear auxE_HMA
clear auxLayersE auxLayersv auxLGMR
clear auxR1 auxR2 auxR3 auxR4 auxR5 auxR6
clear auxAlpha1 auxAlpha2 auxAlpha3 auxAlpha4 auxAlpha5 auxAlpha6
clear auxSigmaR1 auxSigmaR2 auxSigmaR3 auxSigmaR4 auxSigmaR5 auxSigmaR6
clear auxSigmaT1 auxSigmaT2 auxSigmaT3 auxSigmaT4 auxSigmaT5 auxSigmaT6
clear auxSigmaX1 auxSigmaX2 auxSigmaX3 auxSigmaX4 auxSigmaX5 auxSigmaX6
clear auxSigmaY1 auxSigmaY2 auxSigmaY3 auxSigmaY4 auxSigmaY5 auxSigmaY6
clear auxSigmaZ1 auxSigmaZ2 auxSigmaZ3 auxSigmaZ4 auxSigmaZ5 auxSigmaZ6
clear auxTauXY1  auxTauXY2  auxTauXY3  auxTauXY4  auxTauXY5  auxTauXY6
clear aux
clear auxMaxRut auxMaxRutPosition
clear auxrutDepthGranSingleL auxrutDepthHMASingleL auxrutDepthSubGradeSingleL
clear auxrutDepthGranSingle6 auxrutDepthHMASingle6 auxrutDepthSubGradeSingle6
clear auxrutDepthGranSingle10 auxrutDepthHMASingle10 auxrutDepthSubGradeSingle10
clear auxrutDepthGranTandem10 auxrutDepthHMATandem10 auxrutDepthSubGradeTandem10
clear auxrutDepthGranTandem14 auxrutDepthHMATandem14 auxrutDepthSubGradeTandem14
clear auxrutDepthGranTandem18 auxrutDepthHMATandem18 auxrutDepthSubGradeTandem18
clear auxrutDepthGranTridem auxrutDepthHMATridem auxrutDepthSubGradeTridem

%clear stuff from the materials' module
clear previousHumidity
clear numLayersForMoisture
clear moistureLyrJaux
clear currentHumidity
clear downwardRunoff downwardRunoffAux
clear intervalInfiltration intervalRainfall
```

Source Code: avgDown function

```
function tempShort = avgDown(longTemp,longTimestamp,shortTimestamp)
%function tempShort = avgTemp(longTemp,longTimestamp,shortTimestamp)
%This auxiliary function will convert the hourly temperature record stored %in
longTemp (length matching longTimestamp) and calculate the average.

%Assuming all timestamp input vectors are column vectors; longTemp is a ma-
trix, code will perform the averaging over columns. Averages will be reported
at 6AM and 6PM.

nst = length(shortTimestamp);
[rous,cols] = size(longTemp);
tempShort = zeros(nst,cols);

%scan the short timeStamp vector, locate when each short timestamp occurs in
the long vector, and avg. them.
for i = 1:nst
    pos = find(longTimestamp == shortTimestamp(i));
    %now, pick the temeprature records in the longTemp series from which you
need the avg. value. Store in auxVec
    if pos <=11
        auxVec = longTemp(1:pos,:);
    else
        auxVec = longTemp(pos-11:pos,:);
    end
    %now do the avg and store in output
    tempShort(i,:) = mean(auxVec,1);   %perform the mean value for each column
(averages all rows of auxVec)

end


end
```

Source Code: /climate/climateSimulator function

```
function climateMatrix = climateSimulator(projectX,projectY,verbose,forceRecalcMatrix)
%function climateMatrix = climateSimulator(ProjectX,ProjectY,verbose,forceRecalcMatrix)
%
%%front-end function to develop the project site's simulated climate variables
%this function will work as follows
%1 - construct a virtual_ weather station climate record from the INIA db
(years 2011-2017 <future updates to the code may allow to extend the source
data as INIA collects more records>
%%Use Square Dist. interpolation to do so (Wei and McGuinees, 1973; referenced
in hydrology manuals, such as Chow et al., 1988)
%
%The "verbose" input allows to show on screen every calculation step. input 1
if you want to do so. If no input, it assumes zero
%NEW: The forceRecalcMatrix forces the code to generate a new site's climate
record from INIA DB regardless of the cached record being still useful.
%
%Updated Matching V2018-09-05:  Separate lvl2 climate interpolator from scrambler.
% New output will be the interpolated only if a previously executed cached
climateMatrix won't serve (it has been solved for a different place)

%% Preprocc. check if last-generated cached climate records matrix still works
(we are in the same projectX and projectY, and the cached record is the same
length!). If that's the case, spare us interpolating a new climate record.
% note - if forceRecalcMatrix == 1 , ignore matrix recycling, and
% recalculate all the same!

load 'climateMatrixCached.mat'
%the mat file loaded here contains:
%   oldX, oldY = X,Y position of the last project site
%   oldClimateMatrix = cached climate record from previous execution

if oldX == projectX && oldY == projectY == ~forceRecalcMatrix
    %case: don't recalculate the climate matrix, pass on the cached one!
    if verbose
        disp('climateSimulator:: using cached Level-2 site"s climate record')
        %this is the UPDATE 2018-09-05, recalculate the interpolated climate
record only if strictly necessary. (the interpolation
        %procedure takes a while)
    end
    climateMatrix = oldClimateMatrix;
else
    %ok, do a new matrix!. Proceed with the code below
    if verbose && forceRecalcMatrix
        disp('climateSimulator:: order to recalculate site"s climate record
received.')
    end
    %% Part1 - Bring the INIA weather records
    if verbose == 1
```

```
        disp('climateSimulator:: loading INIA Weather stations climate database')
    end
    load './dataFiles/INIAClimate.mat'
    %this .mat file contains structures with the hourly weather data from 8 INIA stations: %INIA_LE INIA_LB INIA_33 INIA_Tbo INIA_Dur INIA_Gle INIA_SG INIA_RO

    %% Part 2 - Create virtual station with distance square interpolation technique.  (Wei & McGuinness 1973)
    %Extract the interpolation factors
    xLB = INIA_LB(1).Location (1);
    yLB = INIA_LB(1).Location (2);
    distLB2 = (projectX - xLB)^2 + (projectY - yLB)^2 ;  %dist to Las Brujas

    xLE = INIA_LE(1).Location (1);
    yLE = INIA_LE(1).Location (2);
    distLE2 =  (projectX - xLE)^2 + (projectY - yLE)^2;  %dist to La Estanzuela

    xTbo = INIA_Tbo(1).Location (1);
    yTbo = INIA_Tbo(1).Location (2);
    distTbo2 = (projectX - xTbo)^2 +(projectY - yTbo)^2;  %dist to Tacuarembo

    xGle = INIA_Gle(1).Location (1);
    yGle = INIA_Gle(1).Location (2);
    distGle2 = (projectX - xGle)^2 +(projectY - yGle)^2;  %dist to Glencoe

    xDur = INIA_Dur(1).Location (1);
    yDur = INIA_Dur(1).Location (2);
    distDur2 = (projectX - xDur)^2 +(projectY - yDur)^2;  %dist to Durazno

    xRo = INIA_RO(1).Location (1);
    yRo = INIA_RO(1).Location (2);
    distRo2 =  (projectX - xRo)^2 + (projectY - yRo)^2;  %dist to Rocha

    xSg = INIA_SG(1).Location (1);
    ySg = INIA_SG(1).Location (2);
    distSG2 =  (projectX - xSg)^2 + (projectY - ySg)^2;  %dist to Salto Grande

    x33 = INIA_33(1).Location (1);
    y33 = INIA_33(1).Location (2);
    dist332 = (projectX - x33)^2 + (projectY - y33)^2;  %dist to Treinta y Tres

    %% Create the variables between 2010 and 2017; %Define timestamp range; %Use find function to locate the variables' values from each station for that timestamp %Q MEASURE against missing values; %construct the vectors.

    if verbose
        disp('climateSimulator:: initialize virtual weather station variables. Simulating 2010-2018 period')
    end
    timestamp = datenum(2010,1,1,0,0,0):1/24:datenum(2019,1,1,0,0,0);
    timestamp = timestamp';
    nle = length(timestamp);

    %the 5 climatic variables - initialize as 0, then I will fill the vectors
    temp = zeros(nle,1);     % interpolate point avg. temperatures [in deg. C]
    rain = zeros(nle,1);   % interpolate accum. rainfall     [in mm]
    wspd = zeros(nle,1);   % interpolate point wind speed    [in m/sec]
    srad = zeros(nle,1);   % interpolate accumulated sun radiation [in MJ/m2]
NEED TO CHECK MEPDG EICM GUIDE FOR USED UNITS...
    hum  = zeros(nle,1);   %interpolate point air humidity   [in %]

    disp('climateSimulator:: calculating virtual weather station variables')

    %% Part 3 - And here I fill up the variables. Use interpolation
    for i = 1:nle
        if verbose && i/1000 == round(i/1000)
            fprintf('climateSimulator:: Completed %g percent  \n', 100*i/nle);
%for Verbose mode, I will print on screen calculation progress every 1000
steps
        end

        %%variables from LE
        position = find(INIA_LE(1).timestamp == timestamp(i));
        %Gotta check that posLE returned something (i.e. LE has a non-NaN
weather record for that timestamp). In the NaN cases, Matlab will output a NaN
value in the design weather variable)
        %Default value (either no record, or record which is NaN
        tempLE = 0;
        rainLE = 0;
        wspdLE = 0;
        sradLE = 0;
        humLE =  0;
        binLE  = 0;   %This binLe term is a binary (==1 only if there is ac-
tual record of the 5 climate variables), I will use it to kill unnecessary
terms in the interpolation
        if ~isempty(position)   %if no record, all values set to 0,then the
interp must ignore them when doing the sum. Case there is a record at that
timestamp, bring in the values only if they are non-NaN
            if ~isnan(INIA_LE(1).temp(position))
                tempLE = INIA_LE(1).temp(position);
            end
            if ~isnan(INIA_LE(1).rain(position))
                rainLE = INIA_LE(1).rain(position);
            end
            if ~isnan(INIA_LE(1).wspd(position))
                wspdLE = INIA_LE(1).wspd(position);
            end
            if ~isnan(INIA_LE(1).srad(position))
                sradLE = INIA_LE(1).srad(position);
            end
            if ~isnan(INIA_LE(1).hum(position))
                humLE =  INIA_LE(1).hum(position);
            end
            if ~isnan(INIA_LE(1).temp(position)) && ~isnan(INIA_LE(1).rain(position)) && ~isnan(INIA_LE(1).wspd(position)) && ~isnan(INIA_LE(1).srad(position)) && ~isnan(INIA_LE(1).hum(position))
                binLE = 1;   %if there is a "valid" record for the 5 variables, utilize them (put the bin == 1)
            end
        end

        %%variables from LB
        position = find(INIA_LB(1).timestamp == timestamp(i));
        %Gotta check that posLB returned something (i.e. LB has a non-NaN weather record for that timestamp). In the NaN cases, Matlab will output a NaN value in the design weather variable)
        %Default value (either no record, or record which is NaN
        tempLB = 0;
        rainLB = 0;
        wspdLB = 0;
        sradLB = 0;
        humLB =  0;
        binLB  = 0;   %This binLB term is a binary (==1 only if there is actual record of the 5 climate variables), I will use it to kill unnecessary terms in the interpolation
        if ~isempty(position)   %if no record, all values set to 0,then the interp must ignore them when doing the sum. Case there is a record at that timestamp, bring in the values only if they are non-NaN
            if ~isnan(INIA_LB(1).temp(position))
                tempLB = INIA_LB(1).temp(position);
            end
            if ~isnan(INIA_LB(1).rain(position))
                rainLB = INIA_LB(1).rain(position);
            end
            if ~isnan(INIA_LB(1).wspd(position))
                wspdLB = INIA_LB(1).wspd(position);
            end
            if ~isnan(INIA_LB(1).srad(position))
                sradLB = INIA_LB(1).srad(position);
            end
            if ~isnan(INIA_LB(1).hum(position))
                humLB =  INIA_LB(1).hum(position);
            end
            if ~isnan(INIA_LB(1).temp(position)) && ~isnan(INIA_LB(1).rain(position)) && ~isnan(INIA_LB(1).wspd(position)) && ~isnan(INIA_LB(1).srad(position)) && ~isnan(INIA_LB(1).hum(position))
                binLB = 1;   %if there is a "valid" record for the 5 variables, utilize them (put the bin == 1)
            end
        end

        %%variables from Dur
        position = find(INIA_Dur(1).timestamp == timestamp(i));
        %Gotta check that posDur returned something (i.e. Dur has a non-NaN weather record for that timestamp). In the NaN cases, Matlab will output a NaN value in the design weather variable)
        %Default value (either no record, or record which is NaN
        tempDur = 0;
        rainDur = 0;
        wspdDur = 0;
        sradDur = 0;
        humDur =  0;
        binDur  = 0;   %This binDur term is a binary (==1 only if there is ac-
tual record of the 5 climate variables), I will use it to kill unnecessary
terms in the interpolation
        if ~isempty(position)   %if no record, all values set to 0,then the
interp must ignore them when doing the sum. Case there is a record at that
timestamp, bring in the values only if they are non-NaN
            if ~isnan(INIA_Dur(1).temp(position))
                tempDur = INIA_Dur(1).temp(position);
            end
            if ~isnan(INIA_Dur(1).rain(position))
                rainDur = INIA_Dur(1).rain(position);
            end
            if ~isnan(INIA_Dur(1).wspd(position))
                wspdDur = INIA_Dur(1).wspd(position);
            end
            if ~isnan(INIA_Dur(1).srad(position))
                sradDur = INIA_Dur(1).srad(position);
            end
            if ~isnan(INIA_Dur(1).hum(position))
                humDur =  INIA_Dur(1).hum(position);
            end
            if ~isnan(INIA_Dur(1).temp(position)) &&
~isnan(INIA_Dur(1).rain(position)) && ~isnan(INIA_Dur(1).wspd(position)) &&
~isnan(INIA_Dur(1).srad(position)) && ~isnan(INIA_Dur(1).hum(position))
                binDur = 1;   %if there is a "valid" record for the 5 vari-
ables, utilize them (put the bin == 1)
            end
        end

        %%variables from Tbo
        position = find(INIA_Tbo(1).timestamp == timestamp(i));
        %Gotta check that posDur returned something (i.e. Dur has a non-NaN
weather record for that timestamp). In the NaN cases, Matlab will output a NaN
value in the design
        %weather variable)
        %Default value (either no record, or record which is NaN
        tempTbo = 0;
        rainTbo = 0;
        wspdTbo = 0;
        sradTbo = 0;
        humTbo =  0;
        binTbo  = 0;   %This binTbo term is a binary (==1 only if there is ac-
tual record of the 5 climate variables), I will use it to kill unnecessary
terms in the interpolation
        if ~isempty(position)   %if no record, all values set to 0,then the
interp must ignore them when doing the sum.
            %case there is a record at that timestamp, bring in the values
only if they are non-NaN
            if ~isnan(INIA_Tbo(1).temp(position))
```

```
                tempTbo = INIA_Tbo(1).temp(position);
            end
            if ~isnan(INIA_Tbo(1).rain(position))
                rainTbo = INIA_Tbo(1).rain(position);
            end
            if ~isnan(INIA_Tbo(1).wspd(position))
                wspdTbo = INIA_Tbo(1).wspd(position);
            end
            if ~isnan(INIA_Tbo(1).srad(position))
                sradTbo = INIA_Tbo(1).srad(position);
            end
            if ~isnan(INIA_Tbo(1).hum(position))
                humTbo =  INIA_Tbo(1).hum(position);
            end
            if ~isnan(INIA_Tbo(1).temp(position)) &&
~isnan(INIA_Tbo(1).rain(position)) && ~isnan(INIA_Tbo(1).wspd(position)) &&
~isnan(INIA_Tbo(1).srad(position)) && ~isnan(INIA_Tbo(1).hum(position))
                binTbo = 1;   %if there is a "valid" record for the 5 vari-
ables, utilize them (put the bin == 1)
            end
        end

        %%variables from 33
        position = find(INIA_33(1).timestamp == timestamp(i));
        %Gotta check that pos33 returned something (i.e. 33 has a non-NaN
weather record for that timestamp). In the NaN cases, Matlab will output a NaN
value in the design weather variable)
        %Default value (either no record, or record which is NaN
        temp33 = 0;
        rain33 = 0;
        wspd33 = 0;
        srad33 = 0;
        hum33 =  0;
        bin33  = 0;   %This bin33 term is a binary (==1 only if there is ac-
tual record of the 5 climate variables), I will use it to kill unnecessary
terms in the interpolation

        if ~isempty(position)   %if no record, all values set to 0,then the
interp must ignore them when doing the sum. Case there is a record at that
timestamp, bring in the values only if they are non-NaN
            if ~isnan(INIA_33(1).temp(position))
                temp33 = INIA_33(1).temp(position);
            end
            if ~isnan(INIA_33(1).rain(position))
                rain33 = INIA_33(1).rain(position);
            end
            if ~isnan(INIA_33(1).wspd(position))
                wspd33 = INIA_33(1).wspd(position);
            end
            if ~isnan(INIA_33(1).srad(position))
                srad33 = INIA_33(1).srad(position);
            end
            if ~isnan(INIA_33(1).hum(position))
                hum33 =  INIA_33(1).hum(position);
            end
            if ~isnan(INIA_33(1).temp(position)) && ~isnan(INIA_33(1).rain(position)) && ~isnan(INIA_33(1).wspd(position)) && ~isnan(INIA_33(1).srad(position)) && ~isnan(INIA_33(1).hum(position))
                bin33 = 1;   %if there is a "valid" record for the 5 variables, utilize them (put the bin == 1)
            end
        end
        %%variables from Rocha
        position = find(INIA_RO(1).timestamp == timestamp(i));
        %Gotta check that posRo returned something (i.e. Ro has a non-NaN weather record for that timestamp). In the NaN cases, Matlab will output a NaN value in the design weather variable)
        %Default value (either no record, or record which is NaN
        tempRo = 0;
        rainRo = 0;
        wspdRo = 0;
        sradRo = 0;
        humRo =  0;
        binRo  = 0;   %This binRo term is a binary (==1 only if there is actual record of the 5 climate variables), I will use it to kill unnecessary terms in the interpolation

        if ~isempty(position)   %if no record, all values set to 0, then the interpolation must ignore them when doing the sum. Case there is a record at that timestamp, bring in the values only if they are non-NaN
            if ~isnan(INIA_RO(1).temp(position))
                tempRo = INIA_RO(1).temp(position);
            end
            if ~isnan(INIA_RO(1).rain(position))
                rainRo = INIA_RO(1).rain(position);
            end
            if ~isnan(INIA_RO(1).wspd(position))
                wspdRo = INIA_RO(1).wspd(position);
            end
            if ~isnan(INIA_RO(1).srad(position))
                sradRo = INIA_RO(1).srad(position);
            end
            if ~isnan(INIA_RO(1).hum(position))
                humRo =  INIA_RO(1).hum(position);
            end
            if ~isnan(INIA_RO(1).temp(position)) && ~isnan(INIA_RO(1).rain(position)) && ~isnan(INIA_RO(1).wspd(position)) && ~isnan(INIA_RO(1).srad(position)) && ~isnan(INIA_RO(1).hum(position))
                binRo = 1;   %if there is a "valid" record for the 5 variables, utilize them (put the bin == 1)
            end
        end
        %%variables from Glencoe
        position = find(INIA_Gle(1).timestamp == timestamp(i));
        %Gotta check that posGle returned something (i.e. Gle has a non-NaN weather record for that timestamp).In the NaN cases, Matlab will output a NaN value in the design weather variable)
        %Default value (either no record, or record which is NaN
        tempGle = 0;
        rainGle = 0;
        wspdGle = 0;
        sradGle = 0;
        humGle =  0;
        binGle  = 0;   %This binGle term is a binary (==1 only if there is ac-
tual record of the 5 climate variables), I will use it to kill unnecessary 
terms in the interpolation
        if ~isempty(position)   %if no record, all values set to 0,then the 
interp must ignore them when doing the sum. Case there is a record at that 
timestamp, bring in the values only if they are non-NaN
            if ~isnan(INIA_Gle(1).temp(position))
                tempGle = INIA_Gle(1).temp(position);
            end
            if ~isnan(INIA_Gle(1).rain(position))
                rainGle = INIA_Gle(1).rain(position);
            end
            if ~isnan(INIA_Gle(1).wspd(position))
                wspdGle = INIA_Gle(1).wspd(position);
            end
            if ~isnan(INIA_Gle(1).srad(position))
                sradGle = INIA_Gle(1).srad(position);
            end
            if ~isnan(INIA_Gle(1).hum(position))
                humGle =  INIA_Gle(1).hum(position);
            end
            if ~isnan(INIA_Gle(1).temp(position)) && 
~isnan(INIA_Gle(1).rain(position)) && ~isnan(INIA_Gle(1).wspd(position)) && 
~isnan(INIA_Gle(1).srad(position)) && ~isnan(INIA_Gle(1).hum(position))
                binGle = 1;   %if there is a "valid" record for the 5 vari-
ables, utilize them (put the bin == 1)
            end
        end
        %%variables from Salto Grande
        position = find(INIA_SG(1).timestamp == timestamp(i));
        %Gotta check that posSG returned something (i.e. SG has a non-NaN 
weather record for that timestamp). (in the NaN cases, Matlab will output a 
NaN value in the design weather variable). Default value (either no record, or 
record which is NaN
        tempSG = 0;
        rainSG = 0;
        wspdSG = 0;
        sradSG = 0;
        humSG =  0;
        binSG  = 0;   %This binSG term is a binary (==1 only if there is ac-
tual record of the 5 climate variables), I will use it to kill unnecessary 
terms in the interpolation

        if ~isempty(position)   %if no record, all values set to 0, then the 
interpolation must ignore them when doing the sum. Case there is a record at 
that timestamp, bring in the values only if they are non-NaN
            if ~isnan(INIA_SG(1).temp(position))
                tempSG = INIA_SG(1).temp(position);
            end
            if ~isnan(INIA_SG(1).rain(position))
                rainSG = INIA_SG(1).rain(position);
            end
            if ~isnan(INIA_SG(1).wspd(position))
                wspdSG = INIA_SG(1).wspd(position);
            end
            if ~isnan(INIA_SG(1).srad(position))
                sradSG = INIA_SG(1).srad(position);
            end
            if ~isnan(INIA_SG(1).hum(position))
                humSG =  INIA_SG(1).hum(position);
            end
            if ~isnan(INIA_SG(1).temp(position)) && ~isnan(INIA_SG(1).rain(position)) && ~isnan(INIA_SG(1).wspd(position)) && ~isnan(INIA_SG(1).srad(position)) && ~isnan(INIA_SG(1).hum(position))
                binSG = 1;   %if there is a "valid" record for the 5 variables, utilize them (put the bin == 1)
            end
        end
        %%Now interpolate using inverse of reciprocal distance (Wei & Guinness, 1971?). Those terms of the equation corresponding to no-Data or NaN-data will be cancelled out automatically (mult. by 0)
        temp(i) = (binLE*tempLE/distLE2 + binLB*tempLB/distLB2 + binDur*tempDur/distDur2 + bin33*temp33/dist332 + binRo*tempRo/distRo2 + binGle*tempGle/distGle2 + binTbo*tempTbo/distTbo2 + binSG*tempSG/distSG2)/ ...
            (binLE/distLE2 + binLB/distLB2 + binDur/distDur2 + bin33/dist332 + binRo/distRo2 + binGle/distGle2 + binTbo/distTbo2 + binSG/distSG2);
        rain(i) = (binLE*rainLE/distLE2 + binLB*rainLB/distLB2 + binDur*rainDur/distDur2 + bin33*rain33/dist332 + binRo*rainRo/distRo2 + binGle*rainGle/distGle2 + binTbo*rainTbo/distTbo2 + binSG*rainSG/distSG2)/ ...
            (binLE/distLE2 + binLB/distLB2 + binDur/distDur2 + bin33/dist332 + binRo/distRo2 + binGle/distGle2 + binTbo/distTbo2 + binSG/distSG2);
        wspd(i) = (binLE*wspdLE/distLE2 + binLB*wspdLB/distLB2 + binDur*wspdDur/distDur2 + bin33*wspd33/dist332 + binRo*wspdRo/distRo2 + binGle*wspdGle/distGle2 + binTbo*wspdTbo/distTbo2 + binSG*wspdSG/distSG2)/ ...
            (binLE/distLE2 + binLB/distLB2 + binDur/distDur2 + bin33/dist332 + binRo/distRo2 + binGle/distGle2 + binTbo/distTbo2 + binSG/distSG2);
        srad(i) = (binLE*sradLE/distLE2 + binLB*sradLB/distLB2 + binDur*sradDur/distDur2 + bin33*srad33/dist332 + binRo*sradRo/distRo2 + binGle*sradGle/distGle2 + binTbo*sradTbo/distTbo2 + binSG*sradSG/distSG2)/ ...
            (binLE/distLE2 + binLB/distLB2 + binDur/distDur2 + bin33/dist332 + binRo/distRo2 + binGle/distGle2 + binTbo/distTbo2 + binSG/distSG2);
        hum(i)  = (binLE*humLE/distLE2  + binLB*humLB/distLB2  + binDur*humDur/distDur2  + bin33*hum33/dist332  + binRo*humRo/distRo2  + binGle*humGle/distGle2  + binTbo*humTbo/distTbo2  + binSG*humSG/distSG2 )/ ...
            (binLE/distLE2 + binLB/distLB2 + binDur/distDur2 + bin33/dist332 + binRo/distRo2 + binGle/distGle2 + binTbo/distTbo2 + binSG/distSG2);
    end
    %% Part 4
    %Output the climateMatrix of interpolated weather records
    if verbose
        disp('climateSimulator:: Exporting simulated climate variables')
    end
    %%export as output...
    climateMatrix = [timestamp temp hum wspd rain srad];
    %UPDATE 2018-09-05 - Store the newly generated climateMatrix in Cache
    oldY = projectY;
    oldX = projectX;
    oldClimateMatrix = climateMatrix;
    save './dataFiles/climateMAtrixCached.mat' oldX oldY oldClimateMatrix

end %close the if loop in ln30


end   %%endFunction
```

Source Code: /climate/climateScrambler function

```
function climateMatrix = climateScrambler(climateRecord,climateRandomSeed,de-
signLife,dateOpenToTraffic,verbose)
%function climateMatrix = climateScrambler(climateRecord,climateRandomSeed,de-
signLife,dateOpenToTraffic,verbose)
%
%%front-end function to develop the project site's simulated climate variables
from interpolated INIA records or Level-1 climate record.
%%%This function will work as follows
%1 - construct a virtual_ weather station climate record from the INIA db
(years 2011-2017 <future updates to the code may allow to extend the source
data as INIA collects more records>. Use Square Dist. interpolation to do so
(bring ref from paper from the 70s).
%2 - simulate future climate:
  %a) create a random vector with the length of design life,
  %b) convert these random numbers to integer years, spanning between 2011 and
2017 (last year of INIA Data)
  %c) construct simulated climate using the interpolated weather station,
scrambling the variables follwoing the random vector.
%
%The "verbose" input allows to show on screen every calculation step. input 1
if you want to do so. If no input, it assumes zero
%
%Matching V2018-09-05:  Separate lvl2 climate interpolator from scrambler.
%                       Add randomSeed integer for repeatability of climate
%                       scrambles
%
%% Preprocessing - unpack the climateRecord matrix
timestamp = climateRecord(:,1);
temp = climateRecord(:,2);
hum  = climateRecord(:,3);
wspd = climateRecord(:,4);
rain = climateRecord(:,5);
srad = climateRecord(:,6);

%Scramble random number vector and obtain randomly chosen years.
if verbose == 1
    disp('climateScrambler:: shuffling random years for climate simulation')
end
load './dataFiles/climateRandomSeed.mat';   %file created! contains last used
random Seed, last generated randomSeedMatrix, and first/last years in that
randomSeedMatrix
%
%variables in climateRandomSeed:
    % lastRandomSeed - matrix with 50 rows and colums equal to last used
    % design life +1
    % lastUsedYear - vector with last used initial and end years
    % lastRandomSeed - last Used random seed (if I ever need it...)

%So, here's the logic for the scrambler.
%1 - given the randomSeed and the pavement's design life
```

```
%   IF design life + 1 = columns in randomSeed matrix, AND first & last years match what previously used. Recycle the matrix for current run
  % ELSE (either column mismatch or years mismatch)
  %     recalculate the matrix; and store in the climateRandomSeed.mat file
        %use an auxiliary function for that!

lastYear = datevec(timestamp(end-1));
lastYear = lastYear(1);  %%Using DateVec function (compatibility issue with Matlab -year function only runs on integers-
firstYear =  datevec(timestamp(1));
firstYear = firstYear(1);

%First, check if I can use the cached randomSeed Matrix
[~,bb] = size(randomSeedMatrix);   %aa should be 50 -fixed by programming; replaced by ~ to ignore it (thanks Matlab!)
if bb == designLife+1 && firstYear == lastUsedYear(1) && lastYear == lastUsedYear(2)
    %recycle matrix, do nothing new
else
    %create a new randomSeedMatrix
    lastUsedYear = [firstYear lastYear];
    lastRandomSeed = climateRandomSeed;
    randomSeedMatrix = random('unif',firstYear,lastYear,[50,designLife+1]);
%OCTAVE USERS - INSTALL STATISTICS PACKAGE TO USE RANDOM FUNCTION!!
    %randomSeedMatrix is a vector with the length of designLife which has the available climate years selected randomly. %But thing is I have to round up the years to perform the climate simulation [I cannot simulate year 2013.445]
    %Other detail: length of the random vector is designLife + 1, I've done this on purpose cause the pavement project may not start on jan. 1st., so, in order to wrap all the design period with the simulated climate.
    randomSeedMatrix = round(randomSeedMatrix);
    save './dataFiles/climateRandomSeed.mat' lastUsedYear lastRandomSeed randomSeedMatrix
end

%% 2 - now that I have a valid (recycled or new) randomSeedMatrix,
%collect the row of random years with the randomSeed

randomYears = randomSeedMatrix(climateRandomSeed,:);

if verbose == 1
    disp('climateSimulator:: randomYears vector ready, constructing simulated climate')
end
%%Define final simulated weather variables
%%1-locate the positions on the simulated weather station where each randomly picked year occurs - store in "position"
auxYear = datevec(timestamp);
auxYear = auxYear(:,1);  %get the "year" column from all the simulated timestamps
%first element in randomYears
position = find(auxYear == randomYears(1));
yearOpToTraffic = datevec(dateOpenToTraffic);
yearOpToTraffic = yearOpToTraffic(1);
```

```
designYear = yearOpToTraffic*ones(length(position),1);
%remaining elements
for k = 2:length(randomYears)
    aux = find(auxYear == randomYears(k));
    if isempty(aux)
        %small workaround when aux comes empty (a year-wide void in the
weather data, may occur when using non-continuous level-1 weather
        %data
        if k<length(randomYears)
         aux = find(auxYear == randomYears(k+1));
        else
         aux = find(auxYear == randomYears(k-1));
        end
    end

    position = [position; aux];   %output of "find" function is a column vec-
tor!
    designYear = [designYear; (designYear(1)+k-1)*ones(length(aux),1)];
end
clear aux;   %don't need it any longer.

%% 3 - construct the simulated weather variables
timestamp = timestamp(position);   %timestamp vector with the same length as
the variables. Need to change year only - will use designYear to do so-)  cor-
rect timestamp to design years;
auxTimestamp = datevec(timestamp);
timestamp = datenum(designYear(:),auxTimestamp(:,2),auxTimestamp(:,3),aux-
Timestamp(:,4),0,0);  %this is the timestamp vector with all the dates but the
year (it starts in the design year)

temp = temp(position);
rain = rain(position);
wspd = wspd(position);
srad = srad(position);
hum  =  hum(position);   %THESE VARIABLES HAVE COMPLETE YEARS ACCORDING TO THE
YEAR SCRAMBLER!

%%3- Remove those timestamps that are from before the dateOpenToTraffic
datesToRemove = find(timestamp<dateOpenToTraffic);  %all the positions herein
are dates that happen before the date the pave. is open to traffic. Remove
them!
dtrLast = datesToRemove(end);  %this is the last position to remove. Redefine
the export variables as var = var(dtrL+1:end)

timestamp = timestamp(dtrLast+1:end);
temp = temp(dtrLast+1:end);
rain = rain(dtrLast+1:end);
wspd = wspd(dtrLast+1:end);
srad = srad(dtrLast+1:end);
hum  =  hum(dtrLast+1:end);

if verbose == 1
    disp('climateSimulator:: Exporting simulated climate variables')
end
```

```
%%export as output...
climateMatrix = [timestamp temp hum wspd rain srad];
%Send also randomYears vector as output? or not nec.?

end   %%endFunction
```

Source Code: /climate/climateModule.m

```
%%%% - PRODUCT-1 - M.E. Pavement Design tool - %%%%
%%% CLIMATE EFFECTS MODULE for flexible pavements%%%
%
% Version 0.3 2019-01-25 - - bug-fixed all throughout
% Version 0.2 2018-10-22 - - added layers moisture module.
% Version 0.1 2018-06-20
%
%%THIS SCRIPT IS MEANT TO RUN AS WITHIN THE PAVEMENT DISTRESS SIMULATION PRO-
GRAMMED IN THE MAIN CODE. %EACH INSTANCE OF THIS SCRIPT OCCURS AT "timeStamp =
k"
%
%%NOTE: A MAJOR HYPOTHESIS FOR THIS DESIGN TOOL: NO FREEZING OCCURS THROUGHOUT
THE PAVEMENT STRUCTURE.  FROST HEAVE PHENOMENA SUCH AS THOSE MODELED WITH %
%THE CRREL "FROST" PROGRAM (1986?) WILL BE DISREGARDED.
%
%units for all values = m3/12h/m2 or m3/m2!  (for volume calculations, assum-
ing 1m x 1m cell)

%% -- RAINFALL-INFILTRATION-BASE DRAINAGE MODULE
%Calculate entering rainfall, evaluate how much infiltrates, calculate infil-
tration through base and subbase, and down to subgrade. Use equations from the
I-D model, deterministic version. Calculate infiltration rate to pavement
%analysis cell of 1m x 1m (1mm of rain equals 0.001m3 of entering water)
infRate_HMA = 0.1*0.305^2;     %unit infiltration rate (m3/h) per meter of
cracks - HMA pavements. Source FHWA RD 90-033, eqn 21  [LATER ON I ACCUMULATE
TO INF_RATE X 12H]
%infRate_PCC = 0.03*0.305^2;   %unit infiltration rate (m3/h) per meter of
cracks - PCC pavements. Source FHWA RD 90-033

%Sum the length of all cracks at time k-1 (to calculate how much water will
infiltrate. %ASSUMPTION: alligator cracking are as cells 0.10mx0.10m; reflec-
tive cracks are cells 3.6x4.5m  (typical PCC slab size in Uruguayan roads)
if k ==1
    totalCracks = 0;   %%CORRECT THIS LINE WITH THE "AS-BUILT" DISTRESS RATES!
else
    totalCracks = 0.001*topDownCrack(k-1,1)+alligatorCrack(k-1,1)*(4*0.1);%
+0.001*transvCrack(k-1)+reflectiveCrack(k-1)*2*(4.5+3.6)*(1/(4.5*3.6));
%this is the total length of all cracks in the pavement / proportionally dis-
tributed over sq. m. of pave.
end

%UPDATE 2018-06-20 - LOCATE THE RAINFALL intake from the period past and ana-
lyze it (compare individual hourly rain rates to inf. rate and deduct infil-
tration and runoff)
%for k == 1 (start of code), rainfallIntake = zeros(12,1)  - let's assume it
wasn't raining when the last HMA layer was placed.
%k follows shortTimestamp!!!
whereInTime = find(longTimestamp == shortTimestamp(k));
if k ==1
    %%accumulate last 12h of rainfall!  in m3/m2/12h
    intervalRainfall = 0.001*12/k*sum(rainFall(1:whereInTime));  %in here I'm
summing up less rainfall than 12h, I'm just extrapolating by multiplying by
12/k
else
    intervalRainfall = 0.001*sum(rainFall(whereInTime-11:whereInTime));
%accumulated rain in the last 12 hours, converted to m3/12h
end

%deduct the infiltration rate to intervalRainfall
intervalInfiltration = min([intervalRainfall,totalCracks*infRate_HMA*12]);%in-
filtration is over a 12-hour period!!!!
surfaceRunoff(k) = intervalRainfall - intervalInfiltration;     %the water
that doesn't infiltrate must leave   [m3/m2/12h]  [I'll convert it to mm/12h
later on <ln. 123>

surfaceInfiltration(k) = 1000*intervalInfiltration;                     %store
interval infiltration in mm/12h
shortRainfall(k) = 1000*intervalRainfall;                               %store
the accumulated rainfall. [mm/12h]

%% -- CALCULATE HUMIDITY AT BASE/SUBBASE LAYERS
%%Update 2019-01-25 - Calculate the kUnsat with the "initial time moisture
content. Use lower layer's kUnsat to calculate downward runoff.Limit downward
runoff also to the available void space between saturation and current mois-
ture in the lower layer.
%%UPDATE 2018-06-20 - THIS SCRIPT WILL RUN IN THE 1/2-DAY-LONG SIMULATION,
get the amount of water that entered throughout the period from --->> "inter-
valInfiltration"
% <<--%%UPDATE 2-18-10-19 - get the hydraulic conductivity from moisture con-
tent

deltaTime = 3600*12;       %time interval to sum the escaping runoff that en-
tered the pavement
numLayersForMoisture = 1+paveLayersNumber-ACLayersNumber;  %layers for mois-
ture analysis are the granular layers + subgrade!
layersMoistureThickness = paveDepths(end-numLayersForMoisture+1:end)/100; %get
the thicknesses of the non-HMA layers and convert to meters
layersSaturation = 0.01*granHumidity(:,3).*layersMoistureThickness;
%saturation vol. content of the gran. layers (m3/m2).

  %Compare against humidity at any moment.  [remember that granHumidity(:,3)
comes in PERCENTAGE VOLUME]
% get previous-time moisture over each layer at the subgrade
if k == 1
    previousHumidity = granHumidity(:,1);
%%initial water content [percentage]. Defined in the materialsParametersLvlXX
script
    previousHumidity =
(1/100).*previousHumidity.*layersMoistureThickness.*1.*1;  %m3/m2
    %IN PERCENT! MUST CONVERT TO VOLUME (MULTIPLY PER SIZE OF LAYER!, IN THE
CASE OF SUBGRADE, REDUCE TO ANALYZING A 3m-thick zone.%(MAY NEED TO JUSTIFY
THIS CHOICE FROM THE STRESS-STRAIN MODEL (HOW DEEP DO THE STRESSES GO...)
else
    previousHumidity = layersMoisture(k-1,:)';
    %row vector cointaining the calculated humidity values for time = k-1 for all the non-HMA layers AND THE SUBGRADE! in percent volume!
    %%Bug correction 2019-01-25:: transpose that row vector to column to avoid issues when converting to m3/m2 (matlab doesn't go entry by entry)
    previousHumidity = (1/100).*previousHumidity.*layersMoistureThickness; %convert it to m3/m2
end

%get each material's kUnsat fusing the "previousHumidity IN PERCENTAGE VOLUME!
kValue = zeros(numLayersForMoisture,1);                    %store each material's unsat hidr. conductivity here.
for j = 1:numLayersForMoisture
   %get the layer's hydraulic conductivity from SWCC calculations done in preprocessing [COMES IN M/SEC]!!!!
    %REPAIR IT WITH A SMALL IF SENTENCE NO NOT UNDO THE previousHumidity unit conversion above
    % and solve now
    if k ==1
        kValue(j) = interp1(granularKSWCC(:,2,j),granularKSWCC(:,3,j),granHumidity(j,1));       %get the layer's hydraulic conductivity from SWCC calculations done in preprocessing. Enter in % volume, output in M/SEC]!!!!
    else
        kValue(j) = interp1(granularKSWCC(:,2,j),granularKSWCC(:,3,j),layersMoisture(k-1,j)');  %get the layer's hydraulic conductivity from SWCC calculations done in preprocessing. Enter in % volume, output in M/SEC]!!!!
    end
    %%end of correction
end

%let's solve separately for each layer one by one, open a For loop
downwardRunoff = intervalInfiltration;                     %start this variable (i will replace it as I calculate over each cell).

