Chapter 6

Parameter Estimation

6.1 General

A problem where the output response to a given input is known but parameters of the process equations are unknown, is called an inverse problem (Sage 1972). This is in contrast to a direct problem, where some input is given and the response is quantified. The problem to be solved in the present work is an inverse parameter estimation problem also referred to as a system identification problem since it is desired to estimate parameters that determine a mathematical description of a hysteretic system. Specifically, parameters are estimated to minimize the difference between experimental results and model output. Here, the objective is to calibrate the hysteresis model introduced in Chapter 5 by minimizing the difference between the computed force and the measured force of a single-bolt joint, given the same displacing function (Figure 6.1).

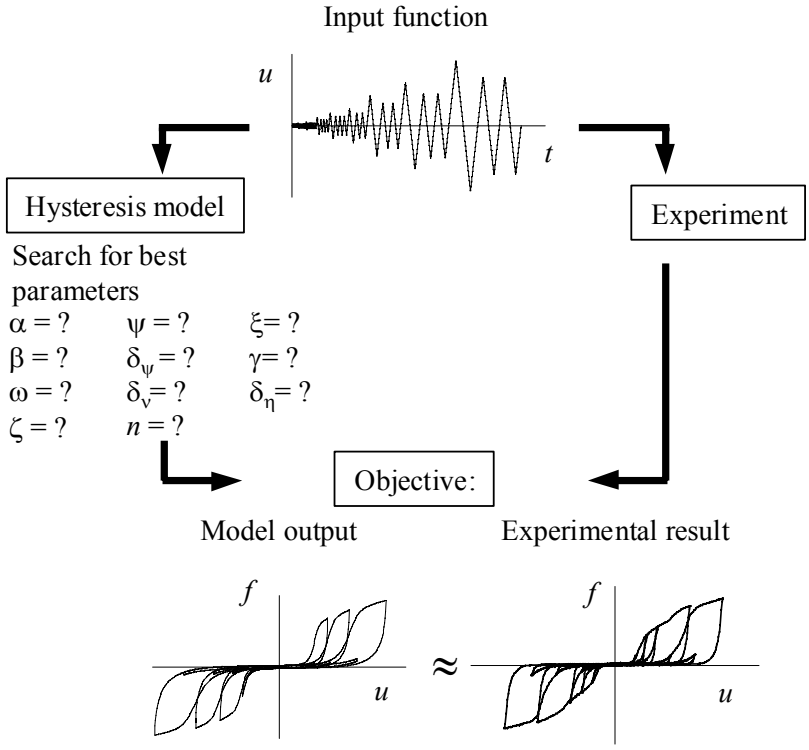


Figure 6.1: System identification problem.

A parameter estimation routine is essential to making the proposed hysteresis model practical and applicable to a wide range of problems. Not only is the hysteresis model extremely sensitive to some parameters, but it is also sensitive to their interaction making it is almost impossible to accurately identify parameters without some kind of systematic search.

6.2 Parameter Estimation: Formulation and Methodology

6.2.2 Formulation

The attempt to minimize a given objective function makes parameter estimation an optimization problem (Figure 6.2). Because any measured output contains errors, the objective function may be defined by one of the two main stochastic estimators available, including Maximum Likelihood and Ordinary Least Squares. Based on its robustness and its positive definite characteristic, Ordinary Least Squares was used in this research.

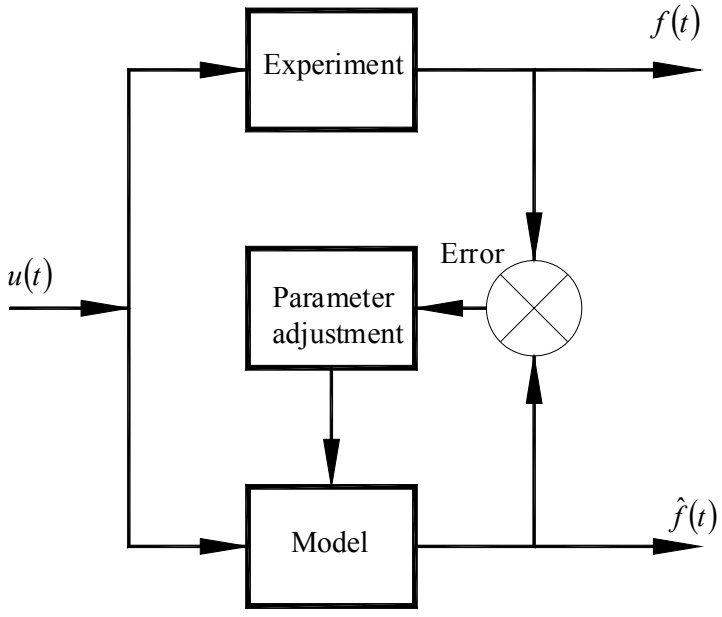


Figure 6.2: Schema of parameter estimation method (after Sage 1972)

6.2.2.1 Maximum Likelihood (ML)

In short, ML estimation consists of the search for the parameter that would have most likely produced the measured data. The ML estimator may be used if one has specific knowledge about distribution and variance of the error (not the case in the present work). Maximum likelihood is achieved if the partial derivatives of the probability density functions of the modeled dependent variables reach zero. If the error is normal and independent with expected mean of zero then ML reverts back to OLS (Beck and Arnold 1977).

6.2.2.2 Ordinary Least Squares (OLS)

The OLS estimator is the by far most widely used method, mostly due to its simplicity and robustness. It is assumed that the error is independent and normally distributed with expected value of zero. OLS works without prior knowledge of both parameters estimated and variances of measurement errors (Garcia 1999). Using OLS estimation, the objective function becomes

$$S = \sum_{i=1}^{N} \left[f\left(p_1, p_2, \cdots, p_j, \cdots p_n; u_i\right) - F_i \right]^2 \tag{6.1}$$

where i represents a measurement point and f is the mathematically modeled relation (hysteresis model in this case) estimating the dependent variable F_i based on measurements of the independent variable, x_i. Adjustment of parameters p_j of the model equations, such that S is minimized, is the fitting process. It is important that n be not confused with N. N is the number of data points measured whereas n denotes the number of unknown parameters. In case of model fitting, the problem is over-determined with N > n (Lybonon and Messa 1999).

6.2.3 Optimization Methods

There is a wealth of information reported in the literature on optimization techniques in general and system identification or parameter estimation methods in particular. Some of the methods that pertain most to this research were reported by Collins et al. (1972), Sage (1972), Lin (1988), Goldberg (1989), Ghanem and Shinozuka (1995), Shinozuka and Ghanem (1995), Carroll (1996), Garcia (1999), and Lybanon and Messa (1999).

While this work does not intend to give an exhaustive review of the methods used in the past and present, it is important to appreciate how the method selected for this work fits in the order of, and compares with, other optimization algorithms. But merely the attempted categorization of existing optimization schemes is a tall order in view of the fact that after decades of intense study all over the world, many different methods and even more hybrids of the

methods devised can be found. Derived from literature, Figure 6.3 presents a sketchy "optimization tree" depicting some of the most widely used algorithms.

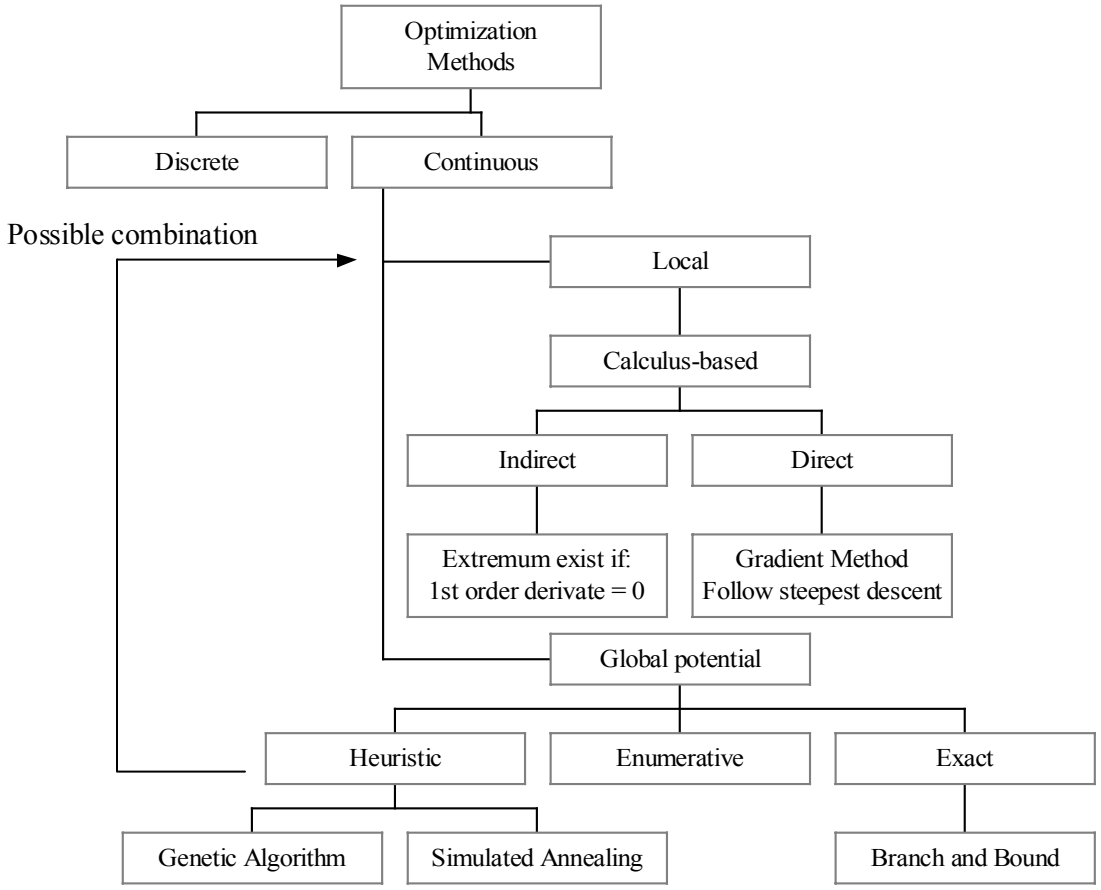


Figure 6.3: The "optimization tree"

Essentially, the world of optimization is a divided one, split between continuous and discrete methods, local and global. Discrete optimization, as the name implies, is concerned with integers and used mostly to solve scheduling problems, network optimization, traveling salesman problems and the like. The most frequently used methods, however, are continuous optimization methods. Continuous optimization methods are generally categorized as local or global. Local optimizers find extrema that are not necessarily the highest peak or lowest valley in the solution space, but in the vicinity of the search area. Almost all calculus-based techniques such as gradient methods are local in nature. Local optimizers have been studied heavily, not least because results are unambiguous, mathematically provable, and comparisons on relative efficiency can be easily made. The science of global optimizers, on the other hand, is somewhat fuzzy to this date. Global optimizers strive to find the absolute best set of parameters to satisfy the objective function. The problem is difficult to solve exactly since there is no simple

mathematical proof of global optimality. The most popular approximate methods are Genetic Algorithms and Simulated Annealing. One of the few exact methods, if not the only exact method to date is the Branch and Bound method. The difficulty the user faces, however, is that there are practically no comparisons of relative efficiency between global optimizers (Neumaier 2001). Application of most global optimizers is more art than science as their problem-specific adaptation and the subject matter knowledge by the user can make a big difference regarding results and efficiency.

6.3 Genetic Algorithm (GA)

6.3.1 What It Is And How It Works

GA's have been around for decades and numerous works detail the theoretical foundation and explain the mechanism behind the algorithm. This is an informal introduction to GA, describing briefly the fundamentals. For more information the reader is encouraged to consult one of the many excellent tutorials put forth by, for example, Goldberg (1989), Heitkoetter and Beasley (1994), Davis (1991), and Garcia (1999). In addition, there is an abundance of resources available on the Internet. One of the most notable is a source code collection of GA routines accessible under http://www.aic.nrl.navy.mil/galist/src/.

