Multigrid Accelerated Cellular Automata for Structural Design Optimization: A 1-D Implementation

Sunwook Kim

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
Aerospace and Ocean Engineering

APPROVED:

Zafer Gürdal, Chair

Mark Jones

Rakesh K. Kapania

June, 8, 2004
Blacksburg, Virginia

Keywords: Cellular Automata, Multigrid, Structural Optimization

Multigrid Accelerated Cellular Automata for Structural Design Optimization: A 1-D Implementation

Sunwook Kim

(ABSTRACT)

Multigrid acceleration is typically used for the iterative solution of partial differential equations in physics and engineering. A typical multigrid implementation uses a base discretization method, such as finite elements or finite differences, and a set of successively coarser grids that is used for accelerating the convergence of the iterative solution on the base grid. The presented thesis extends the use of multigrid acceleration to the design optimization of a sample structural system and demonstrates it within the context of the recently introduced Cellular Automata paradigm for design optimization. Within the design context, the multigrid scheme is not only used for accelerating the analysis iterations, but is also used to help refine the design across multiple grid levels to accelerate the design convergence. A comparison of computational efficiencies achieved by different multigrid implementations, including the multigrid accelerated nested design iteration scheme, is presented. The method is described in its generic form which can be applicable not only to the Cellular Automata paradigm but also to more general finite element analysis based design schemes as well.

Acknowledgments

First and foremost, I'd like to express appreciation to Dr. Gürdal, my advisor for his constant guidance and support over the last two years. Sincere thanks also go to Dr. Jones, Dr. J. Sobieski, and my colleague, Mostafa Abdalla for valuable advices and discussions through all phases of the research and I appreciate Dr. Kapania for serving as a committee member and his help for finalizing this thesis.

There are friends to whom I want to extend thanks. They include Omprakash, Hyun-sik, So-yeon, and Imsoo.

Lastly, I extend my appreciation to my beloved family for their support and love.

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

Cellular Automata (CA) paradigm has recently been successfully applied to structural analysis and design [4, 7]. CA uses a lattice of regularly spaced cells to model a physical phenomenon. Each cell contains all the information needed to update its own state. This information includes both field variables (e.g., displacements or stresses) as well as local design variables (e.g., local cross-sectional area or thickness). The only external information to the cell comes directly from adjacent cells that form the cell neighborhood along with the cell itself. By limiting computations to local neighborhoods and using identical update rules for cell variables in the entire lattice, the CA paradigm proves to be inherently parallel. Moreover, since both field and design variables can be simultaneously updated, CA allows combined analysis and design.

In earlier studies, it has been observed [4, 7] that the CA convergence rate deteriorates considerably as the lattice is refined. It has been recognized that this deterioration of the convergence rate is due to slow propagation of information across the domain; that is, as the cell spacing is reduced, more iterations are needed to update all the cells in the domain.

In this paper, we will present an implementation of Multigrid (MG) acceleration of CA algorithms. Multigrid is a traditional approach for accelerating the convergence of relaxation methods relying on local stencils [13, 14] through accelerating information transfer across the domain. The application of multigrid acceleration to the solution of field problems is a well-researched area [13, 14]. In this work, a multigrid algorithm is proposed that incorporates both analysis and design.

1.2 Objective and scope of the research

Rapid advances in high performance computing architecture and reconfigurable hardware raise a new standard for optimization algorithms. They are now required well-adapted to machine. In this regards, the CA is a promising candidate due to its inherent parallelism and simplicity in implementation. The main objective of this study is to investigate the applicability of the MG accelerated CA for distributed parameter optimization problems on hardware, which is FPGA (Field Programmable Gate Arrays). FPGA has drawn much attention because of its flexibility and potential usage. The secondary purpose is to develop the CA for structural optimization to be more favored in hardware. For this purpose, the detailed convergence studies will be carried out. Acceleration methods will be sought based on the obtained convergence characteristics of CA and also parametric studies are completed for the proposed algorithms.

The shape design of Euler-Bernoulli beams for minimum compliance is selected for this demonstration for two reasons. First, the implementation of CA and MG is fairly straightforward for this one-dimensional problem. Second, exact solutions can be easily obtained and provide a systematic way to carry out a convergence study. The proposed multigrid accelerated CA algorithm shares the same motivation as previous studies using multigrid in topology optimization and the MG approach. However, the proposed CA approach is distinguished by its emphasis on localized computations for both analysis and design, and the

use of rigorous optimality criteria. In addition, we present detailed convergence studies and demonstrate the ability of the MG acceleration to achieve optimal computational complexity for combined analysis and design. The MG approach can be further accelerated by applying the design rule in successively refined grids. The design process on the finer grid is initialized using the converged solution (both field and design variables) from the coarse grid, which is usually referred to as full multigrid (FMG).

1.3 Outline of the thesis

Chapter 1: Chapter 1 presents the introduction and literature review.

Chapter 2: The optimality criterion of the minimum compliance is discussed. The 1-D cell neighborhood is defined for Euler-Bernoulli beam and the analysis update rule for CA is constructed based on the principle of minimum potential energy. The analysis update rule is, moreover, recast into residual correction form. The optimality criterion is defined as the design update rule and the combined analysis and design algorithm for CA is presented.

Chapter 3: The principle of the multigrid algorithm is given in simple terms and the transfer operators for the multigrid algorithm are defined. The design algorithm for CA is modified for the MG acceleration and also the design algorithm for the FMG acceleration is presented in the form of a pseudo code.

Chapter 4: Numerical results and convergence studies are presented for MG/FMG accelerated CA and the standard CA algorithms. Parametric studies for the FMG algorithms are additionally included.

Chapter 5: The study is summarized and conclusions are provided in this chapter.

1.4 Literature review

The review covered Cellular Automata approaches for structural optimization and the multigrid methods in the topology optimization.

The general concepts of Cellular Automata are considered introduced by Ulam [1] and von Neumann [2] in order to model real-life systems. Wolfam [3] introduced the CA approach as a powerful tool for the analysis and design of complex systems.

Gürdal and Tatting [4, 5] introduced a novel CA scheme in which the analysis and design are updated simultaneously, and showed the proposed CA scheme is fully capable of modelling linear, nonlinear truss structures, and also 2-D continuum structures. Their works and references therein give detailed illustrations on CA in structural optimization.

The CA paradigm was successfully extended to solve the design of continuum structures for a specific eigenvalue by Abdalla and Gürdal [6] and the same authors [7] developed the local design update rule based on the optimality criteria in the light of the method of calculus of variations.

Samy, *et al* [8] applied the CA to the topology design of trusses exhibiting geometric and material nonlinearities. The design update rule was derived based on the minimum compliance design principle.

Shahriar and Guŕdal [9] extended the CA paradigm to the design of fiber reinforced laminated composite plates for in-plane properties. The authors demonstrated that the CA is well suited on parallel machines while taking manufacturing considerations into account.

Hajela and Kim [10] and Kita and Toyota [11] applied the CA approach to design elastic structures in which the CA based design update rules are implemented in the design phase, while the analysis of the structures is carried out using traditional techniques.

Multigrid methods have been developed for elliptic partial differential equations, integral equation, etc in the field of physics and engineering. Briggs [12] published the introductory

tutorial for MG in which he deliberated to describe MG methods in simple terms. Hackbush [13] and Wesseling [14] presented in-depth illustrations and applications of the multigrid algorithm. However, a few attempts were made to extend the idea of the multigrid to enhance the optimization methods.

The first application of multigrid methods to large-scale structural optimization problems is the work of Maar and Schulz [15], where the multigrid method is incorporated to accelerate the convergence of nonlinear interior point optimization methods. The effectiveness of this method was demonstrated through two-dimensional topology design example. Kim and Yoon [16] introduced a new approach to topology optimization, multi-resolution multi-scale topology optimization (MTOP). They transformed density design variables of a selected 2-D problem into a multi-scale of design domains in the wavelet space and used a design captured at a coarse resolution for the design at a finer resolution. As a result, the rate of convergence was enhanced, and numerical instabilities were significantly suppressed. Kwon *et al* [17] incorporated the multigrid method into MTOP to improve the efficiency of the numerical computations. The study showed the potential advantage of implementation of the multigrid method in the multiscale topology optimization.

Chapter 2

Cellular Automata (CA) Implementation

The constrained design optimization problem is posed for an Euler-Bernoulli beam and it is solved by the method of Lagrange multipliers in calculus of variations and an optimality condition is obtained in the close form. The optimality condition will be employed as the design update rule in the CA formulation. An analysis update rule for CA will be first derived, applying the principle of minimum potential energy and will be re-constructed in the form of residual correction. With defined analysis and design update rule, a nested analysis and design update algorithm for CA is presented.

2.1 Minimum Compliance Design

The governing equation for an Euler-Bernoulli beam is,

$$\frac{d^2}{dx^2}\left(EI(x)\frac{d^2w}{dx^2}\right) = p(x), \qquad 0 \leq x \leq L, \tag{2.1}$$

where $w(x)$ is the transverse deflection, E is the modulus of elasticity, $I(x)$ is the principal moment of inertia, and $p(x)$ is the distributed loading.