% start the water flow balance / GO LAYER BY LAYER
for j = 1:numLayersForMoisture

    moistureLyrJaux = previousHumidity(j);      %get the volume of water in layer j at time k (per sq. m of area) [from previousHumidity vector]
    %assuming isotropic material kValue is the same for flow in any direction
    lateralRunoff(k,j) =
min([deltaTime*kValue(j)*layersMoistureThickness(j)*paveCrossSlope/100,moistureLyrJaux+downwardRunoff]);
    %lateral runoff layer j = kValue * thickness Lyr j * Pavement SX (absolute)* delta Time. NO GREATER THAN CURRENT MOISTURE + INPUT.  %VALUE CALCULATED HEREIN IS IN MR/M2/12H
    moistureLyrJaux = moistureLyrJaux - lateralRunoff(k,j);
    if j<numLayersForMoisture
        downwardRunoffAux = min([deltaTime*kValue(j+1),moistureLyrJaux,layersSaturation(j+1)-previousHumidity(j+1)]);
    else
        downwardRunoffAux = min([deltaTime*kValue(j),moistureLyrJaux]);
    end
    moistureLyrJaux = moistureLyrJaux - downwardRunoffAux;
    %%%%following simplification in Davidson et al (1969)/ low flow phenome-
na.NO GREATER THAN CURRENT MOISTURE (after pulling lateral flow.
    %%BUG CORRECTION 2019-1-24 (above), add also resetriction to downward flow
not be greater than the capacity to saturation of the underlying layer.
    %%BUG CORRECTION 2019-1-25        , correct downward runoff to make it de-
pendant on the lower layer's kUnsat (more restrictive than upper layer //
lower layer is comparatively impervious

    %Save the final moisture in layer j, and pass on the downward Runoff to
the next layer...
    currentHumidity = min([moistureLyrJaux,layersSaturation(j)]);
%in volume,. layer j  %%%LIMIT IT TO BE NO GREATER THAN SATURATION HUMIDITY!!
    %>>>>>>>>>>>>>>> surfaceRunoff(k) = surfaceRunoff(k) +
max(moistureLyrJaux,layersSaturation(j)) - currentHumidity;  %any additional
water that can neither flow downwards nor sideways nor be retained must go as
surface Runoff.
    downwardRunoff = downwardRunoffAux;
   layersMoisture(k,j) = currentHumidity/(layersMoistureThickness(j)*1*1)*100;
%store the volumetric moisture in layer j in percent volume content

end
%Store surfaceRunoff in mm/12h (to compare with rainfall and infiltration)
surfaceRunoff(k) = 1000*surfaceRunoff(k);

%% -- PAVEMENT TEMPERATURE PROFILE MODULE FOR FLEXIBLE PAVEMENTS
%Calculate instant temperature profile throughout structure. Use equations
from the 1993 paper and LTPP to calculate temperature profile through the Apsh
layers.

%%UPDATE 2018-06-20 - THIS MODULE HAS BEEN REMOVED FROM THE SCRIPT AND PUT AS
A FUNCTION IN THE PRE-PROCESSING (DOESN'T NEED TO RUN ALONG WITH THE DESIGN
SIMULATION)

%% -- SUB-GRADE WATER TABLE MODULE
%%Update 2018-06-20 - THIS MODULE WILL NOT BE IMPLEMENTED. SIMPLIFICATION:
SUB-GRADE WILL HAVE A SINGLE-VALUE RESILIENT MODULUS (CORRESPONDING TO NORMAL
CONDITIONS) - LEVEL-3 ACCORDING TO NCHRP 1-37?.
%%UNDERLYING ASSUMPTION: URUGUAYAN ROADS'SUBGRADES ARE EITHER ALWAYS SOAKED OR
ALWAYS "DRY". THAT IS, RISE OF THE WATER TABLE MAY NOT MODIFY THE STRENGTH OF
THE PORTION OF SUBGRADE THAT RECEIVES LOADS.

%% --END OF SCRIPT
```

Source Code: /dataImporting/climateLoadLocal

```
function [climateRecord] = climateLoadLocalXLS(srcSpreadsheet,verbose)
%function [climateRecord] = climateLoadLocalXLS(srcSpreadsheet,verbose)
%   this function will load a Level 1 climate record from the source Excel spreadsheet srcSpreadsheet
% - made to run in MATLAB ONLY!  (OCTAVE USERS WILL HAVE THE  climateLoadLocalOCT!

if verbose
    disp('Climate records import starting....')
end
%1 - ask for cheking the spreadsheet name is ok
checkFilename = input(sprintf('Please confirm the name of the spreadsheet to import from... (1 = yes, currently %s) \n',srcSpreadsheet));
if ~checkFilename
    srcSpreadsheet = input('Please give <between single brackets> the name of the spreedsheet...');
end

%2 ask for the sheet name  (default is "climate"
defaultSheetname = 'climate';
checkSheet = input(sprintf('Please confirm the name of the spreadsheet to import from... (1 = yes, currently %s)\n',defaultSheetname));
if ~checkSheet
     defaultSheetname= input('Please give (between single brackets) the name of the sheet to read from...');
end

%3 ask for range to import!
importRange = input('Please type in the range to import from (between single brackets)... ');

%4 import with xlsRead!
auxClimateRecord = xlsread(srcSpreadsheet,defaultSheetname,importRange);
[a,b] = size(auxClimateRecord);
climateRecord = zeros(a,6);

%5 Create the timestamp vector with columns 1--4
climateRecord(:,1) = datenum(auxClimateRecord(:,3),auxClimateRecord(:,1),auxClimateRecord(:,2),auxClimateRecord(:,4),0,0);
climateRecord(:,2:6) = auxClimateRecord(:,5:b);
end
```

Source Code: /dataImporting/mainDataImport

```
% PRODUCT-ONE PAVEMENT DESIGN TOOL
%%----------------------------------
%%master importer script to import traffic data
%%Matlab-only imports - uses XLSREAD function on "dataImport" file (defined
from the main script

%% ---PROBLEM SETTINGS READ ------
runVerbose = xlsread(dataImport,'info','b10:b11');        %boolean variable,
if = 1, run in verbose mode, detailing every calc. step
%exportToPDF = runVerbose(2);          %boolean variable. if = 1, make pdf
files with the many graphs to use. Use exportPDF (dependency:: GhostScript).
Disabled in this 1st release of Product-One
runVerbose = runVerbose(1);
createFigures = xlsread(dataImport,'info','b14:b18');
[~,saveDataName,~] = xlsread(dataImport,'info','b21'); %name of the output
files  (code will produce a .mat file with the calculation results and pdf
files with graphs and many reports)
saveDataName = string(saveDataName); %name of the output files  (code will
produce a .mat file with the calculation results and pdf files with graphs and
many reports)

%% ---PROBLEM DATA IMPORT----

%1 - Project location
if runVerbose
    disp('dataImport:: importing Project"s general info')
end
Loc = xlsread(dataImport,'info','e10:e11');
locX = Loc(1);
locY = Loc(2);
clear Loc;
startDate = xlsread(dataImport,'info','h10:h12');
startTimeStamp = datenum(startDate(1), startDate(2), startDate(3),18,0,0);
%fixed to start at daytime (it may save problems when computing design 1/2 day
traffic)
designLife = xlsread(dataImport,'info','h16');

%2 - Trial pav. structure and subgrade
if runVerbose
    disp('dataImport:: importing pavement trial structure info')
end
getPaveSummary = xlsread(dataImport,'pavStructure','m10:m14');
paveCrossSlope = xlsread(dataImport,'pavStructure','d13');
%paveType = getPaveSummary(1);       %%Boolean variable 1 = flexible, 2 = con-
crete
paveChipSeal = getPaveSummary(1);   %%Boolean variable 1 = chip seal surface,
0 = strong HMA or granular pavement (concrete by default will make 0)
paveLayersNumber = getPaveSummary(2);
ACLayersNumber = getPaveSummary(3);
%2.1 - Locate depth of asph layers and total depth of pavement
totalDepth = getPaveSummary(4);
```

```
ACtotalDepth = getPaveSummary(5);
clear getPaveSummary;

paveStructure = xlsread(dataImport,'pavStructure','g10:j15');  %%Read a matrix
loaded with the needed info about the pavement structure and the subgrade!
paveLevelInput = paveStructure(1,end);  %will read the Input level of the up-
permost layer (will tell if using default values or local ones)
paveID = paveStructure(:,1);            %Get the ID of each material used (so
that I can get its material properties from
paveDepths = paveStructure(:,3);
ACPaveDepth  = paveStructure(1:ACLayersNumber,3);  %this column will contain
the depth of each AC layer (needed in temperature profile)
clear paveStructure

%3 - Site climate
climateInputs = xlsread(dataImport,'info','m11:m13');  %boolean variable. if =
1, use local climate and don't construct simulation from INIA
climateRandomSeed = climateInputs(3);
climateLvl2ForceRecalc = climateInputs(2);
climateLevel1 = climateInputs(1);  %boolean variable. if = 1, use local cli-
mate and don't construct simulation from INIA
clear climateInputs

%NOTE: climate info for a Level 1 input (local from project's site) will be
imported by the climateLoadLocal function

%5 - Read the user's settings on distress thresholds---
%Adopt distresses from NCHRP 1-37A
readDistresses = xlsread(dataImport,'info','r9:r10');
maxRutDepth = readDistresses(1);
maxIRI = readDistresses(2);
maxCracking = xlsread(dataImport,'info','r13:r16');   %cracking entered as
transvsersal/longitudinal/alligator/reflective.
clear readDistresses;
initialDistress = xlsread(dataImport,'info','w9:w10');
initialIRI = initialDistress(2);
initialRutDepth = initialDistress(1);
clear initialDistress
```

Source Code: /dataImporting/trafficDataImporter

```
%% PRODUCT-ONE PAVEMENT DESIGN TOOL
%%---------------------------------
%%master importer script to import traffic data
%%Matlab-only imports - uses XLSREAD function on "dataImport" file (defined
%%from the main script).
%
%V2 2019-02-22: RESPONDS TO MODIFIED DATAIMPORT SPREADSHEET.
%The dataImport spreadsheet now contains a selection of distr. factors for
level2 or level3 design; the importer will read an automatically-populated
%table based on the user's Level of Input and functional classification. Only
need to distinguish level1 from level 2/3

%%this was originally a part from the main code, but i move it aside Because
% a) it was using too many lines in the main code
% b) i need a parallel code for Octave


AADTbaseYear = xlsread(dataImport,'trafficCountInput','g19:g41');
AADTGrowthRate = xlsread(dataImport,'trafficCountInput','e19:e41');
AADTSpeed = xlsread(dataImport,'trafficCountInput','i19:i41');
AADTLevelOfInput = xlsread(dataImport,'trafficCountInput','m10:m11');

if AADTLevelOfInput(1) == 1  %inputs for Level 1
    AADTMonthlyDistr = xlsread(dataImport,'trafficCountInput','c49:n71');
    AADTDailyDistr   = xlsread(dataImport,'trafficCountInput','t49:z71');
    AADTHourlyDistr  = xlsread(dataImport,'trafficCountInput','bf49:bh71');
%update jun 20th, 2018 - Use daytime/nighttime traffic distribution now.
else %inputs for Level 2 // 3 AUTOMATICALLY SELECTED IN THE SPREADSHEET!
    AADTMonthlyDistr = xlsread(dataImport,'trafficCountInput','c79:n101');
    AADTDailyDistr   = xlsread(dataImport,'trafficCountInput','t79:z101');
    AADTHourlyDistr  = xlsread(dataImport,'trafficCountInput','bf79:bh101');
end

%B) import weight record per category
if AADTLevelOfInput(1) == 1  %inputs for Level 1
    trafficAxleLoad = xlsread(dataImport,'trafficLoadInput','g13:s104');
    trafficLoadPerc = xlsread(dataImport,'trafficLoadInput','e13:e105');
elseif AADTLevelOfInput(1) == 2  %inputs for Level 2
    trafficAxleLoad = xlsread(dataImport,'trafficLoadInput','g113:s204');
    trafficLoadPerc = xlsread(dataImport,'trafficLoadInput','e113:e204');
elseif AADTLevelOfInput(1) == 3   %inputs for Level 3
    trafficAxleLoad = xlsread(dataImport,'trafficLoadInput','g213:s304');
    trafficLoadPerc = xlsread(dataImport,'trafficLoadInput','e213:e304');
end
```

Source Code: /dataImporting/dataExport

```
% PRODUCT-ONE PAVEMENT DESIGN TOOL
%%----------------------------------
%%master exporter script to export select simulation results to a spreadsheet.
%Matlab-only imports - uses XLSREAD function on file with name:
finalName"saveDataName.xlsx" file (defined from the main script)

if runVerbose; disp('Spreadsheet Export:: Exporting simulated climate variables'); end
%% ---Export simulated climate---
%Target sheet: climate
auxDate = datevec(longTimestamp);

finalRowToExport = 9+length(longTimestamp)-1;                           %%
%I'll program all outputs to go from row 9 downward
rangeToExport    = strcat('a9:d',string(finalRowToExport)) ;
export = xlswrite(char(finalName),{saveDataName},'climate','g2');
%export a tag with the name of the simulation run and date of completion
export = xlswrite(char(finalName),{datestr(now,23)},'climate','g3');
export = xlswrite(char(finalName),auxDate(:,1:4),'climate',char(rangeToExport));   %export timeStamp

rangeToExport    = strcat('e9:e',string(finalRowToExport));
export = xlswrite(char(finalName),airTemp,'climate',char(rangeToExport));
%export temperature

rangeToExport    = strcat('f9:f',string(finalRowToExport));
export = xlswrite(char(finalName),humidity,'climate',char(rangeToExport));
%export humidity

rangeToExport    = strcat('g9:g',string(finalRowToExport));
export = xlswrite(char(finalName),windSpd,'climate',char(rangeToExport));
%export wind speed

rangeToExport    = strcat('h9:h',string(finalRowToExport));
export = xlswrite(char(finalName),rainFall,'climate',char(rangeToExport));
%export rainfall

rangeToExport    = strcat('i9:i',string(finalRowToExport));
export = xlswrite(char(finalName),sunRad,'climate',char(rangeToExport));  %export solar radiation

%% ---Export predicted traffic
if runVerbose; disp('Spreadsheet Export:: Exporting traffic prediction results'); end
%%%sheet = traffic
%%use 'simpleTrafficReport': column 1==years. Columns 2:6 - AADT cars, buses,
trucks lt,md,hv

%%export AADT per category - - maybe this one can pass through from the
dataInput sheet...   I'll think about it
```

```
finalRowToExport = 9+length(designAADT(1,:))-1;  %%%I'll program all outputs to go from row 9 downward
rangeToExport    = strcat('b9:b',string(finalRowToExport)) ;
export = xlswrite(char(finalName),designAADT(1,:)','traffic',char(rangeToExport));  %export years...

rangeToExport    = strcat('d9:z',string(finalRowToExport)) ;
export = xlswrite(char(finalName),(designAADT(2:end,:))','traffic',char(rangeToExport));  %export ADT by category

rangeToExport    = strcat('ab9:af',string(finalRowToExport)) ;
export = xlswrite(char(finalName),simpleTrafficReport','traffic',char(rangeToExport));  %export simplified cat. counts


%% ---Export simulated HMA layers temperature
if runVerbose; disp('Spreadsheet Export:: Exporting HMA temperature simulation results'); end
%%%sheet HMA Layers Temperature

finalRowToExport = 9+length(airTemp)-1;                                  %%%I'll program all outputs to go from row 9 downward
if length(airTemp)>65500
   disp('Spreadsheet Export:: WARNING! Extremely long series to be exported, older XLS versions may truncate the data!')
end
rangeToExport    = strcat('a9:d',string(finalRowToExport)) ;             %%%I'll program all outputs to go from row 9 downward
export = xlswrite(char(finalName),auxDate(:,1:4),'HMA Layers Temperature',char(rangeToExport));  %export short timestamp
clear auxDate;   %%i won't need the long timeSeries any longer

rangeToExport    = strcat('e9:e',string(finalRowToExport)) ;
export = xlswrite(char(finalName),airTemp,'HMA Layers Temperature',char(rangeToExport));  %export air temperature

rangeToExport    = strcat('f9:i',string(finalRowToExport)) ;
export = xlswrite(char(finalName),asphLyrTemp,'HMA Layers Temperature',char(rangeToExport));  %export surface and HMA layers temperature.


if runVerbose; disp('Spreadsheet Export:: Exporting Infiltration & Runoff module results'); end
%% ---Export results from the infiltration analysis
%%%sheet Infiltration Runoff

auxDate = datevec(shortTimestamp);

finalRowToExport = 9+termination-1;       %%%I'll program all outputs to go from row 9 downward
rangeToExport    = strcat('a9:d',string(finalRowToExport)) ;
export = xlswrite(char(finalName),auxDate(:,1:4),'Infiltration Runoff',char(rangeToExport));  %export short timestamp
```

```
rangeToExport    = strcat('e9:e',string(finalRowToExport)) ;
export = xlswrite(char(finalName),shortRainfall,'Infiltration
Runoff',char(rangeToExport));  %export short Rainfall

rangeToExport    = strcat('f9:f',string(finalRowToExport)) ;
export = xlswrite(char(finalName),surfaceInfiltration,'Infiltration
Runoff',char(rangeToExport));  %export infiltration through surface

rangeToExport    = strcat('g9:g',string(finalRowToExport));
export = xlswrite(char(finalName),surfaceRunoff,'Infiltration
Runoff',char(rangeToExport));   %export surface runoff

rangeToExport    = strcat('i9:m',string(finalRowToExport));
export = xlswrite(char(finalName),layersMoisture,'Infiltration
Runoff',char(rangeToExport));   %export moisture content by vol

rangeToExport    = strcat('o9:s',string(finalRowToExport));
export = xlswrite(char(finalName),MR,'Infiltration Runoff',char(rangeToExport));   %export resilient modulus

%% ---Export results from distress prediction
if runVerbose; disp('Spreadsheet Export:: Exporting Predicted Distresses');
end
%%%sheet Distress Prediction
%
finalRowToExport = 9+termination-1;    %%%I'll program all outputs to go from
row 9 downward
rangeToExport    = strcat('a9:d',string(finalRowToExport)) ;
export = xlswrite(char(finalName),auxDate(:,1:4),'Infiltration
Runoff',char(rangeToExport));  %export short timestamp

rangeToExport    = strcat('e9:e',string(finalRowToExport));
export = xlswrite(char(finalName),rutDepth(:,end),'Distress Prediction',char(rangeToExport));  %export rut depth

rangeToExport    = strcat('f9:f',string(finalRowToExport));
export = xlswrite(char(finalName),topDownCrack(:,1),'Distress Prediction',char(rangeToExport));  %export long cracking

rangeToExport    = strcat('g9:g',string(finalRowToExport)) ;
export = xlswrite(char(finalName),alligatorCrack(:,1),'Distress Prediction',char(rangeToExport));  %export alligator cracking

rangeToExport    = strcat('j9:j',string(finalRowToExport)) ;
export = xlswrite(char(finalName),IRI,'Distress Prediction',char(rangeToExport));  %export IRI

rangeToExport    = strcat('k9:k',string(finalRowToExport)) ;
export = xlswrite(char(finalName),PSI,'Distress Prediction',char(rangeToExport));  %export PSI
```

Source Code: /distressCalculator/alligatorCrackFrontEnd.m

```
%% PRODUCT-ONE PAVEMENT DESIGN TOOL
%FRONT END CODE FOR THE DISTRESS CALCULATIONS
%
%This script will direct the calls to auxiliary functions to compute the
%intended distresses over the pavement structure for each time moment, each
%level of load and each axle type
%
%%THESE ARE THE VARIABLES THAT ARE TO BE FILLED.
%EACH CALL TO THIS SCRIPT WILL FILL UP ROW (k) OF THESE MATRICES/VECTORS.
%
% alligatorCrack = zeros(termination,length(HMAPaveDepth));   % units: per-
centage of lane area
% topDownCrack = zeros(termination,legnth(HMAPaveDepth));     % units: m/km
[MEPDG's are ft/mile]
% BOTH ARE COMPUTED WITH THE MEPDG'S RE-FIT OF THE ASPHALT INSTITUTE
% EQUATIONS (DEFAULT CALLIBRATION PARAMETERS)
%
%V0.2 2019-05-20
%   Changelog: topDownDamage variables to have only 1 column
%   (representative of distress calculation at the surface only)
%V0.1 2019-05-14
%   Changelog: adapted input to the alligatorCracking and TopDownCracking
%   compute functions to comply with the 4-dimensional arrays for strains
%   and stresses.

%% PART 0 - INITIALIZE THE DAMAGE VECTORS (AT TIMESTAMP 1), WHICH WILL BE USED
TO COMPUTE DAMAGE
if k ==1
    %these are the damage ratios for the axle passes at timestamp K
    bottomUpDamageSingleL  = zeros(termination,ACLayersNumber);
    bottomUpDamageSingle6  = zeros(termination,ACLayersNumber);
    bottomUpDamageSingle10 = zeros(termination,ACLayersNumber);
    bottomUpDamageTandem10 = zeros(termination,ACLayersNumber);
    bottomUpDamageTandem14 = zeros(termination,ACLayersNumber);
    bottomUpDamageTandem18 = zeros(termination,ACLayersNumber);
    bottomUpDamageTridem   = zeros(termination,ACLayersNumber);

    topDownDamageSingleL  = zeros(termination,1);
    topDownDamageSingle6  = zeros(termination,1);
    topDownDamageSingle10 = zeros(termination,1);
    topDownDamageTandem10 = zeros(termination,1);
    topDownDamageTandem14 = zeros(termination,1);
    topDownDamageTandem18 = zeros(termination,1);
    topDownDamageTridem   = zeros(termination,1);
end


%% PART 1 - - - BOTTOM UP ALLIGATOR CRACKING
%% compute the degree of damage for each axle type (sum of all load levels)
```

```
%%%PASS TO FUNCTION "alligatorCracking" for every axle type size of auxDamage-
BUSingleL: [HMA Layers  x 1] - its the terms of the Miner's law for all axle
load levels
%%update V2019-05-14: Correct epsHXXX from 4-D to 3-D array for a correct call
to alligatorCracking)

[aaa,bbb,ccc,~] = size(epsHsingleL);
epsHSingleLk = reshape(epsHsingleL(:,:,:,k),[aaa,bbb,ccc]);
[aaa,bbb,ccc,~] = size(epsHsingle6);
epsHSingle6k = reshape(epsHsingle6(:,:,:,k),[aaa,bbb,ccc]);
[aaa,bbb,ccc,~] = size(epsHsingle10);
epsHSingle10k = reshape(epsHsingle10(:,:,:,k),[aaa,bbb,ccc]);
[aaa,bbb,ccc,~] = size(epsHtandem10);
epsHTandem10k = reshape(epsHtandem10(:,:,:,k),[aaa,bbb,ccc]);
[aaa,bbb,ccc,~] = size(epsHtandem14);
epsHTandem14k = reshape(epsHtandem14(:,:,:,k),[aaa,bbb,ccc]);
[aaa,bbb,ccc,~] = size(epsHtandem18);
epsHTandem18k = reshape(epsHtandem18(:,:,:,k),[aaa,bbb,ccc]);
[aaa,bbb,ccc,~] = size(epsHtridem);
epsHTridemk = reshape(epsHtridem(:,:,:,k),[aaa,bbb,ccc]);


[~, auxDamage] = alligatorCracking(epsHSingleLk, axlesSingleLight(k,:),z,AC-
PaveDepth(:),EDynSingleLight(:,:,k),HMAparameters);
bottomUpDamageSingleL(k,:) = auxDamage';
[~, auxDamage] = alligatorCracking(epsHSingle6k, axlesSingle6(k,:),z,AC-
PaveDepth(:),EDynSingle6(:,:,k),HMAparameters);
bottomUpDamageSingle6(k,:) = auxDamage';
[~, auxDamage] = alligatorCracking(epsHSingle10k, axlesSingle105(k,:),z,AC-
PaveDepth(:),EDynSingle105(:,:,k),HMAparameters);
bottomUpDamageSingle10(k,:) = auxDamage';
[~, auxDamage] = alligatorCracking(epsHTandem10k, axlesTandem10(k,:),z,AC-
PaveDepth(:),EDynTandem10(:,:,k),HMAparameters);
bottomUpDamageTandem10(k,:) = auxDamage';
[~, auxDamage] = alligatorCracking(epsHTandem14k, axlesTandem14(k,:),z,AC-
PaveDepth(:),EDynTandem14(:,:,k),HMAparameters);
bottomUpDamageTandem14(k,:) = auxDamage';
[~, auxDamage] = alligatorCracking(epsHTandem18k, axlesTandem18(k,:),z,AC-
PaveDepth(:),EDynTandem18(:,:,k),HMAparameters);
bottomUpDamageTandem18(k,:) = auxDamage';
[~, auxDamage] = alligatorCracking(epsHTridemk, axlesTridem(k,:),z,AC-
PaveDepth(:),EDynTridem(:,:,k),HMAparameters);
bottomUpDamageTridem(k,:) = auxDamage';

%% compute the degree of alligator cracking for all layers at timestamp K
% alligatorCrack = zeros(termination,ACLayersNumber);   % units: percentage of
lane area // k is from 0--termination, following the shortTimestamp    %%DE-
FINED IN THE MAINCODE

c2p = -2.40874-39.748*(1+ACtotalDepth/2.54)^-2.85609;   %ACtotalDepth is in
centimeters!
c1p = -1*c2p;
cummDamage = zeros(1,ACLayersNumber);
for i = 1:ACLayersNumber
```

```
    cummDamage(i) = sum(bottomUpDamageSingleL(1:k,i)) + sum(bottomUpDamageSingle6(1:k,i)) + sum(bottomUpDamageSingle10(1:k,i)) + ...
        sum(bottomUpDamageTandem10(1:k,i)) +
sum(bottomUpDamageTandem14(1:k,i)) + sum(bottomUpDamageTandem18(1:k,i)) +
sum(bottomUpDamageTridem(1:k,i));
    alligatorCrack(k,i) = (6000/60).*(1+exp(c1p +
c2p.*log10(100*cummDamage(i)))).^-1;
end

%% PART 2 - - - TOP DOWN ALLIGATOR CRACKING
%% compute the degree of damage for each axle type
%%%PASS TO FUNCTION "topDownCracking" for every axle type
[~, auxDamage] = topDownCracking(epsHSingleLk, axlesSingleLight(k,:),z,ACPaveDepth(:),EDynSingleLight(:,:,k),HMAparameters);
topDownDamageSingleL(k,:) = auxDamage';
[~, auxDamage] = topDownCracking(epsHSingle6k, axlesSingle6(k,:),z,ACPaveDepth(:),EDynSingle6(:,:,k),HMAparameters);
topDownDamageSingle6(k,:) = auxDamage';
[~, auxDamage] = topDownCracking(epsHSingle10k, axlesSingle105(k,:),z,ACPaveDepth(:),EDynSingle105(:,:,k),HMAparameters);
topDownDamageSingle10(k,:) = auxDamage';
[~, auxDamage] = topDownCracking(epsHTandem10k, axlesTandem10(k,:),z,ACPaveDepth(:),EDynTandem10(:,:,k),HMAparameters);
topDownDamageTandem10(k,:) = auxDamage';
[~, auxDamage] = topDownCracking(epsHTandem14k, axlesTandem14(k,:),z,ACPaveDepth(:),EDynTandem14(:,:,k),HMAparameters);
topDownDamageTandem14(k,:) = auxDamage';
[~, auxDamage] = topDownCracking(epsHTandem18k, axlesTandem18(k,:),z,ACPaveDepth(:),EDynTandem18(:,:,k),HMAparameters);
topDownDamageTandem18(k,:) = auxDamage';
[~, auxDamage] = topDownCracking(epsHTridemk,
axlesTridem(k,:),z,ACPaveDepth(:),EDynTridem(:,:,k),HMAparameters);
topDownDamageTridem(k,:) = auxDamage';

%% compute the degree of top down cracking for all layers at timestamp K
% topDownCrack = zeros(termination,ACLayersNumber);     % units: m/km
% [MEPDG's are ft/mile]. 1 m/km = 1.61/0.305 ft/mi

cummDamage = zeros(1,1);
% for i = 1:ACLayersNumber UPDATE V2019-05-09. ONLY COMPUTE TOPDOWNCRACKING AT
THE SURFACE!
for i = 1:1
    cummDamage(i) = sum(topDownDamageSingleL(1:k,i)) + sum(topDownDamageSingle6(1:k,i)) + sum(topDownDamageSingle10(1:k,i)) + ...
        sum(topDownDamageTandem10(1:k,i)) + sum(topDownDamageTandem14(1:k,i))
+ sum(topDownDamageTandem18(1:k,i)) + sum(topDownDamageTridem(1:k,i));
    topDownCrack(k,i) = 10.56*1000.*(1+exp(7.0 -
3.5.*log10(100*cummDamage(i)))).^-1;
    topDownCrack(k,i) = 0.305/1.61.*topDownCrack(k,i);    %%convert ft/mi -->
m/km
end
```

Source Code: /distressCalculator/alligatorCracking

```
function [alligatorNi,alligatorDamage] = alligatorCracking(MLEstrain,
axlePasses,z,HMAlayerDepth,EDyn,HMAProperties)
%function  alligatorDamage = alligatorCracking(MLEstrain, axlePasses,z,layerDepth,
EDyn,HMAProperties)
%
%Compute the degree of Alligator cracking (BOTTOM UP) damage on each HMA layer
for a given axle type (compute at all depths of interest (bottoms of the HMA
layers), for all the weight ranges) for a given shortTimestamp(k)
%%
%Use MEPDG's original equations(eqns. 3.3.29/30 and so on). See Pt3 Chapter 3
of NCHRP 2004.
%
%Inputs:
% MLEstrain,   MAx horizontal strain matrix at each (z,r) location -as computed
by the MLE.
% axlePasses,  number of axle passes in the period of interest
% z            vector of depths tied to MLEStrain     [m]
% HMAlayerDepth   vector stating the thickness of the HMA layers [cm]
% EDyn         dynamic modulus [in PSI] for the HMA layers for the given axle
type and load value (compatible with MLEStrain) AT A SINGLE TIMESTAMP (t(k)).
% HMAPropertiesVector with the Mix properties for the HMA layers as imported
from the dataImport sheet. Need Air voids and bitumen content in perc. volume
%
%Outputs:
% alligatorNi     [size numHMALayers x numLoadLevels]: number of admissible
passes for HMA layer (i) and load level (j)
% alligatorDamage [size numHMALayers x 1]: degree of damage done by all the
passes of the axle type of the given type (sum of all weights) at timeStamp
t(k).
%
%%Assumption: for a given weight range, the radial position that will be
summed to the admissible number of passes (and so to the alligatorDamage
value) is that with the least admissible number of passes (Nf) on all the HMA
layers.
%
%THIS IS THE "UNCALLIBRATED" VERSION OF THE RUTTING MODEL, THE EQUATIONS WERE
PROGRAMMED HEREIN AS THEY HAVE BEEN REPORTED IN THE MEPDG GUIDE. THE CALLIBRATION
EFFORT MAY EVENTUALLY LEAD TO RE-WRITING THIS COMPUTER CODE!
%
%V0.2 - 2019-04-04:
%   Changelog: corrected error in k1 computation (it was summing HMALayerDepth
twice!)
%v0.1 - 2019-04-03:
%   Changelog: Corrected equation for k1 -> it had the top-dn equation and not
the bottom-up equation
%   Corrected M parameter equation, Vb and Va were reverted!
%   Force the Eps_H term in the N_admissible equation to use the absoulte
%   value of the horizontal strain (cause if Eps_H is negative it goes to the
Cmplx plane)
%V0.0 - 2019-02-22
```

```
%% code begins
airVoids = HMAProperties(:,2);  %retrieve air voids vol.. from import.
bitCont  = HMAProperties(:,3);  %retrieve bitumen content from import.
M        = 4.84.*(bitCont./(airVoids+bitCont) - 0.69);     %parameter to equa-
tion 3.3.29

%do some needed calculations
HMAlayerDepth = HMAlayerDepth*.01;   %%convert from cm to meters to compare
with Z
HMAlayerDepth = cumsum(HMAlayerDepth); %and convert thickness to depth.
k1            = 0.000398 + 0.003602*(1+exp(11.02-3.49*(HMAlayerDepth(end))/
0.0254))^(-1);  %%patched v2019-04-04. Since HMAlayerDepth is cumsummed above,
the last entry is the total HMA layer depth!
k1            = 1/k1;             %parameter k1 FOR BOTTOM-UP CRACKING. AS PER
EQUATION 3.3.30.  NOTE: k1 is a scalar (or should be...)

%Locate the MLE strain rows that correspond to the base of the HMA layers.
auxPosZ = zeros(length(HMAlayerDepth),1);
for i = 1:length(HMAlayerDepth)
    aux = find(abs(z-HMAlayerDepth(i))<0.001);
    if ~isempty(aux)
        auxPosZ(i)  = aux;
    end
end
strainHMA = MLEstrain(auxPosZ,:,:);   %get the strain values where z is the
base of the HMA layers, all r positions and all levels of load!
[numHMALayers,radioPositions,numLoadLevels] = size(strainHMA);

%% get the admissible number of passes (NF(z,r,load)) for each z and radial
position and each load level. [z = each HMA layer]

NF = zeros(size(strainHMA));    %size is HMA LAYERS X RADIAL POSITIONS X LOAD
LEVEL
alligatorNi = zeros(numHMALayers,numLoadLevels);

%solve for each load level (size of the axlePasses vector) AND radial position
(size of retrieved from the strainHMA 3-D array). Patched v2019-04-04. Expo-
nent to strainHMA badly written!
for i = 1:numLoadLevels
    for j = 1:radioPositions
        NF(:,j,i) = (0.00432*k1).*10.^M .*(abs(strainHMA(:,j,i))).^(-
3.9492).*EDyn(:,i).^(-1.281);
    end
    %%now, for each load level, get a single set of NF -> alligatorNi
    %get the radial position that brings the smallest NF,
    minNF = min(NF(:,:,i));
    [~,minCol,~] = ind2sub(size(NF(:,:,i)),find(NF(:,:,i) == minNF));   %this
function call gives the coordinates x,y (row/col/stack??) for the position in
NF(:,:,i) where the minNF is located

    %and pass that column for all stacks to alligatorNi below. Just in case,
put the first entry of the minNF case there are many occasions.
    alligatorNi(:,i) = NF(:,minCol(1),i);
end
```

```
%% compute the degree of damage done by these axles

alligatorRatios = zeros(numHMALayers,numLoadLevels);  %compute these sepa-
rately for each HMA layer
alligatorDamage = zeros(numHMALayers,1);

for i = 1:numHMALayers
    alligatorRatios(i,:) = axlePasses./alligatorNi(i,:);
    alligatorDamage(i) = sum(alligatorRatios(i,:));
end

end
```

Source Code: /distressCalculator/IRIclimateSiteFactor

```
function [monthlyRainStd,avgAnnualRain,tempFI] = IRIclimateSiteFactor(time-
stamp,rain,temperature)
% function [monthlyRainStd,avgAnnualRain,tempFI] = IRIclimateSiteFactor(time-
stamp,rain,temperature)
%
%Ancillary function to the IRI calculation module. It computes the climate-
based terms of the Site-Factor equation from the project's site simulated cli-
mate records.
%
%INPUT:
%timestamp - time vector of the climate variables [in Matlab's timestamp for-
mat - 1 hour long!]
%rain      - rainfall record [mm]
%temperature-air temperature record [degC]
%
%OUTPUT:
% monthlyRainStd - standard deviation of the monthly total rainfall [mm]
% avgAnnualRain  - average annual rain for the entire input period [mm]
% tempFI         - average annual freezing index (max avg daily temp - min avg
daily temp) [deg C]
%
%V0.0 2019-03-04

timeMatrix = datevec(timestamp);
yearStart = timeMatrix(1,1);
yearEnd   = timeMatrix(end,1);

%% 1 - compute the avg yearly rain
if yearEnd>yearStart
    auxYears = yearStart:1:yearEnd;
else
    auxYears = yearStart;  %just-in-case if statement, case I'm runnign a 1-yr
design.
end
annRain   = zeros(length(auxYears),1);

if length(auxYears) ==1
    annRain   = sum(rain)*365*24/length(timestamp);  %if I'm modeling a sin-
gle-year design, the annRain is just the sum of the rain entries. This ratio
multiplier here will extrapolate teh average annual rain to a 365-day year
(case I'm modelling less than 1 year)
    avgAnnualRain = annRain;
else
    %start accumulating rain for all the years. Add the trick for the 1-yr
%case (to extrapolate the annual rain in the cases of timestamps starting
half-way through a year)
    yearNo      = 1;
    auxDayCounter = 1;
    annRain(yearNo) = rain(1);
    auxYearID = auxYears(1);
    for j = 2:length(timestamp)
```

```
        dateTime = timeMatrix(j);
        if j<length(timestamp)
            if auxYearID == dateTime(1)   %the year of the date of the j-th
record in timestamp matches the year I'm summing up
                annRain(yearNo) = annRain(yearNo) + rain(j);
                auxDayCounter = auxDayCounter+1;
            else
                annRain(yearNo) = annRain(yearNo)*365*24/auxDayCounter;
                %change year / reset variables
                yearNo = yearNo+1;
                auxDayCounter   = 1;
                annRain(yearNo) = rain(j);
            end
        else
            %termination [CASE J == last] - pull out results
            annRain(yearNo) = annRain(yearNo)*365*24/auxDayCounter;
        end
    end
    avgAnnualRain = mean(annRain);
end

%% 2 - compute monthly rainfall records. I'm lazy and will copy the loop above
and modify accordingly (I don't feel like merging all the searches together)
%
monthStart = timeMatrix(1,2);
monthEnd   = timeMatrix(end,2) + 12*(yearEnd-yearStart);
auxMonths = monthStart:1:monthEnd;

monRain  = zeros(length(auxMonths),1);

%%ASSUMING MORE THAN 1 MONTH IS BEING MODELED HERE!