In short, GA is a numeric copy of natural evolution. GAs are modeled after the belief that natural evolution is the ultimate optimizer based on the idea of Darwin's revolutionary writing *Origin of Species* which first appeared in 1859. According to Darwin, the power of evolution and subsequent species creation lies in the continuing struggle for survival. An organism with some "variation" that is significant enough to increase its chances for survival is hence more likely to survive, reproduce and pass on the favorable variation to its descendants. Key to producing variations – or otherwise called mutations – is the concept of large numbers and randomness. Thus, eventually over time a given environment will contain organisms that are optimally adapted, and hence a global optimum is found.

From this, the key elements of a GA optimizer become clear. Its core is a random number generator. But rather than just producing a large set of random parameters, GAs make use of probabilistic laws to generate and select parameters. The second key element of GAs is the ability to create mutations and "crossover" (crossing of a pair to create an individual similar to the parents-child concept) on a random basis. Just like mutations in nature, this feature enables the GA to 'climb' out of a local minimum and search for the global optimum.

GAs make use of the black-box approach. The only information the algorithm employs is the goodness-of-fit (or fitness) of a given parameter set, which is similar to the live-or-die principle in nature. To enable the features of mutation and crossover, GAs do not work with parameters directly. Instead, GAs work on a lower level by coding parameters following binary principles or double-precision representation (explained later). To gain insight into the workings of a GA, a short example is presented next (based on Goldberg (1989) and elaborated where appropriate).

6.3.1.1 Example

Assume a roulette wheel that contains a large number of string combinations (a string is often referred to as "chromosome" in GA terminology) (Figure 6.4). For the purpose of demonstration, the "large" number is ten. Suppose a string is an arrangement of zeros and ones representing a real number.

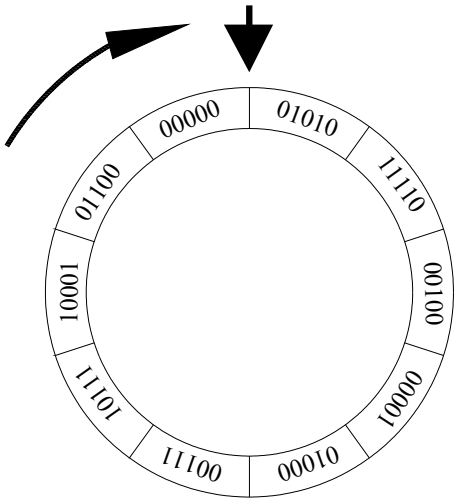


Figure 6.4: Roulette wheel serving as simplification of a random number generator

Each spin of the wheel randomly produces a combination. After spinning the wheel five times we obtain a population of strings and compute a fitness value using some objective function. In addition, we compute the *relative fitness* of each string as a measure of total fitness. The results are presented in Table 6.1.

Table 6.1: Sample strings and fitness values

No.	String	Fitness	% Total
1	00000	123	7.3
2	01000	502	29.8
3	01010	658	39.0
4	11110	12	0.7
5	01100	392	23.2
Total		1687	100

We can use the fitness information of the current set of strings to "reproduce", that is, produce a new set of strings. A simple rule to produce a better population of strings *from* the population we produced randomly is to assign higher probability of creation to the strings with higher fitness values. Thus, our new roulette wheel contains only five slots each of which holds one string of the population depicted in Table 6.1. However, the size of each slot now corresponds to the percentage (relative fitness) that was computed for every string (Figure 6.5). Subsequently the wheel is spun again five times and we obtain a new population, most likely consisting of many redundant strings. The process repeats until the population contains only one string type, which is the optimum.

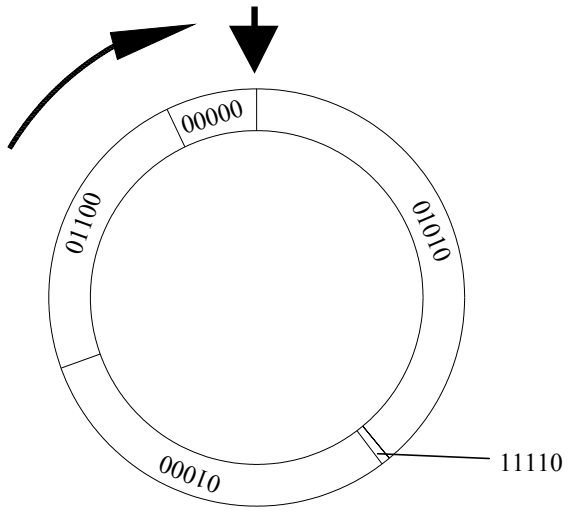


Figure 6.5: New "weighted" roulette wheel with slots sized corresponding to fitness.

From this it is obvious that the size of the population has a great influence on its diversity. Thus, the greater the population size, the higher is the probability that it contains the

one and only best string. Clearly, with only the reproduction mechanism, the algorithm would quickly find an optimal string. However, it would evidently be a local optimizer since we cannot be sure if the optimal string was contained in the population. To circumvent this, the GA employs crossover and mutation.

Crossover takes place after a new population was generated. Simple crossover is a two step process. First, individual strings are paired or "mated" at random. For instance, we may pair strings 3 and 4 of Table 6.1:

3: 01010

4: 11110

Next, an integer position X within the string is chosen, again at random, and the characters at positions after X are swapped. If l is the length of the sting, we find for X

$$1 \leq X < l \tag{6.2}$$

Assume X = 2 then, after swapping, the new string pair becomes

3*:01110

4*:11010

As can be seen from this simple example, entirely new string pairs can be created by crossover. But to increase the GA's pool of diverse strings even more, mutation is introduced. The purpose of mutation is twofold. One is that although new strings can be created by crossover, they still come from old information, which reduces the potential to find entirely new variations. To see this, try to cross over the strings 1 and 2 at position 2. No new string is created since the characters after position 2 are identical in both strings. Second, when strings are crossed over sometimes information is lost (in terms of an old string). In our example mutation is the act of switching a one into a zero at a random location with small probability.

The example demonstrates, though in a rather coarse manner, how GAs work. The fundamentals behind GAs are really quite simple, which may be one reason for their increasing popularity. Other advantages are discussed next.

6.3.2 Why Genetic Algorithm?

The main motivation as to why the GA was selected as parameter estimation method is its versatility. The level of accuracy of the result can be adjusted at will. Furthermore, GAs can be combined with calculus-based techniques such as gradient methods if needed. The mere

option of hybridization opens up enormous possibilities. For instance, the GA may be used to obtain an initial guess of parameters needed for calculus-based techniques (if the problem is differentiable and continuous), which then quickly converge to the optimum. GAs allow different fitting criteria to be easily incorporated in a subroutine without changing the main algorithm. But the flexibility doesn't end there. GAs can execute in parallel, evaluating multiple populations concurrently, which lends itself to efficient computing if multiple processors are at disposal.

But most important, GAs always converge and do not require the objective function to be differentiable. The advantage of this cannot be overstressed. Most test data contain significant error or noise. Calculus-based methods are ill-suited for noisy data. GAs on the other hand can handle any type of data, adding to their robustness. What is more, the use of GA enabled the inclusion of slack growth in the modified hysteresis model. Recall that slack growth was related to the discontinuous displacement amplitude function. While the model is still differentiable at each time step, it cannot be solved in closed-form, and it is therefore much more difficult to develop a stable algorithm based on the gradient method. Because GAs "are blind", as Goldberg (1989) rightly stated, they can handle any objective function with or without constraints, linear or nonlinear. The solution space may be n-dimensional where n can theoretically reach infinity.

Another great advantage of GAs is that no initial point estimates of parameters need to be made. Instead, a range within which the best parameter is likely to lie is input. The range can be virtually from + infinity to – infinity, which means that the initial guess does not affect the outcome. This is a helpful feature especially for new models where it is hard to guess the approximate value of the parameters as was true in this work. The successful convergence of gradient methods, on the other hand, largely depends on the initial guess of the parameters, their sensitivity, and their inter-correlation.

Further motivation for using a GA was its effectiveness as a tool to obtain a better understanding about parameter sensitivity, interaction and range. While GAs take much longer to converge than calculus-based methods, a trend is recognizable relatively fast and the user quickly gains more insight into the problem at hand.

6.3.3 The Costs of Using Genetic Algorithms

Obviously all this flexibility and robustness must come at a price. And costs can be substantial. Compared to calculus-based methods GAs are inherently inefficient. Depending on solution space dimension, population size, and other parameters, the difference in computing time may even be as high as *several orders* of magnitude. The number of function evaluations can

reach millions. Hence, small differences in CPU time to solve any given model can be greatly amplified.

To get a better grasp on the number of function evaluations involved, assume that the input function is 5,000 data points long. This implies that for *each* parameter set *per* population, the LSODE solver (refer to Section 5.3) is called 5,000 times. For problems of this magnitude, good results are usually obtained with a population size of 100 and higher. In other words, to generate *one* population 5,000 x 100 = 500,000 calls to the solver are made, not counting the mating and crossing-over processes to form new generations, which are substantial. Until convergence, hundreds of populations may need to be generated.

Because GAs are easily hybridized and adopted to specific problems, many different versions of GAs exist. But at the same time no guidance as to their problem-specific performance compared with other algorithms can be found. Consequently, the user is left with little choice but to experiment with the algorithm.

While GAs claim to be a global optimizer that find the global optimum with high probability, their solution is never exact. However, exact solutions may not be needed for most problems. But if an exact solution is desired, GAs can always be combined with other methods. In fact, Davis (1991) demonstrated that the combination of GAs with gradient methods is more efficient than each method used separately (Garcia 1999).

6.4 An Efficient Genetic Algorithm For Parameter Estimation and Model Fitting in FORTRAN

Although a wealth of GAs exist in the literature and on the Internet, only one could be found that was written in FORTRAN by Caroll (1996). Of late, the algorithm could be downloaded free of charge from the Internet. However, while Caroll's algorithm was written to solve parameter estimation, it is not optimally suited for this work. A GA recently published by Lybanon and Messa (1999) was specifically developed and optimized for model fitting to solve a satellite altimetry problem. Although the algorithm is written in C (MULTBOLT was developed in FORTRAN), the GA was found to be more appropriate for this research. Consequently, the entire algorithm was rewritten in FORTRAN and modified to solve the problem at hand. The result is an efficient and highly robust GA written in FORTRAN that successfully estimates parameters for the modified hysteresis model developed in Chapter 5. The discussion that follows in Sections 6.4.1 to 6.4.7 is largely based on Lybanon and Messa (1999) and elaborated where appropriate.

6.4.1 Overview

The program organization is depicted in Figure 6.6. The structure is characterized by four main nested loops. The innermost loop is the actual GA that generates a population, checks the solver calculation (LSODE see Section 5.3), and selects and mates pairs to crossover and mutate. Solver checking is necessary because parameters are generated at random. Some parameter combinations trip the solver, and an excessive number of steps is needed to converge. The result is that the solver returns an error flag and stops prematurely with some data points not computed, causing the sums of squares to be too low. To prevent the GA from falsely recognizing the erroneous sums of squares as better fit, the solver computation is checked after each run. If the error flag is detected, the particular parameter set that tripped the solver is assigned the highest sums of squares of the population, which is associated with the worst fitness encouraging the GA to quickly drop the set.

```
Start
Initialize variables
Read parameters and data
LOOP 1: Enter on first pass.  Loop if parameter intervals were shifted. But
Loop a maximum of MAX_PASS times.
        Set interval bounds (equal to user defined on first pass)
        LOOP 2: Enter on first pass. Stop Loop if  largest parameter interval <
        user-specified spread (EPS). But Loop a maximum of maxCycle times.
                LOOP 3: Enter on first pass. Loop timesRepeat times.
                        Generate initial GA population
                        Check solver calculation
                        LOOP 4: GA Loop. Enter on first pass. Loop num_gen
                        times.
                                Generate
                                Check solver calculation
                                Select
                                Crossover
                                Mutate
                        END LOOP 4
                END LOOP 3
                Print final values
                Shrink parameter intervals
                Find largest parameter interval
        END LOOP 2
        Shift intervals if necessary
        Print report
END LOOP 1
End
```

Figure 6.6: Program Organization (after Lybanon and Messa (1999))

Loop 3 executes the GA a user-specified number of times, each time with a different, randomly chosen initial population. Loop 3 is used to increase the "reach" of the GA. Similar effects could be achieved by increasing the population. However, given how numeric random number generators work, it does make a difference if the population is generated twice at different times or if a population twice as big is generated all at once.