The formulation of the design optimization problem is posed as a minimum compliance design. For simplicity, we assume that the cross section is rectangular with a constant width (assumed unity), while the height of the cross section is the design variable. This minimum compliance problem is formulated as,

$$\min_{h(x)} \frac{1}{2} \int_0^L EI(x) w''^2 dx, \tag{2.2}$$

subject to

$$V = \int_0^L a(x) dx \leq V_o.$$

where the width of a beam is chosen to be a unity and $a(x)$ is the height of a beam. Using the definition of the bending moment,

$$M = EI(x) \frac{d^2 w}{dx^2}, \tag{2.3}$$

the Lagrangian is written as,

$$\mathcal{L} = \frac{1}{2} \int_0^L \left[\frac{M^2}{EI(x)} + \lambda(a - a_o) \right] dx, \tag{2.4}$$

where a is the cross sectional area of the beam, M is the bending moment and λ is the Lagrange multiplier associated with the volume constraint. Stationary condition for the Lagrangian is the first variation of Equation (2.4) about the design variable, a, must be zero. Since the cross section of the beam is assumed to be rectangular, the bending stiffness, EI, is expressed as $a^3/12$. Then, the condition that the variation of the Lagrangian, $\delta\mathcal{L}$, is zero can be written as,

$$\delta\mathcal{L} = \int_0^L \left[-\frac{36M^2}{a^4} + \lambda \right] \delta a dx = 0, \tag{2.5}$$

and since δa is arbitrary, the coefficient of δa in (2.5) should be set to be zero, that is, the lagrange multiplier is determined by substituting this equation into the constraint. Accordingly, the optimality condition is obtained that yields the following update rule,

$$a(x) = \alpha \sqrt{|M(x)|}, \tag{2.6}$$

where

$$\alpha = \sqrt{\frac{6}{\sqrt{\lambda}}} \quad \text{and} \quad \sqrt{\lambda} = \frac{6|M(x)|}{a_o^2}.$$

Here, it can be easily shown that Equation (2.6) is essentially equivalent to the fully stressed design using the flexural formula. That is, the required cross section is determined by dividing the maximum bending moment by the allowable stress. Since the maximum bending moment occurs at the farthest point from the neutral axis, the required height is for the rectangular cross section when the width is a unity,

$$a(x) = \sqrt{\frac{6}{\sigma_{allow}}}\sqrt{|M(x)|}$$

where σ_{allow} is an allowable stress, and a quick comparison with the design update rule tells that the square root of the lagrange multiplier, λ, corresponds to the allowable stress. The fully stressed design is a common practice to design a beam by using this relation when the bending moment is only considered in the design.

In addition, a lower bound constraint on the beam height is set in order to avoid numerical problems caused by zero bending moment.

2.2 Cellular Automata (CA) Formulation

A Cellular Automata (CA) is generally defined to be a discrete system where the problem space under consideration is represented by a regular lattice. A CA space can be one-, two-, and three-dimensional and is not limited to be regular. In the given CA space, the neighborhood structure becomes essential in the sense that it directly corresponds to the nature of a physical model that is being solved, and CA update rules are uniformly applied to a cell and its neighboring cells. That is, CA update rules are derived by defining a relationship between a cell and its neighboring cells, and accordingly, the update rules become uniformly applicable to every cell in the lattice. This uniformity in applying update rules render the structure of CA algorithm simple. In light of detailed derivations, we will seek an analysis update rule by the static equilibrium of an Euler-Bernoulli beam and a design update rule by the optimality criterion of the minimum compliance.

The cell definition for structural problems typically includes field variables, applied loads, and design variables. A one-dimensional cell neighborhood and the field variables associated with the cells for an Euler-Bernoulli beam problem are shown in Figure A.1. The neighborhood is composed of a center cell (C) with two neighbors, the left (L) and the right (R) cells. Each cell has two kinematic variables, w and θ, where w is the transverse deflection and θ is the rotation, an externally acting force, F, and moment, M. The state of a cell can be represented as

$$\phi_C = \{\{w, \theta\}, \{F, M\}, \{a\}\}, \tag{2.7}$$

where the subscript, C, denotes a particular cell in the domain.

2.2.1 Analysis Update Rule

An analysis update rule is derived by approximating the equilibrium equations using the minimization of the total potential energy inside the control volume in Figure A.1. The domain of structure is presented in the vertex-centered grid. The displacement field for the Euler-Bernoulli beam between two cells is approximated as,

$$\begin{aligned} w(\xi) &= w_i H_1(\xi) + h\theta_i H_2(\xi) + w_j H_3(\xi) + h\theta_j H_4(\xi) \\ &= [H_1(\xi)\ H_2(\xi)] \cdot \boldsymbol{u}_i + [H_3(\xi)\ H_4(\xi)] \cdot \boldsymbol{u}_j, \end{aligned} \tag{2.8}$$

where ξ is a non-dimensional variable defined by,

$$\xi = \frac{x}{h}, \quad 0 \leq \xi \leq 1 \tag{2.9}$$

H_i are Hermite cubic interpolation functions, $\boldsymbol{u}_i$ is the field variable vector of the ith cell, and h is the cell spacing. The strain energy of the beam inside the control volume is given by,

$$U = \frac{1}{2} \int_{CV} EI(x) \left(\frac{d^2 w}{dx^2} \right)^2 dx. \tag{2.10}$$

The integration is evaluated by substituting the displacement field (2.8) into Equation (2.10), and using an average value of the flexural rigidity, EI, for the integration over the left and

right portions, similar to finite volume discretization. The strain energy is re-written as follows,

$$\begin{aligned} U &= \frac{EI_L^*}{2h^3}\int_0^1 w(\xi)_L''^2 d\xi + \frac{EI_R^*}{2h^3}\int_0^1 w(\xi)_R''^2 d\xi \\ &= \frac{EI_L^*}{2h^3}\int_0^1 ([-6(1-2\xi)\ -2(3\xi-2)]\cdot \boldsymbol{u}_L + [6(1-2\xi)\ -2(3\xi-1)]\cdot \boldsymbol{u}_C)^2 d\xi + \\ &\quad \frac{EI_R^*}{2h^3}\int_0^1 ([-6(1-2\xi)\ -2(3\xi-2)]\cdot \boldsymbol{u}_C + [6(1-2\xi)\ -2(3\xi-1)]\cdot \boldsymbol{u}_R)^2 d\xi, \quad (2.11) \end{aligned}$$

where,

$$EI_L^* = \frac{1}{2}(EI_L + EI_C), \text{ and } EI_R^* = \frac{1}{2}(EI_C + EI_R)$$

The equilibrium equations are obtained by minimizing the total potential energy with respect to the displacements of the center cell, $\boldsymbol{u}_C$, and have the form,

$$\begin{bmatrix} \boldsymbol{k}_{11} & \boldsymbol{k}_{12} & \boldsymbol{k}_{13} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_L \\ \boldsymbol{u}_C \\ \boldsymbol{u}_R \end{bmatrix} = \boldsymbol{f}_C, \quad \boldsymbol{f}_C = [F|M]^T, \tag{2.12}$$

where $\boldsymbol{u}_L$ and $\boldsymbol{u}_R$ are the displacement vector for the left cell and the right cell, respectively, $\boldsymbol{f}_C$ is the vector of concentrated forces and moments; and $\boldsymbol{k}_{ij}$ are 2×2 stiffness matrices given by

$$\boldsymbol{k}_{11} = \frac{\partial^2 U}{\partial \boldsymbol{u}_C \boldsymbol{u}_C}, \quad \boldsymbol{k}_{12} = \frac{\partial^2 U}{\partial \boldsymbol{u}_C \boldsymbol{u}_L}, \quad \boldsymbol{k}_{13} = \frac{\partial^2 U}{\partial \boldsymbol{u}_C \boldsymbol{u}_R}, \tag{2.13}$$

Each variable of the center cell is solved for in terms of its neighbors from Equation (2.12) to form the relation,

$$\boldsymbol{k}_{11} \cdot \boldsymbol{u}_C = \boldsymbol{f}_C + \boldsymbol{f}_g, \tag{2.14}$$

where $\boldsymbol{f}_g$ accounts for the influence of neighboring cells on the center cell, and is given by

$$\boldsymbol{f}_g = -\boldsymbol{k}_{12} \cdot \boldsymbol{u}_L - \boldsymbol{k}_{13} \cdot \boldsymbol{u}_R. \tag{2.15}$$

Thus, Equation (2.14), which is the local equilibrium update rules for a beam, can be shown to be

$$\frac{1}{h^3}\begin{bmatrix} 12(EI_L^* + EI_R^*) & -6(EI_L^* - EI_R^*) \\ -6(EI_L^* - EI_R^*) & 4(EI_L^* + EI_R^*) \end{bmatrix} \begin{bmatrix} w_C \\ h\theta_C \end{bmatrix} = \begin{bmatrix} F_C \\ \dfrac{M_C}{h} \end{bmatrix} + \boldsymbol{f}_g, \tag{2.16}$$

where $\boldsymbol{f}_g$ can be simplified to

$$\boldsymbol{f}_g = -\frac{EI_L^*}{h^3}\begin{bmatrix}-12 & -6\\ 6 & 2\end{bmatrix}\begin{bmatrix}w_L\\ h\theta_L\end{bmatrix} - \frac{EI_R^*}{h^3}\begin{bmatrix}-12 & 6\\ -6 & 2\end{bmatrix}\begin{bmatrix}w_R\\ h\theta_R\end{bmatrix}. \tag{2.17}$$

The local analysis update rule is derived over the domain divided using the vertex-centered discretization scheme. However, an alternate discretization scheme can be used to divide the domain into a set of elements and the analysis update rule can be defined in a similar procedure that we discussed above. In the earlier stage of the research, the cell-centered grid was employed, and it rendered a local analysis update rule complicated, and moreover, the implementation of the multigrid became too cumbersome. The derivation of a local analysis update rule for the cell-centered grid is briefly discussed in Section A.1.