%start accumulating rain for all the years. Add the trick for the 1-yr case
(to extrapolate the annual rain in the cases of timestamps starting half-way
through a year)
monthNo     = 1;
auxDayCounter = 1;
monRain(monthNo) = rain(1);
auxMonID = auxMonths(1);
for j = 2:length(timestamp)
    dateTime = timeMatrix(j,:);
    if j<length(timestamp)
        if auxMonID == dateTime(2)+12*(dateTime(1)-yearStart)   %the month of
the date of the j-th record in timestamp matches the year I'm summing up
            monRain(monthNo) = monRain(monthNo) + rain(j);
            auxDayCounter = auxDayCounter+1;
        else
            monRain(monthNo) = monRain(monthNo)*30*24/auxDayCounter;
            %change month / reset variables
            monthNo = monthNo+1;
            auxMonID = auxMonths(monthNo);
            auxDayCounter   = 1;
            monRain(monthNo) = rain(j);
        end
    else
        %termination [CASE J == last] - pull out results
        monRain(monthNo) = monRain(monthNo)*30*24/auxDayCounter;
    end
end
monthlyRainStd = std(monRain);

%% 3 - do the freezing index calculations.
%a) compute maximum and minimum temp for every day
%b) get the avg daily temp as (0.5* [max+min])
%c) and then for each year FI(year) = max(avgTemp)-min(avg(Temp)

%%important note: when I have truncated years (simulation started on a date
other than Jan 1 and ended other than dec 31), the Avg FI would not be exact
(the truncated years would not give appropriate FI values)

dayStart = floor(timestamp(1));
dayEnd   = floor(timestamp(end));
auxDays  = dayStart:1:dayEnd;

tmaxDay = temperature(1);
tminDay = temperature(1);
tavgDay  = zeros(length(auxDays),1);
tavgDay(1) = mean([tmaxDay,tminDay]);

dayNo      = 1;
auxDayID   = dayStart;

%part A) get the min, max, avg temperature for each day

for j = 2:length(timestamp)
%     dateTime = timeMatrix(j);
    if j<length(timestamp)
        if auxDayID == floor(timestamp(j))
            tmaxDay = max([tmaxDay;temperature(j)]);
            tminDay = min([tminDay;temperature(j)]);
        else
            tavgDay(dayNo)  = (tmaxDay+tminDay)/2;
            %change year / reset variables
            dayNo = dayNo+1;
            auxDayID = auxDays(dayNo);
            tmaxDay = temperature(j);
            tminDay = temperature(j);
        end
    else
        %termination [CASE J == last] - pull out results
        tavgDay(dayNo) = (tmaxDay+tminDay)/2;
    end
end

% part B) get the FI for each year

yearRange = datevec(auxDays);
yearRange = yearRange(:,1);    %%vector with all the year values in yearRange
should pair with  - - auxYears

annualFI = zeros(length(auxYears),1);

for i = 1:length(annualFI)
    rangeToSearch = find(yearRange ==auxYears(i));
    maxTavg = max(tavgDay(rangeToSearch));
    minTavg = min(tavgDay(rangeToSearch));
    annualFI(i) = maxTavg - minTavg;
end

tempFI = mean(annualFI);

end %endfunction
```

Source Code: /distressCalculator/IRIPSICalcFIrontEnd.m

```
%% PRODUCT-ONE PAVEMENT DESIGN TOOL
%FRONT END CODE FOR THE DISTRESS CALCULATIONS
%
%This script will direct the calls to auxiliary functions to compute the in-
tended distresses over the pavement structure for each time moment, each level
of load and each axle type.
%
%
%V0.1 2019-04-04:
%Changelog: Corrected rut depth term, that was in milimiters (while rut depth
was returning from the rutDepthCompute in meters).
%V0.0 2019-03-04
%Feature: Replaced default MEPDG's formula for HMA on unbound base IRI. [eqn
3.3.74 and branches] with simplified expression featured in newer MEPDg imple-
mentations (in AASHTO 2015; Garber & Hoel XXX)

%%code begins
%% compute climate variables for site factor. If already exist (i computed
them on a previous loop), do not calculate
if ~exist('tempFI','var')
    [rainStDev,rainAvg,tempFI] =
IRIclimateSiteFactor(longTimestamp,rainFall,airTemp);
    %climateSateFactor gives rainAVG and rainStDev in mm, tempFI in degC-days
else
   %do nothing
end

%% COMPUTE IRI
%part A) Site Factor
%use the simplified AASHTO (2008) formula instead of the newest v2015 [doesn't
need the P02 of the subgrade soil, which is hardly ever reported in sieving
results]
paveAge = (1/365)*(shortTimestamp(k)-shortTimestamp(1));  %age of the pave-
ment, in years
% subGrPI  = granPI(end);      %% PLASTICITY INDEX OF THE SUBGRADE
% subGrP200= granP200(end);    %% P200 of the subgrade.
SF = 0.02003*(1+granPI(end))+0.007947*(1+rainAvg/
25.4)+0.000636*(1+tempFI*9/5+32);
SF = paveAge*SF;

% topDownCracking is received in m/km. Convert to length in ft/mi!!!.
%<rutDepth is received in mm - convert to inches!
IRI(k) = initialIRI*1.61/0.0254 + 0.0150*SF+0.400*alligatorCrack(k,1) +
0.400*topDownCrack(k,1)*1.61/0.305*1 + 0.0080*0 + 40.0*rutDepth(k,end)/0.0254;
%THE EQUATION ABOVE GIVES IRI(k) in inches/mile; CONVERT UNITS to m/km
 IRI(k) = IRI(k)*0.0254/1.61;

%% compute PSI(IRI). Use Al-Omari and Darter's (1992/4?) formula
PSI(k) = 5.00*exp(-0.26*IRI(k));%IRI assumed to be here in m/km.
```

Source Code: /distressCalculator/RutDepthCalcFrontEnd.m

```
%% PRODUCT-ONE PAVEMENT DESIGN TOOL
%FRONT END CODE FOR THE DISTRESS CALCULATIONS
%
%This script will direct the calls to auxiliary functions to compute the intended distresses over the pavement structure for each time moment, each level of load and each axle type.
%%THESE ARE THE VARIABLES THAT ARE TO BE FILLED. EACH CALL TO THIS SCRIPT WILL FILL UP ROW (k) OF THESE MATRICES/VECTORS.

% rutDepth=zeros(termination,1);           % metric Units [m]
%
%V0.4 2019-05-14
%   Changelog: adapt code to new 4-D stress and strain matrices (added time variable as 4th dimension)
%V0.3 Mother's Day: 2019-05-12
%   Changelog: Simplified rut-depth calculator fore each axle.
%   Using only the location UNDER ONE OF THE AXLE'S WHEELS
%   >>Need to clean up non-nec variables if successful<<
%V0.2 St.Patrick's Day: 2019-03-17
%   Changelog: Tuned up to account for the "strain hardening?" methodology to account for loads on previous passes.
%V0.1 2019-02-22
%---first debugging completed 2019-02-26
%   Changelog: First complete beta. Separate Rut depth calculations from other distresses
%V0.0 Valentine's Day: 2019-02-14

%% CODE BEGINS
%% Update V2019-03-17:
%If first run of the code, initialize the "cache" of plastic strains from previous runs and fill with zeros. The code will retrieve the numbers from %here as needed.
%Initialize with -9999s cause this number the rutDepthCompute will understand refers to "no previous iteration, Neq = 0"
%%update V2019-05-14: convert to 4-D arrays (store time variable; calculation results will be kept for quality check).
if k ==1
   eplCacheSingleL   = -9999*ones(length(paveDepths),nrsL,length(axlesSingleLWeights),termination);
   eplCacheSingle6   = -9999*ones(length(paveDepths),nrs6,length(axlesSingle6Weights),termination);
   eplCacheSingle10  = -9999*ones(length(paveDepths),nrs10,length(axlesSingle10Weights),termination);
   eplCacheTandem10  = -9999*ones(length(paveDepths),nrTa10,length(axlesTandem10Weights),termination);
   eplCacheTandem14  = -9999*ones(length(paveDepths),nrTa14,length(axlesTandem14Weights),termination);
   eplCacheTandem18  = -9999*ones(length(paveDepths),nrTa18,length(axlesTandemWeights),termination);
   eplCacheTridem    = -9999*ones(length(paveDepths),nrTr,length(axlesTridemWeights),termination);
   %%UPDATE V2019-05-14: Add 4th dimension (time) to all rut-depth variables. I'm storing the results for post-proc and quality check
    auxrutDepthHMASingleL        = zeros(ACLayersNumber,nrsL,length(axlesSingleLWeights),termination);
    auxrutDepthGranSingleL       = zeros(length(paveDepths)-1-ACLayersNumber,nrsL,length(axlesSingleLWeights),termination);
    auxrutDepthSubGradeSingleL   = zeros(1,nrsL,length(axlesSingleLWeights),termination);
    auxMaxRutSingleL             = zeros(nrsL,length(axlesSingleLWeights),termination);    %%put here the sum of all rut depths
    % auxMaxRutPosition            = zeros(length(axlesSingleLWeights)); %%get the r position of the maximum rut depth for all weights  %%V2019-05-14: NO LONGER NEEDED

    auxrutDepthHMASingle6        = zeros(ACLayersNumber,nrs6,length(axlesSingle6Weights),termination);
    auxrutDepthGranSingle6       = zeros(length(paveDepths)-1-ACLayersNumber,nrs6,length(axlesSingle6Weights),termination);
    auxrutDepthSubGradeSingle6   = zeros(1,nrs6,length(axlesSingle6Weights),termination);
    auxMaxRutSingle6             = zeros(nrs6,length(axlesSingle6Weights),termination);    %%put here the sum of all rut depths
% auxMaxRutPosition            = zeros(length(axlesSingle6Weights)); % %%get the r position of the maximum rut depth for all weights %%V2019-05-14: NO LONGER NEEDED.

    auxrutDepthHMASingle10        = zeros(ACLayersNumber,nrs10,length(axlesSingle10Weights),termination);
    auxrutDepthGranSingle10       = zeros(length(paveDepths)-1-ACLayersNumber,nrs10,length(axlesSingle10Weights),termination);
    auxrutDepthSubGradeSingle10   = zeros(1,nrs10,length(axlesSingle10Weights),termination);
    auxMaxRutSingle10             = zeros(nrs10,length(axlesSingle10Weights),termination);    %%put here the sum of all rut depths

    auxrutDepthHMATandem10        = zeros(ACLayersNumber,nrTa10,length(axlesTandem10Weights),termination);
    auxrutDepthGranTandem10       = zeros(length(paveDepths)-1-ACLayersNumber,nrTa10,length(axlesTandem10Weights),termination);
    auxrutDepthSubGradeTandem10   = zeros(1,nrTa10,length(axlesTandem10Weights),termination);
    auxMaxRutTandem10             = zeros(nrTa10,length(axlesTandem10Weights),termination);    %%put here the sum of all rut depths
    % auxMaxRutPosition            = zeros(length(axlesTandem10Weights));  % %get the r position of the maximum rut depth for all weights  <<NO LONGER NEEDED.

    auxrutDepthHMATandem14        = zeros(ACLayersNumber,nrTa14,length(axlesTandem14Weights),termination);
    auxrutDepthGranTandem14       = zeros(length(paveDepths)-1-ACLayersNumber,nrTa14,length(axlesTandem14Weights),termination);
    auxrutDepthSubGradeTandem14   = zeros(1,nrTa14,length(axlesTan-
dem14Weights),termination);
    auxMaxRutTandem14             =
zeros(nrTa14,length(axlesTandem14Weights),termination);    %%put here the sum
of all rut depths

    auxrutDepthHMATandem18        = zeros(ACLayersNumber,nrTa18,length(axle-
sTandemWeights),termination);
    auxrutDepthGranTandem18       = zeros(length(paveDepths)-1-
ACLayersNumber,nrTa18,length(axlesTandemWeights),termination);
    auxrutDepthSubGradeTandem18   =
zeros(1,nrTa18,length(axlesTandemWeights),termination);
    auxMaxRutTandem18             =
zeros(nrTa18,length(axlesTandemWeights),termination);    %%put here the sum of
all rut depths
    % auxMaxRutPosition             = zeros(length(axlesTandemWeights));
%%get the r position of the maximum rut depth for all weights

    auxrutDepthHMATridem        =
zeros(ACLayersNumber,nrTr,length(axlesTridemWeights),termination);
    auxrutDepthGranTridem       = zeros(length(paveDepths)-1-
ACLayersNumber,nrTr,length(axlesTridemWeights),termination);
    auxrutDepthSubGradeTridem   = zeros(1,nrTr,length(axlesTridemWeights),ter-
mination);
    auxMaxRutTridem             = zeros(nrTr,length(axlesTridemWeights),termi-
nation);    %%put here the sum of all rut depths
    % auxMaxRutPosition             = zeros(length(axlesTridemWeights));
%%get the r position of the maximum rut depth for all weights

end

%% light weight single axle
%%NOTE: nrsL and all the nr**; z; epsZ*** have been defined in the MLE_fron-
tEnd
%a) compute rutting for each type of axle.

maxRutDepthHMASingleL        = zeros(ACLayersNumber,1,length(axlesSingleL-
Weights));
maxRutDepthGranSingleL       = zeros(length(paveDepths)-1-
ACLayersNumber,1,length(axlesSingleLWeights));
maxRutDepthSubGradeSingleL   = zeros(1,1,length(axlesSingleLWeights));

for j = 1:length(axlesSingleLWeights)
    for r = 1:nrsL   %%<< update v2019-05-14:: Compute all positions, but pass
POSITION r = 1 ONLY  as maxRutDepth (under the wheel)
        %%update V2019-03-17:: pass on previous-iteration to rutDepthCompute
(strain hardening approach)
        eplHMAprev = eplCacheSingleL(1:ACLayersNumber,r,j,k);
        eplGranprev= eplCacheSingleL(ACLayersNumber+1:end-1,r,j,k);
        eplSGprev  = eplCacheSingleL(end,r,j,k);
        %%update V2019-05-14: Add 4th-dimension to call to rutDepthCompute
%and to eplCacheXXX

[auxrutDepthHMASingleL(:,r,j,k),auxrutDepthGranSingleL(:,r,j,k),auxrutDepth-
```

```
SubGradeSingleL(:,r,j,k),eplHMA,eplGran,eplSG] =
rutDepthCompute(epsZsingleL(:,r,j,k),axlesSingleLight(k,j),z,paveDepths,shortAsphLyrTemp(k,2:ACLayersNumber+1),layersMoisture(k,:),granDens,eplHMAprev,eplG
ranprev,eplSGprev);
        eplCacheSingleL(:,r,j,k+1) = [eplHMA;eplGran;eplSG];

        %apply simplification 2019-02-20: locate point where overall maximum
rutting occurs (sum all stacks)
        auxMaxRutSingleL(r,j,k) = sum(auxrutDepthHMASingleL(:,r,j,k))
+sum(auxrutDepthGranSingleL(:,r,j,k))
+sum(auxrutDepthSubGradeSingleL(:,r,j,k));  %Get the total rut depth at each
radial location and each load level
    end
   %get the position where the maximum rutting occurs for each weight level
   %%UPDATE V2019-05-14: store all auxMatRutPosition, but save the r==1 %case
as maxRutDepth (under the wheel)
%    aux =  find(auxMaxRut(:,j)==max(auxMaxRut(:,j)));
%    if ~isempty(aux)
%        auxMaxRutPosition(j) = aux(1);
%    else
%        auxMaxRutPosition(j) = 1;
%    end
    maxRutDepthHMASingleL(:,1,j)        = auxrutDepthHMASingleL(:,1,j,k);
%    maxRutDepthHMASingleL(:,1,j)        = auxrutDepthHMASingleL(:,auxMaxRutPosition(j),j);
   maxRutDepthGranSingleL(:,1,j)       = auxrutDepthGranSingleL(:,1,j,k);
   maxRutDepthSubGradeSingleL(:,1,j)   = auxrutDepthSubGradeSingleL(:,1,j,k);
end

%% single-wheel single axle

maxRutDepthHMASingle6        = zeros(ACLayersNumber,1,length(axlesSingle6Weights));
maxRutDepthGranSingle6       = zeros(length(paveDepths)-1-
ACLayersNumber,1,length(axlesSingle6Weights));
maxRutDepthSubGradeSingle6   = zeros(1,1,length(axlesSingle6Weights));

for j = 1:length(axlesSingle6Weights)
    for r = 1:nrs6  %<< update v2019-05-14:: Compute all positions, but pass
POSITION r = 1 ONLY  as maxRutDepth (under the wheel)
        %%updateV2019-03-17:: pass on previous-iteration
        eplHMAprev = eplCacheSingle6(1:ACLayersNumber,r,j,k);
        eplGranprev= eplCacheSingle6(ACLayersNumber+1:end-1,r,j,k);
        eplSGprev  = eplCacheSingle6(end,r,j,k);

[auxrutDepthHMASingle6(:,r,j,k),auxrutDepthGranSingle6(:,r,j,k),auxrutDepthSubGradeSingle6(:,r,j,k),eplHMA,eplGran,eplSG] =
rutDepthCompute(epsZsingle6(:,r,j,k),axlesSingle6(k,j),z,paveDepths,shortAsphLyrTemp(k,2:end),layersMoisture(k,:),granDens,eplHMAprev,eplGranprev,eplSGprev);
        eplCacheSingle6(:,r,j,k+1) = [eplHMA;eplGran;eplSG];

        %apply simplification 2019-02-20: locate point where overall maximum
rutting occurs (sum all stacks)
```

```
        auxMaxRutSingle6(r,j,k) = sum(auxrutDepthHMASingle6(:,r,j,k))
+sum(auxrutDepthGranSingle6(:,r,j,k))
+sum(auxrutDepthSubGradeSingle6(:,r,j,k));  %Get the total rut depth at each
radial location and each load level
    end
    %get the position where the maximum rutting occurs for each weight level
    %%UPDATE V2019-05-14: store all auxMatRutPosition, but save the r==1 %case
as maxRutDepth (under the wheel)
%     aux =  find(auxMaxRut(:,j)==max(auxMaxRut(:,j)));
%     if ~isempty(aux)
%        auxMaxRutPosition(j) = aux(1);
%    else
%        auxMaxRutPosition(j) = 1;
%    end
%    maxRutDepthHMASingle6(:,1,j)         = auxrutDepthHMASingle6(:,auxMaxRutPosition(j),j);
   maxRutDepthHMASingle6(:,1,j)        = auxrutDepthHMASingle6(:,1,j,k);
   maxRutDepthGranSingle6(:,1,j)       = auxrutDepthGranSingle6(:,1,j,k);
   maxRutDepthSubGradeSingle6(:,1,j)   = auxrutDepthSubGradeSingle6(:,1,j,k);
end

%% dual-wheel single axle

maxRutDepthHMASingle10        = zeros(ACLayersNumber,1,length(axlesSingle10Weights));
maxRutDepthGranSingle10       = zeros(length(paveDepths)-1-
ACLayersNumber,1,length(axlesSingle10Weights));
maxRutDepthSubGradeSingle10   = zeros(1,1,length(axlesSingle10Weights));

for j = 1:length(axlesSingle10Weights)
    for r = 1:nrs10     %%<< update v2019-05-14:: Compute all positions, but
pass POSITION r = 4 ONLY   (under the wheel)
        %%updateV2019-03-17:: pass on previous-iteration
        eplHMAprev = eplCacheSingle10(1:ACLayersNumber,r,j,k);
        eplGranprev= eplCacheSingle10(ACLayersNumber+1:end-1,r,j,k);
        eplSGprev  = eplCacheSingle10(end,r,j,k);

[auxrutDepthHMASingle10(:,r,j,k),auxrutDepthGranSingle10(:,r,j,k),auxrutDepthSubGradeSingle10(:,r,j,k),eplHMA,eplGran,eplSG] = rutDepthCompute(epsZsingle10(:,r,j,k),axlesSingle105(k,j),z,paveDepths,shortAsphLyrTemp(k,2:end),layersMoisture(k,:),granDens,eplHMAprev,eplGranprev,eplSGprev);
         eplCacheSingle10(:,r,j,k+1) = [eplHMA;eplGran;eplSG];

        %apply simplification 2019-02-20: locate point where overall maximum
rutting occurs (sum all stacks)
        auxMaxRutSingle10(r,j,k) = sum(auxrutDepthHMASingle10(:,r,j,k))
+sum(auxrutDepthGranSingle10(:,r,j,k))
+sum(auxrutDepthSubGradeSingle10(:,r,j,k));  %Get the total rut depth at each
radial location and each load level
    end
     %get the position where the maximum rutting occurs for each weight level
    %%UPDATE V2019-05-14: store all auxMatRutPosition, but save the r==4
   maxRutDepthHMASingle10(:,1,j)        = auxrutDepthHMASingle10(:,4,j,k);
   maxRutDepthGranSingle10(:,1,j)       = auxrutDepthGranSingle10(:,4,j,k);
   maxRutDepthSubGradeSingle10(:,1,j)    =
auxrutDepthSubGradeSingle10(:,4,j,k);
end

%% single-wheel tandem axle

maxRutDepthHMATandem10        = zeros(ACLayersNumber,1,length(axlesTan-
dem10Weights));
maxRutDepthGranTandem10       = zeros(length(paveDepths)-1-
ACLayersNumber,1,length(axlesTandem10Weights));
maxRutDepthSubGradeTandem10   = zeros(1,1,length(axlesTandem10Weights));

for j = 1:length(axlesTandem10Weights)
    for r = 1:nrTa10    %%update v2019-05-12::  USE POSITION r = 4 ONLY   (un-
der the wheel)
        %%updateV2019-03-17:: pass on previous-iteration
        eplHMAprev = eplCacheTandem10(1:ACLayersNumber,r,j,k);
        eplGranprev= eplCacheTandem10(ACLayersNumber+1:end-1,r,j,k);
        eplSGprev  = eplCacheTandem10(end,r,j,k);

[auxrutDepthHMATandem10(:,r,j,k),auxrutDepthGranTandem10(:,r,j,k),auxrutDepth-
SubGradeTandem10(:,r,j,k),eplHMA,eplGran,eplSG] = rutDepthCompute(epsZtan-
dem10(:,r,j,k),axlesTandem10(k,j),z,paveDepths,shortAsphLyrTemp(k,2:end),lay-
ersMoisture(k,:),granDens,eplHMAprev,eplGranprev,eplSGprev);
         eplCacheTandem10(:,r,j,k+1) = [eplHMA;eplGran;eplSG];
        %apply simplification 2019-02-20: locate point where overall maximum
rutting occurs (sum all stacks)
        auxMaxRutTandem10(r,j,k) = sum(auxrutDepthHMATandem10(:,r,j,k))
+sum(auxrutDepthGranTandem10(:,r,j,k))
+sum(auxrutDepthSubGradeTandem10(:,r,j,k));  %Get the total rut depth at each
radial location and each load level
    end
    %get the position where the maximum rutting occurs for each weight level
   %%UPDATE V2019-05-14: store all auxMatRutPosition, but save the r==4 %case
as maxRutDepth (under the wheel)
%      aux =  find(auxMaxRut(:,j)==max(auxMaxRut(:,j)));
%      if ~isempty(aux)
%         auxMaxRutPosition(j) = aux(1);
%     else
%         auxMaxRutPosition(j) = 1;
%     end
%     maxRutDepthHMATandem10(:,1,j)
%        = auxrutDepthHMATandem10(:,auxMaxRutPosition(j),j);
   maxRutDepthHMATandem10(:,1,j)        = auxrutDepthHMATandem10(:,4,j,k);
   maxRutDepthGranTandem10(:,1,j)       = auxrutDepthGranTandem10(:,4,j,k);
   maxRutDepthSubGradeTandem10(:,1,j)   =
auxrutDepthSubGradeTandem10(:,4,j,k);
end

%% non-homogeneous tandem axle

maxRutDepthHMATandem14        = zeros(ACLayersNumber,1,length(axlesTan-
dem14Weights));
```

```
maxRutDepthGranTandem14       = zeros(length(paveDepths)-1-
ACLayersNumber,1,length(axlesTandem14Weights));
maxRutDepthSubGradeTandem14   = zeros(1,1,length(axlesTandem14Weights));

for j = 1:length(axlesTandem14Weights)
    for r = 1:nrTa14
        eplHMAprev = eplCacheTandem14(1:ACLayersNumber,r,j,k);
        eplGranprev= eplCacheTandem14(ACLayersNumber+1:end-1,r,j,k);
        eplSGprev  = eplCacheTandem14(end,r,j,k);

[auxrutDepthHMATandem14(:,r,j,k),auxrutDepthGranTandem14(:,r,j,k),auxrutDepth-
SubGradeTandem14(:,r,j,k),eplHMA,eplGran,eplSG] = rutDepthCompute(epsZtan-
dem14(:,r,j,k),axlesTandem14(k,j),z,paveDepths,shortAsphLyrTemp(k,2:end),lay-
ersMoisture(k,:),granDens,eplHMAprev,eplGranprev,eplSGprev);
        eplCacheTandem14(:,r,j,k+1) = [eplHMA;eplGran;eplSG];
        %apply simplification 2019-02-20: locate point where overall maximum
rutting occurs (sum all stacks)
        auxMaxRutTandem14(r,j,k) = sum(auxrutDepthHMATandem14(:,r,j,k))
+sum(auxrutDepthGranTandem14(:,r,j,k))
+sum(auxrutDepthSubGradeTandem14(:,r,j,k));  %Get the total rut depth at each
radial location and each load level
    end
    %get the position where the maximum rutting occurs for each weight level
    %%UPDATE V2019-05-14: store all auxMatRutPosition, but save the r==1 %case
as maxRutDepth (under the wheel)
%     aux =  find(auxMaxRut(:,j)==max(auxMaxRut(:,j)));
%     if ~isempty(aux)
%        auxMaxRutPosition(j) = aux(1);
%    else
%        auxMaxRutPosition(j) = 1;
%    end
%    maxRutDepthHMATandem14(:,1,j)        = auxrutDepthHMATandem14(:,aux-
MaxRutPosition(j),j);
   maxRutDepthHMATandem14(:,1,j)        = auxrutDepthHMATandem14(:,1,j,k);
   maxRutDepthGranTandem14(:,1,j)       = auxrutDepthGranTandem14(:,1,j,k);
   maxRutDepthSubGradeTandem14(:,1,j)   =
auxrutDepthSubGradeTandem14(:,1,j,k);
end

%% dual wheel tandem axle

maxRutDepthHMATandem18        = zeros(ACLayersNumber,1,length(axle-
sTandemWeights));
maxRutDepthGranTandem18       = zeros(length(paveDepths)-1-
ACLayersNumber,1,length(axlesTandemWeights));
maxRutDepthSubGradeTandem18   = zeros(1,1,length(axlesTandemWeights));

for j = 1:length(axlesTandemWeights)
    for r = 1:nrTa18
        eplHMAprev = eplCacheTandem18(1:ACLayersNumber,r,j,k);
        eplGranprev= eplCacheTandem18(ACLayersNumber+1:end-1,r,j,k);
        eplSGprev  = eplCacheTandem18(end,r,j,k);

[auxrutDepthHMATandem18(:,r,j,k),auxrutDepthGranTandem18(:,r,j,k),auxrutDepth-
SubGradeTandem18(:,r,j,k),eplHMA,eplGran,eplSG] = rutDepthCompute(epsZtan-
dem18(:,r,j,k),axlesTandem18(k,j),z,paveDepths,shortAsphLyrTemp(k,2:end),lay-
ersMoisture(k,:),granDens,eplHMAprev,eplGranprev,eplSGprev);
        eplCacheTandem18(:,r,j,k+1) = [eplHMA;eplGran;eplSG];
        %apply simplification 2019-02-20: locate point where overall maximum
rutting occurs (sum all stacks)
        auxMaxRutTandem18(r,j,k) = sum(auxrutDepthHMATandem18(:,r,j,k))
+sum(auxrutDepthGranTandem18(:,r,j,k))
+sum(auxrutDepthSubGradeTandem18(:,r,j,k));  %Get the total rut depth at each
radial location and each load level
    end
    %get the position where the maximum rutting occurs for each weight level
    %%UPDATE V2019-05-14: store all auxMatRutPosition, but save the r==10
%case as maxRutDepth (under a wheel)
%     aux =  find(auxMaxRut(:,j)==max(auxMaxRut(:,j)));
%     if ~isempty(aux)
%        auxMaxRutPosition(j) = aux(1);
%    else
%        auxMaxRutPosition(j) = 1;
%    end
%    maxRutDepthHMATandem18(:,1,j)        = auxrutDepthHMATandem18(:,aux-
MaxRutPosition(j),j);
   maxRutDepthHMATandem18(:,1,j)        = auxrutDepthHMATandem18(:,10,j,k);
   maxRutDepthGranTandem18(:,1,j)       = auxrutDepthGranTandem18(:,10,j,k);
   maxRutDepthSubGradeTandem18(:,1,j)   =
auxrutDepthSubGradeTandem18(:,10,j,k);
end

%% tridem axle

maxRutDepthHMATridem        =
zeros(ACLayersNumber,1,length(axlesTridemWeights));
maxRutDepthGranTridem       = zeros(length(paveDepths)-1-
ACLayersNumber,1,length(axlesTridemWeights));
maxRutDepthSubGradeTridem   = zeros(1,1,length(axlesTridemWeights));

for j = 1:length(axlesTridemWeights)
    for r=1:nrTr    %
        eplHMAprev = eplCacheTridem(1:ACLayersNumber,r,j,k);
        eplGranprev= eplCacheTridem(ACLayersNumber+1:end-1,r,j,k);
        eplSGprev  = eplCacheTridem(end,r,j,k);
        [auxrutDepthHMATridem(:,r,j,k),auxrutDepthGranTridem(:,r,j,k),auxrut-
DepthSubGradeTridem(:,r,j,k),eplHMA,eplGran,eplSG] = rutDepthCompute(epsZtri-
dem(:,r,j,k),axlesTridem(k,j),z,paveDepths,shortAsphLyrTemp(k,2:end),lay-
ersMoisture(k,:),granDens,eplHMAprev,eplGranprev,eplSGprev);
        eplCacheTridem(:,r,j,k+1) = [eplHMA;eplGran;eplSG];
        %apply simplification 2019-02-20: locate point where overall maximum
rutting occurs (sum all stacks)
        auxMaxRutTridem(r,j,k) = sum(auxrutDepthHMATridem(:,r,j,k))
+sum(auxrutDepthGranTridem(:,r,j,k))+sum(auxrutDepthSubGradeTridem(:,r,j,k));
%Get the total rut depth at each radial location and each load level
    end
   %get the position where the maximum rutting occurs for each weight level
   %%UPDATE V2019-05-14: store all auxMatRutPosition, but save the r==4 %case as maxRutDepth (under a wheel)
%      aux =  find(auxMaxRut(:,j)==max(auxMaxRut(:,j)));
%      if ~isempty(aux)
%         auxMaxRutPosition(j) = aux(1);
%     else
%         auxMaxRutPosition(j) = 1;
%     end
%     maxRutDepthHMATridem(:,1,j)        = auxrutDepthHMATridem(:,auxMaxRutPosition(j),j);
   maxRutDepthHMATridem(:,1,j)         = auxrutDepthHMATridem(:,4,j,k);
   maxRutDepthGranTridem(:,1,j)        = auxrutDepthGranTridem(:,4,j,k);
   maxRutDepthSubGradeTridem(:,1,j)    = auxrutDepthSubGradeTridem(:,4,j,k);
end

%% sum them all together!
%%sum over the " stack dimension" (I'll get small vectors with the sum of all rut depths for all the load levels, each the size of the HMA, gran, and subgrade). hen I must transpose them to row vector and stack together in row kth of rutDepth.
rutDepth(k,1:end-1) = [(sum(maxRutDepthHMASingleL,3))' (sum(maxRutDepthGranSingleL,3))' (sum(maxRutDepthSubGradeSingleL,3))'] + ...
     [(sum(maxRutDepthHMASingle6,3))' (sum(maxRutDepthGranSingle6,3))' (sum(maxRutDepthSubGradeSingle6,3))'] + ...
     [(sum(maxRutDepthHMASingle10,3))' (sum(maxRutDepthGranSingle10,3))' (sum(maxRutDepthSubGradeSingle10,3))'] + ...
     [(sum(maxRutDepthHMATandem10,3))' (sum(maxRutDepthGranTandem10,3))' (sum(maxRutDepthSubGradeTandem10,3))'] + ...
     [(sum(maxRutDepthHMATandem14,3))' (sum(maxRutDepthGranTandem14,3))' (sum(maxRutDepthSubGradeTandem14,3))'] + ...
     [(sum(maxRutDepthHMATandem18,3))' (sum(maxRutDepthGranTandem18,3))' (sum(maxRutDepthSubGradeTandem18,3))'] + ...
     [(sum(maxRutDepthHMATridem,3))' (sum(maxRutDepthGranTridem,3))' (sum(maxRutDepthSubGradeTridem,3))'];

 %get the total depth of rutting
rutDepth(k,end) = sum(rutDepth(k,1:end-1),2);   %sum over the second dimension of rutDepth (sum by rows.)
```

Source Code: /distressCalculator/rutDepthCompute

```
function
[rutDepthHMALayers,rutDepthGranLayers,rutDepthSubGrade,eplHMA,eplGran,eplSG] =
rutDepthCompute(MLEstrain,axlePasses,z,layerDepth,asphTemp,granMoisture,granDe
nsity,eplHMAPrev,eplGranPrev,eplSGPrev)
%function
[rutDepthHMALayers,rutDepthGranLayers,rutDepthSubGrade,eplHMa,eplGran,eplSG] =
rutDepthCompute(MLEstrain,axlePasses,z,layerDepth,asphTemp,granMoisture,granDe
nsity,eplHMAPrev,eplGranPrev,eplSGPrev)
%
%compute the vertical deformation (rutting) on each layer and over the
%entire structure for the strain produced by a certain axle type (thus
%load- and axle-type-related) for each single moment in time
shortTimestamp(k),
%
%Use MEPDG's original equations(eqns. 3.3.10 and so on). See Pt3 Chapter 3 of
NCHRP 2004.
%
%Inputs
% MLEstrain,   vertical strains as computed in the MLE module [VECTOR!]
% axlePasses,  number of axle passes (of a given axle type and load level) in
the period of interest [scalar]
% z            vector of depths tied to MLEStrain     [m]
% layerDepth   vector stating the depth of all pavement layers [cm]
% asphTemp:    temperature at each of the HMA layers [C]. DON'T PASS the sur-
face temperature!
% granMoisture volumetric moisture content of the granular layers [as received
from the infiltration module (PERCENTAGE), CONVERTED TO RATIO HERE]
% granDensity  granular materials' density [in g/cm3] (which is equal (in num-
ber value) to their bulk unit weight)
% eplHMAprev   plastic strain at the asphalt layers due to the input axle type
and load computed in previous iteration (if first iteration = -9999)
% eplGranPrev  plastic strain at the granular layers due to the input axle
type and load computed in previous iteration (if first iteration = -9999)
% eplGranSG    plastic strain at the subgrade due to the input axle type and
load computed in previous iteration (if first iteration = -9999)
%
%Outputs
% rutDepthHMALayers   = HMA layers only rut depth [m] - COLUMN VECTOR FORMAT!
% rutDepthGranLayers  = rut depth in the granular aggregate layers [m] -  COL-
UMN VECTOR FORMAT
% rutDepthSubGrade    = rut depth in the subgrade [m] - COLUMN VECTOR FORMAT
% eplHMA              = plastic deformation in the asphalt layers (same format
as rutDepthHMALayers)
% eplGran             = plastic deformation in the granular layers (same for-
mat as rutDepthHMALayers)
% eplSG               = plastic deformation in the subgrade (same format as
rutDepthHMALayers)%
%
%THIS IS THE "UNCALLIBRATED" VERSION OF THE RUTTING MODEL, THE EQUATIONS
%WERE PROGRAMMED HEREIN AS THEY HAVE BEEN REPORTED IN THE MEPDG GUIDE.
%THE CALLIBRATION EFFORT MAY EVENTUALLY LEAD TO RE-WRITING THIS COMPUTER CODE!
```

```
%
%v0.7 - 2019-05-21
%   Changelog: stability correction for granular materials and subgrade: At
%   times the epl/ez ratio to compute Neq gives an impossible Neq number
%   [the Log(epl/(b1*ez*eoer) becomes >1 and then crashes into CMPLx].
%   These translate to moments in which no increase in Epl occurs. FORCE
%   SUCH CASES TO EPL = EPL_PREV
%   Also, major cleanup to the unbound-materials calculators.
%V0.6 - 2019-05-20
%   Changelog: bug detected on SubGrade epl prediction (missing 1.35 factor
added)
%V0.5 - 2019-05-09
    %Changelog: use absolute value of strain in the Epl equations to prevent
them from going to the Cmplx dimension. Revert with signum function once cal-
culation completed to assign proper sign to rut depth
%V0.4 - 2019-04-03
    %Changelog: Added missing term 10^-3.4488 and exponent to temperature temp
in HMA plastic def. equation
%V0.3 - Delayed April's fools 2019-04-02:
    % *Changelog: Corrected calculation of eplPlastic for subgrade and unbound
materials: when traffic = 0,  %it will hit a division by 0 and return NaN.
Force a 0 deflection in such cases
    %**Corrected the expression for rut depth in granular and sub-grade soils,
%it was falling to the Complex numbers (having a negative base powered to a <1
exponent)
    %**Corrected expression to compute (eps0/epsR) for granular and subgrade -
use that of Appendix GG in MEPDG guide (remove the log10 relationship, which
was causing overly huge deformation values)...
%V0.2 - StPatrick's Hangover. 2019-03-18: Stability update added to Neq calcu-
lations when no strain occured on the previous iteration (assume 0 passes)
%V0.1 - StPatrick's day. 2019-03-17
%%changelog: programmed "strain hardening?" methodology to compute the amount
of plastic strain and number of passes, as stated in the 1-37A, P3C3
% - Cached content will be retrieved upon each call
%V0.0 - Valentine's day. 2019-02-14

%% Certical deformation in the HMA layers
%rutDepthHMALayers = zeros(length(asphTemp),1);
%initialize the vector for rutDepthHMALayers  [Matlab suggests against doing
so, that's why I commented this line]

layerDepthHMA = layerDepth(1:length(asphTemp));
layerDepthHMA = layerDepthHMA/2.54;         %convert the vector of layer
depths from cm. to inches.
auxTotalDepthHMA = sum(layerDepthHMA);      %get the total depth of the HMA
layers, inch

%locate the midDepth positions in "z" that correspond to the HMA layers mid-
points
midpointsHMA = zeros(length(layerDepthHMA),1);
auxPosZ = zeros(length(layerDepthHMA),1);
for i = 1:length(layerDepthHMA)
    if i ==1
        midpointsHMA(i) = 0.5*layerDepthHMA(1);
    else
        midpointsHMA(i) = sum(layerDepthHMA(1:i-1))+0.5*layerDepthHMA(i);
    end
    aux                 = find(abs(z-midpointsHMA(i)*0.0254)<0.001);
    if ~isempty(aux)
        auxPosZ(i)      = aux;
    end
end

strainHMA = MLEstrain(auxPosZ);   %get the strain values where z is a midpoint
of the HMA layers (remember that z is in meters and midpointsHMA in inches!!)

asphTemp = asphTemp*18/10+32;      %convert the HMA temperature vector from
deg. C to deg F for calculation purposes.
asphTemp = asphTemp(:);
C1 = -0.1039*auxTotalDepthHMA^2+2.4868*auxTotalDepthHMA-17.342;
C2 =  0.0172*auxTotalDepthHMA^2-1.7331*auxTotalDepthHMA+27.428;
k1z = (C1 + C2.*midpointsHMA).*0.328196.^midpointsHMA;

%update v2019-03-17:: get the Neq (equivalent number of passes of the previous
season)
if eplHMAPrev(1) ~= -9999
    %%%the -9999 case is reserved for case timestamp (k = 1), casue there's no
previous iteration.
    %%update V2019-03-18 (Stability update) -> force if strain = 0, Neq = 0
    %%update V2019-04-03 Added missing exponent in asphTemp  and 10^3.4488
term here below!
    if strainHMA(1) ~= 0
%         Neq = (10^3.4488).*(eplHMAPrev./strainHMA).*(1./k1z).*(asphTemp.^-
1.5606);
        Neq = (10^3.4488).*(abs(eplHMAPrev)./abs(strainHMA)).*(1./k1z).*(asph-
Temp.^-1.5606);  %%update V0.5 - use absolute vaules in this equation to pre-
vent cmplx. Neq values (all variables must be positive before being raised at
the 1/whatever.
        Neq = Neq.^(1/0.479244);
    else
        Neq = 0*ones(size(strainHMA));
    end
else
    Neq = 0*ones(size(strainHMA));
end
%update v2019-03-17:: correct the axle passes by summing the N_equivalent
%from previous iteration (if apply)
%Update V2019-04-02:
%   add this just in case (prevent a negative Neq from entering there and
eventually taking eplHMA into the Cmplx plane...
if Neq >0
    axlePassesHMA = axlePasses + Neq;
else
    axlePassesHMA = axlePasses;
end
```

```
eplHMA = abs(strainHMA).*k1z.*10^(-
3.4488).*asphTemp.^1.5606.*axlePassesHMA.^0.479244;  %%%this is the equation
to obtain the plastic deformation at each HMA layer
eplHMA = eplHMA.*sign(strainHMA); %<<update V0.5: compute epl with absolute
value and assign sign with te signum function.
auxRutDepthHMA = layerDepthHMA.*eplHMA;        %get a vector with each terms
of the Permanent Def. equation [inches]
rutDepthHMALayers = 0.0254*auxRutDepthHMA;     %pass all the Permanent defor-
mation terms. Convert from Inches back to meters!!