Loop 2 progressively decreases or "shrinks" the parameter interval. The GA is an adaptive algorithm in the sense that it is able to "discover" erroneous initial input ranges for the parameters. If the wrong interval is specified and the optimal parameter lies outside the interval, results tend to be clustered near the one side of the interval that should be readjusted. The GA subsequently shifts the interval in the direction of the clustering and starts over, which is the task of Loop 1. Hence, the interval selection for each parameter does not affect the end result, but can make a significant difference in CPU time needed to reach the solution.

For problems that tend to have many local optima in the vicinity of the global optimum, the GA offers an alternate generation model named "crowding model", which replaces only a certain percentage of the original population for the next generation (specified by *genGap*). The crowding model was not used here (*genGap* = 1.0).

6.4.2 Coding

The solution to the problem may be expressed as

$$f\left(p_1, p_2, \cdots, p_j, \cdots p_n; u_i\right) \tag{6.3}$$

where p_j denote the parameters to be estimated. Thus, the p_js are replaced by real numbers r_j and the vector

$$\hat{R} = \begin{Bmatrix} r_1 \\ r_2 \\ \vdots \\ r_j \\ \vdots \\ r_n \end{Bmatrix} \tag{6.4}$$

represents a possible solution. n describes the total number of parameters to be estimated. As elaborated in Section 6.3, GAs do not use real numbers to find the solution but an encoded form that represents the real number, similar to a chromosome that forms the signature of a living cell.

When GAs were first developed, the encoding scheme was a binary string consisting of 0s and 1s. Here, floating-point coding or double-precision representation is used rather than binary coding. Binary coding is less suited for numerical optimization problems (Garcia 1999). If u_j and l_j specify the upper and lower bounds to r_j, then r_j can be represented such that

$$r_j = d_j \cdot \left(u_j - l_j\right) + l_j \tag{6.5}$$

and

$$0.0 \le d_j \le 1.0 \tag{6.6}$$

is a floating-point number corresponding to r_j in the interval $[l_j, u_j]$. Hence, the solution vector can be reconstructed to

$$\hat{R} \triangleq \hat{D} = \begin{Bmatrix} d_1 \\ d_2 \\ \vdots \\ d_j \\ \vdots \\ d_n \end{Bmatrix} \tag{6.7}$$

Since d_j varies from 0 to 1, crossover and mutation schemes can be applied using random number generator functions.

6.4.3 Objective and Fitness Function

Global optimum is reached if Equation (6.1) is minimized. Since a solution vector in its encoded form is represented by $\hat{D}$, it is reconverted to real numbers by Equation (6.5). An initial fitness, the *prefitness* is obtained by (all symbols are as specified earlier)

$$prefitness\left(\hat{R}\right) = \sum_{i=1}^{N} \left[f\left(r_1, r_2, \cdots, r_j, \cdots r_n; u_i\right) - F_i \right]^2 \tag{6.8}$$

GAs work by maximization rather than minimization. In other words, the higher the fitness the better the solution. To solve Equation (6.1) the *rawfitness* is computed using the expression

$$rawfitness = MAX_{\,prefitness} - prefitness\left(\hat{R}\right) \tag{6.9}$$

which turns the minimization problem into a maximization problem. $MAX_{prefitness}$ stands for the maximum prefitness (worst fit) of a population. It is clear then that the best-fit parameter set has a rawfitness value which is closest to $MAX_{prefitness}$.

6.4.4 Scaling

The rawfitness is not the fitness employed by the GA. Instead, the GA scales the rawfitness to obtain the final *fitness* that is utilized for selection purposes and reproduction. Scaling is used by most GAs to avoid premature convergence in the early stage and encourage convergence at later stages. During the exploration stage, few "superindividuals" tend to dominate the selection process leading to inappropriate levels of competition, whereas later in the simulation process, when the population is less diverse, the simulation tends to lose focus (Goldberg 1989). Hence, early on, negative scaling should be used and positive scaling of fitness values should be employed as the simulation reaches a more mature state. The GA applied in this research makes use of linear scaling.

$$fitness\left(\hat{R}\right)\Big|_{gen} = m_p \cdot rawfitness\left(\hat{R}\right) + b_p \tag{6.10}$$

for

$$gen > beginScale \cdot num\_gen \tag{6.11}$$

with

$$m_p = \frac{\left(scale\_fac \cdot rawfitness_{mean} - rawfitness_{mean}\right)}{rawfitness_{\max} - rawfitness_{mean}} \tag{6.12}$$

$$b_p = \frac{rawfitness_{mean}}{rawfitness_{\max} - rawfitness_{mean}} \cdot \left(rawfitness_{\max} - scale\_fac \cdot rawfitness_{mean}\right) \tag{6.13}$$

m_p, b_p	constants computed for each population
fitness	fitness computed for each population and generation
$rawfitness_{mean}$	average rawfitness of a population
$rawfitness_{max}$	maximum rawfitness of a population
scale_fac	user-specified scale factor
num_gen	number of generations to be formed per population
gen	generation count
beginScale	percentage of generations before scaling starts (*beginScale < 1.0)*

Good results were achieved with scale-fac = 1000, *beginScale* = 0.5 and *num_gen* = 10. The user is advised, however, that the higher the number of generations the higher are CPU requirements. Each generation is a new population and the number of calls to the solver increases accordingly. It was found that with 10 generations per *timesRepeat*, time requirements were at the practical limit of the computer equipment available (Pentium III, 800 MHz). For their problem, Lybanon and Messa used *num_gen* of up to 200. Larger scale factors (1000 – 100000) increase accuracy of the solution, but the probability that premature convergence is reached increases accordingly. While the probability decreases with smaller factors (10 – 1000), accuracy of results suffers.

Negative scaling is executed during the initial stage of the program run.

$$scale\_fac\big|_{gen} = nscale \tag{6.14}$$

for

$$gen < 0.1 \cdot num\_gen \tag{6.15}$$

To encourage exploration early, *nscale* = 0.1 was used.

6.4.5 Selection Strategy

Selection is based on "stochastic sampling without replacement" (Lybanon and Messa 1999) also described by Goldberg (1989) and referred to as the "expected value model". Suppose the vector (organism) $\hat{D}$ represents a candidate solution with fitness f_i. Then the expected number of offspring (children) generated from the organism is equal to the truncated division

$$E = \frac{f_i}{f_{mean}} \tag{6.16}$$

f_i	fitness of organism i
f_{mean}	average fitness of entire population
E	expected value of number of offspring (integer)

If an offspring is selected E number of times (at random), the organism is dropped and no longer available for selection. Any unfilled slots in the new population are filled based on Equation (6.16). Furthermore, an "elitist strategy" (De Jong 1975) is used where the best organism of each population is selected and placed unchanged into the next generation preventing early loss of information through crossover or mutation.

6.4.6 Crossover, Mutation and Seeding

The GA allows the user to choose between four crossover strategies: arithmetic crossover, one-point crossover, combined one-point and arithmetic, and combined arithmetic and two-point crossover. To solve the modified hysteresis model, the last option, combined arithmetic and two-point crossover, was chosen as it works better for problems with many parameters. The user can also specify the crossover rate, which is typically 0.9 (90% of pairs are crossed over). Since the data representation is double precision and not binary, the crossover scheme differs. Let

$$\widehat{D}_1 = \begin{Bmatrix} d_{1,1} \\ d_{2,1} \\ \vdots \\ d_{j,1} \\ \vdots \\ d_{n,1} \end{Bmatrix} \qquad \widehat{D}_2 = \begin{Bmatrix} d_{1,2} \\ d_{2,2} \\ \vdots \\ d_{j,2} \\ \vdots \\ d_{n,2} \end{Bmatrix} \tag{6.17}$$

be candidate solutions, which were selected and paired for crossover. Then

$$\widehat{D}_{12} = \begin{Bmatrix} d_{1,1} \\ d_{2,1} \\ \vdots \\ d_{i-1,1} \\ t \\ d_{i+1,2} \\ d_{i+2,2} \\ \vdots \\ d_{n,2} \end{Bmatrix} \tag{6.18}$$

is the crossed over offspring. Where

$$t = p \cdot d_{i,1} + (1.0 - p) \cdot d_{i,2} \tag{6.19}$$

and

$$i = random(1,2,\cdots,n) \tag{6.20}$$

and p is a random floating point between 0.0 and 1.0.

In addition to providing several crossover strategies, the GA incorporates four different mutation strategies including:

- Uniform mutation: the value of a single d_j is randomly changed to a floating-point number between 0.0 and 1.0. User-specified variable: *pr_u*
- Boundary mutation: the value of a a single d_j is randomly changed to *either* 0.0 *or* 1.0. User specified variable: *pr_b*
- Non-uniform mutation: A small value is added to or subtracted from d_j but d_j must not exceed 1.0 or fall below 0.0. The effect is decreased (decayed) at higher generation count. User specified variable: *pr_n*
- Non-uniform mutation without decay: Same as above but without decay. User specified variable: *pr_n*

The variables *pr_u*, *pr_b*, and *pr_n* designate probabilities set by the user. In this work, the probabilities were assigned values of 0.1, 0.01, and 0.1, respectively. For more information on both crossover and mutation strategies for floating-point representation, the reader is referred to Michalewicz (1992).

Seeding is used to make the GA more efficient by using information from a previous run for the offspring generation of the next run. With seeding turned on, not all solution vectors in

the population are generated at random. Rather, the best vectors from the runs of the previous cycle (cycle is one pass through Loop 2 in Figure 6.6) are used as a pool to generate a single new organism, which gives the next run a lead over the previous run. To give the GA plenty of exploration runs, seeding was not started until Cycle 8.

6.4.7 Shrinkage

Shrinkage is used to narrow the parameter intervals set initially by the user. This process also determines the convergence criterion. The criterion used here is uniformity of the population. That is, the algorithm progressively narrows the interval of each parameter and the user can specify the interval spread that must be reached by all parameters before the algorithm is stopped. The amount of shrinkage can be directly controlled by the user. Here, *shrinkage* = 0.5.

6.4.8 User Input

To summarize the preceding discussion, all parameters that must be specified by the user and their values as used in this research are listed in Table 6.2. The parameters are stored in an external file and read in when the program is started.

Table 6.2: User input parameters

Name	Description	Values as used
pop_size	population size	400
num_gen	number of generations	10
scale_fac	scale factor	1000
nscale	negative scale factor for first 10% of generations	0.1
beginScale	initiates scaling after this percent of generations	0.5
genGap	initiates crowding model. 1.0 turns off	1.0
prob_xover	probability of crossover	0.9
xover_type	specifies type of crossover	4
pr_u	probability of uniform mutation	0.10
pr_b	probability of boundary mutation	0.01
pr_n	probability of non-uniform mutation	0.10
timesRepeat	number of runs per cycle	5
maxCycles	maximum number of cycles to be performed	50
shrinkage	shrinkage of parameter intervals for each cycle	0.5
num_unknown	number of parameters to be estimated	11
convert_to_max	1 if objective function is to be minimized, 0 if maximized	1
freq-full	frequency of full report	10
Freq_brief	frequency of short report	5
	Required for each parameter*:*	
lower-bound(i)	lower endpoint of interval within which parameter *i* is assumed to lie	
upper_bound(i)	upper endpoint of interval within which parameter *i* is assumed to lie	

The parameter *convert_to_max* is changed to zero if one solves an objective function that is to be maximized.