2.2.2 Residual-Correction Form

The equilibrium update rule is recast in the residual-correction form, which is sometimes called iterative refinement form. This form of the update rule will be useful later in developing the multigrid acceleration scheme. In the residual-correction approach, the residual forces at the cell are first computed, and then used to obtain the needed correction of cell displacements to restore equilibrium. It is performed for successive approximations until required accuracy is obtained. The residual, $\mathbf{r}$, is computed as

$$\boldsymbol{r} = \boldsymbol{f_g} + \boldsymbol{f_C} - \boldsymbol{k_{11}} \cdot \boldsymbol{u_C}, \tag{2.18}$$

the correction, $\mathbf{e}$, is calculated from,

$$\boldsymbol{e} = \boldsymbol{k_{11}}^{-1} \cdot \boldsymbol{r}, \tag{2.19}$$

and added back into the previous solution

$$\boldsymbol{u_C}^{k+1} = \boldsymbol{u_C}^{k} + \boldsymbol{e}. \tag{2.20}$$

In the residual-correction form the successive computation of the three equations above is equivalent to the equilibrium update rule.

There are two generally known numerical advantages of the residual-correction method. First, it can possibly perform effectively for correcting ill-conditioned systems[18], e.g. the condition number of a system matrix becomes too large. Second, the correction in the method can be computed using a low-precision arithmetic without the loss of accuracy in the overall solutions. The second feature becomes significantly advantageous to CA implementation on hardware capable of variable precision computations such as Field Programmable Arrays (FPGAs). The residual can be calculated in higher precision, while the correction can be calculated at a low-precision without compromising the convergence rate [19]. In addition to this thesis, a graduate student in Electrical Computer Engineering has worked on hardware implementation of the proposed algorithm, where the authors are studying the feasibility of designing a configurable computing hardware for a massively parallel implementation of CA using FPGAs [20].

2.2.3 Design Update Rule

As established in Section 2.1, the beam height at any given point will depend on the bending moment as in Equation (2.6). The bending moment at a cell is recovered by twice differentiating the interpolation functions as shown in Equation (2.3), and simplifies to

$$M_C^- = \frac{EI_L^*}{h^2}\left([6\ 2]\begin{bmatrix} w_L \\ h\theta_L \end{bmatrix} + [-6\ 4]\begin{bmatrix} w_C \\ h\theta_C \end{bmatrix}\right), \tag{2.21}$$

$$M_C^+ = -\frac{EI_R^*}{h^2}\left([6\ 4]\begin{bmatrix} w_C \\ h\theta_C \end{bmatrix} + [-6\ 2]\begin{bmatrix} w_R \\ h\theta_R \end{bmatrix}\right). \tag{2.22}$$

where M_C^- and M_C^+ are the bending moments to the left and to the right of the cell, respectively. M_C^- and M_C^+ can be different if a concentrated moment is applied at a cell. The design at each cell is updated by the average of the absolute value of M_C^- and M_C^+ in Equation 2.6.

2.3 CA Design Algorithm

In a nested analysis and design algorithm a design step is performed after an analysis step until a given convergence criteria are satisfied. CA design algorithm adopts a modified version of this scheme. The flowchart of the design algorithm is presented in Figure A.2. The innermost loop indicates that the field variables of all N cells of the grid are updated for a certain number of times, N_a, so that a cell reaches a reasonable equilibrium state. This is done to ensure that the calculated cell bending moment is realistic and ensures the convergence of the design iteration. As will be discussed in the result section, there exists the optimal number of analysis updates, N_a^*. After the analysis update is repeated for N_a times, the design is then updated for all the cells in the grid. A stopping criterion is checked to terminate the design update scheme. A simple stopping criterion will be likely to be adequate such as any norm of the field variable correction, any norm of the design variable correction, or any norm of the residual. However, the primary variables in the system of equations are the field variables, and as discussed in the previous section, the design update rule uses moments recovered from the field variables. Besides, it seems reasonable to consider that checking the norm of the field variable correction is equivalent to checking the norm of the residual in the following reasoning.

For simplicity, Equation (2.14) is represented in the simple form as

$$A\mathbf{x} = \mathbf{b}, \tag{2.23}$$

where A corresponds to $\boldsymbol{k}_{11}$, $\mathbf{x}$ corresponds to the field variable vector, $\boldsymbol{u}_C$, and $\mathbf{b}$ does to $\boldsymbol{f}_C + \boldsymbol{f}_g$. Suppose that $\tilde{\mathbf{x}}$ is an approximate solution of Equation (2.23), A is non-singular, and $\mathbf{r}$ is the residual for the approximate solution, $\tilde{\mathbf{x}}$, then for any norm

$$||\mathbf{x} - \tilde{\mathbf{x}}|| \leq ||\mathbf{r}|| \cdot ||A^{-1}|| \tag{2.24}$$

That is, the norm of field variable correction, $||\mathbf{x} - \tilde{\mathbf{x}}||$ is bounded by the norm of the residual multiplied by a certain constant. Thus, it seems to be reasonable that the norm of the field

variable correction is related to the norm of the residual unless the system matrix, A becomes significantly ill-conditioned. More mathematically rigorous proof for error bounds is found in the reference [18]. In this manuscript, the infinity norm of the field variable correction will be employed as the stopping criterion for numerical experiments, however, the results for the different stopping criteria will be also presented for a simple comparison.

The CA algorithm presented here differs from traditional optimization methods in two significant features. First, the CA algorithm does not require a fully converged analysis (determination of displacements) before the design is updated. The analysis is converged within the design iterations. Second, the design step is directly based on the latest analysis results, and is applied locally.

Chapter 3

Full Multigrid / Multigrid Acceleration

The fundamental ideas of the Multigrid and Full Multigrid method will be described, respectively. The transfer operators (the prolongation and the restriction operators) are defined by mainly using a one-dimensional example and they are the essential components of the multigrid method. Even though the implementation of the multigrid acceleration algorithms are discussed within the context of the Cellular Automata, the derivation of the transfer operators are purely based on the structure of the discretized domain. Furthermore, the design scheme for the multigrid will follow the nested analysis and design algorithm discussed in Chapter 2 without any significant modification.

3.1 Multigrid Acceleration

In earlier studies, it was observed that the computational effort associated with CA iterations increases substantially as the number of cells is increased. The deterioration of the convergence rate of CA algorithms is due to slow propagation of information across the domain.

As the lattice is refined, the cell spacing becomes significantly less than the characteristic length of the problem. Given that cells can only communicate locally, relevant information is slowly transferred across the domain leading to the observed poor convergence.

It is generally known that fixed grid algorithms eliminates error at high spatial frequencies, but low frequency errors are damped slow [14]. A traditional solution to this problem is to use multigrid/full multigrid acceleration schemes. Even though a one-dimensional example is the focus in the paper, the acceleration algorithms can be easily extended to higher dimensional problems. Furthermore, it is a general conception that the multigrid can be efficiently implemented into parallel computing hardware. This will be potentially beneficial when the proposed multigrid accelerated CA method is implemented on FPGAs hardware in the near future.

3.1.1 Multigrid (MG) Analysis Acceleration

As the name implies, the motivation behind the multigrid is to define a hierarchy of lattices from the finest to the coarsest as shown in Figure A.3.The color of a cell in the figure is used to emphasize which status of a cell is prescribed and the dummy cells have no physical properties, which are the representation of an infinite domain [4]. In that figure, the different resolutions of grids represent a fixed-fixed beam while a force is applied to the center cell and each colored cell represents prescribed conditions and the dummy cells refers to the outside of the domain. The residual of the governing equation, (2.18), can be regarded as a superposition of Fourier components of different wavelengths. The short wave length components of the residual can be effectively eliminated using local iterative techniques, while long wavelength components show a slower decay since their elimination requires the information to travel between a large number of cells. Long wavelength components of residual become short wavelength components when the lattice is coarsened, and can thus be more effectively eliminated. The Multigrid principle is to approximate and reduce the long wavelength error of a solution on a coarse grid, while reducing the short wavelength

components on the fine grid. Consequently, all wavelength components of residual become effectively suppressed, which results in remedying a slow convergence [12, 14].

This fundamental principle of Multigrid algorithms can be applied to accelerate the convergence of CA algorithms. In implementing MG algorithm for CA, the equilibrium update rule is applied to the fine lattice for a fixed number of times,S, to smooth the residual by eliminating short wavelength components. The smoothed residual on the fine lattice is mapped to the next coarser lattice through a restriction operator. The correction for the residual is obtained on the coarse lattice. Then, the correction from the coarse lattice is mapped to the fine lattice through a prolongation operator. This idea can be applied recursively using multiple levels of lattices. For the beam problem, we use a very simple coarsening scheme in which alternate cells are dropped to form a coarser lattice. Thus, the lattice size is reduced roughly by a factor of two each time the lattice is coarsened.

The MG algorithm starts from the finest lattice and visits all coarse lattices. The order in which lattices are visited is called the multigrid schedule. According to the shape of the multigrid algorithm diagrams, algorithms are called V-, W-, or F-cycle [14]. As examples, V- and W-cycle are illustrated in Figure A.4.

3.1.2 Prolongation and Restriction Operators

The key components of the multigrid algorithm are the prolongation and the restriction operators acting between fine and coarse lattices. In particular, the prolongation operator maps corrections of a solution in a coarse lattice to a fine lattice as

$$\boldsymbol{e}^h = I_{2h}^h \boldsymbol{e}^{2h}, \tag{3.1}$$

in which the subscript $2h$ indicates a coarse grid quantity and the superscript h indicates a fine grid quantity. The restriction operator is needed for transferring the residual from a fine lattice to a coarse lattice as

$$\boldsymbol{r}^{2h} = I_h^{2h} \boldsymbol{r}^h. \tag{3.2}$$

These transfer operators are straightforwardly fitted into the residual-correction equation (2.20),

$$\begin{aligned} {\boldsymbol{u}_C}^{k+1} &= {\boldsymbol{u}_C}^{k} + e, \\ &= {\boldsymbol{u}_C}^{k} + I_{2h}^{h} \cdot k_{11}^{2h} \cdot I_{h}^{2h} \boldsymbol{r}^{h}. \end{aligned} \tag{3.3}$$

This equation implies that the multigrid method can be integrated into the existing analysis update rule without causing significant modifications.