%% vertical deformation in granular layers
layerDepthGranular = layerDepth(length(asphTemp)+1:end-1)/2.54;   %get the
depth of the granular layers, convert it to inches
%get the strain at each layer's midpoint!. Use same methodology than for the
HMA layers to locate the midpoints

midpointsGranular = zeros(length(layerDepth)-1-length(layerDepthHMA),1);
auxPosZ = zeros(length(layerDepth)-1-length(layerDepthHMA),1);

for i = 1:length(layerDepthGranular)
    if i == 1
        midpointsGranular(1) = 0.5*layerDepthGranular(i) + auxTotalDepthHMA;
%depth of the midpoint of the 1st granular layer [inches]
    else
       midpointsGranular(i) = auxTotalDepthHMA + sum(layerDepthGranular(1:i-
1)) + 0.5*layerDepthGranular(i);
    end
    aux                  = find(abs(z-midpointsGranular(i)*0.0254)<0.001);
    if ~isempty(aux)
        auxPosZ(i)       = aux;
    end
end
%%%I NEED TO RETRIEVE THE MID/LYR MLE_strain for the Gran layers!
strainGranular = MLEstrain(auxPosZ);   %get the strain values where z is a
midpoint of the granular layers (remember that z is in meters and midpointsHMA
in inches!!)
% strainGranular = strainGranular(:);

%IMPORTANT:::  granMoisture must be converted back to moisture by weight to be
used in beta!!! (though the MEPDG guide doesn't state that it's by weight, RE-
FER TO FHWA, 2006).
%BUT I'M REPLACING THE ORIGINALLY GIVEN EQUATION (cause it won't account for
moisture variations due to infiltration, only the ground water table depth)
%%conversion to moisture by weight
betaGran = 10.^(-0.61119-0.017638.*granMoisture(1:end-
1)'.*1.00.*(granDensity(1:end-1)).^-1) ;              %%Eq 3.3.10.a. Convert-
ing moisture content in % volume to weight [need to divide by material's bulk
gravity]
betaGran = betaGran(:);
%use the MEPDG globally-callibrated formula to compute rutting depth in each
layer
CoGranular   = log(0.15/20);       %%Eq 3.3.10c after simplification [no extra
cal. factors]   NATURAL LOGARITHM!!!!
rhoGranular  = CoGranular./(1-(10^9).^betaGran);
rhoGranular  = rhoGranular.^(1./betaGran);
rhoGranular  = 10^9 .*rhoGranular;                        %%as per eq. 3.3.10c
eoer         = 0.5.*(0.15.*exp(rhoGranular.^betaGran) + 20.*exp((rhoGranular.*10^-9).^betaGran));  %%as per eq. 3.3.10b
% eoer         = 10.^eoer;

%update V2019-03-17 get the Neq (equivalent number of passes of the previous season)
%update v2019-05-20: Added the abs(eplSGPrev>0) condition for stability, if not i'd get a Neq becoming Nan (log(0))
if eplGranPrev(1) == -9999
    %%%the -9999 case is reserved for case timestamp (k = 1), casue there's no previous iteration.
    Neq = 0* ones(size(eplGranPrev));
    axlePassesGran = axlePasses+Neq;
    eplGran = 1.673.*strainGranular.*eoer.*exp(-1*((rhoGranular./axlePassesGran).^betaGran));   %equation 3.3.11. giving the rut depth in each layer in INCHES!

else %case eplGranPrev(1) >0 (must do strain hardening to get neq and compute epl)
    %update v2019-03-17:: correct the axle passes by summing the N_equivalent from previous iteration (if apply)
    %update V2019-04-02:: Corrected exp(-1(rho/N)^b). Previously I had the -1 inside the term to be powered up, that was going to the Cmplx plane...
    %%UPDATE V2019-05-21 - STABILITY UPDATE: Force that if log(xxxx) [below] is <1. if log(xxx) is > 1, epl gran = eplGranPrev.! MUST HAVE TO FORCE THIS FOR EACH LAYER IN THE SUBGRADE
    Neq = zeros(size(betaGran));
    axlePassesGran = axlePasses+Neq;
    eplGran = eplGranPrev;

    for nn = 1:length(betaGran)  %must treat the granular layers separately for the stability condition.
        if  (eplGranPrev(nn)/(1.673*eoer(nn)*abs(strainGranular(nn))))<1   %%stability condition! When this doesn't hold, Neq goes to the cmplx plane.
            Neq(nn) = -1*log(eplGranPrev(nn)./(1.673*eoer(nn).*abs(strainGranular(nn))));   %%update v0.5: use abs. value of eplGranPrev and strainGranular to prevent weird negatives from appearing
            Neq(nn) = Neq(nn).^(-1./betaGran(nn));
            Neq(nn) = rhoGranular(nn).*Neq(nn);
            axlePassesGran(nn) = axlePasses+Neq(nn);
            eplGran(nn) = 1.673*strainGranular(nn)*eoer(nn)*exp(-1*((rhoGranular(nn)/axlePassesGran(nn))^betaGran(nn)));   %equation 3.3.11. giving the rut depth in each layer in INCHES!
        else
            %stability condition doesn't hold. In this timestamp, no increase in plastic strain occurs. force plastic strain to match the previous value
            eplGran(nn) = eplGranPrev(nn);
        end
    end %%by now I should get the eplGran for all granular layers properly calculated.
end
auxRutDepthGranular = eplGran.*layerDepthGranular;
```

```
rutDepthGranLayers  = 0.0254*auxRutDepthGranular;  %pass them all, convert to
meters

%% vertical deformation in the Subgrade - use the MEPDG approach for integrat-
ing the exp. decaying rutting over infinite depth [equations 3.3.60-3.3.62]

%get the eo/er parameter
% strainSubGrade = MLEstrain;
strainSubGrade = MLEstrain(end-1:end);  %two last values in the strain vector,
correspond to top-most level of subgrade and 15cm underneath the surface.
%rutDepthSubGrade requires the deformation at the subgrade start and 15cm un-
derneath for calculation!
betaSubgrade = 10.^(-0.61119-0.017638.*(granMoisture(end)./granDensity(end)));
%https://en.wikipedia.org/wiki/Water_content  Converting moisture content in %
volume to weight [need to divide by material's bulk gravity]

%use the MEPDG globally-callibrated formula to compute rutting depth in each
layer
CoSubgrade    = log(0.15/20);                               %%Eq 3.3.10c after
simplification
rhoSubgrade   = CoSubgrade./(1-(10^9).^betaSubgrade);
rhoSubgrade   = (rhoSubgrade).^(1./betaSubgrade);           %%as per eq. 3.3.10c
rhoSubgrade   = 10^9 .* rhoSubgrade;
eoer          = 0.5.*(0.15.*exp(rhoSubgrade.^betaSubgrade) + 20.*exp((rhoSub-
grade.*10^-9).^betaSubgrade));     %%as per eq. 3.3.10b
% eoer          = 10.^eoer;
%%update V2019-03-17 get the Neq (equivalent number of passes of the previous
season)
%%update v2019-05-20: ADDED A MISSING 1/1.35. Also added the abs(eplSGPrev>0)
condition for stability, if not i'd get a Neq becoming Nan (log(0))

if(eplSGPrev(1) == -9999)
    %%first timestamp iteration case. No Neq. computation. Do the calculations
straightforward
    Neq = zeros(size(strainSubGrade));
    axlePassesSG = axlePasses+Neq;
    auxRutSubgrade = 1.35*strainSubGrade.*eoer.*exp(-1*((rhoSubgrade./
axlePassesSG).^betaSubgrade));   %equation 3.3.19. giving the rut depth in
each layer in INCHES!
    %% BUG-PREVENTION. IF auxRutSubgrade == 0; force kappaCoeff to 1 (because
if not I'll get a log of 0 [-inf]
    %6 comes for 6 inches, and auxRutSubgrade(2) has been computed 6 inches
below the surface of the subgrade.
    if auxRutSubgrade(1) ~= 0
        kappaCoeff = 1/6.*log(auxRutSubgrade(1)/auxRutSubgrade(2));  %unit =
[1/in]
    else
        kappaCoeff = 1;
    end
    eplSG = auxRutSubgrade(1);
    rutDepthSubGrade = eplSG.*(1/kappaCoeff).*(1-exp(-
kappaCoeff*(layerDepth(end))/2.54));   %%eqn. 3.3.23
    rutDepthSubGrade = 0.0254*rutDepthSubGrade;
else
```

```
    %case eplGranPrev(1) >0 (must do strain hardening to get neq and compute 
epl)
    %update v2019-03-17:: correct the axle passes by summing the N_equivalent 
from previous iteration (if apply)
    %update V2019-04-02:: Corrected exp(-1(rho/N)^b). Previously I had the -1 
inside the term to be powered up, that was going to the Cmplx plane...
    %%UPDATE V2019-05-21 - STABILITY UPDATE: Force that if log(xxxx) [below] 
is <1. if log(xxx) is > 1, epl gran = eplGranPrev.! MUST HAVE TO FORCE THIS 
FOR EACH LAYER IN THE SUBGRADE

    if (eplSGPrev/(1.35*eoer(1)*strainSubGrade(1)))<1   %%stability condition! 
When this doesn't hold, Neq goes to the cmplx plane.
       Neq = -log((1/1.35)*abs(eplSGPrev)./(eoer(1).*strainSubGrade(1)));
       Neq = Neq.^(-1/betaSubgrade(1));
       Neq = rhoSubgrade(1).*Neq;
       axlePassesSG = axlePasses+Neq;

       auxRutSubgrade = 1.35*strainSubGrade.*eoer.*exp(-1*((rhoSubgrade./
axlePassesSG).^betaSubgrade));   %equation 3.3.19. giving the rut depth in 
each layer in INCHES!
       if auxRutSubgrade(1) ~= 0
           kappaCoeff = 1/6.*log(auxRutSubgrade(1)/auxRutSubgrade(2));  %unit 
= [1/in]
       else
           kappaCoeff = 1;
       end
       eplSG = auxRutSubgrade(1);
       rutDepthSubGrade = eplSG.*(1/kappaCoeff).*(1-exp(-kappaCoeff*(lay-
erDepth(end))/2.54));   %%eqn. 3.3.23
       rutDepthSubGrade = 0.0254*rutDepthSubGrade;

    else
         %stability condition doesn't hold. In this timestamp, no increase in 
plastic strain occurs. force plastic strain to match the previous value and a 
kappa of 1 just to say sth.
         eplSG = eplSGPrev;
         kappaCoeff = 1;
         rutDepthSubGrade = eplSG.*(1/kappaCoeff).*(1-exp(-kappaCoeff*(lay-
erDepth(end))/2.54));   %%eqn. 3.3.23
         rutDepthSubGrade = 0.0254*rutDepthSubGrade;
    end
end   %%this completes all cases for the subgrade.


end
```

Source Code: /distressCalculator/ TopDownCracking

```
function [alligatorNi,alligatorDamage] = topDownCracking(MLEstrain,
axlePasses,z,HMAlayerDepth,EDyn,HMAProperties)
%function  [topDownNi,topDownDamage] = topDownCracking(MLEstrain,
axlePasses,z,layerDepth,EDyn,HMAProperties)
%
%compute the degree of TOP-DOWN cracking  damage on each HMA layer for a given
axle type
%(compute at all depths of interest (bottoms of the HMA layers), for all the
weight ranges) for a given shortTimestamp(k)
%%
%Use MEPDG's original equations(eqns. 3.3.29/30 and so on). See Pt3 Chapter 3
of NCHRP 2004.
%
%Inputs
% MLEstrain,   MAx horizontal strain matrix at each (z,r) location -as com-
puted by the MLE
% axlePasses,  number of axle passes in the period of interest
% z            vector of depths tied to MLEStrain     [m]
% HMAlayerDepth   vector stating the thickness of the HMA layers [cm]
% EDyn         dynamic modulus [in PSI] for the HMA layers for the given axle
type and load value (compatible with MLEStrain) AT A SINGLE TIMESTAMP (t(k)).
% HMAPropertiesVector with the Mix properties for the HMA layers as imported
from the dataImport sheet. Need Air voids and bitumen content in perc. volume
%
%Outputs
% topDownNi     [size numHMALayers x numLoadLevels]: number of admissible
passes for HMA layer (i) and load level (j)
% topDownDamage [size numHMALayers x 1]: degree of damage done by all the
passes of the axle type of the given type (sum of all weights).
%
%%Assumption: for a given weight range, the radial position that will be
summed to the admissible number of passes (and so to the topDownDamage value)
is that with the least admissible number of passes (Nf) on all the HMA layers.
%
%THIS IS THE "UNCALLIBRATED" VERSION OF THE RUTTING MODEL, THE EQUATIONS WERE
PROGRAMMED HEREIN AS THEY HAVE BEEN REPORTED IN THE MEPDG GUIDE. THE CALLIBRA-
TION EFFORT MAY EVENTUALLY LEAD TO RE-WRITING THIS COMPUTER CODE!
%
%V0.4 - 2019-05-20
%   Changelog: CORRECTED  EQUATION FOR K1 PARAMETER FOR TOP-DOWN CRACKING.
%   MEPDG'S P3C3 (2004) HAS WRONG EQUATION, CORRECT ONE IS MEPDG'S APPENDIX
II, EQN. 27
%V0.3 - 2019-05-19:
%   Changelog: Compute top-dn cracking at surface level only.
%   Modify computations to only regard top-most layer
%V0.2 - 2019-04-04:
%   Changelog: corrected error in k1 computation (it was summing
%   HMALayerDepth twice!)
%v0.1 - 2019-04-03:
%   Changelog: Corrected M parameter equation, Vb and Va were reverted!
%   Force the Eps_H term in the N_admissible equation to use the absoulte
%   value of the horizontal strain (cause if Eps_H is negative it goes to the Cmplx plane)
%V0.0 - 2019-02-22

%% code begins

%%update V2019-05-19: Properties for Layer 1 only
airVoids = HMAProperties(1,2);  %retrieve air voids vol.. from import.
bitCont  = HMAProperties(1,3);  %retrieve bitumen content from import.
M        = 4.84.*(bitCont./(airVoids+bitCont) - 0.69);     %parameter to equation 3.3.29

%do some needed calculations
HMAlayerDepth = HMAlayerDepth*.01;   %%convert from cm to meters to compare with Z
HMAlayerDepth = cumsum(HMAlayerDepth); %and convert thickness to depth.
%UPDATE V2019-05-20: k1 equation corrected WITH MEPDG'S APPENDIX II EQN 27.
%%K1 EQN. patched v2019-04-04. Since HMAlayerDepth is cumsummed above, the last entry is the total HMA layer depth!
k1            = 0.0001 + 29.844*(1+exp(30.544-5.7357*(HMAlayerDepth(end))/0.0254))^(-1);
k1            = 1/k1;             %parameter k1 FOR TOP-DN CRACKING. AS PER EQUATION 27 in mepdg's appendix II (the one in P3.c3 of the MEPDG is WRONG!).
%NOTE: k1 is a scalar (or should be...)

%Locate the MLE strain rows that correspond to the base of the HMA layers.
% strainHMA = MLEstrain(z==HMAlayerDepth,:,:);   %get the strain values where z is the base of the HMA layers, all r positions and all levels of load!
auxPosZ = zeros(length(HMAlayerDepth),1);
for i = 1:length(HMAlayerDepth)
    aux = find(abs(z-HMAlayerDepth(i))<0.001);
    if ~isempty(aux)
        auxPosZ(i)  = aux;
    end
end

%%updateV2019-05-19: Remove all calculations an do the math for z = first
%%location (surface of pavement)

strainHMA = MLEstrain(1,:,:);   %<<UPDATED get the strain values at the pavement surface, all r positions and all levels of load!

[numHMALayers,radioPositions,numLoadLevels] = size(strainHMA);

%% get the admissible number of passes (NF(z,r,load)) for each z and radial position and each load level. [z = each HMA layer]

NF = zeros(size(strainHMA));    %size is HMA LAYERS X RADIAL POSITIONS X LOAD LEVEL
%alligatorNi = zeros(numHMALayers,numLoadLevels);  <<update v2019-05-19
alligatorNi = zeros(1,numLoadLevels);

%solve for each load level (size of the axlePasses vector) AND radial position (size of retrieved from the strainHMA 3-D array)
```

```
%patched v2019-04-04. Exponent to strainHMA badly written!
%update v2019-05-19:  Use Edyn(1,x) to represent the top-most layer only
for i = 1:numLoadLevels
    for j = 1:radioPositions
        NF(:,j,i) = (0.00432*k1).*10.^M .*(abs(strainHMA(:,j,i))).^(-
3.9492).*EDyn(1,i).^(-1.281);
    end
    %%now, for each load level, get a single set of NF -> alligatorNi
    %get the radial position that brings the smallest NF,
    minNF = min(NF(:,:,i));
    [~,minCol,~] = ind2sub(size(NF(:,:,i)),find(NF(:,:,i) == minNF));   %this
function call gives the coordinates x,y (row/col/stack??) for the position in
NF(:,:,i) where the minNF is located and pass that column for all stacks to
alligatorNi below. %Just in case, put the first entry of the minNF case there
are many occasions.
    alligatorNi(:,i) = NF(:,minCol(1),i);
end

%% compute the degree of damage done by these axles

alligatorRatios = zeros(numHMALayers,numLoadLevels);  %compute these sepa-
rately for each HMA layer
alligatorDamage = zeros(numHMALayers,1);

for i = 1:numHMALayers
    alligatorRatios(i,:) = axlePasses./alligatorNi(i,:);
    alligatorDamage(i) = sum(alligatorRatios(i,:));
end

end
```

Source Code: /elasticLinearAnalysis/getAuxEfromE

```
function [ auxE ] = getAuxEfromE(zShort,zLong,E)
%function [ auxE ] = getZfromPaveDepth(zShort, zLong, E)
%MULTI-LYR LINEAR ELASTIC ANALYSIS PROGRAM
%This is an auxiliary function that will build the vector of variable E (with
valuese for each z position in zShort.
%
%Input:
% - zShort vector of points (short series, few points [in this case, the cum-
sum of the layers' thicknesses)
% - zLong  vector of points (long  series, morepoints [all the computation
points in the MLE problem]
% - E      Variable you want to extend from zShort to zLong
%Output:
% - auxE   Extended variable
%
% V0.1: 2019-02-07

%% code begins

%convert thicknesses to depths, and cm to meters. And remove the subgrade
thickness entry (I won't need it)
auxE = zeros(length(zLong),1);

%step 1 - fill in the auxE values for each position Z in zLong. First pass,
smaller than the 1st depth value in zShort
targetPos = find(zLong<=zShort(1));
auxE(targetPos) = E(1);
%second pass, smaller than the ith depth value but greater than the i-1 th
for i = 2:length(zShort)
    targetPos = find(zLong<=zShort(i) & zLong>zShort(i-1));
    auxE(targetPos) = E(i);
end

%%all done!
end
```

Source Code: /elasticLinearAnalysis/getZfromPaveDepth

```
function [ z ] = getZfromPaveDepth(paveDepth)
%function [ z ] = getZfromPaveDepth(paveDepth)
%MULTI-LYR LINEAR ELASTIC ANALYSIS PROGRAM
%This is an auxiliary function that will build the vector of depths at which
to compute stresses and strains from the vector of thickness of each pavement
layer.
%
%Input:
% - paveDepth: vector with the thickness of each layer of pavement (plus the
subgrade) - IN CENTIMETERS (as received from the MainCode)
%
%Output:
% - z: vector with the depth of each computation point, compliant with the
MEPDG manual (NCHRP, 2004); part 3 chap. 3.
% z will contain the positions of [in METERS]:
% [surface, 1cm underneath the surface, 1/2 depth of each layer, bottom of
each layer (until top of subgrade), and a point 15cm underneath the subgrade
surface]
%
% V0.3: 2019-04-03 - - added a z(subgrade)+0.02m point - needed to compute
strain decay in the subgrade for rutting purposes.
% V0.2: 2019-02-22 - - removed the z = 0.01m point (not needed at all).
% V0.1: 2019-02-07

%% code begins

%convert thicknesses to depths, and cm to meters. And remove the subgrade
thickness entry (I won't need it)
paveDepth = paveDepth(1:end-1);  %remove the maybe artificial subgrade thick-
ness
paveDepth = paveDepth(:);        %fix as column vector
z = 0.01*cumsum(paveDepth);      %convert to depths

%add midpoints. Use this workaround: https://www.mathworks.com/matlabcentral/
answers/25536-selecting-mid-points
z = [z;conv(z,[0.5;0.5],'valid')];
%add the 1st layer midpoint (which the convolution above doesn't calculate)
z = [0.5*z(1);z];
z = sort(z,'ascend');
% add surface points [0 and 0.01m], and point 15cm underneath the surface
%z = [0;0.01;z;z(end)+0.15];
%%%update V2019-04-03: Add the z(sub grade)+0.02m [The z(end) is assumed by
Matlab to be in the subbase and not in the subgrade]
z = [0;z;z(end)+0.02;z(end)+0.15];

%%all done!
end
```

Source Code: /elasticLinearAnalysis/MLE_bc

Special thanks to Dr. S. Katicha (VTTI) for the original version of this matlab code.

```
function [A,B,C,D,RM,LM,CM,nLayers,F,dLambda,Lambda] = MLE_bc(m,H,E,nu)
%%function [A,B,C,D,RM,LM,CM,nLayers,F,dLambda,Lambda] = MLE_bc(m,H,E,nu)

%tweaked version V2019-03-29  --- ALL TWEAKS ROLLED BACK!
H = H(:); E = E(:); nu = nu(:);
nLayers = length(E);
sumH = sum(H);
H = [H;max(H)*1e3];
% H = [H;1e5];
Lambda = cumsum(H)/sumH;

LM1 = [exp(-m*Lambda(1)) 1 ; exp(-m*Lambda(1)) -1];
RM1 = [-(1-2*nu(1))*exp(-m*Lambda(1)) 1-2*nu(1) ; 2*nu(1)*exp(-m*Lambda(1))
2*nu(1)];
%%tweak 01 - I can compute [a1--d1] directly with the surface boundary condi-
tion %DON'T DO IT, THIS IS NOT A WELL-DEFINED SYSTEM (4 VARIABLES, 2 EQUA-
TIONS)
% A1 = [LM1 RM1]\[1;0];  %A\B means "solve the system Ax = B. A1 IS THE COLUMN
VECTOR [A1;B1;C1;D1]

dLambda = diff([0;Lambda]);
F = exp(-m*dLambda);
R = E(1:end-1)./E(2:end).*((1+nu(2:end))./(1+nu(1:end-1)));

LM = zeros(4,4,nLayers-1);
RM = zeros(4,4,nLayers-1);
InvLM = zeros(4,4,nLayers-1);
CM = zeros(4,4,nLayers-1);
FM = zeros(4,4);

for i=1:(nLayers-1)
    LM(:,:,i) = [1,  F(i), -(1-2*nu(i)-m*Lambda(i)), (1-2*nu(i)
+m*Lambda(i))*F(i);...
                 1, -F(i), 2*nu(i)+m*Lambda(i),      (2*nu(i)-
m*Lambda(i))*F(i);...
                 1,  F(i), 1+m*Lambda(i),             -(1-m*Lambda(i))*F(i);...
                 1, -F(i), -(2-4*nu(i)-m*Lambda(i)), -(2-4*nu(i)
+m*Lambda(i))*F(i)];

    InvLM(:,:,i) = inv(LM(:,:,i));

    RM(:,:,i) = [F(i+1),         1,     -(1-2*nu(i+1)-m*Lambda(i))*F(i+1),
1-2*nu(i+1)+m*Lambda(i);...
                 F(i+1),        -1,     (2*nu(i+1)+m*Lambda(i))*F(i+1),
2*nu(i+1)-m*Lambda(i);...
                 R(i)*F(i+1),   R(i),   (1+m*Lambda(i))*R(i)*F(i+1),
-(1-m*Lambda(i))*R(i);...
```

```
                 R(i)*F(i+1),  -R(i),  -(2-4*nu(i+1)-m*Lambda(i))*R(i)*F(i+1),
-(2-4*nu(i+1)+m*Lambda(i))*R(i)];

    CM(:,:,i) = InvLM(:,:,i)*RM(:,:,i);
end

FM = CM(:,:,1);

for i=2:(nLayers-1)
    FM = FM*CM(:,:,i);
end
%%Note: For the sake of stability of the equation system (apparently it may
become singular), use teh approach said by the CR CILA paper and other %%refs,
and bother only in Bn and Dn.

FM = FM(:,[2,4]);

%BnDn = (LM1*FM(1:2,:)+RM1*FM(3:4,:))\[1;0];
%%tweak 01, use the [A1...D1] VECTOR TO COMPUTE BnDn   %<< debug results:
%this solution or the CR solutions bring the same boundary cond. values.  %BUT
DONT' USE IT, CAUSE A1--D1 IS NOT DETERMINED (COMES FROM 2EQ/4VAR %SYSTEM)

% BnDn = FM\A1;
%%tweak 01B, use the CR paper's formula to get BnDn
BnDn = ([LM1 RM1]*FM)\[1;0];

A = zeros(nLayers,1); B = zeros(nLayers,1); C = zeros(nLayers,1); D =
zeros(nLayers,1);
B(end) = BnDn(1);
D(end) = BnDn(2);

 %tweak 03 - fill up A--D up to row 2; we have row 1 from the surface boundary
condition
 %DISREGARD, A1 IS NOT A VALID SOLUTION!
% A(1) = A1(1);
% B(1) = A1(2);
% C(1) = A1(3);
% D(1) = A1(4);

%for i=(nLayers-1):-1:1      %bug detected here. A1--D1 COMPUTED IN THIS
%LOOP WOULD NOT VERIFY THE SURFACE BOUNDARY CONDITION. SO WEIRD....
for i=(nLayers-1):-1:1
    BC = CM(:,:,i)*[A(i+1);B(i+1);C(i+1);D(i+1)];
    A(i) = BC(1);
    B(i) = BC(2);
    C(i) = BC(3);
    D(i) = BC(4);
end
```

Source Code: /elasticLinearAnalysis/MLE_sigma

Special thanks to Dr. S. Katicha (VTTI) for the original version of this matlab code.

```
function [sigmaZ,sigmaR,sigmaT] = MLE_sigma(q,a,x,z,H,E,nu)
%function [sigmaZ,sigmaR,sigmaT] = MLE_sigma(q,a,x,z,H,E,nu)
%
%MULTI-LAYER ELASTIC STRESSES CALCULATOR
%Bqased equations in Huang (2004), Appendix B; with update by Caicedo (2018)
%
%Inputs
% q = LOAD pressure              [PA // PSI]
% a = radius of circular load     [METER  // INCH]
% x = location from center of load (radial) [can be a vector] [METER // INCH]
% z = depth from surface [can be a vector]!     [METER // INCH]
% H = matrix with each layers thickness (one less entry than E, nu) [METER //
INCH]
% E = matrix with each layer's Elastic modulus    [PA  // PSI]
% nu= matrix with each layer's Poisson module     [DIMLESS]
%
%NOTE: UNITS OF LOAD, E, H, a, x, z SHOULD BE CONSISTENT AMONG EACH OTHER.
%
%Outputs
% sigmaZ - stress in vertical direction for each (z,x) position (given as a
Z*R matrix)
% sigmaR - stress in radial direction for each (z,x) position (given as a  Z*R
matrix)
% sigmaT - stress in tangential direction for each (z,x) position (given as a
Z*R matrix)
%
%%Dependency: This function calls MLE_bc
%
% Original code by S. K.
% Ported to Product One 2019-02-04.
% Updated V2019-03-31
% Changelog:
%    Debugged against OpenPave and PitraPave: Discovered notation in Huang
(2004) and Caicedo 2018 have reverted signs!. Multiply stresses by -1 to match
actual results.%
%    Use the equations for vertical and radial stress.
%    The m parameter used in calculation is no longer an input but it's de-
fined here as a vector and passed on to the calculator.
%    Improved code to handle zand r as vectors of many positions.

%% code begins
%m = m(:)+1e-16;
m = 0:0.5:50; m(1) = 1e-5; m = m';        %first trial showed that m no larger
than ~40-50 would be necessary, the Rs terms will dilute for greater m (and
Matlab will bring a warining for singular matrices). Also, m(1) so close to 0
will give such an error too
x = x(:);
```

```
H = H(:);
E = E(:);
nu = nu(:);
nLayers = length(nu);
sumH = sum(H);                          %sumH must be THE DISTANCE FROM THE 
SURFACE TO THE TOP OF THE LOWEST LAYER (SUBGRADE)
Lambda = [0;cumsum(H)/sumH;1e3];        %THIS ONE
L = z/sumH;                             %this is "lambda" in Huang's Notation.
ro = x/sumH;                            %rho as defined after eqn B2.f
alpha = a/sumH;                         %defined as Eqn B5 in Huang '04
ind = zeros(size(z));
for i = 1:length(ind)
    ind(i) = find(Lambda>L(i),1);       %ind = first non-zero value of 
Lambda>L   (that is, first position that is below z(i))
end
A = zeros(length(m),nLayers);
B = zeros(length(m),nLayers);
C = zeros(length(m),nLayers);
D = zeros(length(m),nLayers);

for i=1:length(m)
    %solve the boundary conditions for that given m value (obtain A, B, C, D) 
equations B8-B17
    [a,b,c,d] = MLE_bc(m(i),H,E,nu);
    A(i,:) = a(:)';
    B(i,:) = b(:)';
    C(i,:) = c(:)';
    D(i,:) = d(:)';
end

%initialize the output variables
sigmaR = zeros(length(z),length(x));
sigmaT = zeros(length(z),length(x));
sigmaZ = zeros(length(z),length(x));

%% solve for sigma Z
%Get the R* value from equation B4a [vertical stress], for all layers and all
%values of m, for all values of z
for j = 1:length(x)
    for i = 1:length(z)
        Rs = -m.*besselj(0,m*ro(j)).*((A(:,ind(i)-1)-C(:,ind(i)-1).*(1-
2*nu(ind(i)-1)-m*L(i))).*exp(-m*(Lambda(ind(i))-L(i)))...
            +(B(:,ind(i)-1)+D(:,ind(i)-1).*(1-2*nu(ind(i)-1)+m*L(i))).*exp(-
m*(L(i)-Lambda(ind(i)-1))));
         Rs = Rs*-1;   %%update V2019-03-31 [multiply by -1 to match sign con-
vention]
        Rs = Rs(:);
    %
    %integrate R* using equation B7 to get R [sigmaZ].
        %Output here is w at the base of pavement, right?
        if length(m)>1
            sigmaZ(i,j) = q*alpha*sum(((Rs(1:end-1)+Rs(2:end))/2)./((m(1:end-
1)+m(2:end))/2).*besselj(1,((m(1:end-1)+m(2:end))/2)*alpha).*diff(m));
        else
```

```
            sigmaZ(i,j) = Rs;
        end
    end
end
%% solve for sigma R

%get the R* value from equation B4b [radial stress], for all layers and all 
values of m, for all values of z
for j = 1:length(x)
    for i = 1:length(z)
        Rs = (m.*besselj(0,m*ro(j))-(1/
ro(j)).*besselj(1,m*ro(j))).*((A(:,ind(i)-1)+C(:,ind(i)-1).*(1+m*L(i))).*exp(-
m*(Lambda(ind(i))-L(i)))...
            +(B(:,ind(i)-1)-D(:,ind(i)-1).*(1-m*L(i))).*exp(-m*(L(i)-
Lambda(ind(i)-1))))...
            + 2*nu(ind(i)-1).*m.*besselj(0,m*ro(j)).*(C(:,ind(i)-1).*exp(-
m*(Lambda(ind(i))-L(i)))-D(:,ind(i)-1).*exp(-m*(L(i)-Lambda(ind(i)-1))));
        Rs = Rs*-1;      %%update V2019-03-31 [multiply by -1 to match sign 
convention]
        Rs = Rs(:);

    %integrate R* using equation B7 to get R [sigmaZ].
        %Output here is w at the base of pavement, right?
        if length(m)>1
            sigmaR(i,j) = q*alpha*sum(((Rs(1:end-1)+Rs(2:end))/2)./((m(1:end-
1)+m(2:end))/2).*besselj(1,((m(1:end-1)+m(2:end))/2)*alpha).*diff(m));
        else
            sigmaR(i,j) = Rs;
        end
    end
end
%% solve for sigma T

%get the R* value from equation B4c [angular stress], for all layers and all
%values of m, for all values of z
for j = 1:length(x)
    for i = 1:length(z)
        Rs = (1/ro(j)).*besselj(1,m*ro(j)).*((A(:,ind(i)-1)+C(:,ind(i)-
1).*(1+m*L(i))).*exp(-m*(Lambda(ind(i))-L(i)))...
            +(B(:,ind(i)-1)-D(:,ind(i)-1).*(1-m*L(i))).*exp(-m*(L(i)-
Lambda(ind(i)-1))))...
            +2*nu(ind(i)-1).*m.*besselj(0,m*ro(j)).*(C(:,ind(i)-1).*exp(-
m*(Lambda(ind(i))-L(i)))-D(:,ind(i)-1).* exp(-m*(L(i)-Lambda(ind(i)-1))));
        Rs = Rs*-1;   %updave V2019-03-31 [multiply by -1 to match sign con-
vention]
        Rs = Rs(:);

    %integrate R* using equation B7 to get R [sigmaZ].
        %Output here is w at the base of pavement, right?
        if length(m)>1
            sigmaT(i,j) = q*alpha*sum(((Rs(1:end-1)+Rs(2:end))/2)./((m(1:end-
1)+m(2:end))/2).*besselj(1,((m(1:end-1)+m(2:end))/2)*alpha).*diff(m));
        else
            sigmaT(i,j) = Rs;
        end
    end
end
```

Source Code: /elasticLinearAnalysis/MLEFrontEnd.m

```
%%%%  PRODUCT-ONE - M.E. Pavement Design tool - %%%%
%%% FRONT-END SCRIPT for the MULTI-LYR LINEAR ELASTIC ANALYSIS TOOL%%%
%
%%V 0.8 2019-05-20
    %Changelog: Rolled back some angle corrections (was pointing to pi rad - 
actual angle
    %
%%V 0.7 2019-05-14:
    %Changelog: now storing the stress and strain values for all axle types
    %all variables have now a time-dimension  (are R4 matrices)
%%V 0.6 2019-05-09:
    %changelog: corrected bug in calculation of horizontal max strain
    %(epsH), the term with the gammaXY is not a subtraction but a sum!.
    %This was bringing cases to the cmplx world!
%%V 0.5 2019-03-31:
    %Changelog: Corrected call to MLE_sigma: the dependency requires the
    %load pressure instead of the load's total value.
    %Corrected MLE_sigma (see file for description of fixes).
%%V 0.4 2019-03-20:
    %changelog: corrected the 2019-02-19 update (it won't do the MLE
    %calculations for axle categories that don't have traffic). I expect to 
save computation time with this move.
%%V 0.3 2019-03-19
    %Changelog: Removed error in assigning load by wheel in multi-wheel axles
    %Removed r=0 cases and replaced with r=0.01m (when r=0, sigmaR and sigmaT 
diverge to infinity)
    %added epsX epsY epsZ epsH calculations for the SingleL and single6 axles
%%V 0.2 2019-03-12
    %Changelog: major debugging, correctred rdii and angle for most axles (cm 
to m conversion, erroneous angle values)
%%V 0.1 2019-02-05
%
%%THIS SCRIPT IS MEANT TO RUN IN THE MAIN-CODE WORKSPACE. ALL VARIABLES
%%USED HEREIN SHOULD BE DEFINED PREVIOUSLY IN THE MAIN CODE OR ANOTHER SCRIPT
%%EACH INSTANCE OF THIS SCRIPT OCCURS AT "timeStamp = k"
%
%Goal: calculate radial, tangential, and vertical stresses in the pavement
%structure --> with which compute the strains... which are to pass on to
%the distress calculator.

%Will stack stress(r, T, Z) for every axle type on a 3-D array, size
%(Z-domain; R-domain; weight range)

%% code begins
%%Some previous variables I need to fix
tandemAxleSep = 1.20;    %minimum legal axle separation for tandems and 
tridems
dualWheelSep  = 0.68;    %truck-size dual-wheel separation (from Merc.B. com-
mercial brochure)

%% - get the domain variables for all cases: z
```

```
z = getZfromPaveDepth(paveDepths);    %%auxiliary vector to calculate the z positions from the actual pave depths in compliance with MEPDG - program separately! OUTPUT IT GIVEN IN meters!