6.4.9 Problem Specifics

Although the parameter estimation is treated as an unconstrained problem, several constraints do exist. The most obvious are lower bounds for parameters that cannot be negative such as the non-linear frequency, ω. Not permitted values of parameters may actually cause the solver to crash, since singularities may result. Though the user can specify initial bounds of a parameter, the GA may shift the lower bound into negative territory. Hence to avoid crashing the solver, only intervals whose parameters can be negative, such as γ, are permitted to be adjusted below zero or any other lower bound values that apply (for instance n must be bigger or equal to 1.0). Likewise, several parameters have upper bounds and similar limitations apply.

Another important constraint is the shape of the hysteresis. Recall, that for wood structural systems, a weak softening hysteresis model best describes the data, which is true for

$$\gamma - \beta < 0.0 \tag{6.21}$$

and

$$\beta + \gamma > 0.0 \tag{6.22}$$

For the conditions expressed in Equations (6.21) and (6.22) to be met, β may be computed as

$$\beta = p_\beta + \left| p_\gamma \right| \tag{6.23}$$

		Unit
β	parameter input into LSODE to solve the hysteresis model. β exceeds the absolute value of γ	
p_β	GA parameter estimate of β	
p_γ	parameter estimate of γ	

The mathematical hysteresis model is embedded in the PRE_OBJ function, which is called by OBJFUNC function. OBJFUNC was changed to convert the appropriate amount of parameters. PRE_OBJ calls LSODE. The subroutine CALC_CHECK was incorporated to check solver status as explained in Section 6.4.1. The maximum interval spread (*EPS*) that was allowed for parameter estimation was 0.15. Note that this spread applied to the parameter that varied most.

6.5 Remarks

Although satisfactory results were obtained with the current GA, there is still room for improvement, especially concerning CPU time. To obtain an optimum parameter set, a Pentium III, 800Mhz computed for as long as 48 hours. However, computation time varied. Sometimes a good fit was obtained after just 24 hours. The term "good fit" is somewhat fuzzy. Depending on yield mode, noise and other factors such as strain hardening of the bolt, the quality of the fit fluctuated. In addition, without running the analysis several times, no statistics can be revealed as to the probability that a global optimum was really reached. Moreover, significant efficiency increase may be obtained by additional experimentation with crossover probabilities, mutation probabilities, modified selection strategies, and of course population size, number of generations and the like.

While the facts elaborated above seem to pinpoint some limits in the quality of estimated parameters, it is important to take a look at the bigger picture. First, with 5,000 data points and a force amplitude of up to 18 kN, variations in sums-of-errors of as high as 20 (average error variation per data point = 20/5000) hardly affect the final outcome, which is a load-slip plot of the entire joint. Thus, while it is important to be *close* to the global optimum, it is less important to hit it exactly considering the enormous computer resources that would be needed for the present

study (a total of 20 parameter sets were estimated). The GA converges relatively quickly to a useable range for each parameter, but computing time increases exponentially with decreasing final parameter interval requirements. Second, as is shown in Chapter 5, inherently to the variability of wood, load-slip data of the tested joints exhibit significant variation, both in shape and amplitude of the hysteresis. It is therefore questionable whether the extensive CPU time consumed in this study was even necessary or whether a more coarse fit would have been satisfactory, given the excellent results achieved.

Although the science of heuristic optimizers is not new, it is, compared to other fields, still novel and the application of GAs is just beginning to penetrate many aspects of engineering sciences. Therefore, standardization is mostly lacking and efficiency studies are hard to find. Given the late increase in popularity and ever increasing computing power, the future for GAs looks bright, however, and there is hope that wider application will lead to eventual standardization and increased efficiency.

6.6 Results and Discussion

The fitting process yields best results if fasteners are perfectly centered in the hole prior to testing. That is, slacks in positive and negative directions of movement are opposite but equal. A recorded offset in the data is difficult to correct since the input displacement is still symmetric, but the acquired load at each time step is not. In other words, if an offset in slack exists prior to testing such that, for example, there is less slack in positive direction of movement, then load in that direction will pick up earlier in time than in negative direction.

If parameters are not estimated for the entire test data up to failure, the resulting fit may be inaccurate because of lack of information regarding degradation and hysteresis shape. Thus, it is not sufficient to fit the model to just a portion of a test if performance up to failure is to be modeled (Figure 6.7). As general rule, parameters cannot be estimated with a high level of confidence from data of a test or a portion of a test whose energy requirements on the system are lower than the input data used to make predictions with the model because degradation is modeled as a function of dissipated hysteretic energy. A problem that arises with full-test fit, however, are the generally lower forces during the initial testing portion if a stepwise increasing displacing function is used. Obviously, a 10 percent deviation of 10kN results in a higher sums of square error than a 10 percent deviation of 1kN. Hence, using the ordinary least squares estimator, the GA is encouraged to produce a lower relative error at higher loads and associated larger displacements.

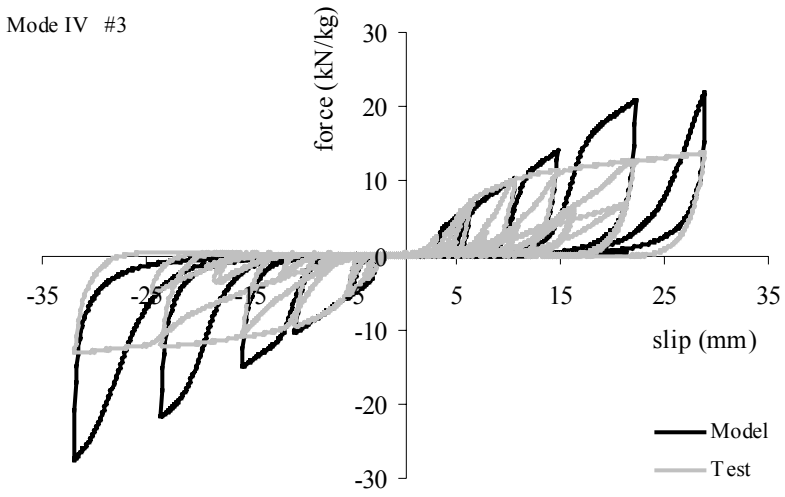


Figure 6.7: Inaccurate fit resulting from insufficient data input. While the model agrees with the initial part of the test, considerable lack of fit can be observed at larger displacements due to erroneous degradation and hysteresis shape parameters.

The GA was not run repeatedly for each fitting process. Repeated runs increase confidence that the true global optimum is obtained based on statistical tests. However, this was not deemed necessary given the good fit reached for all specimens. The results of the fitting process are graphically and numerically depicted in Figures 6.8 through 6.26. Note that Specimen #5 of Mode IV yield failed prematurely and was hence omitted in the analysis. Results should be considered along with descriptions of specimens and procedures as outlined in Chapter10. As elaborated in Chapter 10, the lumber utilized to test Mode IV yield consisted largely of boxed-heart lumber, which contained considerable portions of low-density juvenile wood, and other defects such as splits and cracks, leading to increased variability of results.

In general, excellent fit was obtained for Mode II – type hystereses. The modified hysteresis model had somewhat more difficulties following Mode IV – type loops attributed in part to the mixture of plastic hinge formation, strain hardening and inferior lumber. Because specimens failed at varying displacement levels, the amount of data points input in the GA to estimate parameters varied from specimen to specimen. In addition, data exhibiting excessive strain hardening effects (i.e. significant load recorded when the displacement passes through zero), which occurred close to failure, was not included in the analysis as this would have "misled" the GA. The reason for this is the algorithm would have put too much emphasis on trying to fit the last portion of the test at the expense of initial accuracy. Hence, the sums of squares error achieved, which is clearly a function of the number of data points, is expressed as sums of squares per data point to allow comparisons between specimens.

Parameters	Values
α	0.360775
β	0.802806
ω	1.395263
ζ	0.975164
n	1.776669
ψ_0	0.167594
δ_ψ	0.064088
δ_ν	0.008946
ξ	0.016029
γ	-0.59844
δ_η	0.008754
yield mode	II
specimen #	1
SSE / dp	0.135

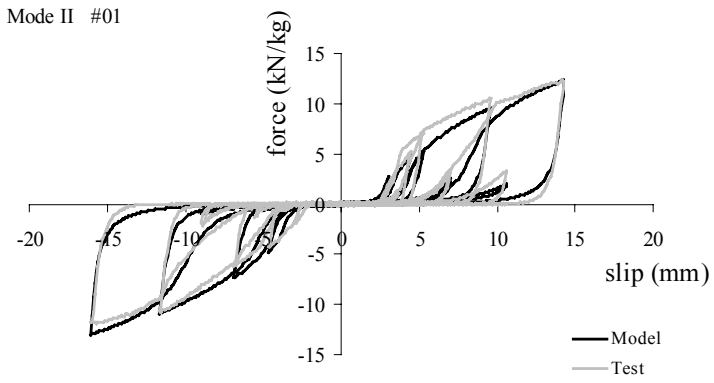


Figure 6.8: Estimated parameters and corresponding model fit of Specimen #1, Mode II

Parameters	Values
α	0.278696
β	0.863364
ω	1.400324
ζ	0.989477
n	1.565305
ψ_0	0.055635
δ_ψ	0.048978
δ_ν	0.008655
ξ	0.009409
γ	-0.5171
δ_η	0.006192
yield mode	II
specimen #	2
SSE / dp	0.049

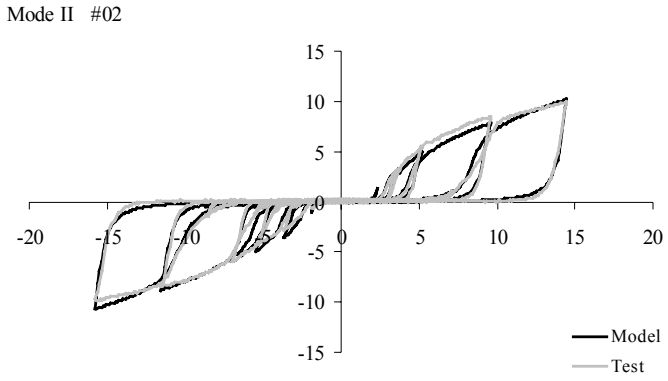


Figure 6.9: Estimated parameters and corresponding model fit of Specimen #2, Mode II

Parameters	Values
α	0.563785
β	0.566122
ω	1.187919
ζ	0.990697
n	1.675955
ψ_0	0.066287
δ_ψ	0.047125
δ_ν	0.00434
ξ	0.015316
γ	-0.17381
δ_η	0.006166
yield mode	II
specimen #	3
SSE / dp	0.330

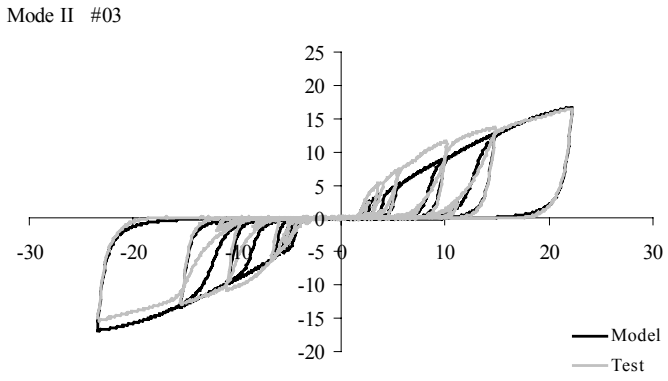


Figure 6.10: Estimated parameters and corresponding model fit of Specimen #3, Mode II

Parameters	Values
α	0.25217
β	1.054192
ω	1.601806
ζ	0.991088
n	1.31411
ψ_0	0.081669
δ_ψ	0.039091
δ_ν	0.009859
ξ	0.010692
γ	-0.70665
δ_η	0.007315
yield mode	II
specimen #	4
SSE / dp	0.065