To retain the symmetry of the multigrid formulation [12], the restriction operator is in terms of the prolongation operator as

$$I_{h}^{2h} = (I_{2h}^{h})^{T}. \tag{3.4}$$

For the prolongation operator, the correction from the coarse lattice cells is either passed to the cells directly above them or to the cells that do not appear in the coarse lattice (see figure A.5(a)). The correction of a cell on a coarse lattice is projected unchanged to a matching cell on a fine lattice as

$$\begin{bmatrix} w_{2i-1}^{h} \\ h\theta_{2i-1}^{h} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} w_{i}^{2h} \\ 2h\theta_{i}^{2h} \end{bmatrix}. \tag{3.5}$$

where, for notational simplicity, only in this section w and θ represent the components of the correction vector e. However, the corrections of a cell on a fine lattice, but not on a coarse lattice are approximated using shape functions on a coarse lattice, and are shown in figure A.5(b). The matrix representation is

$$\begin{bmatrix} w_{2i}^{h} \\ h\theta_{2i}^{h} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{8} \\ \frac{3}{4} & -\frac{1}{8} \end{bmatrix} \begin{bmatrix} w_{i}^{2h} \\ 2h\theta_{i}^{2h} \end{bmatrix} + \begin{bmatrix} \frac{1}{2} & \frac{1}{8} \\ -\frac{3}{4} & -\frac{1}{8} \end{bmatrix} \begin{bmatrix} w_{i+1}^{2h} \\ 2h\theta_{i+1}^{2h} \end{bmatrix}. \tag{3.6}$$

The prolongation operator is therefore expressed as

$$I_{2h}^{h} = \left[\begin{bmatrix} \frac{1}{2} & \frac{3}{4} \\ -\frac{1}{8} & -\frac{1}{8} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} \frac{1}{2} & -\frac{3}{4} \\ \frac{1}{8} & -\frac{1}{8} \end{bmatrix} \right]^{T}. \tag{3.7}$$

The overall structure of the prolongation and restriction is illustrated in Figure A.6

3.1.3 MG Design Algorithm

The MG design algorithm shown in Figure A.7 is similar to the CA design algorithm described in Figure A.2. The main difference is that a multigrid cycle is substituted for the analysis update and repeated for the number of multigrid cycles, N_g, times. The use of multigrid analysis introduces one more algorithm control parameter, the number of smoothing iterations, S, which needs to be prescribed. However, as is pointed out in the previous subsection, the multigrid method is integrated into the CA design algorithm without affecting the structure of the algorithm.

3.2 Nested Iteration (Full MultiGrid, FMG)

Nested Iteration is a numerical technique where the solution is sought on a set of successively finer grids. The converged solution at a given grid level is used as an initial solution for the next finer grid. When applied to multigrid algorithms, nested iteration is referred to as FMG. The basic idea of the FMG algorithm is that because coarse lattices requires less computational time to converge, the computation starts from the coarsest lattice and then recursively interpolates the solution to the next finer lattice. Both field variables and the design variables are interpolated to a finer lattice after obtaining a converged design on the previous level of lattices. The design process is carried out on every level successively starting from the coarse grid, while for MG the design process is carried out only at the finest grid level. In the result section, the proposed FMG algorithm is observed to achieves superior performance to the MG algorithm because FMG accelerates both analysis and design while MG accelerates analysis only.

The merits of FMG over MG are 1) FMG accelerates both analysis and design while Multigrid accelerates analysis only. 2) FMG admits more room for development into an adaptive and fully automated algorithm. The reason behind this is that, FMG progressively generates a converged design from the coarsest to the finest grid. The main purpose of coarse grids is

to provide better initial design and field variables to a fine grid. Coarse grids are, therefore, not converged to the given convergence tolerance but possibly to a tolerance within the discretization error at each grid.

The pseudo code in Figure A.8 illustrates a design algorithm for FMG, where k is the grid level ($k = 1$), S is the smoothing number, and N_g is the number of multigrid cycles between design updates. A multigrid schedule is selected before applying the algorithm. The parametric studies in the result section give guidelines for the choice of N_g and S for maximum efficiency.

Chapter 4

Numerical Results

To demonstrate the potential of MG and FMG accelerated CA algorithms several numerical experiments have been completed utilizing the *Matlab*. This software package is easy and flexible to develop the newly proposed algorithms despite the relative computational inefficiency comparing to other programming languages such as *Fortlan* or *C*. The example model is a cantilevered-pinned beam with a constant distributed load, p over half-span as shown in Figure A.9. For the numerical experiments, the load level p is set to 2.0, the length L is set to 1.0, the Young's modulus E is set to be 1, and the minimum height is set to be 0.1. The stopping tolerance is chosen to be 10^{-6} and the stopping condition is chosen to be the infinity norm of correction of field variables.

4.1 CA implementation

First, we examine the convergence characteristics of the CA algorithm for analysis and design using a direct matrix solver. Therefore, a possible error introduced by using the iterative method is precluded. This study gives us more insight into the CA algorithm. Based on the acquired insight, a general acceleration scheme for the CA is sought and studied.

The convergence of the CA analysis update rule is calculated by explicitly constructing the iteration matrices and calculating the spectral radius, ρ. The convergence rate, (1-ρ), is plotted as a function of the number of cells on a log-log scale in Figure A.10(a). The slope on the plot is constant at -4. Therefore, the analysis convergence rate is inversely proportional to the number of cells raised to the fourth order. The norm of the design variable correction is plotted for the design iteration number in Figure A.10(b). The convergence behavior of the design rule is observed independent of the number of cells.

The CA convergence behavior when analysis and design are combined is shown in Figure A.11 where the total number of cell updates, N_t is shown as a function of the number of analysis updates between the design updates, N_a. Motivated by the convergence characteristics of CA in analysis and design, the figure is illustrated with respect to non-dimensional variables. The vertical axis is the total number of cell updates normalized by the number of cells to the fifth power, $\hat{N}_t = N_t/N^5$. The horizontal axis is the number of analysis updates between the design updates normalized by the number of cells to the fourth power, $\hat{N}_a = N_a/N^4$.

From this figure, it is clear that the total number of cell updates to convergence is large when the number of the analysis updates between design updates is small. This is due to poor convergence of the displacement field. On the other hand, the total number of cell updates to convergence is also large when the number of the analysis updates between design updates is large. Beyond a certain values of $\hat{N}_a^*$, the displacement field is fully converged. Increasing the value of $\hat{N}_a$ increases computational cost without improving convergence rate.There exists an optimal value of $\hat{N}_a$ for which the total number of cell updates is minimum, and the minimum value of $\hat{N}_t$ is nearly proportional to the number of cells raised to the fifth power. Thus CA is polynomial bounded with the rather high computational complexity of $O(N^5)$ for the one-dimensional beam problem. The complexity of CA itself varies according to problems. However, its computational complexity is already prohibitive even for one-dimensional beam problem.

4.2 MG acceleration

Convergence of the combined analysis and design MG accelerated CA algorithm is illustrated in Figure A.12 along with both of the standard CA and the nested design CA algorithms. In short, the nested design CA is the same as to repeat the standard CA successively in each grid, while using the converged design in the previous grid as the initial design. Results were generated for 11 grid levels; the size of the coarsest grid is 5 cells and the finest is 4097 cells, the number of smoothing iterations is 2, the number of multigrid cycles is 1, W-cycle is used. The vertical axis represents the total number of cell updates, N_t, and the horizontal axis represents the number of cells, N. These values are arbitrary chosen, however, the effects of different values of parameters are discussed in Section 4.4. Fist of all, the standard CA and the nested design CA show no significant difference in the slope. The gain from the nested design only is small since the design convergence of CA is grid-independent as shown in the previous section and the analysis convergence of CA makes the CA painfully slow in performance. The plot of MG results shows a linear asymptotic variation on (or in) the log-log scale having a unit slope. Thus, the computational cost is proportional to the number of cells which is the optimal computational complexity. Therefore, this result supports the claim that MG acceleration algorithm successfully remedies the deterioration of CA convergence.

The numerical results for the alternate stopping conditions, the infinity norm of field variable correction and the infinity norm of the residual, are illustrated in Figure A.13. The vertical axis and the horizontal axis are on log-log scale and represent the number of cells, N, and the total number of cell updates, respectively. It is observed that using alternate stopping conditions produced the same results in the sense that the overall behavior of the results are in the good agreement, as is discussed in Section 2.3. Especially, the results for the norm of design variable correction almost exactly matches those for the norm of field variable correction. The results for the norm of the residual, however, take more iterations than those for other stopping criteria. The reason for this can be explained by the fact that the

condition number of the system matrix increases as the grid is refined.

4.3 FMG acceleration

The results of FMG acceleration are shown in Figure A.14 in comparison to MG and the standard CA results. FMG accelerated CA algorithm performs better than MG accelerated CA especially as the size of a lattice grows. The striking fact about FMG performance is that as the lattice is refined, the computational cost almost remains constant. This equally holds for the alternate stopping criteria, illustrated in Figure A.15 where the log-log scale is used on the both axes and the horizontal axis is the number of cells, N and the vertical is the total number of cell updates, N_t. It is once again observed that the computational cost for the norm of the residual is higher than that for other stopping conditions as is observed in the MG acceleration result section.