%% Initialize stress and strain variables
nrsL  = 4;                              %4 radial positions will be analyzed for light and 6-t single axles (all under the axle axis of symmetry).
nrs6  = 4;                              %4 radial positions will be analyzed for single-wheel heavy single axles (all under the axle axis of symmetry).
nrs10 = 6;                              %6 radial positions will be analyzed for dual-wheel heavy single axles (all under the axle axis of symmetry).
nrTa10= 6;                              %8 radial positions will be analyzed for 4-wheel tandem axles (all under the axle-group axis of symmetry).
nrTa14=15;                              %15 radial positions will be analyzed for 6-wheel tandem axles (asymmetric arrangement).
nrTa18=12;                              %12 radial positions will be analyzed for 8-wheel tandem axles (symmetric arrangement).
nrTr  =18;                              %18 radial positions will be analyzed for tridem axles.
if k ==1
    %%UPDATE V2019-05-14: INITIALIZE ALL STRESS AND STRAIN CALC. AT K ==1
    %%for singleLight
    sigmaZsingleL = zeros(length(z),nrsL,length(axlesSingleLWeights),termination);
    % sigmaRsingleL = zeros(length(z),nrsL,length(axlesSingleLWeights));   %%NOT WORTH KEEPING! THE MULTI-WHEEL AXLES WILL HAVE NON-ADDITIVE sigmaT and sigmaR; STORE sigmaX and sigmaY only!
    % sigmaTsingleL = zeros(length(z),nrsL,length(axlesSingleLWeights));
    sigmaXsingleL = zeros(length(z),nrsL,length(axlesSingleLWeights),termination);
    sigmaYsingleL = zeros(length(z),nrsL,length(axlesSingleLWeights),termination);
    tauXYsingleL = zeros(length(z),nrsL,length(axlesSingleLWeights),termination);

    epsXsingleL = zeros(length(z),nrsL,length(axlesSingleLWeights),termination);
    epsYsingleL = zeros(length(z),nrsL,length(axlesSingleLWeights),termination);
    gmXYsingleL = zeros(length(z),nrsL,length(axlesSingleLWeights),termination);
    epsHsingleL = zeros(length(z),nrsL,length(axlesSingleLWeights),termination);        %%only one worth computing, this is the "greatest [or smallest since it's being negative because it's tension]" horizontal tension - the one used in fatigue cracking model [After Huang 04 eq. 3.4]
    epsZsingleL = zeros(length(z),nrsL,length(axlesSingleLWeights),termination);        %% needed for vertical deflection

    %%for single 6 ton
    sigmaZsingle6 = zeros(length(z),nrs6,length(axlesSingle6Weights),termination);
    % sigmaRsingle6 = zeros(length(z),nrs6,length(axlesSingle6Weights));
    sigmaTsingle6 = zeros(length(z),nrs6,length(axlesSingle6Weights),termination);
    sigmaXsingle6 = zeros(length(z),nrs6,length(axlesSingle6Weights),termination);
    sigmaYsingle6 = zeros(length(z),nrs6,length(axlesSingle6Weights),termination);
    tauXYsingle6 = zeros(length(z),nrs6,length(axlesSingle6Weights),termination);

    epsXsingle6 = zeros(length(z),nrs6,length(axlesSingle6Weights),termination);
    epsYsingle6 = zeros(length(z),nrs6,length(axlesSingle6Weights),termination);
    gmXYsingle6 = zeros(length(z),nrs6,length(axlesSingle6Weights),termination);
    epsHsingle6 = zeros(length(z),nrs6,length(axlesSingle6Weights),termination);
    epsZsingle6 = zeros(length(z),nrs6,length(axlesSingle6Weights),termination);

    %%for single 10 ton
    sigmaZsingle10 = zeros(length(z),nrs10,length(axlesSingle10Weights),termination);
    % sigmaRsingle10 = zeros(length(z),nrs10,length(axlesSingle10Weights));
    % sigmaTsingle10 = zeros(length(z),nrs10,length(axlesSingle10Weights));
    sigmaXsingle10 = zeros(length(z),nrs10,length(axlesSingle10Weights),termination);
    sigmaYsingle10 = zeros(length(z),nrs10,length(axlesSingle10Weights),termination);
    tauXYsingle10 = zeros(length(z),nrs10,length(axlesSingle10Weights),termination);

    epsXsingle10 = zeros(length(z),nrs10,length(axlesSingle10Weights),termination);
    epsYsingle10 = zeros(length(z),nrs10,length(axlesSingle10Weights),termination);
    gmXYsingle10 = zeros(length(z),nrs10,length(axlesSingle10Weights),termination);
    epsHsingle10 = zeros(length(z),nrs10,length(axlesSingle10Weights),termination);
    epsZsingle10 = zeros(length(z),nrs10,length(axlesSingle10Weights),termination);

    %%%%%%%%%%%%%%%%%%
    %%for tandem 10 ton
    sigmaZtandem10 = zeros(length(z),nrTa10,length(axlesTandem10Weights),termination);
    sigmaXtandem10 = zeros(length(z),nrTa10,length(axlesTandem10Weights),termination);
    sigmaYtandem10 = zeros(length(z),nrTa10,length(axlesTandem10Weights),termination);
    tauXYtandem10 = zeros(length(z),nrTa10,length(axlesTandem10Weights),termination);
    epsXtandem10 = zeros(length(z),nrTa10,length(axlesTandem10Weights),termination);
    epsYtandem10 = zeros(length(z),nrTa10,length(axlesTandem10Weights),termination);
    gmXYtandem10 = zeros(length(z),nrTa10,length(axlesTandem10Weights),termination);
    epsHtandem10 = zeros(length(z),nrTa10,length(axlesTandem10Weights),termination);
    epsZtandem10 = zeros(length(z),nrTa10,length(axlesTandem10Weights),termination);

    %%for tandem 14 ton
    sigmaZtandem14 = zeros(length(z),nrTa14,length(axlesTandem14Weights),termination);
    sigmaXtandem14 = zeros(length(z),nrTa14,length(axlesTandem14Weights),termination);
    sigmaYtandem14 = zeros(length(z),nrTa14,length(axlesTandem14Weights),termination);
    tauXYtandem14 = zeros(length(z),nrTa14,length(axlesTandem14Weights),termination);

    epsXtandem14 = zeros(length(z),nrTa14,length(axlesTandem14Weights),termination);
    epsYtandem14 = zeros(length(z),nrTa14,length(axlesTandem14Weights),termination);
    gmXYtandem14 = zeros(length(z),nrTa14,length(axlesTandem14Weights),termination);
    epsHtandem14 = zeros(length(z),nrTa14,length(axlesTandem14Weights),termination);
    epsZtandem14 = zeros(length(z),nrTa14,length(axlesTandem14Weights),termination);

    %%for tandem 18 ton
    sigmaXtandem18 = zeros(length(z),nrTa18,length(axlesTandemWeights),termination);
    sigmaYtandem18 = zeros(length(z),nrTa18,length(axlesTandemWeights),termination);
    sigmaZtandem18 = zeros(length(z),nrTa18,length(axlesTandemWeights),termination);
    tauXYtandem18 = zeros(length(z),nrTa18,length(axlesTandemWeights),termination);

    epsXtandem18 = zeros(length(z),nrTa18,length(axlesTandemWeights),termination);
    epsYtandem18 = zeros(length(z),nrTa18,length(axlesTandemWeights),termination);
    gmXYtandem18 = zeros(length(z),nrTa18,length(axlesTandemWeights),termination);
    epsHtandem18 = zeros(length(z),nrTa18,length(axlesTandemWeights),termination);
    epsZtandem18 = zeros(length(z),nrTa18,length(axlesTandemWeights),termination);

    %%%%%%%%%%%%%%%
    %%for tridem
    sigmaZtridem = zeros(length(z),nrTr,length(axlesTridemWeights),termination);
    sigmaXtridem = zeros(length(z),nrTr,length(axlesTridemWeights),termination);
    sigmaYtridem = zeros(length(z),nrTr,length(axlesTridemWeights),termination);
    tauXYtridem = zeros(length(z),nrTr,length(axlesTridemWeights),termination);

    epsXtridem = zeros(length(z),nrTr,length(axlesTridemWeights),termination);
    epsYtridem = zeros(length(z),nrTr,length(axlesTridemWeights),termination);
    gmXYtridem = zeros(length(z),nrTr,length(axlesTridemWeights),termination);
    epsHtridem = zeros(length(z),nrTr,length(axlesTridemWeights),termination);
    epsZtridem = zeros(length(z),nrTr,length(axlesTridemWeights),termination);
end

%% Compute stresses with the MLE_sigma for each family of axles.
%% 1 Singles - light weight, 6-ton and 10.5-ton

%nrsL = length(rSingleL); [4 pos]
nax = length(axlesSingleLWeights);
rSingleL = zeros(nax,nrsL);            %%store the 4 radial positions for each weight combination (they depend on the contact radius, which depends on the load)

%%compose the vector of Es from the HMA layers (temperature and load varying) and the gran. layers MR..
auxE_HMA = EDynSingleLight(:,:,k);
auxv_HMA = HMAPoissonSingleLight(:,:,k);%%get the poisson coefficients for all HMA layers (rows) and all load ranges (cols) for timestamp k (stack)

layersE = MR(k,:)' * ones(1,nax);     %Step 1: retrieve the granular and subgrade's MRs and turn it to a matrix the same size of the E*. The product gives a matrix [layers x axleWeight] in size
layersE = [auxE_HMA;layersE];         %Step 2: stack the two matrices together
%%update v2019-03-19:: unit consistency check, convert layersE from PSI to Pa
layersE = layersE/145.04*1e6;

layersv = granPoiss * ones(1,nax);
layersv = [auxv_HMA; layersv];         %repeat to get the Poissons


for j = 1:nax
    %update v2019-03-19: replaced r=0 with r=0.01m
    rSingleL(j,:) = [0.01 0.5*0.01*aSingleLight(j) 0.01*aSingleLight(j) 0.01*aSingleLight(j) + 0.10];
    %for i = 1:nrsL
        %get axle Load (tons), convert to load by wheel -> That's axleSingleLWeights(j)/2
        %get load radius - - - - computed with wheelFootprint (called from the MainCode) - - IT'S GIVEN IN CM and accounts for the fact that the axle load is equally divided over all the axle's wheels!
        %get E, nu, height for all materials! - watchful for asphalt materi-
als!
       %get the sigmaZ, sigmaR, sigmaT values at each Z,R pair. No need to
%compose calculations for multiple wheels
       %%UPDATE 2019-02-19:: DON'T COMPUTE ANYTHING (AND KEEP ZEROS) IF
THERE'S 0 TRAFFIC IN THIS CATEGORY.
       %%UPDATE 2019-03-19:: CONVERT LOAD TO NEWTON ( ton x 9800), DISTANCES
TO METERS (cm x 0.01), AND E TO PA (psi x 1.000.000 / 145.04)
       if axlesSingleLight(k,j)~=0   %"if there's actual traffic of these
axles"
           %%update V2019-03-31:: pass pressure instead of total load to
MLE_sigma.
           %%update V2019-05-14: Add 4th dimension to sigmaXXX outputs from
%MLE
           aj = 0.01*aSingleLight(j);
           qj = 1/2*9800*axlesSingleLWeights(j)/(pi*(aj)^2);

[sigmaZsingleL(:,:,j,k),sigmaXsingleL(:,:,j,k),sigmaYsingleL(:,:,j,k)] =
MLE_sigma(qj,aj,rSingleL(j,:),z,0.01*paveDepths(1:end-1),layersE(:,j),lay-
ersv(:,j));
       end
     %create these auxiliar auxE and aux_Poisson variables with the E and v
for this load level to use with the conversion to strain        %equations
     auxLayersE = getAuxEfromE(0.01*cumsum(paveDepths),z,layersE(:,j));   %
%stretch to the z domain, will need it to superimpose the horizontal stresses
[although not needed in single and 6-ton axles...]
     auxLayersv = getAuxEfromE(0.01*cumsum(paveDepths),z,layersv(:,j));
     %%these two above are in vector format [length(z) by 1]. need to convert
     %%them to z x r matrices. Multiply them for ones(1,length(r))
     auxLayersE = auxLayersE * ones(1,nrsL);
     auxLayersv = auxLayersv * ones(1,nrsL);

    %Convert stresses to strains (refer to Huang 04, chap 3)
    %%update V2019-05-14: Add 4th dimension to epsX, epsY, epsZ, epsH
    epsXsingleL(:,:,j,k) = 1./auxLayersE.*(sigmaXsingleL(:,:,j,k) - auxLay-
ersv.*(sigmaYsingleL(:,:,j,k) + sigmaZsingleL(:,:,j,k)));
    epsYsingleL(:,:,j,k) = 1./auxLayersE.*(sigmaYsingleL(:,:,j,k) - auxLay-
ersv.*(sigmaXsingleL(:,:,j,k) + sigmaZsingleL(:,:,j,k)));
    epsZsingleL(:,:,j,k) = 1./auxLayersE.*(sigmaZsingleL(:,:,j,k) - auxLay-
ersv.*(sigmaXsingleL(:,:,j,k) + sigmaYsingleL(:,:,j,k)));
%     gmXYsingleL(:,:,j) = zeros(size(epsSingleL));
%%simplified formular for epsH, since there is no tauXY
    epsHsingleL(:,:,j,k) = 0.5.*(epsXsingleL(:,:,j,k) + epsYsingleL(:,:,j,k))
- 0.5.*(epsXsingleL(:,:,j,k)-epsYsingleL(:,:,j,k));
    %end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%nrs6 = length(rSingle6); [4 pos]
nax = length(axlesSingle6Weights);
rSingle6 = zeros(nax,nrs6);             %%store the 4 radial positions for each
weight combination (they depend on the contact radius, which depends on the
load)
```

```
%%compose the vector of Es from the HMA layers (temperature and load varying)
and the gran. layers MR..
auxE_HMA = EDynSingle6(:,:,k);
auxv_HMA = HMAPoissonSingle6(:,:,k);%%get the poisson coefficients for all HMA
layers (rows) and all load ranges (cols) for timestamp k (stack)

layersE = MR(k,:)' * ones(1,nax);     %Step 1: retrieve the granular and sub-
grade's MRs and turn it to a matrix the same size of the E*. The product gives
a matrix [layers x axleWeight] in size
layersE = [auxE_HMA;layersE];         %Step 2: stack the two matrices together
%%update v2019-03-19:: unit consistency check, convert layersE from PSI to Pa
layersE = layersE/145.04*1e6;

layersv = granPoiss * ones(1,nax);
layersv = [auxv_HMA; layersv];         %repeat to get the Poissons

for j = 1:nax
    %update v2019-03-19: replaced r=0 with r=0.01m
    rSingle6(j,:) = [0.01 0.5*0.01*aSingle6(j) 0.01*aSingle6(j)
0.01*aSingle6(j)+0.10];
    %for i = 1:nrs
        %get axle Load (tons), convert to load by wheel -> That's axleSingleLWeights(j)/2
        %get load radius - - - - computed with wheelFootprint (called from the
MainCode) - - IT'S GIVEN IN CM and accounts for the fact that the axle load is
equally divided over all the axle's wheels!
        %get E, nu, height for all materials! - watchful for asphalt
%materials!
        %get the sigmaZ, sigmaR, sigmaT values at each Z,R pair. No need to
compose calculations for multiple wheels
        %%UPDATE 2019-02-19:: DON'T COMPUTE ANYTHING (AND KEEP ZEROS) IF
THERE'S 0 TRAFFIC IN THIS CATEGORY.
       if axlesSingle6(k,j)~=0   %"if there's actual traffic of these axles"
            %%UPDATE 2019-03-19:: CONVERT LOAD TO NEWTON ( ton x 9800), DISTANCES TO METERS (cm x 0.01), AND E TO PA (psi x 1.000.000 / 145.04)
            %%update V2019-03-31:: pass pressure [qj] instead of total load to
MLE_sigma.
            %%update V2019-05-14: Add 4th dimension to sigmaXXX outputs from
%MLE
            aj = 0.01*aSingle6(j);
            qj = 1/2*9800*axlesSingle6Weights(j)/(pi*(aj)^2);

[sigmaZsingle6(:,:,j,k),sigmaXsingle6(:,:,j,k),sigmaYsingle6(:,:,j,k)] =
MLE_sigma(qj,aj,rSingle6(j,:),z,0.01*paveDepths(1:end-1),layersE(:,j),layersv(:,j));
       end
     %create these auxiliar auxE and aux_Poisson variables with the E and v
for this load level to use with the conversion to strain       %equations
     auxLayersE = getAuxEfromE(0.01*cumsum(paveDepths),z,layersE(:,j));   %
%stretch to the z domain, will need it to superimpose the horizontal stresses
[although not needed in single and 6-ton axles...]
     auxLayersv = getAuxEfromE(0.01*cumsum(paveDepths),z,layersv(:,j));
     %%these two above are in vector format [length(z) by 1]. need to convert
     %%them to z x r matrices. Multiply them for ones(1,length(r))
     auxLayersE = auxLayersE * ones(1,nrs6);
     auxLayersv = auxLayersv * ones(1,nrs6);

    %Convert stresses to strains (refer to Huang 04, chap 3)
    %%update V2019-05-14: Add 4th dimension to epsX, epsY, epsZ, epsH
    epsXsingle6(:,:,j,k) = 1./auxLayersE.*(sigmaXsingle6(:,:,j,k) - auxLayersv.*(sigmaYsingle6(:,:,j,k) + sigmaZsingle6(:,:,j,k)));
    epsYsingle6(:,:,j,k) = 1./auxLayersE.*(sigmaYsingle6(:,:,j,k) - auxLayersv.*(sigmaXsingle6(:,:,j,k) + sigmaZsingle6(:,:,j,k)));
    epsZsingle6(:,:,j,k) = 1./auxLayersE.*(sigmaZsingle6(:,:,j,k) - auxLayersv.*(sigmaXsingle6(:,:,j,k) + sigmaYsingle6(:,:,j,k)));
%     gmXYsingleL(:,:,j) = zeros(size(epsSingleL));
%%simplified formular for epsH, since there is no tauXY
    epsHsingle6(:,:,j,k) = 0.5.*(epsXsingle6(:,:,j,k) + epsYsingle6(:,:,j,k)) - 0.5.*(epsXsingle6(:,:,j,k)-epsYsingle6(:,:,j,k));
    %end
    %end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%nrs = length(rSingles);  [6 pos]
nax = length(axlesSingle10Weights);
rSingle10 = zeros(nax,nrs10);

%%compose the vector of Es from the HMA layers (temperature and load varying) and the gran. layers MR..
auxE_HMA = EDynSingle105(:,:,k);
auxv_HMA = HMAPoissonSingle105(:,:,k);%%get the poisson coefficients for all HMA layers (rows) and all load ranges (cols) for timestamp k (stack)

layersE = MR(k,:)' * ones(1,nax);     %Step 1: retrieve the granular and subgrade's MRs and turn it to a matrix the same size of the E*. The product gives a matrix [layers x axleWeight] in size
layersE = [auxE_HMA;layersE];         %Step 2: stack the two matrices together
%%update v2019-03-19:: unit consistency check, convert layersE from PSI to Pa
layersE = layersE/145.04*1e6;

layersv = granPoiss * ones(1,nax);
layersv = [auxv_HMA; layersv];         %repeat to get the Poissons

for j = 1:nax
    %update v2019-03-19: replaced r=0 in auxR1-auxR2 with r=0.01m
    rSingle10(j,:) = 0:1:5;
    %since I need to compose the effects of the dual wheels (and rSingle10 measures from the midpoint between the two wheels' centers (as in the MEPDG), I need to define respective r vectors for each.
    %auxR1 and auxR2 are the radial distance to each wheel from each rSingle10 position [calculated manually elsewhere]
    auxR1 = [0.5*dualWheelSep 0.75*dualWheelSep-0.5*0.01*aSingle105(j) dualWheelSep-0.01*aSingle105(j) dualWheelSep dualWheelSep+0.01*aSingle105(j) dualWheelSep+0.01*aSingle105(j)+0.10];
    auxR2 = [0.5*dualWheelSep 0.25*dualWheelSep+0.5*0.01*aSingle105(j) 0.01*aSingle105(j) 0.01 0.01*aSingle105(j) 0.01*aSingle105(j)+0.10];
    auxAlpha1 = zeros(1,6);   %%update v2019-05-20... rolled back these angles
    auxAlpha2 = [pi pi pi 0 0 0];
    %for the rotation calculations, extend auxAlpha1 and auxAlpha2 to the size
of auxSigmaX1 and auxSigmaY2 (z x r)
    auxAlpha1 = ones(length(z),1)*auxAlpha1;
    auxAlpha2 = ones(length(z),1)*auxAlpha2;
    %for j = 1:nax
        %get axle Load (tons), convert to load by wheel -> That's axleSingleLWeights(j)/2
        %get load radius - - - - computed with wheelFootprint (called from the
MainCode) - - IT'S GIVEN IN CM and accounts for the fact that the axle load is
equally divided over all the axle's wheels!
        %get E, nu, height for all materials! - watchful for asphalt materials!
        %get the sigmaZ, sigmaR, sigmaT values at each Z,R pair. - COMPOSE THE
EFFECTS OF THE DUAL WHEELS!

        %%UPDATE 2019-02-19:: DON'T COMPUTE ANYTHING (AND KEEP ZEROS) IF
THERE'S 0 TRAFFIC IN THIS CATEGORY.
        %%update 2019-03-19:: Correted it to properly work for each load
%%level and axle type
       if axlesSingle105(k,j)~=0   %"if there's actual traffic of these axles
and load value"
            %%UPDATE 2019-03-19:: CONVERT LOAD TO NEWTON ( ton x 9800), DISTANCES TO METERS (cm x 0.01), AND E TO PA (psi x 1.000.000 / 145.04)
             %update V2019-03-31:: pass pressure [qj] instead of total load to
MLE_sigma.
            aj = 0.01*aSingle105(j);
            qj = 1/4*9800*axlesSingle10Weights(j)/(pi*(aj)^2);
            [auxSigmaZ1,auxSigmaR1,auxSigmaT1] =
MLE_sigma(qj,aj,auxR1,z,0.01*paveDepths(1:end-1),layersE(:,j),layersv(:,j));
            [auxSigmaZ2,auxSigmaR2,auxSigmaT2] =
MLE_sigma(qj,aj,auxR2,z,0.01*paveDepths(1:end-1),layersE(:,j),layersv(:,j));
       else
           auxSigmaZ1 = zeros(length(z),nrs10);
           auxSigmaR1 = zeros(length(z),nrs10);
           auxSigmaT1 = zeros(length(z),nrs10);
           auxSigmaZ2 = zeros(length(z),nrs10);
           auxSigmaR2 = zeros(length(z),nrs10);
           auxSigmaT2 = zeros(length(z),nrs10);
       end
        %%update V2019-05-14: Add 4th dimension to sigmaXXX outputs from %MLE
       sigmaZsingle10(:,:,j,k) = auxSigmaZ1 + auxSigmaZ2;   %these sums here
are the superposition of the sigmas of both wheels at each rSingle10 position
        %rotate the sigmaR and sigmaT to cartesian coordinates, sum them together
        auxSigmaX1 = auxSigmaR1.*(cos(auxAlpha1).^2) + auxSigmaT1.*(sin(auxAlpha1).^2);
        auxSigmaX2 = auxSigmaR2.*(cos(auxAlpha2).^2) + auxSigmaT2.*(sin(auxAlpha2).^2);
        auxSigmaY1 = auxSigmaR1.*(sin(auxAlpha1).^2) + auxSigmaT1.*(cos(auxAlpha1).^2);
        auxSigmaY2 = auxSigmaR2.*(sin(auxAlpha2).^2) + auxSigmaT2.*(cos(auxAlpha2).^2);
        auxTauXY1  = (auxSigmaR1 - auxSigmaT1).*sin(auxAlpha1).*cos(auxAl-
pha1);
        auxTauXY2  = (auxSigmaR2 - auxSigmaT2).*sin(auxAlpha2).*cos(auxAl-
pha2);

        %%update V2019-05-14: Add 4th dimension to sigmaXXX outputs from %MLE
        sigmaXsingle10(:,:,j,k) = auxSigmaX1 + auxSigmaX2;
        sigmaYsingle10(:,:,j,k) = auxSigmaY1 + auxSigmaY2;
        tauXYsingle10(:,:,j,k) = auxTauXY1 + auxTauXY2;

        %create these auxiliar auxE and aux_Poisson variables with the E
%and v for this load level to use with the conversion to strain        %equa-
tions
        auxLayersE = getAuxEfromE(0.01*cumsum(paveDepths),z,layersE(:,j));   %
%stretch to the z domain, will need it to superimpose the horizontal stresses
[although not needed in single and 6-ton axles...]
        auxLayersv = getAuxEfromE(0.01*cumsum(paveDepths),z,layersv(:,j));
        %%these two above are in vector format [length(z) by 1]. need to con-
vert
        %%them to z x r matrices. Multiply them for ones(1,length(r))
        auxLayersE = auxLayersE * ones(1,nrs10);
        auxLayersv = auxLayersv * ones(1,nrs10);

        %Convert stresses to strains (refer to Huang 04, chap 3)
        %%update V2019-05-14: Add 4th dimension to epsX, epsY, epsZ, epsH
        epsXsingle10(:,:,j,k) = 1./auxLayersE.*(sigmaXsingle10(:,:,j,k) -
auxLayersv.*(sigmaYsingle10(:,:,j,k) + sigmaZsingle10(:,:,j,k)));
        epsYsingle10(:,:,j,k) = 1./auxLayersE.*(sigmaYsingle10(:,:,j,k) -
auxLayersv.*(sigmaXsingle10(:,:,j,k) + sigmaZsingle10(:,:,j,k)));
        epsZsingle10(:,:,j,k) = 1./auxLayersE.*(sigmaZsingle10(:,:,j,k) -
auxLayersv.*(sigmaXsingle10(:,:,j,k) + sigmaYsingle10(:,:,j,k)));
        gmXYsingle10(:,:,j,k) = 2./
auxLayersE.*(1+auxLayersv).*tauXYsingle10(:,:,j,k);
        %update v2019-03-19:: corrected formula for epsH, missing a ^2
        %UPDATE V2019-05-09:: BUG CORRECTED - the 'minus' sign in -gmXY is in-
correct, should be a sum!
        epsHsingle10(:,:,j,k) = 0.5.*(epsXsingle10(:,:,j,k) +
epsYsingle10(:,:,j,k)) - sqrt(0.25.*(epsXsingle10(:,:,j,k)-
epsYsingle10(:,:,j,k)).^2 + gmXYsingle10(:,:,j,k).^2);
    %end
end

%% 2 Tandems - 10-ton, 14-ton, 18-ton

nax = length(axlesTandem10Weights);
rTandem10 = zeros(nax,nrTa10);

%%compose the vector of Es from the HMA layers (temperature and load varying)
and the gran. layers MR..
auxE_HMA = EDynTandem10(:,:,k);
auxv_HMA = HMAPoissonTandem10(:,:,k);%%get the poisson coefficients for all
HMA layers (rows) and all load ranges (cols) for timestamp k (stack)
```

```
layersE = MR(k,:)' * ones(1,nax);      %Step 1: retrieve the granular and sub-
grade's MRs and turn it to a matrix the same size of the E*. The product gives
a matrix [layers x axleWeight] in size
layersE = [auxE_HMA;layersE];          %Step 2: stack the two matrices together
%%update v2019-03-19:: unit consistency check, convert layersE from PSI to Pa
layersE = layersE/145.04*1e6;

layersv = granPoiss * ones(1,nax);
layersv = [auxv_HMA; layersv];          %repeat to get the Poissons

for j = 1:nax
    %update v2019-03-19: replaced r=0 in auxR1-auxR2 with r=0.01m
    rTandem10(j,:) = 0:1:5;
    %since I need to compose the effects of the dual wheels (and rTandem10
measures from the midpoint between the two axles' centers, I need to define
respective r vectors for each.
    %auxR1 and auxR2 are the radial distance to each wheel from each rSingle10
position [calculated manually elsewhere]
    auxR1 = [0.5*tandemAxleSep
sqrt((0.5*tandemAxleSep)^2+(0.01*aTandem10(j))^2)
sqrt((0.5*tandemAxleSep)^2+(0.01*aTandem10(j)+0.10)^2) tandemAxleSep
sqrt(tandemAxleSep^2+(0.01*aTandem10(j))^2) sqrt(tandemAxleSep^2+(0.01*aTan-
dem10(j)+0.10)^2)];
    auxR2 = [0.5*tandemAxleSep
sqrt((0.5*tandemAxleSep)^2+(0.01*aTandem10(j))^2)
sqrt((0.5*tandemAxleSep)^2+(0.01*aTandem10(j)+0.10)^2) 0.01 0.01*aTandem10(j)
0.01*aTandem10(j)+0.10];
    auxAlpha1 = [0.5*pi atan(tandemAxleSep/(2*0.01*aTandem10(j))) atan(tande-
mAxleSep/(2*0.01*aTandem10(j)+2*0.10)) 0.5*pi atan(tandemAxleSep/(0.01*aTan-
dem10(j))) atan(tandemAxleSep/(0.01*aTandem10(j)+0.10))];
    auxAlpha2 = [-0.5*pi atan(-tandemAxleSep/(2*0.01*aTandem10(j)))  atan(-
tandemAxleSep/(2*0.01*aTandem10(j)+2*0.10))  0  0  0];
    %for the rotation calculations, extend auxAlpha1 and auxAlpha2 to the size
of auxSigmaX1 and auxSigmaY2 (z x r)
    auxAlpha1 = ones(length(z),1)*auxAlpha1;
    auxAlpha2 = ones(length(z),1)*auxAlpha2;
    %get axle Load (tons) -  convert to load by wheel!
    %get load radius - - - - computed with wheelFootprint (called from the
MainCode) - - IT'S GIVEN IN CM and accounts for the fact that the axle load is
equally divided over all the axle's wheels!
    %get E, nu, height for all materials! - watchful for asphalt materials!

     %%UPDATE 2019-02-19:: DON'T COMPUTE ANYTHING (AND KEEP ZEROS) IF THERE'S
0 TRAFFIC IN THIS CATEGORY.
      %%update 2019-03-19:: Corrected it to properly work for each load      %
%level and axle type
       if axlesTandem10(k,j)~=0   %"if there's actual traffic of these axles"
            %%UPDATE 2019-03-19:: CONVERT LOAD TO NEWTON ( ton x 9800), DIS-
TANCES TO METERS (cm x 0.01), AND E TO PA (psi x 1.000.000 / 145.04)
            %update V2019-03-31:: pass pressure [qj] instead of total load to
MLE_sigma.
            aj = 0.01*aTandem10(j);
            qj = 1/4*9800*axlesTandem10Weights(j)/(pi*(aj)^2);
            [auxSigmaZ1,auxSigmaR1,auxSigmaT1] =
MLE_sigma(qj,aj,auxR1,z,0.01*paveDepths(1:end-1),layersE(:,j),layersv(:,j));
            [auxSigmaZ2,auxSigmaR2,auxSigmaT2] =
MLE_sigma(qj,aj,auxR2,z,0.01*paveDepths(1:end-1),layersE(:,j),layersv(:,j));
       else
           auxSigmaZ1 = zeros(length(z),nrTa10);
           auxSigmaR1 = zeros(length(z),nrTa10);
           auxSigmaT1 = zeros(length(z),nrTa10);
           auxSigmaZ2 = zeros(length(z),nrTa10);
           auxSigmaR2 = zeros(length(z),nrTa10);
           auxSigmaT2 = zeros(length(z),nrTa10);
       end
    sigmaZtandem10(:,:,j,k) = auxSigmaZ1 + auxSigmaZ2;   %these sums here are
the superposition of the sigmas of both wheels at each rSingle10 position
    %rotate the sigmaR and sigmaT to cartesian coordinates, sum them together
    auxSigmaX1 = auxSigmaR1.*(cos(auxAlpha1).^2) + auxSigmaT1.*(sin(auxAl-
pha1).^2);
    auxSigmaX2 = auxSigmaR2.*(cos(auxAlpha2).^2) + auxSigmaT2.*(sin(auxAl-
pha2).^2);
    auxSigmaY1 = auxSigmaR1.*(sin(auxAlpha1).^2) + auxSigmaT1.*(cos(auxAl-
pha1).^2);
    auxSigmaY2 = auxSigmaR2.*(sin(auxAlpha2).^2) + auxSigmaT2.*(cos(auxAl-
pha2).^2);
    auxTauXY1  = (auxSigmaR1 - auxSigmaT1).*sin(auxAlpha1).*cos(auxAlpha1);
    auxTauXY2  = (auxSigmaR2 - auxSigmaT2).*sin(auxAlpha2).*cos(auxAlpha2);

    %%update V2019-05-14: Add 4th dimension to sigmaXXX outputs
    sigmaXtandem10(:,:,j,k) = auxSigmaX1 + auxSigmaX2;
    sigmaYtandem10(:,:,j,k) = auxSigmaY1 + auxSigmaY2;
    tauXYtandem10(:,:,j,k) = auxTauXY1 + auxTauXY2;

    %create these auxiliar auxE and aux_Poisson variables with the E
%and v for this load level to use with the conversion to strain        %equa-
tions
    auxLayersE = getAuxEfromE(0.01*cumsum(paveDepths),z,layersE(:,j));   %
%stretch to the z domain, will need it to superimpose the horizontal stresses
[although not needed in single and 6-ton axles...]
    auxLayersv = getAuxEfromE(0.01*cumsum(paveDepths),z,layersv(:,j));
    %%these two above are in vector format [length(z) by 1]. need to convert
    %%them to z x r matrices. Multiply them for ones(1,length(r))
    auxLayersE = auxLayersE * ones(1,nrTa10);
    auxLayersv = auxLayersv * ones(1,nrTa10);

    %Convert stresses to strains (refer to Huang 04, chap 3)
    %%update V2019-05-14: Add 4th dimension to epsX, epsY, epsZ, epsH
    %%BUG DETECTED V2019-05-14: CORRECT FORMULA FOR gmXY (was calling
    %%"single10")
    epsXtandem10(:,:,j,k) = 1./auxLayersE.*(sigmaXtandem10(:,:,j,k) - auxLay-
ersv.*(sigmaYtandem10(:,:,j,k) + sigmaZtandem10(:,:,j,k)));
    epsYtandem10(:,:,j,k) = 1./auxLayersE.*(sigmaYtandem10(:,:,j,k) - auxLay-
ersv.*(sigmaXtandem10(:,:,j,k) + sigmaZtandem10(:,:,j,k)));
    epsZtandem10(:,:,j,k) = 1./auxLayersE.*(sigmaZtandem10(:,:,j,k) - auxLay-
ersv.*(sigmaXtandem10(:,:,j,k) + sigmaYtandem10(:,:,j,k)));
```

```
    gmXYtandem10(:,:,j,k) = 2./
auxLayersE.*(1+auxLayersv).*tauXYtandem10(:,:,j,k);
    %update v2019-03-19:: corrected formula for epsH, missing a ^2
    epsHtandem10(:,:,j,k) = 0.5.*(epsXtandem10(:,:,j,k) +
epsYtandem10(:,:,j,k)) - sqrt(0.25.*(epsXtandem10(:,:,j,k)-
epsYtandem10(:,:,j,k)).^2 + gmXYtandem10(:,:,j,k).^2);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nax = length(axlesTandem14Weights);
rTandem14 = zeros(nax,nrTa14);

%%compose the vector of Es from the HMA layers (temperature and load varying)
and the gran. layers MR..
auxE_HMA = EDynTandem14(:,:,k);
auxv_HMA = HMAPoissonTandem14(:,:,k);%%get the poisson coefficients for all
HMA layers (rows) and all load ranges (cols) for timestamp k (stack)

layersE = MR(k,:)' * ones(1,nax);      %Step 1: retrieve the granular and sub-
grade's MRs and turn it to a matrix the same size of the E*. The product gives
a matrix [layers x axleWeight] in size
layersE = [auxE_HMA;layersE];          %Step 2: stack the two matrices together
%%update v2019-03-19:: unit consistency check, convert layersE from PSI to Pa
layersE = layersE/145.04*1e6;

layersv = granPoiss * ones(1,nax);
layersv = [auxv_HMA; layersv];          %repeat to get the Poissons

for j = 1:nax
    %update V2019-05-20: bug detected in auxAlpha1. Corrected. Also rolled
back angle convention
    %update v2019-03-19: replaced r=0 in auxR1-auxR2 with r=0.01m
    rTandem14(j,:) = 0:1:14;
    %since I need to compose the effects of the THREE wheels (and rTandem14
measures from the center of the outermost wheel in the dualwheel axle, I need
to define respective r vectors for each.
    %auxR1 and auxR2 are the radial distance to each wheel from each rSingle10
position [calculated manually elsewhere]
    auxR1 = [0.01 0.01*aTandem14(j) 0.25*dualWheelSep+0.5*0.01*aTandem14(j)
0.5*dualWheelSep 0.75*dualWheelSep-0.5*0.01*aTandem14(j) dualWheelSep-
0.01*aTandem14(j) dualWheelSep dualWheelSep+0.01*aTandem14(j)
dualWheelSep+0.01*aTandem14(j)+0.10 ...
        0.5*tandemAxleSep sqrt((0.5*tandemAxleSep)^2+(0.01*aTandem14(j))^2)
sqrt((0.5*tandemAxleSep)^2+(0.5*dualWheelSep)^2) tandemAxleSep
sqrt((0.01*aTandem14(j))^2+tandemAxleSep^2) sqrt(tandemAxleSep^2+(0.01*aTan-
dem14(j)+0.10)^2)];
    auxR2 = [dualWheelSep dualWheelSep-0.01*aTandem14(j) 0.75*dualWheelSep-
0.5*0.01*aTandem14(j) 0.5*dualWheelSep 0.25*dualWheelSep+0.01*0.5*aTandem14(j)
0.01*aTandem14(j) 0.01 0.01*aTandem14(j) 0.01*aTandem14(j)+0.10 ...
        sqrt(dualWheelSep^2+(0.5*tandemAxleSep)^2) sqrt((0.5*tandemAxleSep)^2+
(dualWheelSep-0.01*aTandem14(j))^2) sqrt((0.5*tandemAxleSep)^2+(0.5*dual-
WheelSep)^2) sqrt(tandemAxleSep^2+dualWheelSep^2) sqrt(tandemAxleSep^2+(dual-
WheelSep-0.01*aTandem14(j))^2) sqrt(tandemAxleSep^2+(dualWheelSep-0.01*aTandem14(j)-0.10)^2)];
    auxR3 = [tandemAxleSep sqrt(tandemAxleSep^2+(0.01*aTandem14(j))^2)
sqrt(tandemAxleSep^2+(0.25*dualWheelSep+0.5*0.01*aTandem14(j))^2) sqrt(tandemAxleSep^2+(0.5*dualWheelSep)^2) sqrt(tandemAxleSep^2+(0.75*dualWheelSep-
0.5*0.01*aTandem14(j))^2) sqrt(tandemAxleSep^2+(dualWheelSep-
0.01*aTandem14(j))^2) sqrt(tandemAxleSep^2+dualWheelSep^2)
sqrt(tandemAxleSep^2+(dualWheelSep+0.01*aTandem14(j))^2) sqrt(tandemAxleSep^2+
(dualWheelSep+0.01*aTandem14(j)+0.10)^2)...
        0.5*tandemAxleSep sqrt((0.01*aTandem14(j))^2+(0.5*tandemAxleSep)^2)
sqrt((0.5*tandemAxleSep)^2+(0.5*dualWheelSep)^2) 0.01 0.01*aTandem14(j)
0.01*aTandem14(j)+0.10];
    auxAlpha1 = [0 0 0 0 0 0 0 0 0 ...
        0.5*pi atan(tandemAxleSep/(2*0.01*aTandem14(j))) atan(tandemAxleSep/
dualWheelSep) 0.5*pi atan(tandemAxleSep/(0.01*aTandem14(j))) atan(2*tandemAxleSep/dualWheelSep)];
    auxAlpha2 = [pi pi pi pi pi pi 0 0 0 ...
        pi-atan(0.5*tandemAxleSep/dualWheelSep) pi-atan(0.5*tandemAxleSep/(dualWheelSep-0.01*aTandem14(j))) pi-atan(tandemAxleSep/dualWheelSep) pi-
atan(tandemAxleSep/dualWheelSep) pi-atan(tandemAxleSep/(dualWheelSep-
0.01*aTandem14(j))) pi-atan(2*tandemAxleSep/dualWheelSep)];
    auxAlpha3 = [-pi/2 atan(-tandemAxleSep/auxR1(2)) atan(-tandemAxleSep/
auxR1(3)) atan(-tandemAxleSep/auxR1(4)) atan(-tandemAxleSep/auxR1(5)) atan(-
tandemAxleSep/auxR1(6)) ...
        atan(-tandemAxleSep/auxR1(7)) atan(-tandemAxleSep/auxR1(8)) atan(-
tandemAxleSep/auxR1(9)) -pi/2 atan(-0.5*tandemAxleSep/(0.01*aTandem14(j)))
atan(-tandemAxleSep/dualWheelSep) 0 0 0];
    %for the rotation calculations, extend auxAlpha1 and auxAlpha2 to the size
of auxSigmaX1 and auxSigmaY2 (z x r)
    auxAlpha1 = ones(length(z),1)*auxAlpha1;
    auxAlpha2 = ones(length(z),1)*auxAlpha2;
    auxAlpha3 = ones(length(z),1)*auxAlpha3;

    %get axle Load (tons) - convert to load by wheel!
    %get load radius - - - - computed with wheelFootprint (called from the
MainCode) - - IT'S GIVEN IN CM and accounts for the fact that the axle load is
equally divided over all the axle's wheels!
    %get E, nu, height for all materials! - watchful for asphalt %materials!
         %%UPDATE 2019-02-19:: DON'T COMPUTE ANYTHING (AND KEEP ZEROS) IF
THERE'S 0 TRAFFIC IN THIS CATEGORY.
          %%update 2019-03-19:: Corrected it to properly work for each load
level and axle type
          %%UPDATE 2019-03-19:: CONVERT LOAD TO NEWTON ( ton x 9800), DISTANCES TO METERS (cm x 0.01), AND E TO PA (psi x 1.000.000 / 145.04)
       if axlesTandem14(k,j)~=0   %"if there's actual traffic of these axles"
           %update V2019-03-31: pass pressure [qj] instead of total load to
MLE_sigma.
            aj = 0.01*aTandem14(j);
            qj = 1/6*9800*axlesTandem14Weights(j)/(pi*(aj)^2);
            [auxSigmaZ1,auxSigmaR1,auxSigmaT1] =
MLE_sigma(qj,aj,auxR1,z,0.01*paveDepths(1:end-1),layersE(:,j),layersv(:,j));
            [auxSigmaZ2,auxSigmaR2,auxSigmaT2] =
MLE_sigma(qj,aj,auxR2,z,0.01*paveDepths(1:end-1),layersE(:,j),layersv(:,j));
```