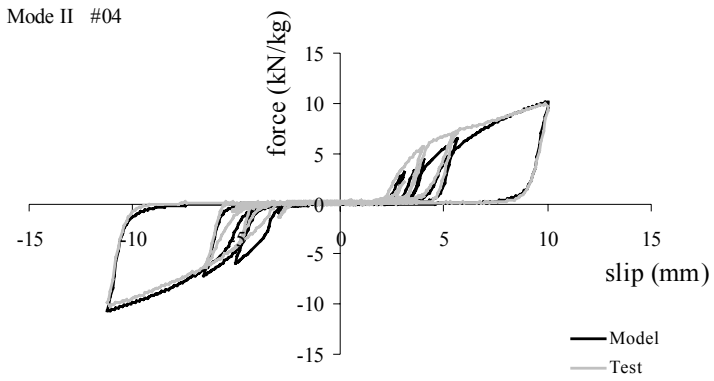


Figure 6.11: Estimated parameters and corresponding model fit of Specimen #4, Mode II

Parameters	Values
α	0.32046
β	0.918625
ω	1.282484
ζ	0.988079
n	1.469065
ψ_0	0.05772
δ_ψ	0.05759
δ_ν	0.004813
ξ	0.010737
γ	-0.54426
δ_η	0.004622
yield mode	II
specimen #	5
SSE / dp	0.084

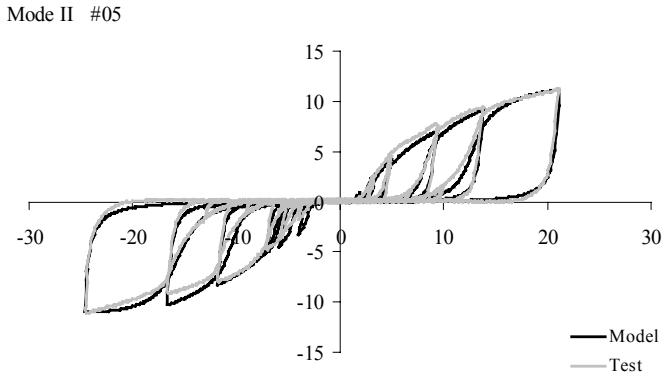


Figure 6.12: Estimated parameters and corresponding model fit of Specimen #5, Mode II

Parameters	Values
α	0.340381
β	0.901119
ω	1.334404
ζ	0.985081
n	1.492587
ψ_0	0.065836
δ_ψ	0.064879
δ_ν	0.003058
ξ	0.015166
γ	-0.53403
δ_η	0.003979
yield mode	II
specimen #	6
SSE / dp	0.091

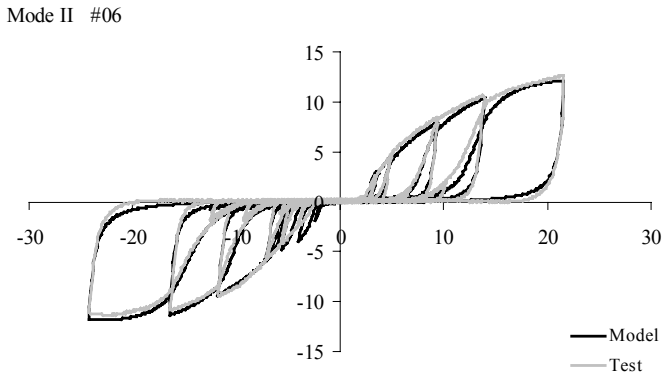


Figure 6.13: Estimated parameters and corresponding model fit of Specimen #6, Mode II

Parameters	Values
α	0.264234
β	0.620275
ω	1.243159
ζ	0.987878
n	1.84719
ψ_0	0.077806
δ_ψ	0.049747
δ_ν	0.005439
ξ	0.019109
γ	-0.53394
δ_η	0.004376
yield mode	II
specimen #	7
SSE / dp	0.084

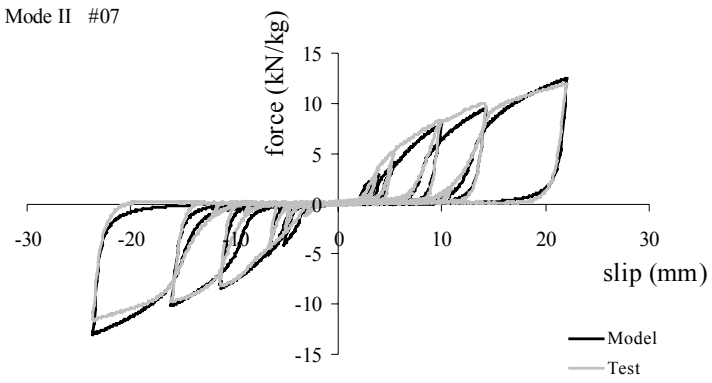


Figure 6.14: Estimated parameters and corresponding model fit of Specimen #7, Mode II

Parameters	Values
α	0.230357
β	0.463618
ω	1.592574
ζ	0.986914
n	1.789566
ψ_0	0.057774
δ_ψ	0.053153
δ_ν	0.005599
ξ	0.016994
γ	-0.23696
δ_η	0.00261
yield mode	II
specimen #	8
SSE / dp	0.146

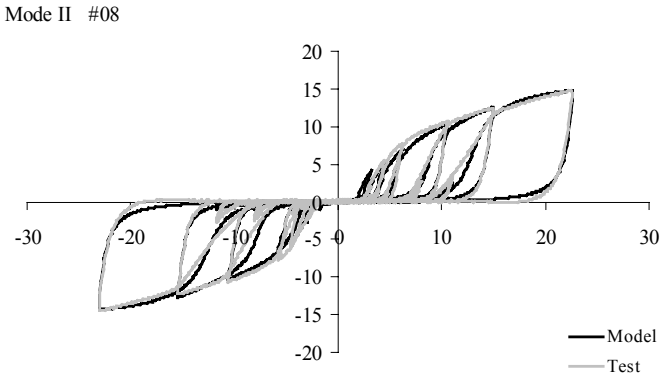


Figure 6.15: Estimated parameters and corresponding model fit of Specimen #8, Mode II

Parameters	Values
α	0.25776
β	0.862885
ω	1.545841
ζ	0.985677
n	1.325332
ψ_0	0.071291
δ_ψ	0.049684
δ_ν	0.003487
ξ	0.011852
γ	-0.66068
δ_η	0.008827
yield mode	II
specimen #	9
SSE / dp	0.090

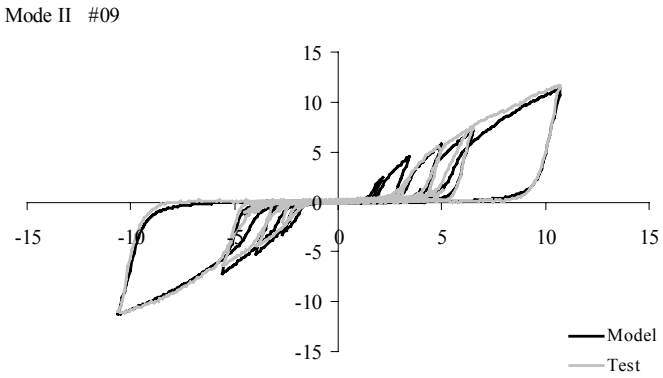


Figure 6.16: Estimated parameters and corresponding model fit of Specimen #9, Mode II

Parameters	Values
α	0.354352
β	1.065951
ω	1.465699
ζ	0.982779
n	1.481031
ψ_0	0.065623
δ_ψ	0.066371
δ_ν	0.00757
ξ	0.013481
γ	-0.53604
δ_η	0.007798
yield mode	II
specimen #	10
SSE / dp	0.096

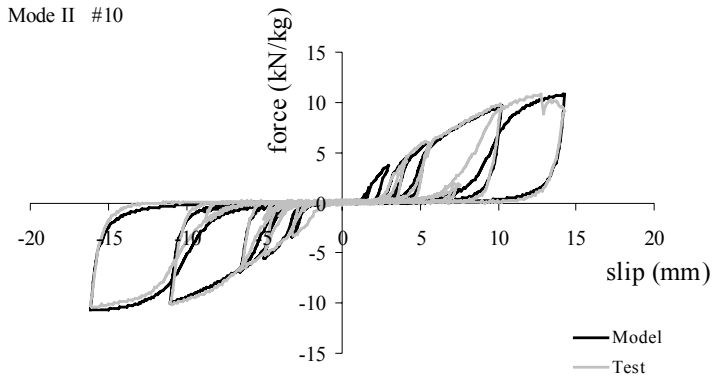


Figure 6.17: Estimated parameters and corresponding model fit of Specimen #10, Mode II

Parameters	Values
α	0.29514
β	2.293517
ω	1.748329
ζ	0.979875
n	2.552348
ψ_0	0.366356
δ_ψ	0.00523
δ_ν	0.083865
ξ	0.025456
γ	-0.49942
δ_η	0.068443
yield mode	IV
specimen #	1
SSE / dp	0.178

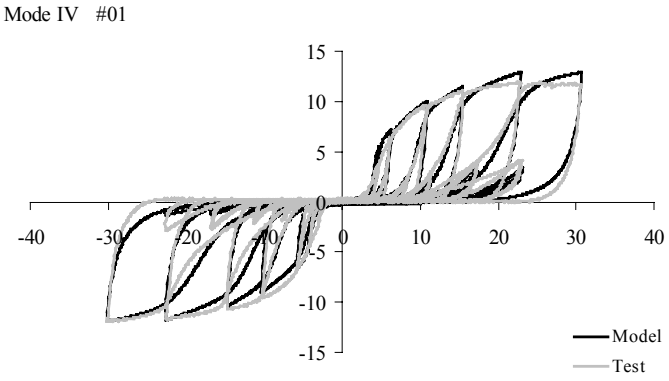


Figure 6.18: Estimated parameters and corresponding model fit of Specimen #1, Mode IV

Parameters	Values
α	0.480918
β	7.501033
ω	1.907905
ζ	0.973032
n	3.070036
ψ_0	0.539356
δ_ψ	0.001747
δ_ν	0.091581
ξ	0.033164
γ	-2.53669
δ_η	0.077982
yield mode	IV
specimen #	2
SSE / dp	0.352

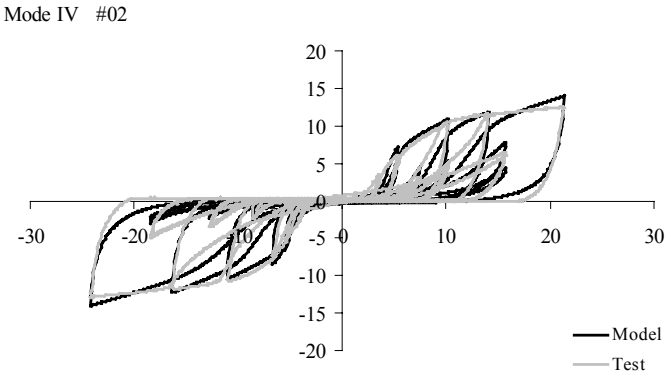


Figure 6.19: Estimated parameters and corresponding model fit of Specimen #2, Mode IV

Parameters	Values
α	0.403979
β	11.40256
ω	1.889841
ζ	0.976872
n	3.229515
ψ_0	0.484147
δ_ψ	0.000691
δ_ν	0.094627
ξ	0.034248
γ	-4.55995
δ_η	0.076046
yield mode	IV
specimen #	3
SSE / dp	0.409