To study the convergence characteristics of the MG and FMG schemes, the infinity norm of the design variable correction is plotted as a function of the number of design updates, N_d for different values of N in figure A.16(a) and A.16(b), respectively, The design convergence of MG acceleration is shown in figure A.16(a), and is independent of a number of cells. This is to be expected since MG accelerates analysis convergence only. In case of FMG acceleration, Figure A.16(b), we can draw out a relatively simple account for the superior performance of FMG. The initial height correction for a finer lattice is less than that of coarser lattices, fewer design updates are, hence, needed as the lattice is refined. Since the largest computational cost is incurred on fine lattices, this leads to a significant decrease of the overall computational cost of the FMG algorithm. The marked decrease in the number of design steps to convergence as the lattice is refined can also be explained by the fact that the solution to the design problem becomes grid-independent as the lattice is refined, shown in Figure A.17 where the design progress of FMG acceleration is compared to the analytic solution. The displacement profiles of FMG acceleration are illustrated in Figure

A.18. With observation on the figures for the design progress and the displacement profile progress, respectively, it is clear that the FMG acceleration capable to supply a better design and displacement profile to a finer grid. Figure A.19, moreover, manifest the fact that the FMG algorithm successively starts from a better design profile as the grid level goes up.

4.4 Parametric studies for FMG acceleration

The MG and FMG accelerations have the total of three algorithm control parameters which are the smoothing number, S, the number of multigrid cycles, N_g, and the parameter to select either of V cycle or W cycle. The performance of the FMG acceleration previously proves to be better than the MG acceleration. Thus, the parametric studies are performed using the FMG algorithm. There is no discrepancy in the general behavior of MG and FMG.

The parameter for either V or W cycle and the number of multigrid cycles are first examined while the smoothing number, S remains constant. The V and W cycle results for different numbers of N_g are shown in Figure A.20, where the vertical axis is in log scale and the total number of cell updates is plotted for different numbers of multigrid cycles, N_g. In general, the performance of the V cycle is better than that of the W cycle with respect to N_g. However, in case of $S = 2$ and $N_g = 1$, the W cycle is more efficient than V cycle, that is, one W cycle is sufficient to convey corrections to a fine grid. When the V cycle and $S = 2$, the computational efforts for $N_g = 1$ and $N_g = 2$ appear almost the same, which implies the inefficiency of the V cycle in $N_g = 1$ since there is no significant gain between $N_g = 1$ and $N_g = 2$. Results for different smoothing iterations, S, and N_g are shown in Figure A.21 where the vertical axis is also in log scale and W cycle is only considered. When $S = 1$, the result of $N_g = 1$ is almost the same as that of $N_g = 2$, and otherwise, the total number of cell updates increases as N_g increases or S increases. Figure A.22 shows that when $S = 1$ and $N_g = 1$, the graph is observed to be slightly unstable. The number of N_g, however, causes no distinct difference in the graphs when $S = 1$. Therefore, it can be concluded that a minimal

number of the smoothing and one or two W cycles are sufficient to obtain converged design.

Chapter 5

Conclusions

5.1 Summary

The main goal of this study is to demonstrate the applicability of MG and FMG accelerations to the CA method for structural optimization. To achieve the goal, this study started with posing the minimum compliance design problem for an Euler-Bernoulli beam. In Chapter 2, the CA formulations for the given problem is consistently derived such as the analysis update rule, the residual correction formulation, the design update rule, and the nested analysis and design update scheme for CA is illustrated. The residual correction formulation is used in favor of a hardware implementation. In the following chapter, the transfer operators for the multigrid are defined over the discretized domain and showed that the MG and FMG accelerations can be seamlessly incorporated into the CA method. The numerical experiments were performed to investigate CA convergence characteristics, the MG acceleration, and the FMG acceleration. The convergence of the MG and FMG acceleration were discussed to explain the better performance of the FMG over the MG algorithm. Additionally, the parametric studies on the algorithm control parameter were discussed and shown that the minimal number of the smoothing and one or two W cycles are sufficient to solve the given problem.

In the thesis, the proposed algorithms are developed, giving the priority to the implementation of the algorithms on hardware. The residual correction formulation is used in favor of hardware. Moreover, the acceleration methods, the multigrid, are chosen not to harm the positive features of CA such as inherent parallelism and simplify in implementation.

5.2 Conclusions

This study has demonstrated that MG and FMG accelerations successfully accelerate CA convergence for combined analysis and design problems. Numerical experiments prove that MG acceleration improves the analysis convergence and that FMG acceleration improves both of the analysis and the design convergence. The MG acceleration algorithm achieves optimal computational complexity in the sense that the computational cost is approximately proportional to the number of cells, and the FMG algorithm achieves even greater computational savings for a given convergence tolerance. Even though the potential of MG and FMG accelerations in structural optimization is demonstrated with the relatively simple beam problem, the presented multigrid acceleration schemes are expected to be applicable to more general structural optimization problems. Further investigation into the application of FMG accelerated CA, especially on FPGA hardware, will potentially provide powerful computational tools for large-scale computationally demanding problems.

Bibliography

[1] S. Ulam,"Random Processes and Transformations", Proceedings of the International Congress of Mathematicians, 2, 1992, pp. 85-87.

[2] J. von Neumann, "Theory of Self-Reproducing Automata", University of Illinois Press, 1996.

[3] S. Wolfram, "Cellular Automata and Complexity: collected Papers", Addison-Wesley Publishing Company, 1994.

[4] Gürdal, Z. and Tatting, B., "Cellular Automata for Design of Truss Structures with Linear and Nonlinear Response", AIAA-2000-1580, Proceedings of the 41st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Atlanta, GA, April 3-6, 2000.

[5] Tatting, B. and Gürdal, Z., "Cellular Automata for Design of Two-Dimensional Continuum Structures", AIAA-2000-4832, 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach CA, September 6-8, 2000.

[6] Abdalla, M. and Gürdal, Z., "Structural Design using Cellular Automata for Eigenvalue Problems", Structural and Multidisciplinary Analysis and Optimization, Vol. 26, No. 3-4, pp. 200-208.

[7] Abdalla, M. and Gürdal, Z., "Structural Design Using Optimality Based Cellular Automata", AIAA-2002-1676, Proceedings of the 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Denver CO, April 22-25.

[8] Samy Missoum, Abdalla, M. and Gürdal, Z., "Nonlinear Topology Design of Trusses using Cellular Automata", AIAA-2003-1445, the 44th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Norfork VA, April 2003 7-10.

[9] Shahriar and Gürdal, Z., "Design of Composite Layers with Curvillinear Fiber Paths Using Cellular Automata", AIAA-2003-2003, the 44th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Norfork VA, April 2003 7-10.

[10] Hajela, R., J., and Kim, B., "On the Use of Energy Minimization for CA based Analysis in Elasticity", Proceedings of the 41st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Atlanta, GA, April 3-6, 2000.

[11] Kita, E. and Toyota, T., "Structural Design using Cellular Automata", Structural and Multidisciplinary Analysis and Optimization, Vol. 19, 2000, pp. 64-73.

[12] Briggs, W. L., "A Multigrid Tutorial", SIAM, 1987.

[13] Hackbush, W., "Multigrid methods and applications", Springer, 1985.

[14] Wesseling, "An Introduction to Multigrid Methods", Wiley, 1992.

[15] Maar, B,. and Schulz, V., "Interior point multigrid methods for topology optimization. Technical Report 98-57", IWR, University of Heidelberg, 1998

[16] Kim, Y. Y. and Yoon, G. H., "Multi-resolution Multi-scale Topology Optimization - a New Paradigm", Int. J. Solids and Structure, 37, 5529-5559, 2000.

[17] Kwon, K., Kim, J. E., Jang, G. W.and Kim Y. Y, "Multigrid approach for efficient integrated multiscale analysis and design optimization", 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization 4-6, 2000.

[18] Burden, Richard L., and Faires J., Douglas, "Numerical Analysis", Brooks/cole, 2001.

[19] Jones, M., Plassmann, P. and Gürdal, Z., "Efficient Algorithms for Iterative Refinement Based on Configurable Precision Arithmetic", Submitted for publication in SIAM Journal On Matrix Analysis and Applications.

[20] Gürdal, Z., Hartka, T., Jones, M., and Kim, S. W., "A Reconfigurable Approach to Structural Engineering Design Computations", Accepted for publication in Engineering of Reconfigurable Systems and Algorithms, 2004.

[21] Janmes M., Gere and Stephen P., Timoshenko, "Mechanics of Materials", Third SI Edition, Chapman & Hall, 1991.

Appendix

A.1 CA formulation for the cell-centered grid

A.1.1 Analysis Update

The beam domain is presented in the cell-centered grid, illustrated in Figure A.23. The general cell definition and variables are the same as described in the CA formulation section. However, it is clearly distinguished in Figure A.23 that a cell is placed at the midpoint of a constant material property region and a fictitious cell is introduced between cells. Each cell reflects the segment of physical domain, which is different from the vertex-centered grid.