```
            [auxSigmaZ3,auxSigmaR3,auxSigmaT3] = 
MLE_sigma(qj,aj,auxR3,z,0.01*paveDepths(1:end-1),layersE(:,j),layersv(:,j));
       else
           auxSigmaZ1 = zeros(length(z),nrTa14);
           auxSigmaR1 = zeros(length(z),nrTa14);
           auxSigmaT1 = zeros(length(z),nrTa14);
           auxSigmaZ2 = zeros(length(z),nrTa14);
           auxSigmaR2 = zeros(length(z),nrTa14);
           auxSigmaT2 = zeros(length(z),nrTa14);
           auxSigmaZ3 = zeros(length(z),nrTa14);
           auxSigmaR3 = zeros(length(z),nrTa14);
           auxSigmaT3 = zeros(length(z),nrTa14);
       end

    sigmaZtandem14(:,:,j,k) = auxSigmaZ1 + auxSigmaZ2 + auxSigmaZ3;   %these 
sums here are the superposition of the sigmas of both wheels at each rSingle10 
position

     %rotate the sigmaR and sigmaT to cartesian coordinates, sum them together
    auxSigmaX1 = auxSigmaR1.*(cos(auxAlpha1).^2) + auxSigmaT1.*(sin(auxAlpha1).^2);
    auxSigmaX2 = auxSigmaR2.*(cos(auxAlpha2).^2) + auxSigmaT2.*(sin(auxAlpha2).^2);
    auxSigmaX3 = auxSigmaR3.*(cos(auxAlpha3).^2) + auxSigmaT3.*(sin(auxAlpha3).^2);
    auxSigmaY1 = auxSigmaR1.*(sin(auxAlpha1).^2) + auxSigmaT1.*(cos(auxAlpha1).^2);
    auxSigmaY2 = auxSigmaR2.*(sin(auxAlpha2).^2) + auxSigmaT2.*(cos(auxAlpha2).^2);
    auxSigmaY3 = auxSigmaR3.*(sin(auxAlpha3).^2) + auxSigmaT3.*(cos(auxAlpha3).^2);
    auxTauXY1  = (auxSigmaR1 - auxSigmaT1).*sin(auxAlpha1).*cos(auxAlpha1);
    auxTauXY2  = (auxSigmaR2 - auxSigmaT2).*sin(auxAlpha2).*cos(auxAlpha2);
    auxTauXY3  = (auxSigmaR3 - auxSigmaT3).*sin(auxAlpha3).*cos(auxAlpha3);

    %%update V2019-05-14: Add 4th dimension to sgimaXXX outputs
    sigmaXtandem14(:,:,j,k) = auxSigmaX1 + auxSigmaX2 + auxSigmaX3;
    sigmaYtandem14(:,:,j,k) = auxSigmaY1 + auxSigmaY2 + auxSigmaY3;
    tauXYtandem14(:,:,j,k) = auxTauXY1 + auxTauXY2 + auxTauXY3;

    %create these auxiliar auxE and aux_Poisson variables with the E
%and v for this load level to use with the conversion to strain        %equations
    auxLayersE = getAuxEfromE(0.01*cumsum(paveDepths),z,layersE(:,j));   % 
%stretch to the z domain, will need it to superimpose the horizontal stresses 
[although not needed in single and 6-ton axles...]
    auxLayersv = getAuxEfromE(0.01*cumsum(paveDepths),z,layersv(:,j));
    %%these two above are in vector format [length(z) by 1]. need to convert
    %%them to z x r matrices. Multiply them for ones(1,length(r))
    auxLayersE = auxLayersE * ones(1,nrTa14);
    auxLayersv = auxLayersv * ones(1,nrTa14);

    %Convert stresses to strains (refer to Huang 04, chap 3)
    %%update V2019-05-14: Add 4th dimension to epsX, epsY, epsZ, epsH
    epsXtandem14(:,:,j,k) = 1./auxLayersE.*(sigmaXtandem14(:,:,j,k) - auxLay-
ersv.*(sigmaYtandem14(:,:,j,k) + sigmaZtandem14(:,:,j,k)));
    epsYtandem14(:,:,j,k) = 1./auxLayersE.*(sigmaYtandem14(:,:,j,k) - auxLay-
ersv.*(sigmaXtandem14(:,:,j,k) + sigmaZtandem14(:,:,j,k)));
    epsZtandem14(:,:,j,k) = 1./auxLayersE.*(sigmaZtandem14(:,:,j,k) - auxLay-
ersv.*(sigmaXtandem14(:,:,j,k) + sigmaYtandem14(:,:,j,k)));
    gmXYtandem14(:,:,j,k) = 2./
auxLayersE.*(1+auxLayersv).*tauXYtandem14(:,:,j,k);
    %update v2019-03-19:: corrected formula for epsH, missing a ^2
    epsHtandem14(:,:,j,k) = 0.5.*(epsXtandem14(:,:,j,k) +
epsYtandem14(:,:,j,k)) - sqrt(0.25.*(epsXtandem14(:,:,j,k)-
epsYtandem14(:,:,j,k)).^2 + gmXYtandem14(:,:,j,k).^2);
%
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nax = length(axlesTandemWeights);
rTandem18 = zeros(nax,nrTa18);

%%compose the vector of Es from the HMA layers (temperature and load varying)
and the gran. layers MR..
auxE_HMA = EDynTandem18(:,:,k);
auxv_HMA = HMAPoissonTandem18(:,:,k);%%get the poisson coefficients for all
HMA layers (rows) and all load ranges (cols) for timestamp k (stack)

layersE = MR(k,:)' * ones(1,nax);      %Step 1: retrieve the granular and sub-
grade's MRs and turn it to a matrix the same size of the E*. The product gives
a matrix [layers x axleWeight] in size
layersE = [auxE_HMA;layersE];          %Step 2: stack the two matrices together
%%update v2019-03-19:: unit consistency check, convert layersE from PSI to Pa
layersE = layersE/145.04*1e6;

layersv = granPoiss * ones(1,nax);
layersv = [auxv_HMA; layersv];          %repeat to get the Poissons

for j = 1:nax
    %update v2019-05-20: Rolled back angle convention (was incorrect)
    %update v2019-03-19: replaced r=0 in auxR1-auxR2 with r=0.01m
  rTandem18(j,:) = 0:1:11;
    %since I need to compose the effects of the THREE wheels (and rTandem18
measures from the center of the outermost wheel in the dualwheel axle, I need
to define respective r vectors for each.
    %auxR1 and auxR2 are the radial distance to each wheel from each rSingle10
position [calculated manually elsewhere]
    auxR1 = [sqrt((0.5*dualWheelSep)^2+(0.5*tandemAxleSep)^2) sqrt((0.5*dual-
WheelSep)^2+(0.75*tandemAxleSep-0.5*0.01*aTandem18(j))^2) sqrt((0.5*dual-
WheelSep)^2+(tandemAxleSep-0.01*aTandem18(j))^2)
sqrt((0.5*dualWheelSep)^2+tandemAxleSep^2) sqrt((0.5*dualWheelSep)^2+(tande-
mAxleSep+0.01*aTandem18(j))^2) sqrt((0.5*dualWheelSep)^2+
(tandemAxleSep+0.01*aTandem18(j)+0.10)^2)...
        sqrt(tandemAxleSep^2+(0.5*dualWheelSep)^2) sqrt(tandemAxleSep^2+(-
0.5*0.01*aTandem18(j)+0.75*dualWheelSep)^2) sqrt(tandemAxleSep^2+(dual-
WheelSep-0.01*aTandem18(j))^2) sqrt(tandemAxleSep^2+dualWheelSep^2)
sqrt(tandemAxleSep^2+(dualWheelSep+0.01*aTandem18(j))^2) sqrt(tandemAxleSep^2+
(dualWheelSep+0.01*aTandem18(j)+0.1)^2)];
    auxR2 =  [sqrt((0.5*dualWheelSep)^2+(0.5*tandemAxleSep)^2) sqrt((0.5*dual-
WheelSep)^2+(0.25*tandemAxleSep+0.5*0.01*aTandem18(j))^2) sqrt((0.5*dual-
WheelSep)^2+(0.01*aTandem18(j))^2) 0.5*dualWheelSep
sqrt((0.5*dualWheelSep)^2+(0.01*aTandem18(j))^2)
sqrt((0.5*dualWheelSep)^2+(0.01*aTandem18(j)+0.10)^2)...
        sqrt(tandemAxleSep^2+(0.5*dualWheelSep)^2)
sqrt(tandemAxleSep^2+(0.5*0.01*aTandem18(j)+0.25*dualWheelSep)^2) sqrt(tande-
mAxleSep^2+(0.01*aTandem18(j))^2) tandemAxleSep
sqrt(tandemAxleSep^2+(0.01*aTandem18(j))^2) sqrt(tandemAxleSep^2+(0.01*aTan-
dem18(j)+0.1)^2)];
    auxR3 = [auxR1(1) auxR1(2) auxR1(3) auxR1(4) auxR1(5) auxR1(6) dual-
WheelSep/2 0.75*dualWheelSep-0.5*0.01*aTandem18(j) dualWheelSep-0.01*aTan-
dem18(j) dualWheelSep dualWheelSep+0.01*aTandem18(j) dualWheelSep+0.01*aTan-
dem18(j)+0.10];
    auxR4 = [auxR2(1) auxR2(2) auxR2(3) auxR2(4) auxR2(5) auxR2(6) dual-
WheelSep/2 0.25*dualWheelSep+0.5*0.01*aTandem18(j) 0.01*aTandem18(j) 0.01
0.01*aTandem18(j) 0.01*aTandem18(j)+0.10];
    auxAlpha1 = [atan(tandemAxleSep/dualWheelSep) atan(tandemAxleSep/(3*dual-
WheelSep/2-0.01*aTandem18(j))) atan(tandemAxleSep*0.5/(dualWheelSep-0.01*aTan-
dem18(j))) atan(tandemAxleSep*0.5/dualWheelSep) atan(tandemAxleSep*0.5/(dual-
WheelSep+0.01*aTandem18(j))) atan(tandemAxleSep*0.5/(dualWheelSep+0.01*aTan-
dem18(j)+0.10))...
        atan(2*tandemAxleSep/dualWheelSep) atan(tandemAxleSep/(0.75*dual-
WheelSep-0.5*0.01*aTandem18(j))) atan(tandemAxleSep/(dualWheelSep-0.01*aTan-
dem18(j))) atan(tandemAxleSep/dualWheelSep) atan(tandemAxleSep/
(dualWheelSep+0.01*aTandem18(j))) atan(tandemAxleSep/(dualWheelSep+0.01*aTan-
dem18(j)+0.10))];
    auxAlpha2 = [pi-atan(tandemAxleSep/dualWheelSep) pi-atan(tandemAxleSep/
(0.01*aTandem18(j)+0.5*dualWheelSep)) pi-atan(tandemAxleSep*0.5/(0.01*aTan-
dem18(j))) 0.5*pi atan(tandemAxleSep*0.5/(0.01*aTandem18(j))) atan(tandemAxle-
Sep*0.5/(0.10+0.01*aTandem18(j)))...
        pi-atan(2*tandemAxleSep/dualWheelSep) pi-atan(tandemAxleSep/(0.25*du-
alWheelSep+0.5*0.01*aTandem18(j))) pi-atan(tandemAxleSep/(0.01*aTandem18(j)))
pi/2 atan(tandemAxleSep/(0.01*aTandem18(j))) atan(tandemAxleSep/
(0.10+0.01*aTandem18(j)))];
    auxAlpha3 = [atan(-tandemAxleSep/dualWheelSep) atan(-tandemAxleSep/(3*du-
alWheelSep/2-0.01*aTandem18(j))) atan(-tandemAxleSep*0.5/(dualWheelSep-
0.01*aTandem18(j))) atan(-tandemAxleSep*0.5/dualWheelSep) atan(-
tandemAxleSep*0.5/(dualWheelSep+0.01*aTandem18(j))) atan(-tandemAxleSep*0.5/
(dualWheelSep+0.01*aTandem18(j)+0.10))...
        0 0 0 0 0 0 ];
    auxAlpha4 = [pi+atan(tandemAxleSep/dualWheelSep) pi+atan(tandemAxleSep/
(0.01*aTandem18(j)+0.5*dualWheelSep)) pi+atan(tandemAxleSep*0.5/(0.01*aTan-
dem18(j))) -0.5*pi atan(-tandemAxleSep*0.5/(0.01*aTandem18(j))) atan(-tande-
mAxleSep*0.5/(0.10+0.01*aTandem18(j)))...
        pi pi pi 0 0 0];
    %for the rotation calculations, extend auxAlpha1 and auxAlpha2 to the size
of auxSigmaX1 and auxSigmaY2 (z x r)
    auxAlpha1 = ones(length(z),1)*auxAlpha1;
    auxAlpha2 = ones(length(z),1)*auxAlpha2;
    auxAlpha3 = ones(length(z),1)*auxAlpha3;
    auxAlpha4 = ones(length(z),1)*auxAlpha4;
    %get axle Load (tons) -  convert to load by wheel!
    %get load radius - - - - computed with wheelFootprint (called from the
MainCode) - - IT'S GIVEN IN CM and accounts for the fact that the axle load is
equally divided over all the axle's wheels!
    %get E, nu, height for all materials! - watchful for asphalt materials!
     %%UPDATE 2019-02-19:: DON'T COMPUTE ANYTHING (AND KEEP ZEROS) IF THERE'S
0 TRAFFIC IN THIS CATEGORY.
      %%update 2019-03-19:: Correted it to properly work for each load level
and axle type
     %%UPDATE 2019-03-19:: CONVERT LOAD TO NEWTON ( ton x 9800), DISTANCES TO
METERS (cm x 0.01), AND E TO PA (psi x 1.000.000 / 145.04)
       if axlesTandem18(k,j)~=0   %"if there's actual traffic of these axles"
     %update V2019-03-31:: pass pressure [qj] instead of total load to
MLE_sigma.
            aj = 0.01*aTandem18(j);
            qj = 1/8*9800*axlesTandemWeights(j)/(pi*(aj)^2);
            [auxSigmaZ1,auxSigmaR1,auxSigmaT1] =
MLE_sigma(qj,aj,auxR1,z,0.01*paveDepths(1:end-1),layersE(:,j),layersv(:,j));
            [auxSigmaZ2,auxSigmaR2,auxSigmaT2] =
MLE_sigma(qj,aj,auxR2,z,0.01*paveDepths(1:end-1),layersE(:,j),layersv(:,j));
            [auxSigmaZ3,auxSigmaR3,auxSigmaT3] =
MLE_sigma(qj,aj,auxR3,z,0.01*paveDepths(1:end-1),layersE(:,j),layersv(:,j));
            [auxSigmaZ4,auxSigmaR4,auxSigmaT4] =
MLE_sigma(qj,aj,auxR4,z,0.01*paveDepths(1:end-1),layersE(:,j),layersv(:,j));
       else
           auxSigmaZ1 = zeros(length(z),nrTa18);
           auxSigmaR1 = zeros(length(z),nrTa18);
           auxSigmaT1 = zeros(length(z),nrTa18);
           auxSigmaZ2 = zeros(length(z),nrTa18);
           auxSigmaR2 = zeros(length(z),nrTa18);
           auxSigmaT2 = zeros(length(z),nrTa18);
           auxSigmaZ3 = zeros(length(z),nrTa18);
           auxSigmaR3 = zeros(length(z),nrTa18);
           auxSigmaT3 = zeros(length(z),nrTa18);
           auxSigmaZ4 = zeros(length(z),nrTa18);
           auxSigmaR4 = zeros(length(z),nrTa18);
           auxSigmaT4 = zeros(length(z),nrTa18);
       end
    sigmaZtandem18(:,:,j,k) = auxSigmaZ1 + auxSigmaZ2 + auxSigmaZ3 + auxSigmaZ4;
   %these sums here are the superposition of the sigmas of both wheels at
each rSingle10 position

     %rotate the sigmaR and sigmaT to cartesian coordinates, sum them together
    auxSigmaX1 = auxSigmaR1.*(cos(auxAlpha1).^2) + auxSigmaT1.*(sin(auxAlpha1).^2);
    auxSigmaX2 = auxSigmaR2.*(cos(auxAlpha2).^2) + auxSigmaT2.*(sin(auxAlpha2).^2);
    auxSigmaX3 = auxSigmaR3.*(cos(auxAlpha3).^2) + auxSigmaT3.*(sin(auxAlpha3).^2);
    auxSigmaX4 = auxSigmaR4.*(cos(auxAlpha4).^2) + auxSigmaT4.*(sin(auxAlpha4).^2);
    auxSigmaY1 = auxSigmaR1.*(sin(auxAlpha1).^2) + auxSigmaT1.*(cos(auxAlpha1).^2);
    auxSigmaY2 = auxSigmaR2.*(sin(auxAlpha2).^2) + auxSigmaT2.*(cos(auxAl-
pha2).^2);
    auxSigmaY3 = auxSigmaR3.*(sin(auxAlpha3).^2) + auxSigmaT3.*(cos(auxAl-
pha3).^2);
    auxSigmaY4 = auxSigmaR4.*(sin(auxAlpha4).^2) + auxSigmaT4.*(cos(auxAl-
pha4).^2);
    auxTauXY1  = (auxSigmaR1 - auxSigmaT1).*sin(auxAlpha1).*cos(auxAlpha1);
    auxTauXY2  = (auxSigmaR2 - auxSigmaT2).*sin(auxAlpha2).*cos(auxAlpha2);
    auxTauXY3  = (auxSigmaR3 - auxSigmaT3).*sin(auxAlpha3).*cos(auxAlpha3);
    auxTauXY4  = (auxSigmaR4 - auxSigmaT4).*sin(auxAlpha4).*cos(auxAlpha4);

    %%update V2019-05-14: Add 4th dimension to sigmaXXXX
    sigmaXtandem18(:,:,j,k) = auxSigmaX1 + auxSigmaX2 + auxSigmaX3 + auxSig-
maX4;
    sigmaYtandem18(:,:,j,k) = auxSigmaY1 + auxSigmaY2 + auxSigmaY3 + auxSig-
maY4;
    tauXYtandem18(:,:,j,k) = auxTauXY1 + auxTauXY2 + auxTauXY3 + auxTauXY4;

    %create these auxiliar auxE and aux_Poisson variables with the E
%and v for this load level to use with the conversion to strain equations
    auxLayersE = getAuxEfromE(0.01*cumsum(paveDepths),z,layersE(:,j));   %
%stretch to the z domain, will need it to superimpose the horizontal stresses
[although not needed in single and 6-ton axles...]
    auxLayersv = getAuxEfromE(0.01*cumsum(paveDepths),z,layersv(:,j));
    %%these two above are in vector format [length(z) by 1]. need to convert
them to z x r matrices. Multiply them for ones(1,length(r))
    auxLayersE = auxLayersE * ones(1,nrTa18);
    auxLayersv = auxLayersv * ones(1,nrTa18);

    %Convert stresses to strains (refer to Huang 04, chap 3)
    %%update V2019-05-14: Add 4th dimension to epsX, epsY, epsZ, epsH
    epsXtandem18(:,:,j,k) = 1./auxLayersE.*(sigmaXtandem18(:,:,j,k) - auxLay-
ersv.*(sigmaYtandem18(:,:,j,k) + sigmaZtandem18(:,:,j,k)));
    epsYtandem18(:,:,j,k) = 1./auxLayersE.*(sigmaYtandem18(:,:,j,k) - auxLay-
ersv.*(sigmaXtandem18(:,:,j,k) + sigmaZtandem18(:,:,j,k)));
    epsZtandem18(:,:,j,k) = 1./auxLayersE.*(sigmaZtandem18(:,:,j,k) - auxLay-
ersv.*(sigmaXtandem18(:,:,j,k) + sigmaYtandem18(:,:,j,k)));
    gmXYtandem18(:,:,j,k) = 2./
auxLayersE.*(1+auxLayersv).*tauXYtandem18(:,:,j,k);
    %update v2019-03-19:: corrected formula for epsH, missing a ^2
    epsHtandem18(:,:,j,k) = 0.5.*(epsXtandem18(:,:,j,k) +
epsYtandem18(:,:,j,k)) - sqrt(0.25.*(epsXtandem18(:,:,j,k)-
epsYtandem18(:,:,j,k)).^2 + gmXYtandem18(:,:,j,k).^2);
%
end

%% 3 Tridem - 25.5-ton only

nax = length(axlesTridemWeights);
rTridem = zeros(nax,nrTr);

%%compose the vector of Es from the HMA layers (temperature and load varying)
and the gran. layers MR..
auxE_HMA = EDynTridem(:,:,k);
```

```
auxv_HMA = HMAPoissonTridem(:,:,k);%%get the poisson coefficients for all HMA
layers (rows) and all load ranges (cols) for timestamp k (stack)

layersE = MR(k,:)' * ones(1,nax);     %Step 1: retrieve the granular and subgrade's
MRs and turn it to a matrix the same size of the E*. The product gives
a matrix [layers x axleWeight] in size
layersE = [auxE_HMA;layersE];         %Step 2: stack the two matrices together
%%update v2019-03-19:: unit consistency check, convert layersE from PSI to Pa
layersE = layersE/145.04*1e6;

layersv = granPoiss * ones(1,nax);
layersv = [auxv_HMA; layersv];         %repeat to get the Poissons

for j = 1:nax
    rTridem(j,:) = 0:1:17;
    %since I need to compose the effects of the THREE wheels (and rTridem measures
from the center of the outermost wheel in the dualwheel axle, I need to
define respective r vectors for each.
    %auxR1 and auxR2 are the radial distance to each wheel from each rSingle10
position [calculated manually elsewhere]
    auxR1 = [sqrt(tandemAxleSep^2+(0.5*dualWheelSep)^2)
sqrt(tandemAxleSep^2+(0.75*dualWheelSep-0.5*0.01*aTridem(j))^2) sqrt(tandemAxleSep^2+(dualWheelSep-0.01*aTridem(j))^2) sqrt(tandemAxleSep^2+dualWheelSep^2) sqrt(tandemAxleSep^2+(dualWheelSep+0.01*aTridem(j))^2) sqrt(tandemAxleSep^2+(dualWheelSep+0.01*aTridem(j)+0.10)^2)...
        sqrt((1.5*tandemAxleSep)^2+(0.5*dualWheelSep)^2) sqrt((1.5*tandemAxleSep)^2+(0.75*dualWheelSep-0.5*0.01*aTridem(j))^2) sqrt((1.5*tandemAxleSep)^2+
(dualWheelSep-0.01*aTridem(j))^2) sqrt((1.5*tandemAxleSep)^2+dualWheelSep^2)
sqrt((1.5*tandemAxleSep)^2+(dualWheelSep+0.01*aTridem(j))^2) sqrt((1.5*tandemAxleSep)^2+(dualWheelSep+0.01*aTridem(j)+0.10)^2)...
        sqrt((2*tandemAxleSep)^2+(0.5*dualWheelSep)^2)
sqrt((2*tandemAxleSep)^2+(0.75*dualWheelSep-0.5*0.01*aTridem(j))^2)
sqrt((2*tandemAxleSep)^2+(dualWheelSep-0.01*aTridem(j))^2) sqrt((2*tandemAxleSep)^2+dualWheelSep^2) sqrt((2*tandemAxleSep)^2+
(dualWheelSep+0.01*aTridem(j))^2) sqrt((2*tandemAxleSep)^2+
(dualWheelSep+0.01*aTridem(j)+0.10)^2)];
    auxR2 = [sqrt(tandemAxleSep^2+(0.5*dualWheelSep)^2)
sqrt(tandemAxleSep^2+(0.25*dualWheelSep+0.5*0.01*aTridem(j))^2) sqrt(tandemAxleSep^2+(0.01*aTridem(j))^2) tandemAxleSep sqrt(tandemAxleSep^2+(0.01*aTridem(j))^2) sqrt(tandemAxleSep^2+(0.01*aTridem(j)+0.10)^2)...
        sqrt((1.5*tandemAxleSep)^2+(0.5*dualWheelSep)^2) sqrt((1.5*tandemAxleSep)^2+(0.25*dualWheelSep+0.5*0.01*aTridem(j))^2)
sqrt((1.5*tandemAxleSep)^2+(0.01*aTridem(j))^2) 1.5*tandemAxleSep
sqrt((1.5*tandemAxleSep)^2+(0.01*aTridem(j))^2)
sqrt((1.5*tandemAxleSep)^2+(0.01*aTridem(j)+0.10)^2)...
        sqrt((2*tandemAxleSep)^2+(0.5*dualWheelSep)^2)
sqrt((2*tandemAxleSep)^2+(0.25*dualWheelSep+0.5*0.01*aTridem(j))^2)
sqrt((2*tandemAxleSep)^2+(0.01*aTridem(j))^2) 2*tandemAxleSep sqrt((2*tandemAxleSep)^2+(0.01*aTridem(j))^2) sqrt((2*tandemAxleSep)^2+(0.01*aTridem(j)
+0.10)^2)];
    auxR3 = [0.5*dualWheelSep 0.75*dualWheelSep-0.5*0.01*aTridem(j) dualWheelSep-0.01*aTridem(j) dualWheelSep dualWheelSep+0.01*aTridem(j) dualWheelSep+0.01*aTridem(j)+0.10...
```

```
        sqrt((0.5*tandemAxleSep)^2+(0.5*dualWheelSep)^2) sqrt((0.5*tandemAxleSep)^2+(0.75*dualWheelSep-0.5*0.01*aTridem(j))^2) sqrt((0.5*tandemAxleSep)^2+(dualWheelSep-0.01*aTridem(j))^2) sqrt((0.5*tandemAxleSep)^2+dualWheelSep^2) sqrt((0.5*tandemAxleSep)^2+(dualWheelSep+0.01*aTridem(j))^2) sqrt((0.5*tandemAxleSep)^2+(dualWheelSep+0.01*aTridem(j)+0.10)^2)...
        sqrt(tandemAxleSep^2+(0.5*dualWheelSep)^2) sqrt(tandemAxleSep^2+(0.75*dualWheelSep-0.5*0.01*aTridem(j))^2) sqrt(tandemAxleSep^2+(dualWheelSep-0.01*aTridem(j))^2) sqrt(tandemAxleSep^2+dualWheelSep^2) sqrt(tandemAxleSep^2+(dualWheelSep+0.01*aTridem(j))^2) sqrt(tandemAxleSep^2+(dualWheelSep+0.01*aTridem(j)+0.10)^2)];
    auxR4 = [0.5*dualWheelSep 0.25*dualWheelSep+0.5*0.01*aTridem(j) 0.01*aTridem(j) 0.01 0.01*aTridem(j) 0.01*aTridem(j)+0.10...
        sqrt((0.5*tandemAxleSep)^2+(0.5*dualWheelSep)^2) sqrt((0.5*tandemAxleSep)^2+(0.25*dualWheelSep+0.5*0.01*aTridem(j))^2) sqrt((0.5*tandemAxleSep)^2+(0.01*aTridem(j))^2) 1.5*tandemAxleSep sqrt((0.5*tandemAxleSep)^2+(0.01*aTridem(j))^2) sqrt((0.5*tandemAxleSep)^2+(0.01*aTridem(j)+0.10)^2)...
        sqrt(tandemAxleSep^2+(0.5*dualWheelSep)^2) sqrt(tandemAxleSep^2+(0.25*dualWheelSep+0.5*0.01*aTridem(j))^2) sqrt(tandemAxleSep^2+(0.01*aTridem(j))^2) tandemAxleSep sqrt(tandemAxleSep^2+(0.01*aTridem(j))^2) sqrt(tandemAxleSep^2+(0.01*aTridem(j)+0.10)^2)];
    auxR5 = [auxR1(1) auxR1(2) auxR1(3) auxR1(4) auxR1(5) auxR1(6)...
        auxR3(7) auxR3(8) auxR3(9) auxR3(10) auxR3(11) auxR3(12)...
        0.5*dualWheelSep 0.75*dualWheelSep-0.5*0.01*aTridem(j) dualWheelSep-0.01*aTridem(j) dualWheelSep dualWheelSep+0.01*aTridem(j) dualWheelSep+0.01*aTridem(j)+0.10];
    auxR6 = [auxR2(1) auxR2(2) auxR2(3) auxR2(4) auxR2(5) auxR2(6)...
        auxR4(7) auxR4(8) auxR4(9) auxR4(10) auxR4(11) auxR4(12)...
        0.5*dualWheelSep 0.25*dualWheelSep+0.5*0.01*aTridem(j) 0.01*aTridem(j) 0.01 0.01*aTridem(j) 0.01*aTridem(j)+0.10];
    auxAlpha1 = [atan(2*tandemAxleSep/dualWheelSep) atan(tandemAxleSep/(0.75*dualWheelSep-0.5*0.01*aTridem(j))) atan(tandemAxleSep/(dualWheelSep-0.01*aTridem(j))) atan(tandemAxleSep/dualWheelSep) atan(tandemAxleSep/(dualWheelSep+0.01*aTridem(j))) atan(tandemAxleSep/(dualWheelSep+0.01*aTridem(j)+0.10))...
        atan(3*tandemAxleSep/dualWheelSep) atan(1.5*tandemAxleSep/(0.75*dualWheelSep-0.5*0.01*aTridem(j))) atan(1.5*tandemAxleSep/(dualWheelSep-0.01*aTridem(j))) atan(1.5*tandemAxleSep/dualWheelSep) atan(1.5*tandemAxleSep/(dualWheelSep+0.01*aTridem(j))) atan(1.5*tandemAxleSep/(dualWheelSep+0.01*aTridem(j)+0.10))...
        atan(4*tandemAxleSep/dualWheelSep) atan(2*tandemAxleSep/(0.75*dualWheelSep-0.5*0.01*aTridem(j))) atan(2*tandemAxleSep/(dualWheelSep-0.01*aTridem(j))) atan(2*tandemAxleSep/dualWheelSep) atan(2*tandemAxleSep/(dualWheelSep+0.01*aTridem(j))) atan(2*tandemAxleSep/(dualWheelSep+0.01*aTridem(j)+0.10))];
    auxAlpha2 = [pi-atan(2*tandemAxleSep/dualWheelSep) pi-atan(tandemAxleSep/(0.25*dualWheelSep+0.5*0.01*aTridem(j))) pi-atan(tandemAxleSep/(0.01*aTridem(j))) 0.5*pi atan(tandemAxleSep/(0.01*aTridem(j))) atan(tandemAxleSep/(0.01*aTridem(j)+0.10))...
        pi-atan(3*tandemAxleSep/dualWheelSep) pi-atan(1.5*tandemAxleSep/(0.25*dualWheelSep+0.5*0.01*aTridem(j))) pi-atan(1.5*tandemAxleSep/(0.01*aTridem(j))) 0.5*pi atan(1.5*tandemAxleSep/(0.01*aTridem(j))) atan(1.5*tandemAxleSep/(0.01*aTridem(j)+0.10))...
        pi-atan(4*tandemAxleSep/dualWheelSep) pi-atan(2*tandemAxleSep/
(0.25*dualWheelSep+0.5*0.01*aTridem(j))) pi-atan(2*tandemAxleSep/(0.01*aTridem(j)))
0.5*pi atan(2*tandemAxleSep/(0.01*aTridem(j))) atan(2*tandemAxleSep/
(0.01*aTridem(j)+0.10))];
    auxAlpha3 = [0 0 0 0 0 0 ...
        atan(tandemAxleSep/dualWheelSep) atan(tandemAxleSep*0.5/(0.75*dualWheelSep-0.5*0.01*aTridem(j))) atan(tandemAxleSep*0.5/(dualWheelSep-0.01*aTridem(j))) atan(tandemAxleSep*0.5/dualWheelSep) atan(tandemAxleSep*0.5/(dualWheelSep+0.01*aTridem(j))) atan(tandemAxleSep*0.5/
(dualWheelSep+0.01*aTridem(j)+0.10))...
        atan(2*tandemAxleSep/dualWheelSep) atan(tandemAxleSep/(0.75*dualWheelSep-0.5*0.01*aTridem(j))) atan(tandemAxleSep/(dualWheelSep-
0.01*aTridem(j))) atan(tandemAxleSep/dualWheelSep) atan(tandemAxleSep/(dualWheelSep+0.01*aTridem(j))) atan(tandemAxleSep/(dualWheelSep+0.01*aTridem(j)
+0.10))];
    auxAlpha4 = [pi pi pi 0 0 0 ...
        pi-atan(tandemAxleSep/dualWheelSep) pi-atan(tandemAxleSep*0.5/
(0.25*dualWheelSep+0.5*0.01*aTridem(j))) pi-atan(tandemAxleSep*0.5/(0.01*aTridem(j))) 0.5*pi atan(tandemAxleSep*0.5/(0.01*aTridem(j)))
atan(tandemAxleSep*0.5/(aTridem(j)*0.01+0.10))...
        pi-atan(2*tandemAxleSep/dualWheelSep) pi-atan(tandemAxleSep/(0.25*dualWheelSep+0.5*0.01*aTridem(j))) pi-atan(tandemAxleSep/(0.01*aTridem(j)))
0.5*pi atan(tandemAxleSep/(0.01*aTridem(j))) atan(tandemAxleSep/
(aTridem(j)*0.01+0.10))];
    auxAlpha5 = [atan(-2*tandemAxleSep/dualWheelSep) atan(-tandemAxleSep/
(0.75*dualWheelSep-0.5*0.01*aTridem(j))) atan(-tandemAxleSep/(dualWheelSep-
0.01*aTridem(j))) atan(-tandemAxleSep/dualWheelSep) atan(-tandemAxleSep/(dualWheelSep+0.01*aTridem(j))) atan(-tandemAxleSep/(dualWheelSep+0.01*aTridem(j)
+0.10))...
        atan(-tandemAxleSep/dualWheelSep) atan(-tandemAxleSep*0.5/(0.75*dualWheelSep-0.5*0.01*aTridem(j))) atan(-tandemAxleSep*0.5/(dualWheelSep-
0.01*aTridem(j))) atan(-tandemAxleSep*0.5/dualWheelSep) atan(-
tandemAxleSep*0.5/(dualWheelSep+0.01*aTridem(j))) atan(-tandemAxleSep*0.5/(dualWheelSep+0.01*aTridem(j)+0.10))...
        0 0 0 0 0 0];
    auxAlpha6 = [pi+atan(2*tandemAxleSep/dualWheelSep) pi+atan(tandemAxleSep/
(0.25*dualWheelSep+0.5*0.01*aTridem(j))) pi+atan(tandemAxleSep/
(0.01*aTridem(j))) -0.5*pi atan(-tandemAxleSep/(0.01*aTridem(j))) atan(-tandemAxleSep/(0.01*aTridem(j)+0.10))...
        pi+atan(tandemAxleSep/dualWheelSep) pi+atan(tandemAxleSep*0.5/
(0.25*dualWheelSep+0.5*0.01*aTridem(j))) pi+atan(tandemAxleSep*0.5/(0.01*aTridem(j))) -0.5*pi atan(-tandemAxleSep*0.5/(0.01*aTridem(j))) atan(-tandemAxleSep*0.5/(aTridem(j)*0.01+0.10))...
        pi pi pi 0 0 0];
   %for the rotation calculations, extend auxAlpha1 and auxAlpha2 to the size
of auxSigmaX1 and auxSigmaY2 (z x r)
    auxAlpha1 = ones(length(z),1)*auxAlpha1;
    auxAlpha2 = ones(length(z),1)*auxAlpha2;
    auxAlpha3 = ones(length(z),1)*auxAlpha3;
    auxAlpha4 = ones(length(z),1)*auxAlpha4;
    auxAlpha5 = ones(length(z),1)*auxAlpha5;
    auxAlpha6 = ones(length(z),1)*auxAlpha6;

    %get axle Load (tons) -  convert to load by wheel!
    %get load radius - - - - computed with wheelFootprint (called from the
MainCode) - - IT'S GIVEN IN CM and accounts for the fact that the axle load is
equally divided over all the axle's wheels!
    %get E, nu, height for all materials! - watchful for asphalt        %mate-
rials!
     %%UPDATE 2019-02-19:: DON'T COMPUTE ANYTHING (AND KEEP ZEROS) IF THERE'S
0 TRAFFIC IN THIS CATEGORY.
      %%UPDATE 2019-03-19:: CONVERT LOAD TO NEWTON ( ton x 9800), DISTANCES TO
METERS (cm x 0.01), AND E TO PA (psi x 1.000.000 / 145.04)
      %%%BUG DETECTED HERE!. Calling AxlesTandem18 instead of axlesTridem!
      if axlesTridem(k,j)~=0   %"if there's actual traffic of these axles"
            %update V2019-03-31:: pass pressure [qj] instead of total load to
MLE_sigma.
            aj = 0.01*aTridem(j);
            qj = 1/12*9800*axlesTridemWeights(j)/(pi*(aj)^2);
            [auxSigmaZ1,auxSigmaR1,auxSigmaT1] =
MLE_sigma(qj,aj,auxR1,z,0.01*paveDepths(1:end-1),layersE(:,j),layersv(:,j));
            [auxSigmaZ2,auxSigmaR2,auxSigmaT2] =
MLE_sigma(qj,aj,auxR2,z,0.01*paveDepths(1:end-1),layersE(:,j),layersv(:,j));
            [auxSigmaZ3,auxSigmaR3,auxSigmaT3] =
MLE_sigma(qj,aj,auxR3,z,0.01*paveDepths(1:end-1),layersE(:,j),layersv(:,j));
            [auxSigmaZ4,auxSigmaR4,auxSigmaT4] =
MLE_sigma(qj,aj,auxR4,z,0.01*paveDepths(1:end-1),layersE(:,j),layersv(:,j));
            [auxSigmaZ5,auxSigmaR5,auxSigmaT5] =
MLE_sigma(qj,aj,auxR5,z,0.01*paveDepths(1:end-1),layersE(:,j),layersv(:,j));
            [auxSigmaZ6,auxSigmaR6,auxSigmaT6] =
MLE_sigma(qj,aj,auxR6,z,0.01*paveDepths(1:end-1),layersE(:,j),layersv(:,j));
       else
           auxSigmaZ1 = zeros(length(z),nrTr);
           auxSigmaR1 = zeros(length(z),nrTr);
           auxSigmaT1 = zeros(length(z),nrTr);
           auxSigmaZ2 = zeros(length(z),nrTr);
           auxSigmaR2 = zeros(length(z),nrTr);
           auxSigmaT2 = zeros(length(z),nrTr);
           auxSigmaZ3 = zeros(length(z),nrTr);
           auxSigmaR3 = zeros(length(z),nrTr);
           auxSigmaT3 = zeros(length(z),nrTr);
           auxSigmaZ4 = zeros(length(z),nrTr);
           auxSigmaR4 = zeros(length(z),nrTr);
           auxSigmaT4 = zeros(length(z),nrTr);
           auxSigmaZ5 = zeros(length(z),nrTr);
           auxSigmaR5 = zeros(length(z),nrTr);
           auxSigmaT5 = zeros(length(z),nrTr);
           auxSigmaZ6 = zeros(length(z),nrTr);
           auxSigmaR6 = zeros(length(z),nrTr);
           auxSigmaT6 = zeros(length(z),nrTr);

       end
    sigmaZtridem(:,:,j,k) = auxSigmaZ1 + auxSigmaZ2 + auxSigmaZ3 + auxSigmaZ4
+ auxSigmaZ5 + auxSigmaZ6;   %these sums here are the superposition of the
sigmas of both wheels at each rSingle10 position