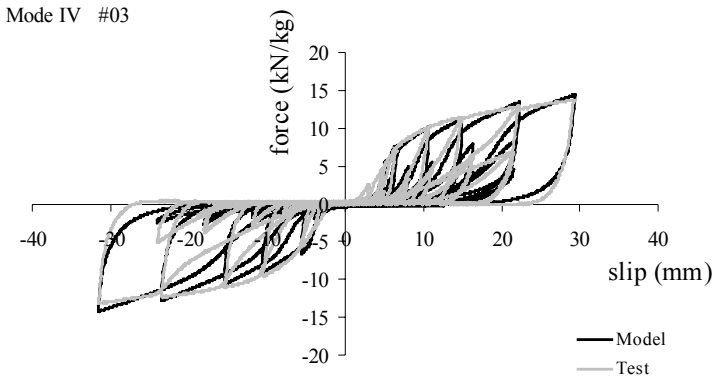


Figure 6.20: Estimated parameters and corresponding model fit of Specimen #3, Mode IV

Parameters	Values
α	0.581105
β	10.36245
ω	2.174239
ζ	0.966376
n	2.741066
ψ_0	0.583732
δ_ψ	0.009962
δ_ν	0.025996
ξ	0.03509
γ	-4.08524
δ_η	0.061014
yield mode	IV
specimen #	4
SSE / dp	0.397

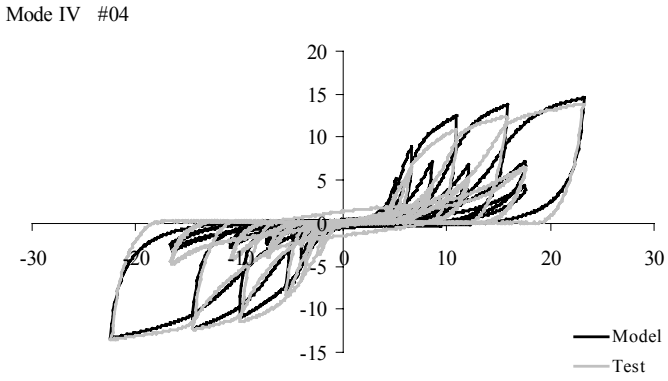


Figure 6.21: Estimated parameters and corresponding model fit of Specimen #4, Mode IV

Parameters	Values
α	0.608406
β	6.301517
ω	1.714059
ζ	0.961484
n	3.781685
ψ_0	0.625969
δ_ψ	0.008698
δ_ν	0.095548
ξ	0.042298
γ	-3.13172
δ_η	0.094712
yield mode	IV
specimen #	6
SSE / dp	0.457

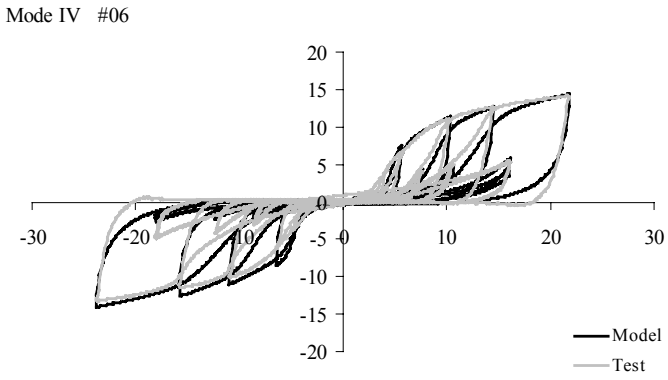


Figure 6.22: Estimated parameters and corresponding model fit of Specimen #6, Mode IV

Parameters	Values
α	0.59479
β	0.703489
ω	1.598973
ζ	0.965015
n	2.030553
ψ_0	0.56163
δ_ψ	0.0023
δ_ν	0.132482
ξ	0.042941
γ	-0.01676
δ_η	0.06269
yield mode	IV
specimen #	7
SSE / dp	0.570

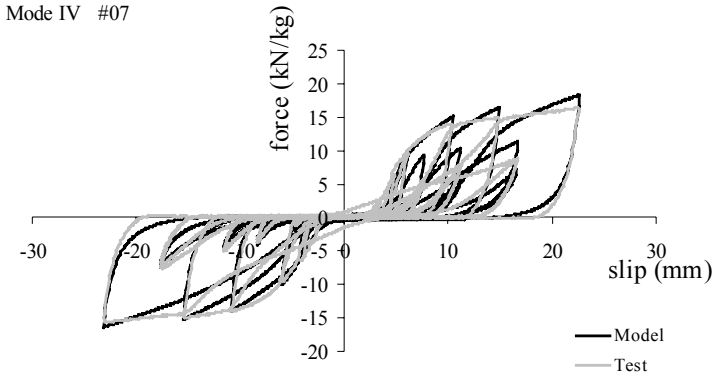


Figure 6.23: Estimated parameters and corresponding model fit of Specimen #7, Mode IV

Parameters	Values
α	0.419649
β	10.07431
ω	1.61656
ζ	0.976856
n	3.579933
ψ_0	0.400914
δ_ψ	0.004095
δ_ν	0.15849
ξ	0.029463
γ	-5.83939
δ_η	0.115226
yield mode	IV
specimen #	8
SSE / dp	0.198

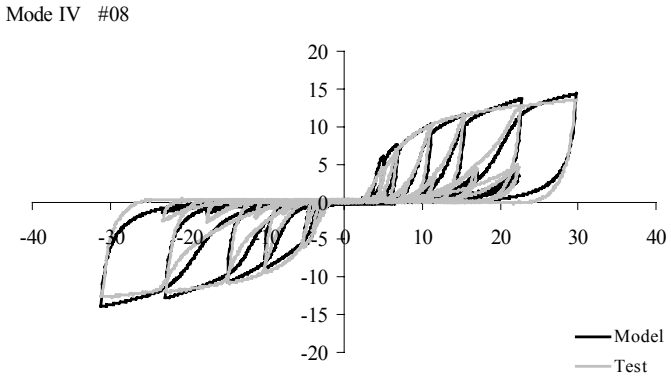


Figure 6.24: Estimated parameters and corresponding model fit of Specimen #8, Mode IV

Parameters	Values
α	0.42827
β	7.512633
ω	1.631748
ζ	0.975224
n	2.60889
ψ_0	0.293016
δ_ψ	0.003863
δ_ν	0.12301
ξ	0.021955
γ	-2.92135
δ_η	0.170246
yield mode	IV
specimen #	9
SSE / dp	0.272

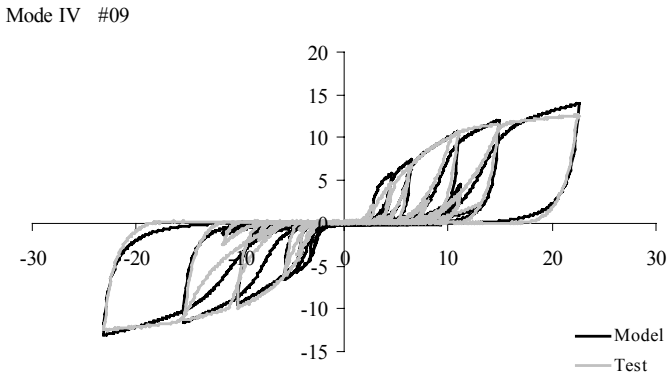


Figure 6.25: Estimated parameters and corresponding model fit of Specimen #9, Mode IV

Parameters	Values
α	0.584155
β	7.42014
ω	1.880311
ζ	0.966404
n	3.47536
ψ_0	0.570648
δ_ψ	0.003383
δ_ν	0.146222
ξ	0.034133
γ	-2.84259
δ_η	0.100281
yield mode	IV
specimen #	10
SSE / dp	0.372

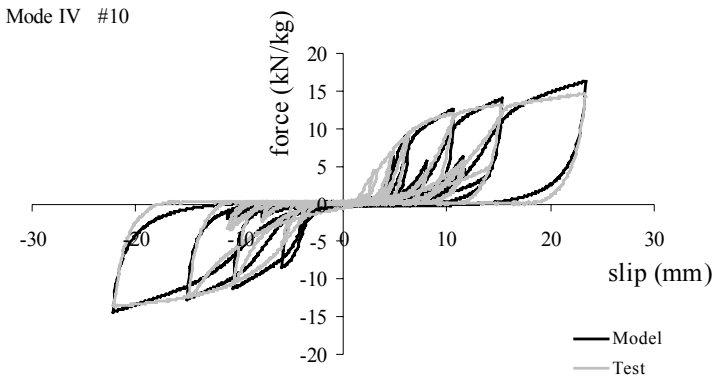


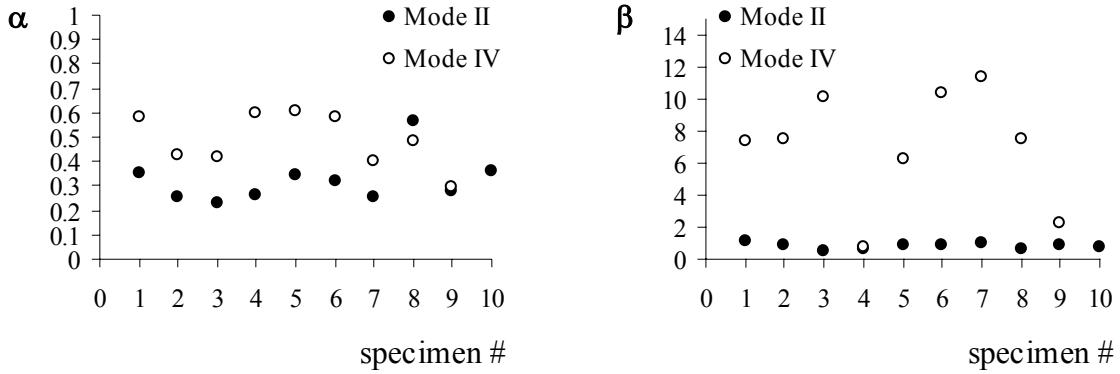
Figure 6.26: Estimated parameters and corresponding model fit of Specimen #10, Mode IV

6.6.1 Mode II Versus Mode IV

Estimated parameters for each specimen and yield mode were plotted and subjected to a simple t-test (Figures 6.27 – 6.32). For convenience, parameters were plotted versus the same specimen numbers. Although it may appear differently from the graphs, specimen numbers of each yield mode have no relation. That is, Specimens 1 through 10 yielding in Mode II are completely different from Specimens 1 through 10 of Mode IV. Several observations can be made from the graphs.

The hysteresis model describing Yield Mode II contains different parameters in comparison to the model describing Mode IV. All parameters proved significantly different when subjected to a two-tailed standard t-test (significance level: 0.05). More specifically, Mode IV hysteresis is more linear than Mode II because α is higher. Mode IV has a higher non-linear pseudo-natural frequency, ω, because the system is stiffer, a lower level of pinching, ζ and slack growth, δ_ψ, because less wood is damaged due to the plastic hinges that develop in the bolt. Mode IV also has a higher damping ration, ξ, because more energy is dissipated attributed to plastic hinges, strain hardening of the bolt, and larger members. While the two shape parameters, β and γ for each yield mode are significantly different, their ratios (β/γ) are not. Both parameters are larger in magnitude for Mode IV to account for the higher ductility of the joint. The same reason explains the higher degradation if the bolt yields in Mode IV. It is not hard to see why the shape parameter n is higher for Mode IV considering the sharper transitions of the loops. The higher slack variable, ψ_0, estimated for the Mode IV hysteresis is somewhat an artifact. Slack was not actually higher, but to better fit the loops at large displacements, the GA selected a larger

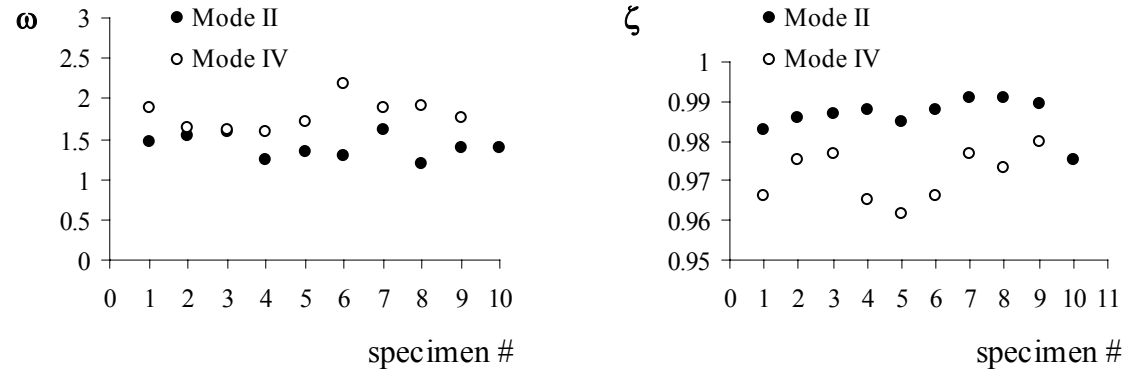
slack variable. The variable reduced substantially when only the initial portion of the test was fit but accuracy of fit at larger displacements was unsatisfactory (Figure 6.7).