As discussed in Section 2.2.1, the first step is to compute the strain energy for four segments in the control volume that is marked by the dashed line. By applying the principle of the minimum potential energy, the equilibrium equation can be expressed with respect to the field variable vector of the center cell and that of auxiliary cells as follows,

$$\begin{bmatrix} \boldsymbol{k}_{11} & \boldsymbol{k}_{12} & \boldsymbol{k}_{13} \\ \boldsymbol{k}_{12}{}^T & \boldsymbol{k}_{22} & 0 \\ \boldsymbol{k}_{13}{}^T & 0 & \boldsymbol{k}_{33} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_C \\ \boldsymbol{u}_{ML} \\ \boldsymbol{u}_{MR} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \boldsymbol{C}_2 & 0 \\ 0 & \boldsymbol{C}_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_L \\ \boldsymbol{u}_R \end{bmatrix} = \boldsymbol{f}_C, \tag{A.1}$$

and

$$\boldsymbol{f}_C = [[F|M]\ [0|0]\ [0|0]]^T.$$

where $\boldsymbol{f}_C$ is the vector of concentrate forces and moments. The assumption is that the

external loads are limited on the center cell, that is, auxiliary cells carry no loads.The 2 x 2 stiffness matrices, $\boldsymbol{k}_{ij}$, are as follows,

$$\begin{aligned} \boldsymbol{k}_{11} &= \frac{\partial^2 U}{\partial \boldsymbol{u}_C \boldsymbol{u}_C}, \quad \boldsymbol{k}_{12} = \frac{\partial^2 U}{\partial \boldsymbol{u}_C \boldsymbol{u}_{ML}}, \quad \boldsymbol{k}_{13} = \frac{\partial^2 U}{\partial \boldsymbol{u}_C \boldsymbol{u}_{MR}}, \\ \boldsymbol{k}_{22} &= \frac{\partial^2 U}{\partial \boldsymbol{u}_{ML} \boldsymbol{u}_{ML}}, \quad \boldsymbol{k}_{33} = \frac{\partial^2 U}{\partial \boldsymbol{u}_{MR} \boldsymbol{u}_{MR}}, \end{aligned} \tag{A.2}$$

and $\boldsymbol{C}_i$ are given by,

$$\boldsymbol{C}_2 = \frac{\partial^2 U}{\partial \boldsymbol{u}_{ML} \boldsymbol{u}_L}, \qquad \boldsymbol{C}_3 = \frac{\partial^2 U}{\partial \boldsymbol{u}_{MR} \boldsymbol{u}_R}. \tag{A.3}$$

Due to the assumption that no load is on fictitious cell, the field variables of auxiliary cells are reduced to,

$$\boldsymbol{u}_{ML} = -{\boldsymbol{k}_{22}}^{-1} \cdot (\boldsymbol{k}_{21} \cdot \boldsymbol{u}_C + \boldsymbol{C}_2 \cdot \boldsymbol{u}_L) \tag{A.4}$$

$$\boldsymbol{u}_{MR} = -{\boldsymbol{k}_{33}}^{-1} \cdot (\boldsymbol{k}_{31} \cdot \boldsymbol{u}_C + \boldsymbol{C}_3 \cdot \boldsymbol{u}_L) \tag{A.5}$$

Accordingly, substitute (A.4) into (A.2) and consistently simplify (A.2) exclusively with respect to the variables of the center cell, similar to the static condensation. After all, the resulting equation are obtained as,

$$\tilde{\boldsymbol{k}} \cdot \boldsymbol{u}_C = \boldsymbol{f}_C + \boldsymbol{f}_g, \tag{A.6}$$

where $\tilde{\boldsymbol{k}}$ are re-defined 2 x 2 stiffness matrices, and $\boldsymbol{f}_g$ is related to the influence of neighboring cells to a center cell. $\tilde{\boldsymbol{k}}$ and $\boldsymbol{f}_g$ are respectively given by,

$$\tilde{\boldsymbol{k}} = \boldsymbol{k}_{11} - \boldsymbol{k}_{12} \cdot {\boldsymbol{k}_{22}}^{-1} \cdot {\boldsymbol{k}_{12}}^T, \tag{A.7}$$

$$\boldsymbol{f}_g = \boldsymbol{k}_{12} \cdot {\boldsymbol{k}_{22}}^{-1} \cdot \boldsymbol{C}_2 \cdot \boldsymbol{u}_L + \boldsymbol{k}_{13} \cdot {\boldsymbol{k}_{33}}^{-1} \cdot \boldsymbol{C}_3 \cdot \boldsymbol{u}_R. \tag{A.8}$$

(A.6) has the same form as the previously derived analysis update rule in section 2.2.1 even though the coefficients of the stiffness matrix and $\boldsymbol{f}_g$ are different. Once again, we have the well defined relation between the center cell and its neighboring cells, which can be expressed in the fully expanded form such as,

$$\frac{1}{f(a)f(b)} \frac{8E_C I_C}{h^3} \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} w_C \\ h\theta_C \end{bmatrix} = \boldsymbol{f}_C + \boldsymbol{f}_g, \tag{A.9}$$

where,

$$a = \frac{E_L I_L}{E_C I_C}, \qquad b = \frac{E_R I_R}{E_C I_C}, \tag{A.10}$$

$$c = a + b, \qquad d = ab, \tag{A.11}$$

$$S_{11} = 12\left(c(1+c) + d(26+15c) + 2d^2\right), \tag{A.12}$$

$$S_{12} = S_{21} = -3(a-b)(3+c+11d), \tag{A.13}$$

$$S_{22} = c(c+7) + 2d(d-1) + d(21c+196), \tag{A.14}$$

$$\boldsymbol{f}_g = \frac{8E_C I_c}{h^3}\left[\begin{bmatrix} \frac{12a(1+a)}{f(a)} & \frac{3a(1+3a)}{f(a)} \\ \frac{-3a(3+a}{f(a)} & \frac{-2a(1+a)}{f(a)} \end{bmatrix}\begin{bmatrix} w_L \\ h\theta_L \end{bmatrix} + \begin{bmatrix} \frac{12b(1+b)}{f(b)} & \frac{-3b(1+3b)}{f(b)} \\ \frac{3b(1+b}{f(b)} & \frac{-2b(1+b)}{f(b)} \end{bmatrix}\begin{bmatrix} w_R \\ h\theta_R \end{bmatrix}\right], \tag{A.15}$$

and lastly, $f(x) = 1 + 14 + x^2$.

The local analysis update rule, (A.9) appears more complicated than it actually is. Since all the coefficients are constant during the analysis, (A.9) become expressed more concisely. It, however, requires more computations to update all the coefficients than the local analysis update rule for the vertex-centered grid. Furthermore, the discontinuity in the design variable between a center cell and its neighboring cells hinders in defining simple transfer operators for the multigrid method.

A.1.2 Design Update Rule

The design is based on the fact that each cell needs the minimum dimension of a height to carry the internal stress in the cell, which is the fully stressed deign. For simplicity, the width of a beam is set to be a unity. A bending moment is used as a stress measure, and is recovered using (2.3), i.e.,

$$\begin{aligned} M_C &= \frac{2EI_C}{h^2 f(a)}\left[[12a(3+a)\ 8a(1+a)] \cdot \boldsymbol{u}_L + [-12a(3+a)\ 4a(7+a)] \cdot \boldsymbol{u}_C\right] \\ &= \frac{-2EI_C}{h^2 f(b)}\left[[12b(3+b)\ 4b(7+b)] \cdot \boldsymbol{u}_C + [-12b(3+3b)\ 8b(1+b)] \cdot \boldsymbol{u}_R\right], \end{aligned} \tag{A.16}$$

where coefficients, a and b, and the function, $f(x)$ are defined in the previous subsection.

A.2 Analytic Solution

A cantilevered-pinned beam with the distributed load is statically indeterminate so that the superposition method and the unit load method are applied to give the analytic solution for the optimized beam. The details of these two methodologies are not given here, yet well explained in any introductory structure text book such as [21]. The procedure of how to solve is described in brief. We first turn a statically indeterminate structure into statically determinate released structures by using the method of superposition, and then apply the unit load method so that the redundants are determined by satisfying the redundant constraints. For the given problem, we need to select one redundant and to meet one redundant constraint.

Here, the beam in A.24 (a) is analyzed by taking the moment, M_a as the redundant. Accordingly, released structures becomes simply supported, shown in A.24 (b and c). The normalized moment distribution of the released beam (b) is,

$$M_o(\xi) = \begin{cases} l^2\xi & \text{if } 0 < \xi \leq 1-l \\ (\xi-1)(1-2l+l^2-\xi) & \text{if } 1-l < \xi \leq 1 \end{cases} \tag{A.17}$$

in which L is the length of beam, q is the distributed load, l is the part on which q acts, $\xi = x/L$, and the normalization is carried out by multiplying the moment by $2/(qL^2)$.

The moment of the released beam (c), M_a is considered as,

$$M_a = -C_q m(\xi) = -C_q(1-\xi) \tag{A.18}$$

where the minus sign appears in opposition to the sign of the moment, M_o.