     %rotate the sigmaR and sigmaT to cartesian coordinates, sum them together
    auxSigmaX1 = auxSigmaR1.*(cos(auxAlpha1).^2) + auxSigmaT1.*(sin(auxAlpha1).^2);
    auxSigmaX2 = auxSigmaR2.*(cos(auxAlpha2).^2) + auxSigmaT2.*(sin(auxAlpha2).^2);
    auxSigmaX3 = auxSigmaR3.*(cos(auxAlpha3).^2) + auxSigmaT3.*(sin(auxAlpha3).^2);
    auxSigmaX4 = auxSigmaR4.*(cos(auxAlpha4).^2) + auxSigmaT4.*(sin(auxAlpha4).^2);
    auxSigmaX5 = auxSigmaR5.*(cos(auxAlpha5).^2) + auxSigmaT5.*(sin(auxAlpha5).^2);
    auxSigmaX6 = auxSigmaR6.*(cos(auxAlpha6).^2) + auxSigmaT6.*(sin(auxAlpha6).^2);
    auxSigmaY1 = auxSigmaR1.*(sin(auxAlpha1).^2) + auxSigmaT1.*(cos(auxAlpha1).^2);
    auxSigmaY2 = auxSigmaR2.*(sin(auxAlpha2).^2) + auxSigmaT2.*(cos(auxAlpha2).^2);
    auxSigmaY3 = auxSigmaR3.*(sin(auxAlpha3).^2) + auxSigmaT3.*(cos(auxAlpha3).^2);
    auxSigmaY4 = auxSigmaR4.*(sin(auxAlpha4).^2) + auxSigmaT4.*(cos(auxAlpha4).^2);
    auxSigmaY5 = auxSigmaR5.*(sin(auxAlpha5).^2) + auxSigmaT5.*(cos(auxAlpha5).^2);
    auxSigmaY6 = auxSigmaR6.*(sin(auxAlpha6).^2) + auxSigmaT6.*(cos(auxAlpha6).^2);
    auxTauXY1  = (auxSigmaR1 - auxSigmaT1).*sin(auxAlpha1).*cos(auxAlpha1);
    auxTauXY2  = (auxSigmaR2 - auxSigmaT2).*sin(auxAlpha2).*cos(auxAlpha2);
    auxTauXY3  = (auxSigmaR3 - auxSigmaT3).*sin(auxAlpha3).*cos(auxAlpha3);
    auxTauXY4  = (auxSigmaR4 - auxSigmaT4).*sin(auxAlpha4).*cos(auxAlpha4);
    auxTauXY5  = (auxSigmaR5 - auxSigmaT5).*sin(auxAlpha5).*cos(auxAlpha5);
    auxTauXY6  = (auxSigmaR6 - auxSigmaT6).*sin(auxAlpha6).*cos(auxAlpha6);

    %%update V2019-05-14: Add 4th dimension to sigmaXXXX
    sigmaXtridem(:,:,j,k) = auxSigmaX1 + auxSigmaX2 + auxSigmaX3 + auxSigmaX4
+ auxSigmaX5 + auxSigmaX6;
    sigmaYtridem(:,:,j,k) = auxSigmaY1 + auxSigmaY2 + auxSigmaY3 + auxSigmaY4
+ auxSigmaY5 + auxSigmaY6;
    tauXYtridem(:,:,j,k) = auxTauXY1 + auxTauXY2 + auxTauXY3 + auxTauXY4 +
auxTauXY5 + auxTauXY6;

    %create these auxiliar auxE and aux_Poisson variables with the E
%and v for this load level to use with the conversion to strain        %equations
    auxLayersE = getAuxEfromE(0.01*cumsum(paveDepths),z,layersE(:,j));   %
%stretch to the z domain, will need it to superimpose the horizontal stresses
[although not needed in single and 6-ton axles...]
    auxLayersv = getAuxEfromE(0.01*cumsum(paveDepths),z,layersv(:,j));
    %%these two above are in vector format [length(z) by 1]. need to convert
    %%them to z x r matrices. Multiply them for ones(1,length(r))
    auxLayersE = auxLayersE * ones(1,nrTr);
    auxLayersv = auxLayersv * ones(1,nrTr);

    %Convert stresses to strains (refer to Huang 04, chap 3)
    epsXtridem(:,:,j,k) = 1./auxLayersE.*(sigmaXtridem(:,:,j,k) - auxLayersv.*(sigmaYtridem(:,:,j,k) + sigmaZtridem(:,:,j,k)));
    epsYtridem(:,:,j,k) = 1./auxLayersE.*(sigmaYtridem(:,:,j,k) - auxLay-
ersv.*(sigmaXtridem(:,:,j,k) + sigmaZtridem(:,:,j,k)));
    epsZtridem(:,:,j,k) = 1./auxLayersE.*(sigmaZtridem(:,:,j,k) - auxLay-
ersv.*(sigmaXtridem(:,:,j,k) + sigmaYtridem(:,:,j,k)));
    gmXYtridem(:,:,j,k) = 2./auxLayersE.*(1+auxLayersv).*tauXYtridem(:,:,j,k);
    %update v2019-03-19:: corrected formula for epsH, missing a ^2
    epsHtridem(:,:,j,k) = 0.5.*(epsXtridem(:,:,j,k) + epsYtridem(:,:,j,k)) -
sqrt(0.25.*(epsXtridem(:,:,j,k)-epsYtridem(:,:,j,k)).^2 +
gmXYtridem(:,:,j,k).^2);
%
end
%% --END OF SCRIPT
```

Source Code: /materials/sin18

```
function y = sin18(x,z)
%function y = sin18(x,z)
%function to calculate the sin of the current hour in the 18-hour cycle, as
defined for the BELLS equations (see FHWA RD-98-085)
% INPUT: x = hour in decimal time <treated as an angle in radians>
%        z = an auxiliary hour (the BELLS equations use 15.5 and 13.5 in dif-
ferent terms). Refer to the RD-98-085 for details.
%OUTPUT: y = value for sin18(x) <adim.>


%first - manage the hours following the RD-98-085 rule (separate cases for z =
15.5 and z = 13.5)

switch z
    case 15.5
        if x >=0 && x<=5
            x = x + 24;  for hours between 0:00 and 5:00, sum 24 hours
        elseif x>5 && x < 11
            x = 11;    %for hours between 5:00 and 11:00, replace for 11:00
        else
            %do nothing
        end
    case 13.5
        if x >=0 && x<=3
            x = x + 24;      %for hours between 0:00 and 3:00, sum 24 hours
        elseif x>3 && x < 9
            x = 9;     %for hours between 3:00 and 9:00, replace for 9:00
        else
            %do nothing
        end
    otherwise
    %do nothing
end

%second, calculate the sin of the hour x
angle = 2 * pi * (x-z)/18;
y = sin(angle);

end
```

Source Code: /materials/vaporPressure

```
function pvap = vaporPressure(temp,hum)
  %function pvap = vaporPressure(temp,hum)
%This auxiliary function will provide the actual vapor pressure for a site at
temperature "temp" [in C] and a humidity value "hum" [percentage]
%Output pvap is in mmHg
%
%Load vapor pressure table - there's a different .mat file for whether code is
running in Matlab (binary .mat) or Octave (text-based .mat)


  isThisOctave = exist('OCTAVE_VERSION') ~=0;  %if the 'OCTAVE VERSION' vari-
able exists, the statement is diff. from 0, the value of isThisOctave is 1
  %(in Matlab, isThisOctave shall equal to 0)
%Ref: https://stackoverflow.com/questions/2246579/how-do-i-detect-if-im-run-
ning-matlab-or-octave

  if isThisOctave
      load('./dataFiles/pvapTable_OCT.mat');   %Text version of the temp |
pvap series  (stored in pvapTable variable)
  else
      load('./dataFiles/pvapTable_MAT.mat');   %Binary version of the temp |
pvap series
  end

  %first, obtain maximum vapor pressure for given temperature Temp
  pvapMaxima = interp1(pvapTable(:,1),pvapTable(:,2),temp);

  %Compute the actual vapor pressure
  pvap = hum/100 .* pvapMaxima;
end
```

Source Code: /materials/calculateMR

```
function MRactual = calculateMR(materialID,moistureContent,MROpt,OptSaturation,OptMoisture)
%function MRactual = calculateMR(materialID,moistureContent,MRSat,OptSaturation,OptMoisture)
%
%% -- GRANULAR LAYERS AND SUBGRADE UNSAT MR MODULE
%
%INPUT:
%materialID: column vector stating for each granular layer its ID (to check if it's fine or coarse)
%MROpt    : column vector stating for each granular layer and the sub-grade what is its MR in optimum compaction conditions [PSI by default].
%moistureContent: vector (row format) containing the percent moisture
%content (by volume!!) of each granular layer and the MR
%
%OUTPUT: MRactual: row vector (to be pasted in the Main Code) with the actual resilient Moduli for each layer, matching the actual moisture content
%
%METHODOLOGY: USE Witczak's formula (in 1-37A //app. DD; also mentioned by Bilodeau & Dore, 2011)
%
%%ASSUMPTIONS:
%a) separate parameter values for fine [M and C families in SUCS] and coarse [S and G families] soils and base materials
%b) use default calibration values only (refer to 1-37A, appendix DD-1)
%V 0.1 Spring semester 2019-01-22.
%V 0.0 Summer solstice 2018-12-22.

%% - code begins

%1 - set the equation's parameters for each material family (separately if these are fine [C, M] or coarse [S, G]
p  = zeros(length(MROpt),1);
q  = zeros(length(MROpt),1);
b  = zeros(length(MROpt),1);
ks = zeros(length(MROpt),1);

for i = 1:length(materialID)
    if materialID(i) > 150
        %coarse materials  (ID > 150)
        p(i) = -0.3123;
        q(i) = 0.3;
        ks(i) = 6.8157;
    else
        %fine materials (ID < 150)
        p(i) = -0.5934;
        q(i) = 0.4;
        ks(i) = 6.1324;
    end
    b(i) = log(-p(i)/q(i));   %%NATURAL (e-based) logarithm. by definition
end
```

```
%2 - get saturation values for the materials' current mositure rate
saturation = moistureContent.*(OptSaturation./OptMoisture);

%3 - calculate the material's resilient modulus with the
aux = 1+ exp(b + ks.*(saturation-OptSaturation));
logMRRatio = p + (q-p)./aux;

MRRatio = 10.^logMRRatio;
MRactual = MROpt.*MRRatio;

end
```

Source Code: /materials/HMALayersTemperature

```
function asphLyrTemp = HMALayersTemperature(timestamp,airTemp,humidity,windSpd,
sunRad,ACPlacementTemp,ACPaveDepth,asphAbsorvipity,asphEmissivity,asphThermalConductivity,
verbose)
%function asphLyrTemp = HMALayersTemperature(timestamp,airTemp,humidity,windSpd,
sunRad,ACPlacementTemp,ACPaveDepth,asphAbsorvipity,asphEmissivity,asphThermalConductivity,
verbose)
%% -- PAVEMENT TEMPERATURE PROFILE MODULE FOR FLEXIBLE PAVEMENTS
%Calculate instant temperature profile throughout structure. Use equations
from the 1993 paper and LTPP Bells Equations to calculate temperature profile
through the HMA layers.
%Update 2018-06-20: Converted to function format from previous version of code
(doesn't need to run along design process, the HMA layers temperature can be
predicted before running the simulation)
%
%INPUT: timestamp: vector with time reference (1 unit is 1 day)
      % airTemp:  air temperature (deg. Celsius)
      % humidity: relative air humidity (perc.)
      % windSpd:  wind speed (m/sec)
      % sunRad:   net solar radiation (Watt/m2)
      % ACPlacementtemp: Temperature at which the last HMA layer has been
placed (deg. C)
      % ACPaveDepth: vector containing the thickness of each HMA layer (cm)
      % asphAbsorvipity: thermal absorvipity of the HMA layers (level 3 - single
default value)
      % asphEmissivity:  thermal emissivity of the HMA layers  (level 3 - single
default value)
      % asphThermalConductivity: thermal conductivity of the HMA layers (level
3 - single default value)
      %verbose: boolean variable: 1 = report each calculation step on screen /
0 = run silently
%
%OUTPUT: asphLyrTemp = Matrix containing column of [HMA surf. temperature, avg
temp in lyr1 ..... avg temp in lyr n]

%% 1 - Initialize variables for the loop
%First let's set up the terms of the equation to solve (after the 1993 paper)
if verbose
   disp('    HMA temperature calculation:: initializing');
end
n = length(timestamp);
sigmaStephBoltz = 5.68E-8;  %Stephan-Boltzmann constant [5.68 x 10-8 W/m2K4]
asphLyrTemp = zeros(n,length(ACPaveDepth)+1);             %Initialize asphLyr
Temp [matrix of size timestamp by number of HMA Layers]
depthTemp = 0.01*ACPaveDepth(1);                          %depth of the uppermost
HMA layer (select value from the imput matrix) -- CONVERT FROM cm TO m
tempPrevDay = temperaturePreviousDay(timestamp,airTemp); %aux. function to
give me the previous day temperature (needed to calculate AC temperature at
time k)
currentHour = datevec(timestamp);                         %extract the hours of
the timestamp vector
currentHour = currentHour(:,4);
```

```
zTemp = [0;cumsum(ACPaveDepth)];                          %column vector with
depth -accumulated thickness- of each AC boundary and surface [in centime-
ters].

%these vectors have values used in the equations within the loop. Since they
don't change as the loop progresses I can pull them out.

qs = asphAbsorvipity.*sunRad;  %short-wave incoming radiation at any time
pVapor = vaporPressure(airTemp,humidity);  %calculate actual vapor pressure
[mmHg] from imput data
emissivityAir = 0.77 -0.28 .* 10.^(-0.074*pVapor);
qa = emissivityAir.*sigmaStephBoltz.*((airTemp+273.15).^4);   %long-wave radi-
ation from atmosphere

%% - 2 Launch the loop: calculate temperature throughout the structure at time
k
for k = 1:n
    if verbose && round(k/1000) == (k/1000)
           fprintf(' \t HMA temperature completed %g percent \n',k/n*100);
    end
    %1.1 - Solve Ts with a Newton-Raphson method. Launch from Ts = Td(t-1)
    %Tolerance: deviate 3% from 0
    tol = 0.03;
    isTsSolved = 0;
    if k == 1
%          tempN = 0.5*( ACPlacementTemp +airTemp(k));     %temp in the surface
at try N
%          tempD = ACPlacementTemp;    temp at a depth D in the pavement (D is
depthTemp = depth of top-most layer, see above in #1)
          tempN = 1.5*airTemp(1);
          tempD = 0.7*airTemp(1);
    else
        tempN = 0.5*( asphLyrTemp(k-1,1) + airTemp(k));
        tempD = asphLyrTemp(k-1,2);   %default value for tempD = temp of pre-
vious hour inside layer 1 (avg estimated value for whole layer)
    end
    while isTsSolved ==0
        hsTn = 698.24*(0.00144*(mean([tempN+273.15,airTemp(k)
+273.15]))^0.3*windSpd(k)^0.7 + 0.00097*abs(tempN-airTemp(k))^0.3);
        %%Tweaked over the original Equation (Vehrencamp 1953; appears in
NCHRP 1-37A as eqn. 2.3.27)  <Use of abs of difference in the non-linear term
to prevent it from going to complex world...>
        fTn = qs(k) + qa(k) - asphEmissivity*sigmaStephBoltz*(tempN+273.15)^4
- ...
            (-1)*asphThermalConductivity*(1/depthTemp)*(tempD - tempN) + (-
1)*hsTn*(tempN-airTemp(k));
        fPrimaTn = -4*asphEmissivity*sigmaStephBoltz*(tempN+273.15)^3 - asph-
ThermalConductivity*(1/depthTemp) + ...
            (-1)*hsTn*1 + (-1)*(tempN-
airTemp(k))*698.24*(0.00144*0.3*(mean([tempN+273.15,airTemp(k)+273.15]))^-
0.3*0.5*windSpd(k)^0.7 + 0.00097*0.3*abs(tempN-airTemp(k))^(-0.3));
        %Now do the check, if abs(fTn) <0.03, tempN is my root; if not, re-
place value
```

```
        if abs(fTn)<tol
            isTsSolved =1;
        else
            tempN = tempN - fTn/fPrimaTn;   %update value of tempN
        end
    end
    asphLyrTemp(k,1) = tempN;

    %% 3 - Solve downward to the end of the asphalt layers. Use the LTPP BELLS
EQUATION (ref: HFWA RD 98-085). %I will calculate temperature on the boundary
of each HMa layer and will average values. %Store them in asphLyrTemp(k,
2:end)

    ACTempAux = zeros(length(zTemp),1);   %in this auxiliary variable I will
store the temperatures at the surface and at the bottom of each layer at time-
stamp k
    ACTempAux(1) = tempN;
    for j = 2:length(zTemp)
      %Now I apply the Bells equation throughout the AC layers. ZTEMP MUST BE
CONVERTED FROM CM TO MM TO BE PROPERLY USED IN THE BELLS EQUATION!!!
      ACTempAux(j) = 2.78+0.912*airTemp(k) + (log10(10*zTemp(j))-1.25)*(-
0.428*airTemp(k) + 0.553*tempPrevDay(k) + 2.63*sin18(currentHour(k),15.5)) ...
      +( 0.027*airTemp(k)*sin18(currentHour(k),13.5));
    end

    %%2.2 - Average the boundary values to get the asphLyrTemp row
    ACAvgTempAux = zeros(length(zTemp)-1,1);
    for j = 1:length(ACAvgTempAux)
      ACAvgTempAux(j) = mean([ACTempAux(j),ACTempAux(j+1)]);  %solve mean
temepration for layers 1:down
    end
    %2.3 - replace aux in the asphLyrTemp matrix
    asphLyrTemp(k,2:end) = ACAvgTempAux;

end  %END FOR-loop for AC surface temp

end %% -- END OF MODULE.
```

Source Code: /materials/HMALoadFrequency

```
function [timeLoad,wLoad] = HMALoadFrequency(HMALyrsDepth,axleSpeed,radiusAxle,HMAModulus,SubGradeMR)
%function [timeLoad,wLoad] = HMALoadFrequency(HMALyrsDepth,axleSpeed,aAxle)
%Auxiliary function that calculates the time and (angular) frequency of load for the passage of a certain type of axle. Time of load is calculated from the 45deg/midlayer methodology used by the MEPDG (reported by AlQadi et al. 2008....). For each layer  (assuming that the method is valid for both single axles and tandem-tridem axles).
%Assuming frequency of load is angular (see Huang 04, chap 7), reported by AlQadi et al. (2008) as being more appropriate than the linear frequency
%
%V1.0 of function - 2018-06-26
%V2.0  - 2019-02-17
%Changelog: Bug corrected in leff formula, it was not considering the Odemark's effective depth approach (conver)


%INPUT: HMALyrsDepth:: Vector containing the depths of each HMA layer [cm]
%       axleSpeed      <Single Value>: the speed at which the axle passes [km/h]
%       radiusAxle     <Vector>      : equivalent circular footprint radius for different weights of the referred axle type [cm]
%       HMAModulus     <vector>      : vector of HMA elastic moduli  (I need at least a preliminary value (set at ambient temperature shortly after placement) for the effective length computations [PSI]
%       SubGradeMR     <Single value>: resilient modulus of the subgrade (for calculation purposes, take only the saturated state - acknowleedging it's a major simplification) [PSI]%
%%Note: this code won't treat differently the single/tandem/tridem axles.
%
%OUTPUT: timeLoad: matrix containing the time of load application (seconds) for each radiusAxle value on each HMA layer
         %size:  [nr. of hma layers x radiusAxle size]
         %wLoad: vector containing the angular frequency for each timeload value.    %w = 1/(2*pi*timeLoad)

%Initialize variables

nHMAL = length(HMALyrsDepth);
nAxles = length(radiusAxle);

%1 calculate timeLoad for each layer
%First: calculate "effective DEPTH" <Zeff> for each layer. Using the Odemark methodology.(Refer to MEPDG APP CC3 for equations)\
z    = zeros(nHMAL,1);
zeff = zeros(nHMAL,1);
%moduliRatio (to compute zeff)
moduliRatio = zeros(nHMAL,1);
moduliRatio = (HMAModulus./SubGradeMR).^(1/3);  %%refer to MEPDG, appendix CC, equation 5
```

```
%compute zeff for mid-layer positions (MEPDG, appendix CC3, eqn 7)    [zeff is calculated in cm here!!!]
zeff(1) = 0.5*HMALyrsDepth(1)*moduliRatio(1);
z(1)    = 0.5*HMALyrsDepth(1);
for i = 2:nHMAL
  %solve each value of zeff
  z(i)    = 0.5*HMALyrsDepth(i) + sum(HMALyrsDepth(1:i-1));
  zeff(i) = 0.5*HMALyrsDepth(i)*moduliRatio(i) + sum(HMALyrsDepth(1:i-1).*moduliRatio(1,i-1));
end
%calculate "effective length" <leff> for each layer. (Refer to MPEDG APPENDIX CC3 for detailed equation).
%MAJOR SIMPLIFICATION HERE: ASSUMING A SINGLE EQUATION FOR ALL AXLE TYPES (theoretical dev. exists for treating each axle type spearately but the outcome wouldn't change significatively - ref. MEPDG App. CC3)

leff = zeros(nHMAL,nAxles);
%solve one column at a time
for j = 1:nAxles
    leff(:,j) = 2.*(radiusAxle(j) + zeff);    %%equation 9 (single-axle case) in Appendix CC3, valid for multi-axles too.
end

%Then calculate the time of passage (leff and timeLoad are the same size). Got to convert leff to inches and axleSpeed to mph to properly use the formula.

% timeLoad = zeros(nHMAL,nAxles);
% wLoad = zeros(nHMAL,nAxles);
timeLoad = (leff./2.54)./(17.6*axleSpeed/1.6);

%2 finally calculate wLoad

wLoad = 1./(timeLoad);
%add the correction factors suggested by AlQadi et al (2008) - go by column in wLoad!!
CorrFactor = 0.03.*z + 0.2333;   %%Interstate-lvl equation by the authors for avg. correction factors. Conveniently tweaked to receive z in cm!
for j = 1:nAxles
    wLoad(:,j) = CorrFactor.*wLoad(:,j);
end

end  %endfunction
```

Source Code: /materials/HMAModulus

```
function E = 
HMAModulus(HMAtemperature,HMAparameters,HMAFrequency,inputLevel,verbose)
%function E = 
HMAModulus(HMAtemperature,HMAparameters,HMAFrequency,inputLevel,verbose)
%
%Auxiliary function that will calculate the HMA mix dynamic modulus off mix 
%or binder properties (according to level of input).
%INTERIM VERSION - It will calculate the Dynamic modulus of the HMA mixes us-
ing the Asphalt Institute (1979) formulas (refer to Huang '04, chap 7). Same 
methodology for the three levels of input.
%INPUT:  HMAtemperature: vector containing the predicted temperature series of 
each HMA layer during the design period [in C]
 %       HMAparameters:  matrix containing, for each HMA layer, P200 <percent-
age of fines passing #200 mesh>, Va <volume of air voids>, Vb <effective bitu-
men volume content %>, P25c <penetration at 25 deg. C>
         %HMAFrequency
         %inputLevel
         %verbose: boolean variable (1 = report on screen progress of calcula-
tions / 0 = run silently)
%OUTPUT: E = 3D array matrix (each layer size number of HMA layers x length of 
load applications) with the E* value for each temperature (timestamp) value. 
[PSI]
%(1 PSI = 6.9 kPa)

%Methodology would vary across levels of input <interim version will only have 
lvl2 case (calculation of E from values pulled from local material library>
switch inputLevel
    case 1  %% Level of input 1: Master curve to be defined from lab testing 
(Ref: NCHRP 1-28A Vol II report -Wictzak, 2003))
        %interim version -  Level of input 1: same case as input level 2 - 
call recursively
        E = HMAModulus(HMAtemperature,HMAparameters,HMAFrequency,2,verbose);
    case 2  %% Level of input 2: Master curve to be defined from material 
properties (NCHRP 1-37A #2 & App. CC)
        %WARNING: HMATemperature matrix has the temperature of each layer PLUS 
the temperature at the surface of the pavement. Peel off 1st column before 
starting the routine!!!!!
        HMAtemperature = HMAtemperature(:,2:end);

        [nTemp,nHMA] = size(HMAtemperature);  %should be timestamp x number of 
HMA layers  (length will give me the largest of both (timestamp))
        [nFreqs] = length(HMAFrequency);      %size of the load frequency ma-
trix is [number of layers / frequencies]
        E = -1*ones(nHMA,nFreqs,nTemp);        %initalize E, each stack is the 
E* for all the asphalt layers on different timestamps
        P200 = HMAparameters(:,1);
        Va = HMAparameters(:,2);
        Vb = HMAparameters(:,3);
        P25c = HMAparameters(:,4);
        for k = 1:nTemp %open the loop to go over all the temperature values (along all timestamps)
            if verbose
                 if round(k/1000) == (k/1000)
                     fprintf(' \t HMA E* calculation completed %g percent \n',k/nTemp*100);
                 end
            end
            %solve E* with the Asphalt Institute (1979) equations for each temperature(timestamp) value

            lambda = 29508.2*P25c.^-2.1929.*ones(size(HMAFrequency)); %matrix the same size as HMAFrequency
            tempK = HMAtemperature(k,:).*1.8 + 32;      %before running convert HMA temperature from deg. C to F!!!!
            tempK = tempK' .*ones(size(HMAFrequency)); %matrix the same size as HMAFrequency
            beta5 = 1.3+0.49825*log10(HMAFrequency);    %matrix the same size as HMAFrequency
            beta4 = 0.483*Vb.*ones(size(HMAFrequency));%matrix the same size as HMAFrequency
            beta3 = 0.533833 + 0.028829*((P200.*ones(size(HMAFrequency))).*HMAFrequency.^-0.1703) - 0.03476.*Va.*ones(size(HMAFrequency)) + 0.070377*lambda + 0.931757.*HMAFrequency.^-0.02774;
            beta2 = sqrt(beta4).*(tempK).^beta5;
            beta1 = beta3 + 0.000005*beta2-0.00189*beta2.*HMAFrequency.^-1.1;
            E(:,:,k) = 100000*(10.^beta1);  %Modulus in PSI!
        end
    case 3  %% Level of input 3: same case as input level 2 - call recursively
       E = HMAModulus(HMAtemperature,HMAparameters,HMAFrequency,2,verbose);
    otherwise
        error('HMA Master Curve Calculator error:: Level of input not supported')  %this error should never pop up (
end

end %endfunction
```

Source Code: /materials/HMAPoisson

```
function nu = HMAPoisson(Edyn,inputLevel,verbose)
%function nu = HMAPoisson(Edyn,HMAparameters,inputLevel)
%
%Auxiliary function that will calculate the HMA mix poisson ratio according to
level of input.
%INTERIM VERSION - Using only Level 2 equation from the MEPDG guide with de-
fault fitting values (P2C2, eq. 2.2.24).
%INPUT:  EDyn: 3-D arrange containing the dynamic modulus  E* of all
%the HMA layers under different load freqs. over time.
%OUTPUT: nu = 3D array matrix (each layer size timestamp x number of HMA lay-
ers) with the Poisson coefficient for each HMA layer at each simulated temp
and load condition, as calculated in Edyn.


switch inputLevel
    case 1  %% Level of input 1: Master curve to be defined from lab testing
(Ref: NCHRP 1-28A Vol II report -Wictzak, 2003))
        %interim version -  Level of input 1: same case as input level 2 -
call recursively
        nu = HMAPoisson(Edyn,2,verbose);
    case 2  %% Level of input 2: Master curve to be defined from material
properties (NCHRP 1-37A #2, eqn 2.2.24 )
        nu = zeros(size(Edyn));
        nu = 0.15 + 0.35./(1+exp(-1.63+3.84E-6.*Edyn));
    case 3  %% Level of input 3: same case as input level 2 - call recursively
       nu = HMAPoisson(Edyn,2,verbose);
    otherwise
        error('HMA Poisson coefficient calculator error:: Level of input not
supported')  %this error should never pop up (
end

end %endfunction
```

Source Code: /materials/kUnsatSWCC

```
function SWCC_matrix = kUnsatSWCC(SWCCparameters,verbose)
%function SWCC_matrix = kUnsatSWCC(SWCCparameters,verbose)
%auxiliary function to the infiltration module, it will calculate the hy-
draulic conductivity of the granular base layers using the SWCC (soil water
char. curve --defined by Xi et al (1994) and adopted by the MEPDG--) and inte-
grating it the way explained in Fredlund et al (1994).
%Inputs:: SWCCparameters, which contain:
%       - Saturated vol. water content
%       - Saturated hydr. conductivity of the granular layers (1st col)
%       -  SWCC parameters (refer to the MEPDG)  (cols 2::2nd)
%Each row of SWCCparameters is one layer%
%
%Outputs:: a 3-D matrix (each "level" is for a single granular layer), which
will contain 3 columns:
%       - the suction domain [0:1000 PSI converted to MPA]
%       - the corresponding vol. moisture content(calculated with the SWCC
curve),
%       - the hydraulic conductivity for that unsaturated conditions (follow-
ing Fredlund et al., 1994)
%
% V01- 2019-01-24
% V0 - 2018-10-18

%NOTE:: 1 MPA = 145.038PSI
%    :: water unit weight = 9800 N/m3 (1000kg/m3)
%    :: kSat must be received in metric units [m/sec]

waterWeight = 9800;

 if verbose
%    disp('Materials Preprocessing:: Calculating hydraulic conductivity func-
tion for unsaturated conditions'). No need to add this msg. line, it aklready
splashes from the main code
 end
[numLayers,~] = size(SWCCparameters);
%%note: parameters' names according to MEPDG (Part 2 vol3)
humSat = SWCCparameters(:,1);  %saturated volumetric water content [%]
kSat   = SWCCparameters(:,2);  %saturated-state hydraulic conductivity [m/sec]
SWCCa  = SWCCparameters(:,3);  %parameter "a" for the SWCC curve [PSI]
SWCCb  = SWCCparameters(:,4);  %parameter "b" for the SWCC curve [adim]
SWCCc  = SWCCparameters(:,5);  %parameter "c" for the SWCC curve [adim]
SWCChr = SWCCparameters(:,6);  %parameter "hr" for the SWCC curve [PSI]

hPSI = 0:10:1000; %%Domain for suction pressure MPA (according to Fredlund et
al. 1994)
hPSI = hPSI';     %%im reporting hPSI as a column!
m = length(hPSI);
SWCC_matrix = zeros(m,3,numLayers); %%for each level (layer), it must contain
hPSI
kUnsat = ones(m:1);%*kSat (remove the product);
```

```
%calculate the SWCC for each layer
%calculate the C(h) parameter of the SWCC
for k = 1:numLayers
    Ch_num = log(1+145.038*hPSI./SWCChr(k));   %log: NATURAL LOGARITHM. the 145.038 is unit conversion factor from MPA to PSI
    Ch_den = log(1+1000*145.038/SWCChr(k));
    Ch = 1-Ch_num/Ch_den;
    %calculate the SWCC (vol. water content (suction))
    SWCC = (log(exp(1)+(hPSI/SWCCa(k)).^SWCCb(k))).^-SWCCc(k);
    SWCC = Ch.*humSat(k).*SWCC;  %volumetric moisture content for a given suction pressure
    SWCC_matrix(:,1,k) = hPSI;     %the name may sound counter-intuitive, but I am outputting in MPA!
    SWCC_matrix(:,2,k) = SWCC;
    %now calculate the unSat hydraulic conductivity - use a for loop, cause the equation looks somehow convoluted.
    %suctionHead2 = (hPSI.^2)/145.038/waterWeight; %get the suction Head [m] squared, but with some minor unit inconsistency (no worries, in future calculations all the unit errors cancel themselves out).
    %%BUG DETECTED 2019-01-23 - unit errors in suctionHead2: hPSI is in MPA (despite the name). no need to divide by 145.038. Actually, Fred&Xi '94 (eqn 7) don't convert hPSI to head. Removing conversion!
    %%update 2019-01-24:: corrected iteration below, wrong term in initialization of auxNum!
    suctionHead2 = hPSI.^2;
    for i = m:-1:1
        %first term of the sumation [eqn 7 in F&Xi '94 (j = m)
        auxNum=1/suctionHead2(m)*(2*(m-i)+1);
        auxDen=1/suctionHead2(m)*(2*m-1);
        if i<m  %i don't want this for loop to run in the last iteration (no terms to add to the auxNum and auxDen values)
            for j = m-1:-1:i+1
                auxNum = auxNum + (1/suctionHead2(j)*(1+2*j-2*i));
                auxDen = auxDen + (1/suctionHead2(j)*(-1+2*j));
            end
        end
       kUnsat(i) = kSat(k)*auxNum/auxDen;
    end
    SWCC_matrix(:,3,k) = kUnsat;
    %and then reset kUnsat
    kUnsat = ones(m,1);
end

end  %endfunction
```

Source Code: /materials/materialsParametersLVL2.m

```
%%%%      - PRODUCT-ONE -         %%%%
%%  - M.E. Pavement Design tool - %%%%
%%% PAVEMENT PROPERTIES IMPORTER  %%%%
%
% Version 0.3 - 2019-03-05  Add import of P200, PI, and P02 for subgrade and
gran. materials [needed for IRI calculations]
% Version 0.2 - 2019-02-11  Update to also use xlsread in Octave (v4.2 and
above)
% Version 0.1 - 2018-10-09  UPDATE: import for granular base materials
% Version 0.0 - 2018-06-26
%materials Parameters for Level 2 - Uy Literature default values
%Note: This script uses the variables in the MainCode environment, and so will
drop variables there

%% - Retrieve materials' props. library
if runVerbose
    disp('importing materials" properties from library')
end
if isThisOctave
    HMALibrary = xlsread(dataImport,'materialsCatalog','b10:l34');
    granularLibrary = xlsread(dataImport,'materialsCatalog','b39:aa83');
    % CTBLibrary
    % subGradeLibrary
else
    HMALibrary = xlsread(dataImport,'materialsCatalog','b10:l34');
    granularLibrary = xlsread(dataImport,'materialsCatalog','b39:aa83');
    % CTBLibrary
    % subGradeLibrary
end


%% -ASPHALT CONCRETE MATERIALS' PROPERTIES

asphAbsorvipity = 0.93;             %Source 1993 paper
asphEmissivity = 0.93;              %Source 1993 paper
asphThermalConductivity = 1.0811;    %Source NCHRP 1-37 A pg 2.3.13 (average
value, converted to Watt/[meter Celsius]

ACPlacementTemp = 160;               %Temperature at which the surface Hot/
Warm Mix Asphalt layer is placed [deg. C] (used in initial temperature calcu-
lations)

asphCrackInfiltrationRate = 0.10;

%%parameters to calculate dynamic modulus E*
%prepare a matrix called "HMAparameters" containing the following
%Percentage passing #200 mesh sieve / Volume of air voids / Volume of
bitumen / Penetration at 25C

%Retrieve imported library of materials from "dataimport" source and then pick
for the project's HMA layers (use HMAID variable, as defined in the Main code)
```

```
HMA_ID = paveID(find(paveID<100));  %all layer IDs lower than 100 are HMA layers.

%(column %vectors for each HMA layer, pick columns 5, 7, 8, and 9 from the HMALibrary)
%Use ismember instead of find to locate elements with multiple criteria. Trick from: https://www.mathworks.com/matlabcentral/answers/32781-find-multiple-elements-in-an-array
%OVERRIDE: Trick above doesn't work when the multiple criteria has 2+ entries with the same value. Using a slower for loop instead.
HMARowsToPick = zeros(length(HMA_ID),1);
for k = 1:length(HMA_ID)
    HMARowsToPick(k) = find(HMALibrary(:,1) == HMA_ID(k));
end
HMAP200 = HMALibrary(HMARowsToPick,7);
HMAVa   = HMALibrary(HMARowsToPick,9);
HMAVb   = HMALibrary(HMARowsToPick,10);
HMAP25c = HMALibrary(HMARowsToPick,11);
HMAparameters = [HMAP200 HMAVa HMAVb HMAP25c];          %Parameters for dynamic analysis

clear HMARowsToPick HMAP200 HMAVa HMAVb HMAP25c HMA_ID
clear HMALibrary

%% -GRANULAR MATERIALS AND SOILS' PROPERTIES
% Make up a table with the variables i need (or read it from an Excel file)
% - -Hydraulic conductivity
% - - resilient modulus (default parameters for the MRvs stress curve)

gran_ID = paveID(find(paveID>100 & paveID<200));
granRowsToPick = zeros(length(gran_ID),1);
for k = 1:length(gran_ID)
    granRowsToPick(k) = find(granularLibrary(:,1) == gran_ID(k));
end
granP200  = granularLibrary(granRowsToPick,5);
granPI    = granularLibrary(granRowsToPick,8);
granMR    = granularLibrary(granRowsToPick,10);
granPoiss = granularLibrary(granRowsToPick,11);
granDens  = granularLibrary(granRowsToPick,14);
granPerc  = granularLibrary(granRowsToPick,16);
granHumidity  = granularLibrary(granRowsToPick,18:20);
granKHidr = granularLibrary(granRowsToPick,22);
granSWCC  = granularLibrary(granRowsToPick,23:26);  %contains hydraulic conductivity [MKS] and SWCC curve parameters [but with suction in PSI]
granSWCCParameters = [granHumidity(:,end) granKHidr granSWCC];
granularParameters = [granMR granPoiss granDens granPerc];  %Parameters for dynamic analysis

clear granRowsToPick
clear granularLibrary
```

Source Code: /materials/temperaturePreviousDay

```
function t = temperaturePreviousDay(hourlyTimestamp,tempRecord)
  %function t = temperaturePreviousDay(i,timeStamp,tempRecord)
  %this auxiliary function will calculate the average temperature for the day
before hourly timestamp(s) i, according to what reads in tempRecord.
  %
  %timeStamp is a hourly timeStamp vector, whose length matches tempRecord.
  %%UPDATE 2018-06-20: convert this function to a Matrix-calculation (relieve
itself from running only within a for loop)
  %%UPDATE 2019-03-15: changed the dayK searcher for a "missed 29th of febru-
ary scenario": on 366-days years made with 365-days registries. I'll have a
blank spot for the 29th feb.; the dayK search will come up null. Force it to
go to the 28th. instead (search for hourlyTimestamp(k)-2 instead)
  %%UPDATE 2018-06-20: convert this function to a Matrix-calculation (relieve
itself from running only within a for loop)

hourlyTimestamp = floor(hourlyTimestamp);  %round down every timestamp to
timestamp value matching year/mo/day @ 0:00hs
  [a,b] = size(hourlyTimestamp);
  t = zeros(a,b);   %initialize output vector

  %% 1 - solve t for the first day (positions where day(timeStamp(:)) =
day(timeStamp(1))
  %Use t = temperature of day i

  locateDay1 = find(hourlyTimestamp(:) == hourlyTimestamp(1));  %locateDay1
will have all the positions in timeSTamp that match day 1
  t(locateDay1,:) = mean(tempRecord(locateDay1));

  %1.2 - solve for day 2: fill 24 positions with avg. from day 1 as well
  locateDay2 = find(hourlyTimestamp(:) == hourlyTimestamp(1)+1); %LocateDay2
will have the positions on timeMat corresponding to the day after day 1
  t(locateDay2,:) = mean(tempRecord(locateDay1));   %fill up with day 1 avg.