α	*Mode II*	*Mode IV*
Mean	0.322	0.488
Variance	0.0093	0.0120
	t-test (two-tail)	
p value	0.0030	

β	*Mode II*	*Mode IV*
Mean	0.812	7.064
Variance	0.041	12.885
	t-test (two-tail)	
p value	0.0008	

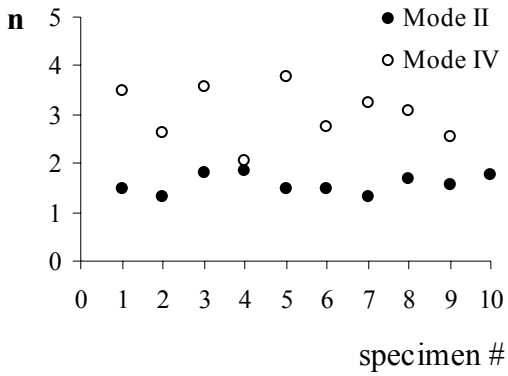
Figure 6.27: Parameters α and β plotted per specimen and yield mode. Parameters for each mode are significantly different according to a two-tailed t-test.

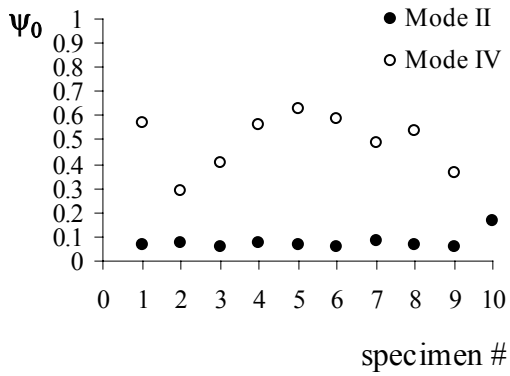


ω	*Mode II*	*Mode IV*
Mean	1.405	1.796
Variance	0.021	0.035
	t-test (two-tail)	
p value	0.0001	

ζ	*Mode II*	*Mode IV*
Mean	0.986	0.971
Variance	0.000022	0.000042
	t-test (two-tail)	
p value	0.0001	

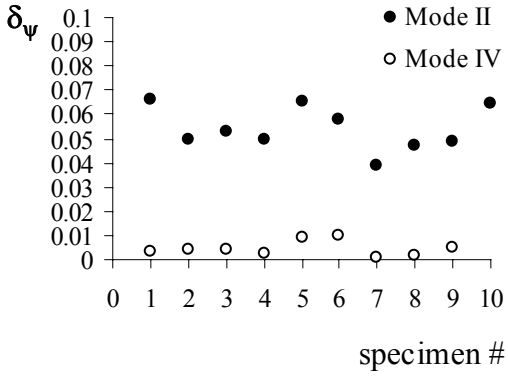
Figure 6.28: Parameters ω and ζ plotted per specimen and yield mode. Parameters for each mode are significantly different according to a two-tailed t-test.





n	*Mode II*	*Mode IV*
Mean	1.574	3.008
Variance	0.036	0.324
t-test (two-tail)		
p value	0.00003	

ψ0	*Mode II*	*Mode IV*
Mean	0.077	0.492
Variance	0.0011	0.0129
t-test (two-tail)		
p value	0.000002	



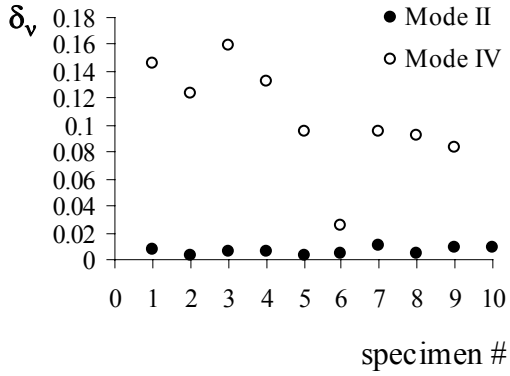
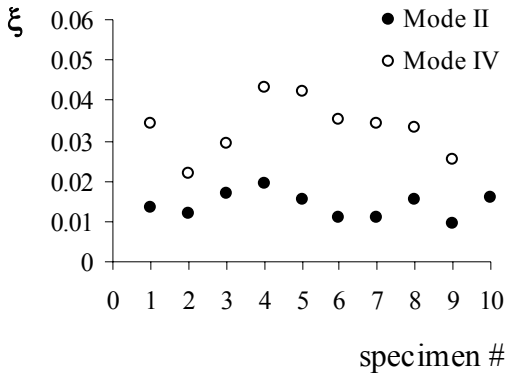


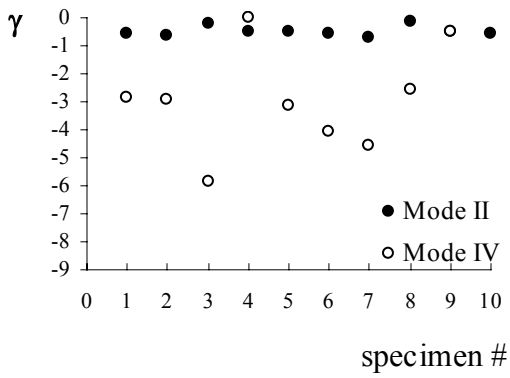
Figure 6.29: Parameters *n* and ψ_0 plotted per specimen and yield mode. Parameters for each mode are significantly different according to a two-tailed t-test.

δ_ψ	*Mode II*	*Mode IV*
Mean	0.0541	0.0044
Variance	0.000080	0.000010
t-test (two-tail)		
p value	0.000000	

δ_ν	*Mode II*	*Mode IV*
Mean	0.0062	0.1058
Variance	0.0000058	0.001589
t-test (two-tail)		
p value	0.00007	

Figure 6.30: Parameters δ_ψ and δ_ν plotted per specimen and yield mode. Parameters for each mode are significantly different according to a two-tailed t-test.

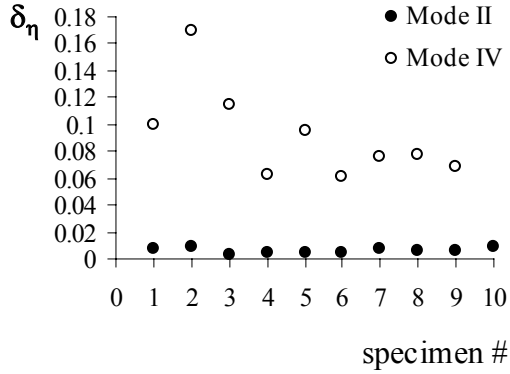




ξ	*Mode II*	*Mode IV*
Mean	0.0139	0.0332
Variance	0.000010	0.000048
	t-test (two-tail)	
p value	0.000010	

γ	*Mode II*	*Mode IV*
Mean	-0.504	-2.937
Variance	0.029	3.382
	t-test (two-tail)	
p value	0.0042	

Figure 6.31: Parameters ξ and γ plotted per specimen and yield mode. Parameters for each mode are significantly different according to a two-tailed t-test.



δ_η	*Mode II*	*Mode IV*
Mean	0.0061	0.0918
Variance	0.0000045	0.0011953
	t-test (two-tail)	
p value	0.00007	

Figure 6.32: Parameter δ_η plotted per specimen and yield mode. Parameters for each mode are significantly different according to a two-tailed t-test.

6.6.2 Parameter Correlation Mode II

Only linear correlations are presented. Scatter plots of parameters showing no significant linear relation did not indicate any other type (non-linear) of correlation. Significant linear correlations were found for a considerable number of parameters estimated for joints yielding in Mode II. This is an important finding since, with the estimated regression equations, the number of parameters that needs to be identified can effectively be reduced to only 4 instead of 11. If additional research confirms the correlations for other joint geometries and wood species, the sizeable reduction in solution space would substantially speed up the parameter estimation process for Mode II – type joints in the future.

The p-value shown in Figures 6.33 through 6.40 is a measure for the significance of the regression performed, based on an F-test testing the null hypothesis that the regression coefficient is zero. If the p-value is significant ($\alpha = 0.05$), the regression coefficient is not zero and it can be concluded that variation in the independent variable contributes to variation in the dependent variable. General inferences from Figures 6.33 through 6.40 are that linearity (α) increases with increasing capacity of the joint, but decreases with increasing pseudo-natural frequency. Loop size, which is increased by increasing n in combination with a decreasing β and increasing γ, is directly related to damping (ξ). And as expected, a relatively strong negative correlation exists between the two hysteresis shape parameters β and γ, which was partly stipulated by boundary conditions.

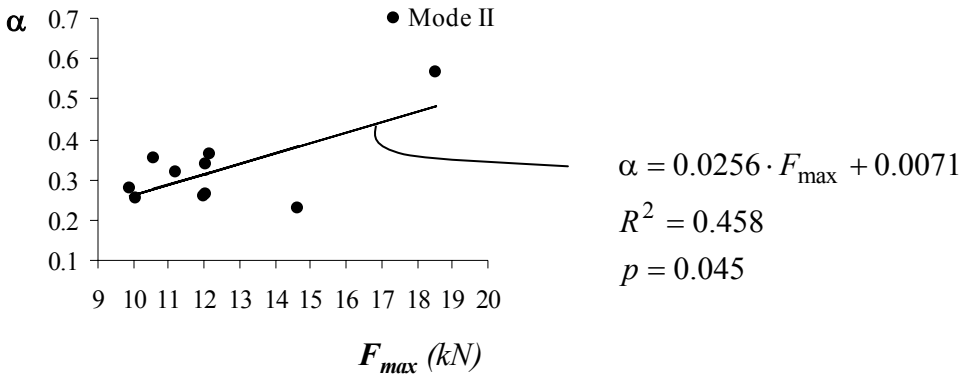


Figure 6.33: Weak but significant linear and positive correlation between α and capacity of joints yielding in Mode II (significance level: $p = 0.05$). Regression equation and percentage of variability explained by the model (R^2) are also shown.

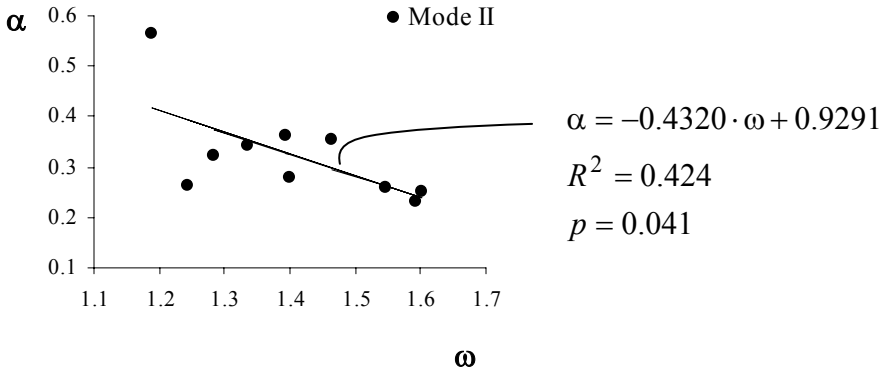


Figure 6.34: Weak but significant linear and negative correlation between α and pseudo-natural frequency ω of joints yielding in Mode II (significance level: $p = 0.05$).