Therefore, the total moment becomes,

$$M_{total} = \begin{cases} l^2\xi - C_q(1-\xi) & \text{if } 0 < \xi \leq 1-l \\ (\xi-1)(1-2l+l^2-\xi) - C_q(1-\xi) & \text{if } 1-l < \xi \leq 1 \end{cases} \tag{A.19}$$

As described in the design rule, the design is based on the bending moment and for conve-

nience the formula is as follows,

$$a(\xi) = \sqrt{|M_{total}|} \tag{A.20}$$

where $a(\xi)$ is the height of beam. Furthermore, we have normalized the bending rigidity for the rectangular cross section so as to be,

$$EI(\xi) = |M_{total}|^{\frac{3}{2}}. \tag{A.21}$$

The bending moment diagram for the total moment is illustrated in A.24 in which C_o is the lower bound and p_i denote the points that the bending moment intersects the lower bound. There are three intersecting points for the given bending diagram, yet the number and location of intersecting points are dependent on the magnitude of bending moment and lower bound. Once intersecting points are found, we can re-calculate the optimal bending rigidity by using A.21. In case of A.24, the bending rigidity becomes,

$$EI(\xi) = \begin{cases} (-M_{total})^{\frac{2}{3}} & \text{if } 0 < \xi \leq p_1 \\ (-C_o)^{\frac{2}{3}} & \text{if } p_1 < \xi \leq p_2 \\ (M_{total})^{\frac{2}{3}} & \text{if } p_2 < \xi \leq p_3 \\ (C_o)^{\frac{2}{3}} & \text{if } p_3 < \xi \leq 1 \end{cases} \tag{A.22}$$

Subsequently, we apply the unit load method to calculate the angle of rotation at the left-hand support, which is as follows,

$$\theta_o = \int_0^1 \frac{M_{total}(\xi)m(\xi)}{EI(\xi)} d\xi \tag{A.23}$$

wherein θ_o is the redundant constraint because we have selected M_a as the redundant, and what we want to do is to find C_q that makes θ_o zero. In the problem solving we used the secant method to find a proper C_q for given distributed load. Appendix B presents the *Mathematica* program exclusively written for the cantilevered-pinned beam design.

A.3 Mathematica Program for Analytic Solution

The *Mathematica* program is coded for the cantilevered-pinned beam and includes four functions in it. We will illustrate the input variables and output variables for each function in simple terms. If run the code once, we will come to more understand how it is implemented and works.

1. The usage of **GBeam[]** is:

$$\{\theta_o, Volume, pi, newDesign, Moment, dw, w\} = \text{GBeam}[C_q, C_o, l]$$

where θ_o is the rotation at the fixed end, $Volume$ is the total volume, pi is the list of intersecting points, $newDesign$ is the new height distribution, $Moment$ is the moment distribution according to the new design, dw is the rotation of the beam, and w is the deflection of the beam. Input variables are already described in Appendix A.

2. The usage of **SecantSolve[]** is:

$$C_q = \text{SecantSolve}[\{\{C_q^a, C_q^b\}, \{\{\theta_q^a, \theta_q^b\}\}, \text{GBeam}[C_q, C_o, l]]$$

where C_q^a and C_q^b are initial approximations of C_q and θ_o^a and θ_o^b are corresponding initial approximate rotations at the fixed end. The function, **SecantSolve[]** returns the proper C_q.

3. The usage of **PlotMSFun[]** and **EvaluatetMSFun[]** are:

$$\text{PlotMSFunp}[pi, func, \xi] \quad \text{and} \quad \text{EvalueateMSFun}[pi, func, \xi, x]$$

where **PlotMSFun[]** and **EvaluatetMSFun[]** return the plot and numerical values for the corresponding function, respectively. The input functions can be any of return variables of **GBeam[]** except θ_o, $Volume$, and pi.

For instance, the example problem is that the distributed load acts over the half of the beam span and the lower bound of the design is 0.1, i.e. $C_o = 0.01$ so that we initialize variables

and solve such as,

$l = 0.5;$

$C_o = 0.01;$

$C_a = 0.1; C_b = 0.125$

$\theta_q^a = \text{GBeam}[C_a, C_o, l][[1]];$

$\theta_q^b = \text{GBeam}[C_b, C_o, l][[1]];$

$C_q = \text{SecantSolve}[\{\{C_q^a, C_q^b\}, \{\theta_q^a, \theta_q^b\}\}, \text{GBeam}[\#, C_o, l][[1]]\&]$

$Sol = \text{GBeam}[C_q, C_o, l];$

Supposed that we want to plot the moment distribution,

$pi = Sol[[3]]; \qquad Moment = Sol[[5];$

$\text{PlotMSFunp}[pi, Moment, \xi]$

The *Mathematica* source code is attached in the next page

```
In[1]:= GBeam[Cq_, Co_, ℓ_] :=
 Module[{IL, F, B, C1, C2, C3, b, c1, c2, As, Ds, θ, vol, wp1Co, wp1p,
   wp1n, wp2Co, wp2p, wp2n, w1Co, w1p, w1n, w2Co, w2p, w2n, Wp, W, W1},
  M2 = Expand[(1 - 2 ℓ + ℓ^2 - ξ) (-1 + ξ) - Cq (1 - ξ)];
  M1 = Expand[ℓ^2 ξ - Cq (1 - ξ)];
  f = 1 + Cq - 2 ℓ + ℓ^2;
  g = (Cq + ℓ^2);
  p = Cq / g;
  wp1Co = -g (2 p - ξ) ξ / (2 Co^(3/2));
  w1Co = g (3 p - 2 ξ) ξ^2 / (6 Co^(3/2));
  wp2Co = 1 / (6 Co^(3/2)) ξ (-6 f + 3 ξ + 3 f ξ - 2 ξ^2);
  w2Co = 1 / (12 Co^(3/2)) ξ^2 (6 f - 4 ξ - 4 f ξ + 3 ξ^2);
  wp2p = -2 ArcTan[Sqrt[(1 - ξ) / (ξ - f)]];
  w2p = Sqrt[(1 - ξ) (ξ - f)] + (1 + f) / 2 ArcTan[((1 + f) / 2 - ξ) / Sqrt[(1 - ξ) (ξ - f)]];
  wp2n = 2 Log[Sqrt[1 - ξ] + Sqrt[f - ξ]];
  w2n = Sqrt[(1 - ξ) (ξ - f)] - (1 + f) Log[Sqrt[1 - ξ] + Sqrt[f - ξ]];
  wp1p = 2 / Sqrt[g] Sqrt[ξ - p];
  w1p = -(2 Sqrt[ξ - p]) / (3 Sqrt[g]) (2 p + ξ);
  wp1n = 2 / Sqrt[g] Sqrt[p - ξ];
  w1n = -(2 Sqrt[p - ξ]) / (3 Sqrt[g]) (2 p + ξ);
  C1 = Co / ℓ + ℓ - ℓ^2;
  C2 = -Co / ℓ + ℓ - ℓ^2;
  C3 = (2 - ℓ) ℓ - 2 Sqrt[Co];
  If[Cq < C1
    , If[Cq < C3
       , Ds = Sqrt[(Cq + (-2 + ℓ) ℓ)^2 - 4 Co];
        As = 2 + Cq + (-2 + ℓ) ℓ;
        c2 = 1 / 2 (As + Ds);
        If[Cq < C2
          , c1 = (Co + Cq) / (Cq + ℓ^2);
           If[Cq < Co
             , IL = {0, c1, 1 - ℓ, c2, 1};
               F = {Co, M1, M2, Co};
               B = {M1, M1, M2, M2};
               Wp = {wp1Co, wp1p, wp2p, wp2Co};
```

```
                W = {w1Co, w1p, w2p, w2Co};
                (*Print["TTTT"];*)
              , b = (-Co + Cq)/(Cq + ℓ^2);
               IL = {0, b, c1, 1 - ℓ, c2, 1};
                F = {-M1, Co, M1, M2, Co};
                B = {M1, M1, M1, M2, M2};
                Wp = {wp1n, wp1Co, wp1p, wp2p, wp2Co};
                W = {w1n, w1Co, w1p, w2p, w2Co};
                (*Print["TTTF"]; (*OKK*)*)
              ];
          , c1 = 1/2 (As - Ds);
            If[Cq < Co
              , IL = {0, 1 - ℓ, c1, c2, 1};
                F = {Co, Co, M2, Co};
                B = {M1, M2, M2, M2};
                (*Print["TTFT"];*)
              , b = (-Co + Cq)/(Cq + ℓ^2);
               IL = {0, b, 1 - ℓ, c1, c2, 1};
                F = {-M1, Co, Co, M2, Co};
                B = {M1, M1, M2, M2, M2};
                Wp = {wp1n, wp1Co, wp2Co, wp2p, wp2Co};
                W = {w1n, w1Co, w2Co, w2p, w2Co};
                (*Print["TTFF"]; (*OKK*)*)
              ];
           ];
        , If[Cq < Co
            , IL = {0, 1 - ℓ, 1};
              F = {Co, Co};
              B = {M1, M2};
              (*Print["TFFT"];*)
            , b = (-Co + Cq)/(Cq + ℓ^2);
             IL = {0, b, 1 - ℓ, 1};
              F = {-M1, Co, Co};
              B = {M1, M1, M2};
              Wp = {wp1n, wp1Co, wp2Co};
              W = {w1n, w1Co, w2Co};
              (*Print["TFFF"]; (*OKK*)*)
            ];
        ];
  , b = 1/2 (2 + Cq + (-2 + ℓ) ℓ - Sqrt[4 Co + (Cq + (-2 + ℓ) ℓ)^2]);
   If[Cq < C3
     , Ds = Sqrt[(Cq + (-2 + ℓ) ℓ)^2 - 4 Co];
      As = 2 + Cq + (-2 + ℓ) ℓ;
      c2 = 1/2 (As + Ds); c1 = 1/2 (As - Ds);
      IL = {0, 1 - ℓ, b, c1, c2, 1};
       F = {-M1, -M2, Co, M2, Co};
       B = {M1, M2, M2, M2, M2};
```

```
          Wp = {wp1n, wp2n, wp2Co, wp2p, wp2Co};
          W = {w1n, w2n, w2Co, w2p, w2Co};
          (*Print["FT"];*)
       , IL = {0, 1 - ℓ, b, 1};
          F = {-M1, -M2, Co};
          (*Print["FF"];*)
          B = {M1, M2, M2};
          Wp = {wp1n, wp2n, wp2Co};
          W = {w1n, w2n, w2Co};
       ];
    ];
  θ = Apply[Plus, MapThread[Integrate[#1 (1 - ξ) / #2^(3/2), {ξ, #3, #4}] &,
      {B, F, Drop[IL, -1], Drop[IL, 1]}]] // Re;
  vol = Apply[Plus, MapThread[Integrate[√#1, {ξ, #2, #3}] &,
      {F, Drop[IL, -1], Drop[IL, 1]}]] // Re;
  Print[Cq, "+++", θ];
  W1 = MapThread[#1 /. ξ → #2 &, {Wp, Drop[IL, -1]}];
  Wp = Wp - W1;
  W1 = MapThread[#1 /. ξ → #2 &, {Wp, Drop[IL, 1]}];
  Wp = Drop[FoldList[Plus, 0, W1], -1] + Wp;
  W = W + Wp ξ;
  W1 = MapThread[#1 /. ξ → #2 &, {W, Drop[IL, -1]}];
  W = W - W1;
  W1 = MapThread[#1 /. ξ → #2 &, {W, Drop[IL, 1]}];
  W = Drop[FoldList[Plus, 0, W1], -1] + W;
  {θ, vol, IL, √F, B, Wp, W}
 ]

SecantStep[{{a_, b_}, {fa_, fb_}}, f_] := Module[{x, fx},
  x = b - (b - a) / (fb - fa) fb;
  fx = f @@ {x};
  Return[{{b, x}, {fb, fx}}];
 ]

SecantSolve[{{a_, b_}, {fa_, fb_}}, fz_] :=
 Last[NestWhileList[SecantStep[#, fz] &, {{a, b}, {fa, fb}},
    (Abs[#[[2, 2]]] > 10^-10) &]][[1, 2]]

PlotMSFun[supp_List, f_List, ξ_] :=
 Show[MapThread[Plot[#1, {ξ, #2, #3}, DisplayFunction → Identity] &,
   {f, Drop[supp, -1], Drop[supp, 1]}], DisplayFunction → $DisplayFunction]

EvaluateMSFun[supp_List, f_List, ξ_, x_] :=
 f[[Max[Length[Select[supp, # < x &]], 1]]] /. {ξ → x}
```