   %% 2 - solve t for day 3 onward: t(day i) = tempRecord (day i-1)
  %start of day 3 is in position locateDay2(end)
  for k = locateDay2(end)+1:a
      dayK = find((hourlyTimestamp) == (hourlyTimestamp(k)-1));   %position in
hourlyTimestamp corresponding to day before day k
      if isempty(dayK)
          %update 2019-03-15--- missing 29thFeb problem!, force to check
          %from the 1st of march to the 28th of feb
         dayK =  find((hourlyTimestamp) == (hourlyTimestamp(k)-2));
      end
      t(k) = mean(tempRecord(dayK));     %do the average of temp records for
the day
      if isnan(t(k))
         disp('stop here')
      end
  end

end
```

Source Code: /plottingTools/plotAlligator

```
function plotSuccess = plotAlligator(timestamp,AlligatorCrack,TopDownCrack)
%function plotSuccess = plotRutDepth(timestamp,rutDepth)
%Plotting Tools - Alligator cracking and top-down cracking
%
%This auxiliary script will plot the predicted extent of alligator (bottom-up)
and longitudinal (top-down) cracking over the HMA layers
%
%%V0.1 - St. Patrick's hangover: 2019-03-18
%   Changelog: x-axis labels in date format

%% code begins
%
auu = isnan(AlligatorCrack(:,end));
figure(43)
if auu(end)==0
    plot(datetime(datevec(timestamp)),
real(AlligatorCrack(:,1)),'b','linewidth',2)
    plotSuccess = 1;
    legendstring = {'Cracking asph. lyr. 1'};
    hold on
    [~,b] = size(AlligatorCrack);
    for i = 2:b
        plot(datetime(datevec(timestamp)),
real(AlligatorCrack(:,i)),'color',rand(1,3),'linewidth',2);
        addlegend = sprintf('Cracking asph. lyr. %g',i);
        legendstring = [legendstring;{addlegend}];
    end
    legend(legendstring{:});
else
    plot(datetime(datevec(timestamp)),
zeros(size(timestamp)),'b','linewidth',2)
    plotSuccess = 0;
end
grid
xlabel('date')
xtickformat('dd-MM-yy')
ylabel('alligator cracking [perc. lane area]')
title('Alligator cracking - all asphalt layers')
hold off

figure(44)
if auu(end)==0
    plot(datetime(datevec(timestamp)),
real(TopDownCrack(:,1)),'b','linewidth',2)
    legendstring = {'Cracking asph. lyr. 1'};
    hold on
    [~,b] = size(AlligatorCrack);
    for i = 2:b
        plot(datetime(datevec(timestamp)),
real(TopDownCrack(:,i)),'color',rand(1,3),'linewidth',2);
        addlegend = sprintf('Cracking asph. lyr. %g',i);
        legendstring = [legendstring;{addlegend}];
    end
    legend(legendstring{:});
    hold off
else
    plot(datetime(datevec(timestamp)),
zeros(size(timestamp)),'b','linewidth',2)
end
grid
xtickformat('dd-MM-yy')
xlabel('date')
ylabel('Top-Dn cracking [m/km]')
title('Top-Down cracking - all asphalt layers')


end
```

Source Code: /plottingTools/plotClimateVariables

```
function plotSuccess = plotClimateVariables(timestamp,temp,hum,wspd,srad,rain)
%function plotSuccess = plotClimateVariables(timestamp,temp,hum,wspd,rad,rain)
%Plotting Tools
%
%SIMULATED CLIMATE VARIABLES PLOTTER
%
%This auxiliary script will plot the resulting site's simulated climate vari-
ables time series throughout the design life
%V0.1 - St. Patrick's hangover: 2019-03-18
%   Changelog: x-axis labels in date format

%% code begins
%close figure 1 figure 2 figure 3 figure 4 figure 5
figure(1)
plot(datetime(datevec(timestamp)), temp,'r-.');
grid
xtickformat('dd-MM-yy')
xlabel('date')
ylabel('air temperature[deg C.]')
title('Air Temperature')
legend('air Temperature');

figure (2)
plot(datetime(datevec(timestamp)), hum,'b-.');
grid
xtickformat('dd-MM-yy')
xlabel('date')
ylabel('Humidity [%]')
title('Air humidity')
legend('Humidity [%]');

figure(3)
plot(datetime(datevec(timestamp)), wspd,'k-.');
grid
xtickformat('dd-MM-yy')
xlabel('date')
ylabel('wind speed [m/s]')
title('Wind Speed')
legend('Wind [m/s]');

figure(4)
plot(datetime(datevec(timestamp)), rain,'b--');
grid
xtickformat('dd-MM-yy')
xlabel('date')
ylabel('rainfall [mm]')
title('Rainfall')
legend('Rain [mm]');

figure(5)
plot(datetime(datevec(timestamp)), srad,'color',rand(1,3));
```

```
 %the rand(1,3) sentence will randomly cycle colors (using 3-value RGB color coordinates) for the different temperature series.
    %Ref: https://www.mathworks.com/matlabcentral/answers/25831-plot-multiple-colours-automatically-in-a-for-loop
grid
xtickformat('dd-MM-yy')
xlabel('date')
ylabel('Avg. Sun Net Radiation [Watt/m^2]')
title('Average Solar Net Radiation')
legend('SRad Net [Watt/m^2]');


plotSuccess = 1;
end
```

Source Code: /plottingTools/plotHMATemperature

Source Code: /plottingTools/plotHMATemperature

```
function plotSuccess = plotHMATemperature(timestamp,airTemp,asphTemp)
%function plotSuccess = plotHMATemperature(timestamp,airTemp,asphTemp)
%Plotting Tools: %HMA TEMPERATURE PROFILE PLOTTER
%
%This auxiliary script will plot the HMA layers temperature time series. Read
values from: timestamp vector, air temperature series, and asphalt layer tem-
perature series
%V0.1 - St. Patrick's hangover: 2019-03-18
%   Changelog: x-axis labels in date format

%% code begins

figure(31)
plot(datetime(datevec(timestamp)), airTemp,'r-.')
grid
xlabel('date')
xtickformat('dd-MM-yy')
ylabel('temperature [deg C.]')
title('AC layers temperature')
legendString = {'air Temperature'};
hold on
plot(datetime(datevec(timestamp)),asphTemp(:,1),'color',rand(1,3));
legendString = [legendString;{'HMA Surface temperature'}];

[~,b] = size(asphTemp); %will have to plot the "b" columns
for k = 2:b
    plot(datetime(datevec(timestamp)),asphTemp(:,k),'color',rand(1,3));
    %the rand(1,3) sentence will randomly cycle colors (using 3-value RGB
color coordinates) for the different temperature series.
    %Ref: https://www.mathworks.com/matlabcentral/answers/25831-plot-multiple-
colours-automatically-in-a-for-loop
    addLegendString = sprintf('HMA Layer %g temperature',k-1);
    legendString = [legendString;{addLegendString}];
end
legend(legendString{:})  %The trick to make the increasing-size legend is from
here: https://www.mathworks.com/matlabcentral/answers/38113-plot-help
hold off

plotSuccess = 1;
end
```

Source Code: /plottingTools/plotIRIPSI

```
function plotSuccess = plotIRIPSI(timestamp,IRI,PSI)
%function plotSuccess =
plotSoilHydraulics(timestamp,rain,soilMoistureMatrix,surfaceInfiltration,surfacerunoff)
%Plotting Tools - IRI and PSI prediction
%
%This auxiliary script will plot the predicted roughness index (IRI) and Serviceability index (PSI), which were computed from the predicted distresses
%%V0.1 - St. Patrick's hangover: 2019-03-18
%   Changelog: x-axis labels in date format

%% code begins
figure(45)
yyaxis left
plot(datetime(datevec(timestamp)), real(IRI),'b','linewidth',2)
grid
xlabel('date')
xtickformat('dd-MM-yy')
ylabel('IRI [m/km]')
title('Predicted IRI and PSI')

yyaxis right
plot(datetime(datevec(timestamp)),real(PSI),'color','r','linewidth',2);
ylabel('PSI [dimless]')
xtickformat('dd-MM-yy')
legendstring = [{'IRI'}, {'PSI'}];
legend(legendstring{:});
hold off

plotSuccess = 1;

end
```

Source Code: /plottingTools/plotMR

```matlab
function plotSuccess = plotMR(timestamp,MRMatrix)
%function plotSuccess = plotMR(timestamp,MRMatrix)
%Plotting Tools - Unbound materials' resilient modulus plotter
%
%This auxiliary script will plot the unsaturated Resilient Modulus timeseries
for each %granular material and the subgrade.
%
%MRMatrix is a [timestamp-long - by - number-of-layers] matrix containing the
layers' MR values at each timestamp
%%V2019-03-18

%% code begins

figure(34)
plot(datetime(datevec(timestamp)), MRMatrix(:,1),'r-.')
grid
xlabel('date')
xtickformat('dd-MM-yy')
ylabel('Reesilient Modulus [PSI]')
title('Granular layers and subgrade resilient Modulus')
legendString = {'Granular layer 1'};
hold on
[~,b] = size(MRMatrix); %will have to plot the "b" columns
for k = 2:b
    plot(datetime(datevec(timestamp)),MRMatrix(:,k),'color',rand(1,3));
    %the rand(1,3) sentence will randomly cycle colors (using 3-value RGB
color coordinates) for the different temperature series.
    %Ref: https://www.mathworks.com/matlabcentral/answers/25831-plot-multiple-
colours-automatically-in-a-for-loop
    if k<b
        addLegendString = sprintf('Granular layer %g',k);
        legendString = [legendString;{addLegendString}];
    else
        addLegendString = sprintf('Subgrade');
        legendString = [legendString;{addLegendString}];
    end
end
legend(legendString{:})  %The trick to make the increasing-size legend is from
here: https://www.mathworks.com/matlabcentral/answers/38113-plot-help
hold off

plotSuccess = 1;
end
```

Source Code: /plottingTools/plotRutDepth

```
function plotSuccess = plotRutDepth(timestamp,rutDepth)
%function plotSuccess = plotRutDepth(timestamp,rutDepth)
%Plotting Tools - Rut-depth plot
%
%This auxiliary script will plot the calculated moisture of the pavement lay-
ers throughout the pavement's design life
%%V0.2 - 2019-04-04
%Changelog: Convert rut depth values to mm (they were passing in m)
%%V0.1 - St. Patrick's hangover: 2019-03-18
%   Changelog: x-axis labels in date format

%% code begins%

rutDepth = rutDepth * 1000;  %convert to mm!

figure(41)
auu = isnan(rutDepth(:,end));
if auu(end) ==0   %not isnan
    plot(datetime(datevec(timestamp)),rutDepth(:,end),'b','linewidth',2)
else
    %%%added this case to sort out when rut depth is a mass of NaNs

plot(datetime(datevec(timestamp)),zeros(size(timestamp)),'b','linewidth',2)
end
grid
xlabel('date')
xtickformat('dd-MM-yy')
ylabel('total rut depth[mm]')
title('Depth of rut - sum of all layers')


if auu(end) ==0
    figure(42)
    plot(datetime(datevec(timestamp)), rutDepth(:,end),'b','linewidth',2)
    grid
    xlabel('date')
    xtickformat('dd-MM-yy')
    ylabel('total rut depth[mm]')
    title('Depth of rut - all layers combined')
    legendstring = {'Total Rut Depth'};
    hold on
    [~,b] = size(rutDepth);
    for i = 1:b-1
        plot(datetime(datevec(timestamp)),
rutDepth(:,i),'color',rand(1,3),'linewidth',2);
        if i<b-1
            addlegend = sprintf('Rut depth at layer %g',i);
        else
            addlegend = sprintf('Rut depth at the sub-grade');
        end
        legendstring = [legendstring;{addlegend}];
    end
    legend(legendstring{:});
    hold off
    plotSuccess = 1;
else  %%rutDepth is a NaN
    plotSuccess = 0;
end


end
```

Source Code: /plottingTools/plotSoilHydraulics

```
function plotSuccess =
plotSoilHydraulics(timestamp,rain,soilMoistureMatrix,surfaceInfiltration,surfaceRunoff)
%function plotSuccess =
plotSoilHydraulics(timestamp,rain,soilMoistureMatrix,surfaceInfiltration,surfacerunoff)
%Plotting Tools: %PREDICTED SOIL MOISTURE PLOTTING TOOL
%
%This auxiliary script will plot the calculated moisture of the pavement layers throughout the pavement's design life
%%V0.1 - St. Patrick's hangover: 2019-03-18
%   Changelog: x-axis labels in date format

%% code begins
%close figure 32 figure 33
figure(32)
plot(datetime(datevec(timestamp)), rain,'b','linewidth',2)
grid
xlabel('timestamp')
xtickformat('dd-MM-yy')
ylabel('unit runoff [mm/12h]')
title('Rainfall and infiltration')

hold on
plot(datetime(datevec(timestamp)),real(surfaceInfiltration),'color',
[0.60,0.33,0.45],'linewidth',2);
ylabel('runoff [mm/12h]')
plot(datetime(datevec(timestamp)),real(surfaceRunoff),'k','linewidth',2);
legend([{'Rainfall'};{'infiltration'};{'surfaceRunoff'}]);
hold off

figure(33)
yyaxis left
plot(datetime(datevec(timestamp)), rain,'b','linewidth',2)
grid
xtickformat('dd-MM-yy')
xlabel('timestamp')
ylabel('rainfall [mm]')
title('Rainfall and granular layers" moisture')
legendstring = {'Rainfall'};

yyaxis right
hold on
[~,b] = size(soilMoistureMatrix);
for i = 1:b

    plot(datetime(datevec(timestamp)),
soilMoistureMatrix(:,i),'color',rand(1,3),'linewidth',2);
    if i<b
        addlegend = sprintf('Granular Layer %g moisture',i);
    else
```

```
        addlegend = sprintf('sub-grade moisture');
    end
    legendstring = [legendstring;{addlegend}];
end
ylabel('Vol. moisture cont. [%]')
legend(legendstring{:});
hold off

plotSuccess = 1;
end
```

Source Code: /plottingTools/plotTrafficData

```
function [plotSuccess,reducedTrafficVars] = plotTrafficData(timestamp,AADTMatrix)
%function plotSuccess = plotTrafficData(timestamp,AADTMatrix)
%Plotting Tools - Yearly Traffic by category
%
%This auxiliary script will plot the predicted AADT values for each R103 and
%R110 category (cars, buses, light-trucks, semi-trucks, heavy-trucks) over
each year

%% 1 - get years' vector from the timeStamp

years = year(datetime(datevec(timestamp(1)))):1:year(datetime(datevec(timestamp(end))));

%% 2  - plot AADT by cat. as is.

figure(21)
plot(years, AADTMatrix(1,:),'color',rand(1,3));
grid
xlabel('year')
ylabel('AADT design lane [vehicles/year]')
title('Traffic by R103 category - design lane only')

hold on

[a,~] = size(AADTMatrix); %will have to plot the "a"rows (each for a vehicle
category
for k = 2:a
    plot(years,AADTMatrix(k,:),'color',rand(1,3));
    %the rand(1,3) sentence will randomly cycle colors (using 3-value RGB
color coordinates) for the different temperature series.
    %Ref: https://www.mathworks.com/matlabcentral/answers/25831-plot-multiple-
colours-automatically-in-a-for-loop
end
legend([{'A11'},{'O11'},{'O12'},{'O22'},{'C11'},{'C12'},{'C22'},{'T11S1'},
{'T11S2'},{'T12S1'},{'T11S11'},{'C11R11'},{'T11S3'},{'T12S2'},{'T11S12'},
{'T12S11'},{'T12S3'},{'C11R12'},{'C12R11'},{'C12R12'},{'T11S111'},{'T12S111'},
{'T12S2S2'}]);
hold off

%% - get reduced plot (5-cat)
cars = AADTMatrix(1,:);
buses = AADTMatrix(2,:)+AADTMatrix(3,:)+AADTMatrix(4,:);
trucksLT = AADTMatrix(5,:)+AADTMatrix(6,:);
trucksMD =
+AADTMatrix(7,:)+AADTMatrix(8,:)+AADTMatrix(9,:)+AADTMatrix(10,:)+AADTMatrix(1
1,:);
trucksHV = sum(AADTMatrix(12:end,:));  %I don't feel like writing all the columns manually. This workaround may do the job.

reducedTrafficVars = [cars; buses; trucksLT; trucksMD; trucksHV];
```

```
figure(22)
plot(years,cars,'r')
grid
title('Traffic by Reduced category - design lane only')
xlabel('year')
ylabel('AADT design lane [vehicles/year]')
hold on
plot(years,buses,'b')
plot(years,trucksLT,'color',rand(1,3))
plot(years,trucksMD,'color',rand(1,3))
plot(years,trucksHV,'color',rand(1,3))
legend('cars','buses','light trucks','mid-size trucks','heavy trucks')
hold off
plotSuccess = 1;
end
```

Source Code: /trafficProcessing/axleSpeedFromVehicleSpeed

```
function speedByAxle = axleSpeedFromVehicleSpeed(AADTSpeed)
%function speedByAxle = axleSpeedFromVehicleSpeed(AADTSpeed)
%This function will estimate the avg. speed of each axle type from the speed
of the different traffic vehicles.
%INPUT: is the "AADT speed" vector column -avg. speed by vehicle type-
%OUTPUT: vector containing the avg. speed of each axle category (assuming %all
weights within each axle cat. run at the same speed)
%Categories as follows: [light-weight single / 6 ton single / 10 ton single %/
18 ton tandem / 10 ton tandem / 14 ton tandem / 22/25ton tridem]

speedByAxle = zeros(7,1);

%speed of light-weight axles (cars only)
speedByAxle(1) = AADTSpeed(1);

%speed of 6-ton single axles (entries 2,3, 5, 6 8:end)
speedByAxle(2) = mean(AADTSpeed([2,3,5,6,8:end]));

%speed of 10-ton single axles (entries 2, 5, 8, 9, 10, 11, 12 13 15 16 18 19
20 21 22
speedByAxle(3) = mean(AADTSpeed([2,5,8,9:13,15,16,18:22]));

%speed of 18-ton tandem axles (entries 6,7,9,10,14,15,16,17,18,19,20,22,23)
speedByAxle(4) = mean(AADTSpeed([6,7,9,10,14:20,22,23]));

%speed of 10-ton tandem axles (entries 4,7 )
speedByAxle(5) = mean(AADTSpeed([4,7]));

%speed of 14-ton tandem axles (entries 3-4)
speedByAxle(6) = mean(AADTSpeed([3,4]));

%speed of tridems (entries 13 17)
speedByAxle(7) = mean(AADTSpeed([13,17]));

end  %endfunction
```

Source Code: /trafficProcessing/axlesWeights.m

```
%axle weights by axle category
%auxiliary script used by the Main Code and the trafficToAxes function

axlesSingleLWeights   = 0.5:0.5:2;   %weights in tons!
axlesSingle6Weights   = 4:0.5:9;     %weights in tons!
axlesSingle10Weights  = 8:0.5:13;    %weights in tons!
axlesTandem10Weights  = 8:1:14;
axlesTandem14Weights  = 9:1:17;
axlesTandemWeights    = 12:1:22;
axlesTridemWeights    = 16:1:30;
```

Source Code: /trafficProcessing/trafficSimulator

```
function [designTraffic,designAADT] = trafficSimulator(timestamp,AADTBaseYear,AADTGrowthRate,AADTMonthlyDistr,AADTDailyDistr,AADTHourlyDistr)
% [designTraffic,designAADT] = trafficSimulator(timeStamp,AADTBaseYear,AADTGrowthRate,AADTMonthlyDistr,AADTDatilyDistr,AADTHourlyDistr);
%Auxiliary function to calculate the total number of vehicles over the pavement's design period
% OUTPUT:
%  designTraffic::  Matrix sized [timestamp x number of vehicle categories]% with the 12-hourly traffic per category
%  designAADT   ::  Matrix sized (1+number of vehicle cats x design years) with the yearly AADTs
%
%update 2019-03-19:
%   Changelog:: Corrected AADT asignation for each timestamp (was only bringing AADT for cars, and not for all vehicle categories)
%update 2019-03-15: Corrected an error in the extrapolation (it was applying the growth rate from the current year traffic to get next year's at a power, when it should be linear...)
%update 2018-06-20: The designTraffic will be nighttime/daytime portions of
%traffic per category instead of hourly vehicle flow  (this code itself should not change)

%% 1 - initialize output
numVehicleCats = length(AADTBaseYear);
n = length(timestamp);

designTraffic = zeros(n,numVehicleCats);
initialYear = datevec(timestamp(1));
lastYear = datevec(timestamp(end));

%% 2 - apply the yearly increase rate to AADT to obtain future traffic volumes store in auxiliary variable...

auxTime = initialYear:1:lastYear;
nAT = length(auxTime);

auxAADT = zeros(numVehicleCats,nAT);
auxAADT(:,1) = AADTBaseYear;
%auxAADT is storing the increased traffic (use along auxTime)
for k = 2:nAT
    auxAADT(:,k) = auxAADT(:,k-1).*((1+AADTGrowthRate));%.^(k-1));
end
%consider transposing auxAADT so that it ends up being each column the AADT for each vehicle category. - not do, I need TPDA's in column format %IN CURRENT FORMAT, auxAADT is rows = vehicle categories; columns = years;

%% 3 - Now scan timestamp vector to get hourly trafffic per category - iteration done over time

for k = 1 :n
```

```
    auxDate = datevec(timestamp(k));   %auxDate is year / mo / day / hour /
min /
    locateTPDA = find(auxTime == auxDate(1));  %locate which year I am using
%%NOTE: Matlab suggests this notation <<locateTPDA = auxTime == auxDate(1)>>;
%Update V2019-03-19:: original statement here only pulled AADT for cars and
not for all categories. Corrected!
    TPDA = auxAADT(:,locateTPDA);  %pull the AADT by category for that year. -
column vector
    %update 2018-06-20 - hourly factor now is a 6Am (night-time subtotal) or
6Pm (18hs, daytime subtotal)
    thisMonthFactor = AADTMonthlyDistr(:,auxDate(2)); %pick the monthly cor-
rection factor [column vector]
    whatDayIsIt = weekday(timestamp(k));
    thisDayFactor   = AADTDailyDistr(:,whatDayIsIt);  %pick the daily correc-
tion factor for day 1-7 [sun-sat]
    if auxDate(4) == 6   %sum late-hour nighttime traffic from current day
plus nighttime traffic from prev. day
        if k > 1
            auxDate2 = datevec(timestamp(k-1));
            whatDayWasYest = weekday(timestamp(k-1));
        else
            auxDate2 = [auxDate(1) auxDate(2) auxDate(3) auxDate(4)-1 0 0];
            whatDayWasYest = weekday(datenum(auxDate2));
        end
        thisHourFactor = AADTHourlyDistr(:,1); %pick the late nighttime hourly
factor from day k (1st column) and the early night factor to apply to day k-1
(3rd column)
        thisHourFactor2 = AADTHourlyDistr(:,3);
        thisMonthFactor2 = AADTMonthlyDistr(:,auxDate2(2)); %pick the monthly
correction factor for k-1 day [column vector]
        thisDayFactor2   = AADTDailyDistr(:,whatDayWasYest);  %pick the daily
correction factor for day 1-7 [sun-sat]
        trafficThisHour = TPDA .* thisMonthFactor .*thisDayFactor .*thisHour-
Factor/100 + TPDA .* thisMonthFactor2 .*thisDayFactor2 .*thisHourFactor2/100 ;
%%the "hour factor" is a percentage value!, should divide by 100 to get the
intended value

    else   %only need to sum up daytime traffic
        thisHourFactor  = AADTHourlyDistr(:,2); %pick the daytime hourly fac-
tor  [2nd column vector]
        trafficThisHour = TPDA .* thisMonthFactor .*thisDayFactor .*thisHour-
Factor/100;   %%the "hour factor" is a percentage value!, should divide by 100
to get the intended value
    end
    designTraffic(k,:) = trafficThisHour';
end

designAADT = [auxTime; auxAADT];

end  %endfunction
```

Source Code: /trafficProcessing/trafficToAxles

```
function [singleL, single6, single10, tandems, tandem10, tandem14, tridem] =
trafficToAxes(designTraffic,loadPercentage,trafficAxleLoadCat)
%function [singleL, singles, tandems, tandem10, tandem14, tridem] = trafficToAxes
(designTraffic,loadPercentage,trafficAxleLoadCat)
%
%this function will convert the design traffic by category to axle counts summing
up the axles from all categories. Outputs are column vectors having
hourly passes of each axle type (length equal to length of timeStamp vector -
given in designTraffic (hourly vehicle traffic)).
%
%V0.1 2019-03-19:
%%Changelog9:: BUG DETECTED: definition of whatWeight
       %%origginally pointed at position (j) in the whatAxleIHave vectors (positions
creating a subset of FL-UL/OL vectors). %Actually, whatWeight must be
the weight in the position said by %whatAxleIHave(j)

%% code begins
[n,vehCats] = size(designTraffic);

%% 1 - variable initialization
run 'axlesWeights.m';
%retrieve the list of axles weight range by axle category.
%get the lengths of the load ranges
nsl = length(axlesSingleLWeights);
ns6 = length(axlesSingle6Weights);
ns10 = length(axlesSingle10Weights);
nt10= length(axlesTandem10Weights);
nt14= length(axlesTandem14Weights);
nt  = length(axlesTandemWeights);
ntr = length(axlesTridemWeights);

singleL = zeros(n,nsl);
single6 = zeros(n,ns6);
single10 = zeros(n,ns10);
tandems = zeros(n,nt);
tandem10 = zeros(n,nt10);
tandem14 = zeros(n,nt14);
tridem = zeros(n,ntr);


%% 2 - fill up the output variables. Go by column, summing up the traffic
across all times!

for k = 1:vehCats
    %A - Get axle data for vehicles in category k
    percentages = loadPercentage(4*k-3:4*k);   %these are the percentage of
unloaded / partially loaded / loaded / overloaded vehicles
    weightUL = trafficAxleLoadCat(4*k-3,:);   %bring the weights by axle for
unloaded vehicles
    weightPL = trafficAxleLoadCat(4*k-2,:);   %bring the weights by axle for
partially loaded vehicles
```

```
    weightFL = trafficAxleLoadCat(4*k-1,:);   %bring the weights by axle for
fully loaded vehicles
    weightOL = trafficAxleLoadCat(4*k,:);   %bring the weights by axle for
overloaded vehicles
    % B - sort out the axles
    % %columns of the axle weight table:
    %col 1-2: single lightWeight // 3 - single single/wheel //4-7 single dual-
wheel // 8 - Tandem single-wheel // 9 - tandem Non-homogeneous //10-12 - tan-
dem dual-wheel // 13 - tridem (assumed as homogeneous
    trafficCatK = [designTraffic(:,k)*percentages(1) designTraffic(:,k)*per-
centages(2) designTraffic(:,k)*percentages(3)
designTraffic(:,k)*percentages(4)];   %this matrix here contains the design
traffic of cat-K vehicles sorted according to weight range

    %--1: sort axles for unloaded vehicles
    whatAxleIHave = find(weightUL ~=0);  %non-zero values in weightUL will
tell me which axles I have - whatAxleIHave contains the position of those non-
zero values
    for j = 1:length(whatAxleIHave)
       axleType =  whatAxleIHave(j);
       %update V2019-03-19
       %whatWeight = weightUL(j);   %%locate axle weight for j-th non-zero el-
ement in the weightUL vector
       whatWeight = weightUL(axleType);   %%locate axle weight for j-th non-
zero element in the weightUL vector+
       switch axleType
           case {1,2}  %axle is a light-weight single axle
               targetColumn = find(axlesSingleLWeights == whatWeight);
%locate what column of the singleL output matrix should I drop the axles to
               singleL(:,targetColumn) = singleL(:,targetColumn) + traffic-
CatK(:,1);    %add the axles.
           case 3      %axle is a single-wheel heavy single axle (legal load
of 6 ton)
               targetColumn = find(axlesSingle6Weights == whatWeight);
%locate what column of the single6 output matrix should I drop the axles to
               single6(:,targetColumn) = single6(:,targetColumn) + traffic-
CatK(:,1);    %add the axles.
           case {4,5,6,7}  %axle is a dual-wheel heavy single axle (legal load
of 10.5 ton)
               targetColumn = find(axlesSingle10Weights == whatWeight);
%locate what column of the single10 output matrix should I drop the axles to
               single10(:,targetColumn) = single10(:,targetColumn) + traffic-
CatK(:,1);    %add the axles.
           case 8       %axle is a single-wheel heavy tandem axle (legal load
of 10 ton)
               targetColumn = find(axlesTandem10Weights == whatWeight);
%locate what column of the tandem10 output matrix should I drop the axles to
               tandem10(:,targetColumn) = tandem10(:,targetColumn) + traffic-
CatK(:,1);    %add the axles.
           case 9       %axle is a Non-homogeneous heavy tandem axle (legal
load of 14 ton)
               targetColumn = find(axlesTandem14Weights == whatWeight);
%locate what column of the tandem14 output matrix should I drop the axles to
```

```
               tandem14(:,targetColumn) = tandem14(:,targetColumn) + traffic-
CatK(:,1);    %add the axles.
           case {10,11,12} %axle is a dual-wheel heavy tandem axle (legal load
of 18 ton)
               targetColumn = find(axlesTandemWeights == whatWeight);
%locate what column of the single10 output matrix should I drop the axles to
               tandems(:,targetColumn) = tandems(:,targetColumn) + traffic-
CatK(:,1);    %add the axles.
           otherwise %case 13, axle is a dual-wheel heavy tridem axle (legal
load of 22/25.5 ton)
               targetColumn = find(axlesTridemWeights == whatWeight);
%locate what column of the single10 output matrix should I drop the axles to
               tridem(:,targetColumn) = tridem(:,targetColumn) +
trafficCatK(:,1);    %add the axles.
       end  %end-switch
    end   %end loop for the unloaded vehicles.

    %--2: repeat for partially loaded vehicles.
     whatAxleIHave = find(weightPL ~=0);  %non-zero values in weightUL will
tell me which axles I have - whatAxleIHave contains the position of those non-
zero values
    for j = 1:length(whatAxleIHave)
       axleType =  whatAxleIHave(j);
       %updave V2019-03-19
       %whatWeight = weightPL(j);   %%locate axle weight for j-th non-zero el-
ement in the weightUL vector
       whatWeight = weightPL(axleType);   %%locate axle weight for j-th non-
zero element in the weightUL vector
       switch axleType
           case {1,2}  %axle is a light-weight single axle
               targetColumn = find(axlesSingleLWeights == whatWeight);
%locate what column of the singleL output matrix should I drop the axles to
               singleL(:,targetColumn) = singleL(:,targetColumn) + traffic-
CatK(:,2);    %add the axles.
           case 3      %axle is a single-wheel heavy single axle (legal load
of 6 ton)
               targetColumn = find(axlesSingle6Weights == whatWeight);
%locate what column of the single6 output matrix should I drop the axles to
               single6(:,targetColumn) = single6(:,targetColumn) + traffic-
CatK(:,2);    %add the axles.
           case {4,5,6,7}  %axle is a dual-wheel heavy single axle (legal load
of 10.5 ton)
               targetColumn = find(axlesSingle10Weights == whatWeight);
%locate what column of the single10 output matrix should I drop the axles to
               single10(:,targetColumn) = single10(:,targetColumn) + traffic-
CatK(:,2);    %add the axles.
           case 8       %axle is a single-wheel heavy tandem axle (legal load
of 10 ton)
               targetColumn = find(axlesTandem10Weights == whatWeight);
%locate what column of the tandem10 output matrix should I drop the axles to
               tandem10(:,targetColumn) = tandem10(:,targetColumn) + traffic-
CatK(:,2);    %add the axles.
           case 9       %axle is a Non-homogeneous heavy tandem axle (legal
load of 14 ton)
```

```
               targetColumn = find(axlesTandem14Weights == whatWeight);
%locate what column of the tandem14 output matrix should I drop the axles to
               tandem14(:,targetColumn) = tandem14(:,targetColumn) + traffic-
CatK(:,2);    %add the axles.
           case {10,11,12} %axle is a dual-wheel heavy tandem axle (legal load
of 18 ton)
               targetColumn = find(axlesTandemWeights == whatWeight);
%locate what column of the single10 output matrix should I drop the axles to
               tandems(:,targetColumn) = tandems(:,targetColumn) + traffic-
CatK(:,2);    %add the axles.
           otherwise %case 13, axle is a dual-wheel heavy tridem axle (legal
load of 22/25.5 ton)
               targetColumn = find(axlesTridemWeights == whatWeight);
%locate what column of the single10 output matrix should I drop the axles to
               tridem(:,targetColumn) = tridem(:,targetColumn) +
trafficCatK(:,2);    %add the axles.
       end  %end-switch
    end   %end loop for the partially loaded vehicles.

    %--3: repeat for fully loaded vehicles.
    whatAxleIHave = find(weightFL ~=0);  %non-zero values in weightUL will
tell me which axles I have - whatAxleIHave contains the position of those non-
zero values
    for j = 1:length(whatAxleIHave)
       axleType =  whatAxleIHave(j);
       %%update V2019-03-19:: BUG DETECTED: definition of whatWeight
       %%origginally pointed at position (j) in the whatAxleIHave vectors (po-
sitions creating a subset of FL-UL/OL vectors.)
       %Actually, %%whatWeight must be the weight in the position said by
%whatAxleIHave(j)
       %WRONG: whatWeight = weightFL(j)
       whatWeight = weightFL(axleType);   %%locate axle weight for j-th non-
zero element in the weightUL vector
       switch axleType
           case {1,2}  %axle is a light-weight single axle
               targetColumn = find(axlesSingleLWeights == whatWeight);
%locate what column of the singleL output matrix should I drop the axles to
               singleL(:,targetColumn) = singleL(:,targetColumn) + traffic-
CatK(:,3);    %add the axles.
           case 3      %axle is a single-wheel heavy single axle (legal load
of 6 ton)
               targetColumn = find(axlesSingle6Weights == whatWeight);
%locate what column of the single6 output matrix should I drop the axles to
               single6(:,targetColumn) = single6(:,targetColumn) + traffic-
CatK(:,3);    %add the axles.
           case {4,5,6,7}  %axle is a dual-wheel heavy single axle (legal load
of 10.5 ton)
               targetColumn = find(axlesSingle10Weights == whatWeight);
%locate what column of the single10 output matrix should I drop the axles to
               single10(:,targetColumn) = single10(:,targetColumn) + traffic-
CatK(:,3);    %add the axles.
           case 8       %axle is a single-wheel heavy tandem axle (legal load
of 10 ton)
```

```
               targetColumn = find(axlesTandem10Weights == whatWeight);
%locate what column of the tandem10 output matrix should I drop the axles to
               tandem10(:,targetColumn) = tandem10(:,targetColumn) + traffic-
CatK(:,3);    %add the axles.
           case 9       %axle is a Non-homogeneous heavy tandem axle (legal
load of 14 ton)
               targetColumn = find(axlesTandem14Weights == whatWeight);
%locate what column of the tandem14 output matrix should I drop the axles to
               tandem14(:,targetColumn) = tandem14(:,targetColumn) + traffic-
CatK(:,3);    %add the axles.
           case {10,11,12} %axle is a dual-wheel heavy tandem axle (legal load
of 18 ton)
               targetColumn = find(axlesTandemWeights == whatWeight);
%locate what column of the single10 output matrix should I drop the axles to
               tandems(:,targetColumn) = tandems(:,targetColumn) + traffic-
CatK(:,3);    %add the axles.
           otherwise %case 13, axle is a dual-wheel heavy tridem axle (legal
load of 22/25.5 ton)
               targetColumn = find(axlesTridemWeights == whatWeight);
%locate what column of the single10 output matrix should I drop the axles to
               tridem(:,targetColumn) = tridem(:,targetColumn) +
trafficCatK(:,3);    %add the axles.
       end  %end-switch
    end   %end loop for the fully loaded vehicles.

    %--4: repeat for overloaded vehicles.
    whatAxleIHave = find(weightOL ~=0);  %non-zero values in weightUL will
tell me which axles I have - whatAxleIHave contains the position of those non-
zero values
    for j = 1:length(whatAxleIHave)
       axleType =  whatAxleIHave(j);
       %update V2019-03-19: Bug correction.
       %whatWeight = weightOL(j);   %%locate axle weight for j-th non-zero el-
ement in the weightUL vector
       whatWeight = weightOL(axleType);   %%locate axle weight for j-th non-
zero element in the weightUL vector
       switch axleType
           case {1,2}  %axle is a light-weight single axle
               targetColumn = find(axlesSingleLWeights == whatWeight);
%locate what column of the singleL output matrix should I drop the axles to
               singleL(:,targetColumn) = singleL(:,targetColumn) + traffic-
CatK(:,4);    %add the axles.
           case 3      %axle is a single-wheel heavy single axle (legal load
of 6 ton)
               targetColumn = find(axlesSingle6Weights == whatWeight);
%locate what column of the single6 output matrix should I drop the axles to
               single6(:,targetColumn) = single6(:,targetColumn) + traffic-
CatK(:,4);    %add the axles.
           case {4,5,6,7}  %axle is a dual-wheel heavy single axle (legal load
of 10.5 ton)
               targetColumn = find(axlesSingle10Weights == whatWeight);
%locate what column of the single10 output matrix should I drop the axles to
               single10(:,targetColumn) = single10(:,targetColumn) + traffic-
CatK(:,4);    %add the axles.
```

```
           case 8       %axle is a single-wheel heavy tandem axle (legal load
of 10 ton)
               targetColumn = find(axlesTandem10Weights == whatWeight);
%locate what column of the tandem10 output matrix should I drop the axles to
               tandem10(:,targetColumn) = tandem10(:,targetColumn) + traffic-
CatK(:,4);    %add the axles.
           case 9       %axle is a Non-homogeneous heavy tandem axle (legal
load of 14 ton)
               targetColumn = find(axlesTandem14Weights == whatWeight);
%locate what column of the tandem14 output matrix should I drop the axles to
               tandem14(:,targetColumn) = tandem14(:,targetColumn) + traffic-
CatK(:,4);    %add the axles.
           case {10,11,12} %axle is a dual-wheel heavy tandem axle (legal load
of 18 ton)
               targetColumn = find(axlesTandemWeights == whatWeight);
%locate what column of the single10 output matrix should I drop the axles to
               tandems(:,targetColumn) = tandems(:,targetColumn) + traffic-
CatK(:,4);    %add the axles.
           otherwise %case 13, axle is a dual-wheel heavy tridem axle (legal
load of 22/25.5 ton)
               targetColumn = find(axlesTridemWeights == whatWeight);
%locate what column of the single10 output matrix should I drop the axles to
               tridem(:,targetColumn) = tridem(:,targetColumn) +
trafficCatK(:,4);    %add the axles.
       end  %end-switch
    end   %end loop for the fully loaded vehicles.

end   %end loop for vehicle categories.


end   %endfunction
```

Source Code: /trafficProcessing/wheelFootprint

```
function a = wheelFootprint(axleWeights,axleWheelCount,tirePressure)
%function a = wheelFootprint(axleWeights,axleWheelCount,tirePressure)
%This auxiliary function calculates the radius of the equivalent footprint for
each wheel belonging to a certain type of axle.
%A circular equivalent shape is assumed for the wheel footprint(Refer to Huang
'04 chapter 1 for details)
%
%Input: axleWeights: Vector of wheights in the axle category [in tonnes]
%       tirePRessure: tire pressure for each wheel in the axle (assumed con-
stant throughout the axles -yet the code will admit a different
%       pressure for each axle weight- [PSI]
%       axle Code: 1 = single axle / 2 = tandem axle / 3 = tridem axle
%       axleWheelCount: count of all the wheels in the axle (both sides)
%
%OUTPUT: a: radius of equivalent circular footprint [cm] in a vector the %same
size as the "axleWeights"
%NOTE: This function needs that all the axle weights included in "axleWeights"
have the same number of wheels!

%0 - convert the axle weights in tonnes to pounds so that I can divide by
%tire pressure in PSI and keep units consistency. 1 ton(metric) = 1000kgf =
2204.623 pounds
axleWeights = axleWeights*2204.623;

%1 - calculate contact area for each wheel (assuming all the wheels are at
%the same pressure and load is distributed equally onto each wheel)

contactArea = (1/axleWheelCount)*(axleWeights./tirePressure);

%2 - calculate the equivalent contact radius "a"
a = sqrt(contactArea/pi);

%2b - convert a to cm  [result above is in inches]
a = a*2.54;

end
```