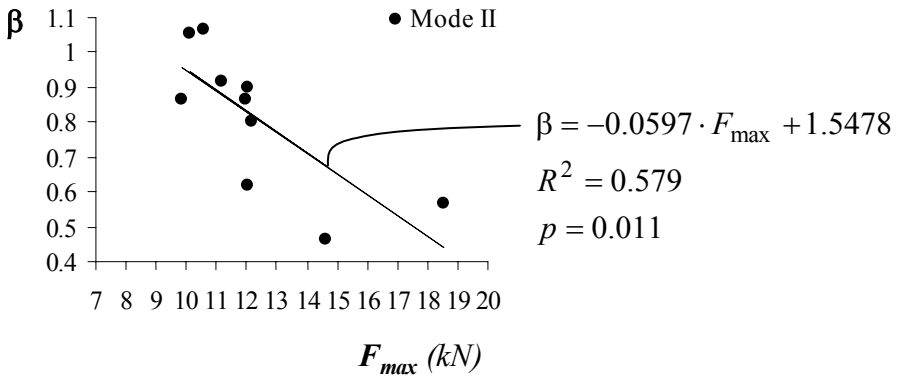


Figure 6.35: Significant linear and negative correlation between the shape parameter β and capacity of joints yielding in Mode II.

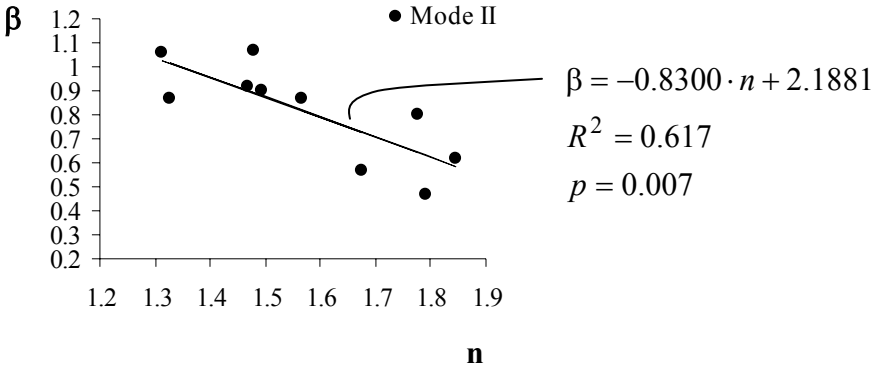


Figure 6.36: Relatively strong linear and negative correlation between the two shape parameters β and n for Mode II joints.

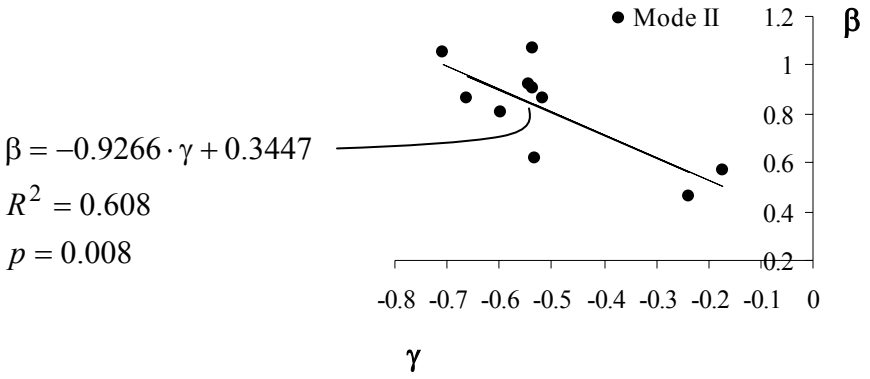


Figure 6.37: Significant linear and negative correlation between shape parameters β and γ for Mode II in part due to boundary conditions.

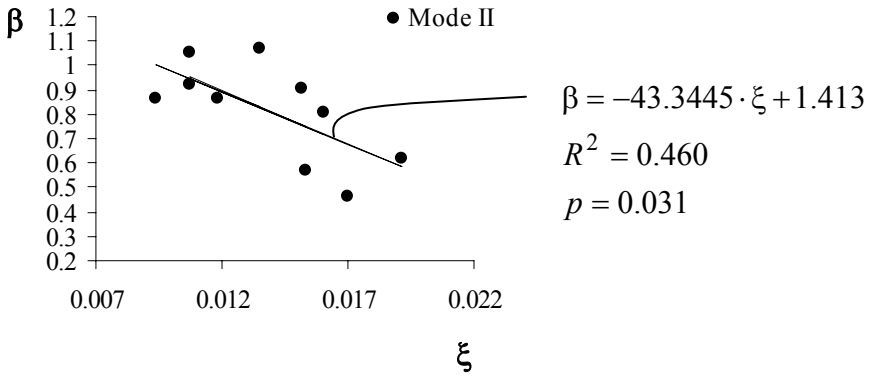


Figure 6.38: Relatively weak linear and negative correlation between the shape parameter β and the damping ratio ξ for Mode II.

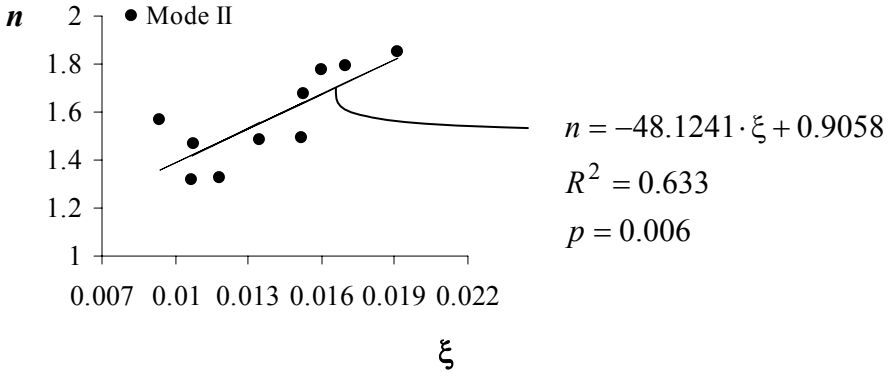


Figure 6.39: Relatively strong linear and positive correlation between the shape parameter n and the damping ratio ξ for Mode II.

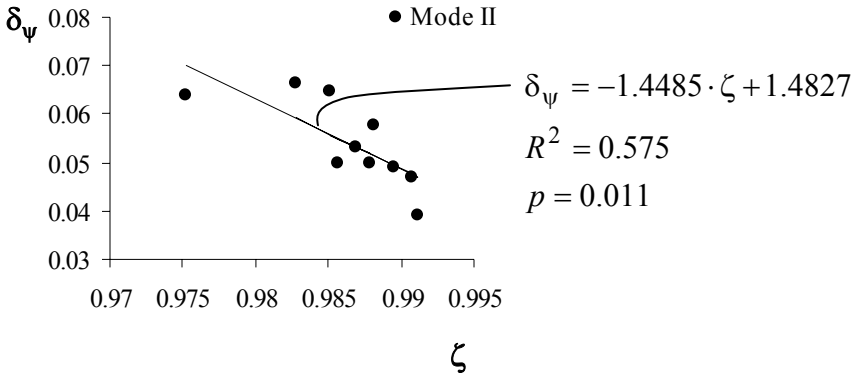


Figure 6.40: Significant linear and negative correlation observed between the slack growth parameter δ_ψ and the pinching parameter ζ.

6.6.3 Parameter Correlation Mode IV

Far fewer significant linear correlations could be identified for parameters of the modified hysteresis model describing joints yielding in Mode IV. The reason may lie in the increased variability of some of the parameters estimated, which may be attributed to the inferior lumber used. In addition, the mechanics of the yielding process are completely different for Modes IV and II. In case of the latter, no bolt bending and strain hardening are involved. However, the correlations found for Mode IV are similar to and do not contradict the linear relations acquired for Mode II (Figures 6.41 through 6.43). As with Mode II, linearity increases with increasing joint capacity. In addition, like Mode II, there is a strong linear relation between the two shape parameters β and γ. An additional correlation was found between shape parameters n and γ. This relation was not significant for Mode II specimens.

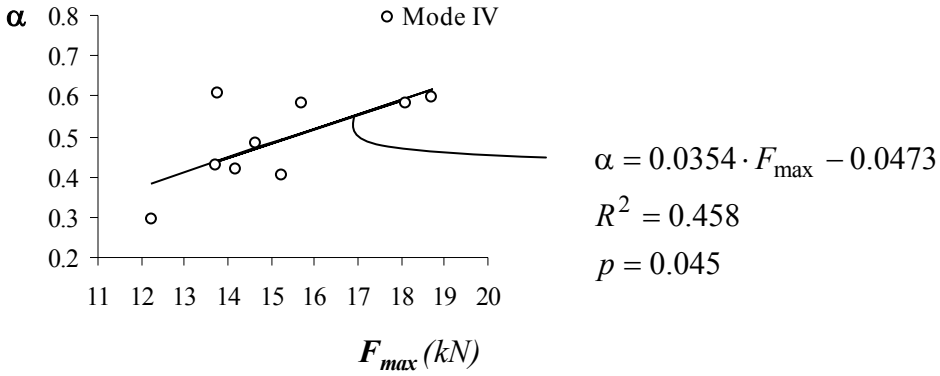


Figure 6.41: Weak but significant linear and positive correlation between α and capacity of joints yielding in Mode IV (significance level: $p = 0.05$). Regression equation and percentage of variability explained by the model (R^2) are also shown.

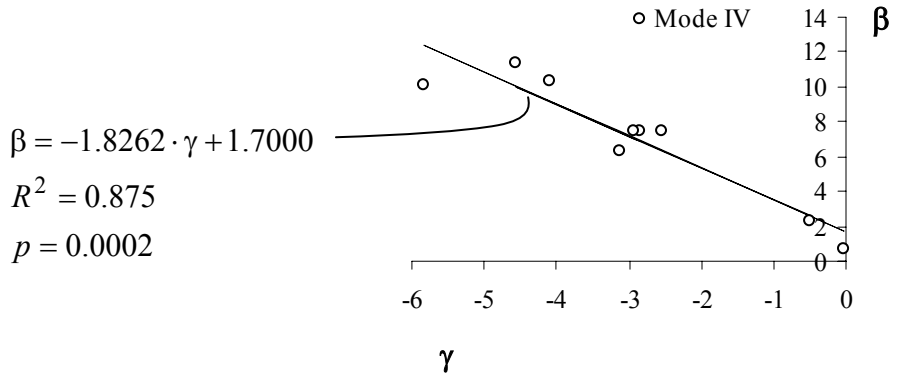


Figure 6.42: Strong linear and negative correlation between shape parameters β and γ for Mode IV in part due to boundary conditions.

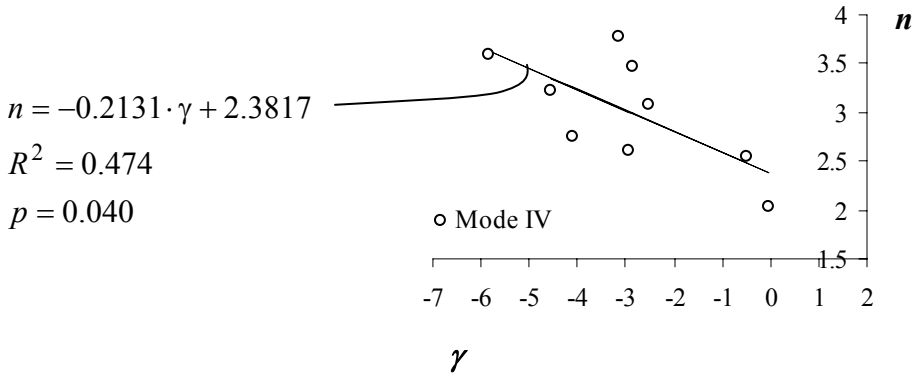


Figure 6.43: Relatively weak linear and negative correlation between the two shape parameters n and γ.