■ **Example**

```
In[6]:= ℓ = 0.5;
        Co = 0.01
        a = 0.1
        b = 0.125
        fa = GBeam[a, Co, ℓ][[1]]
        fb = GBeam[b, Co, ℓ][[1]]

Out[7]= 0.01

Out[8]= 0.1

Out[9]= 0.125

        0.1+++-0.0307286

Out[10]= -0.0307286

        0.125+++-0.205953

Out[11]= -0.205953

In[12]:= Cq = SecantSolve[{{a, b}, {fa, fb}}, GBeam[#, Co, ℓ][[1]] &]

        0.0956158+++0.00687798

        0.0965654+++-0.00147014

        0.0963982+++-8.29901×10^-6

        0.0963972+++1.00819×10^-8

        0.0963972+++-8.10463×10^-14

Out[12]= 0.0963972

In[13]:= Sol = GBeam[Cq, Co, ℓ];

        0.0963972+++-8.10463×10^-14
```

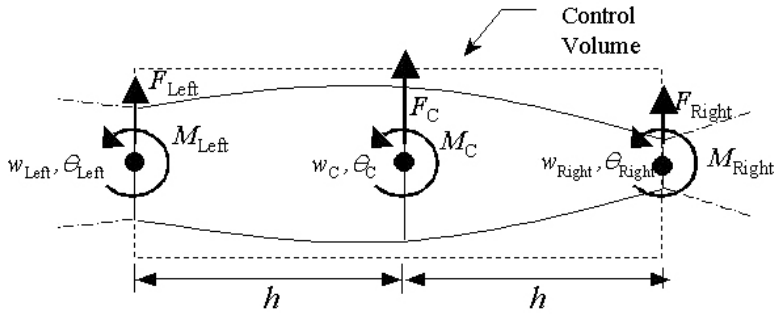


Figure A.1: 1-D Cell neighborhood.

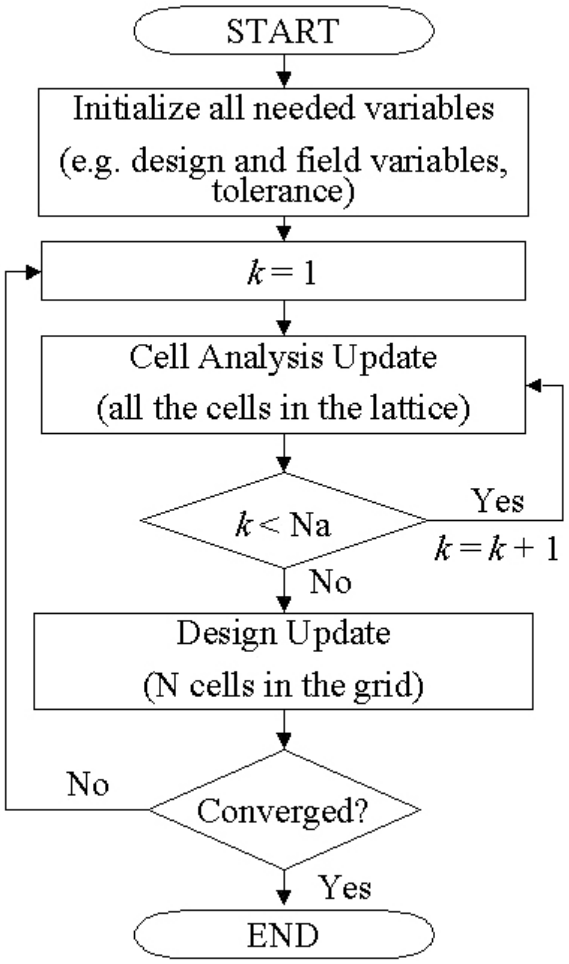


Figure A.2: CA design algorithm.

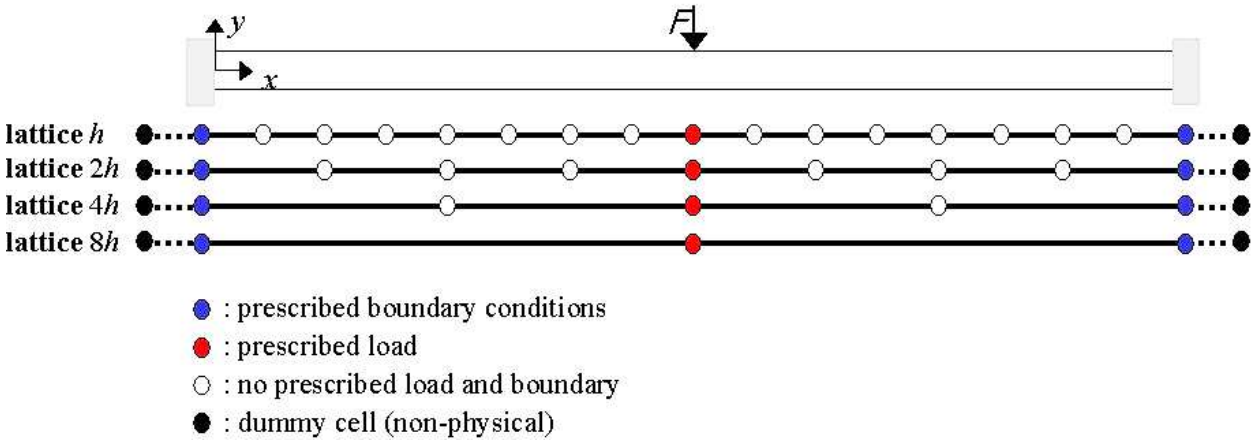


Figure A.3: Multi-level lattices for MG.

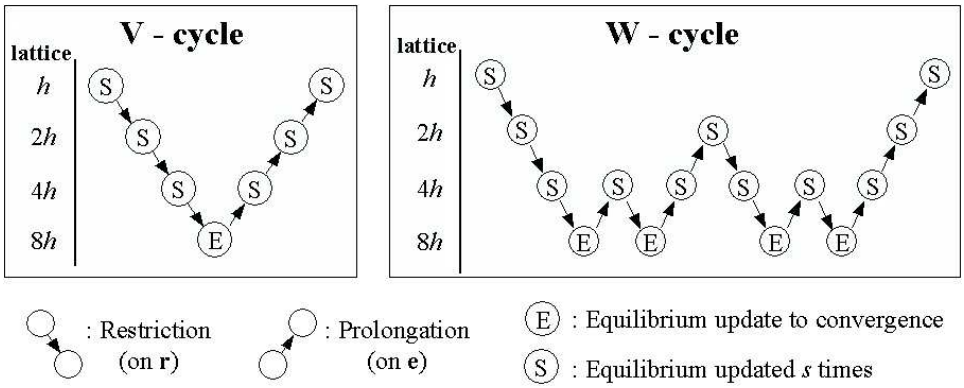
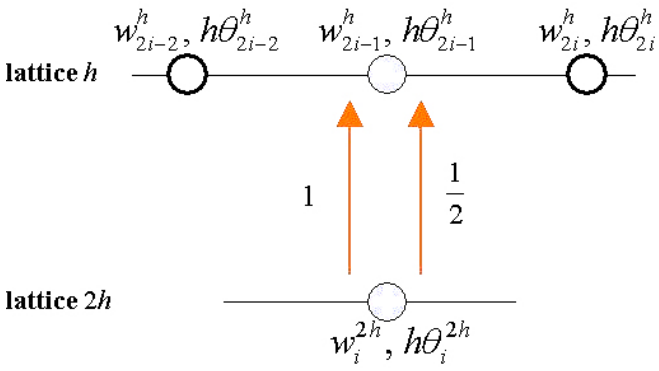
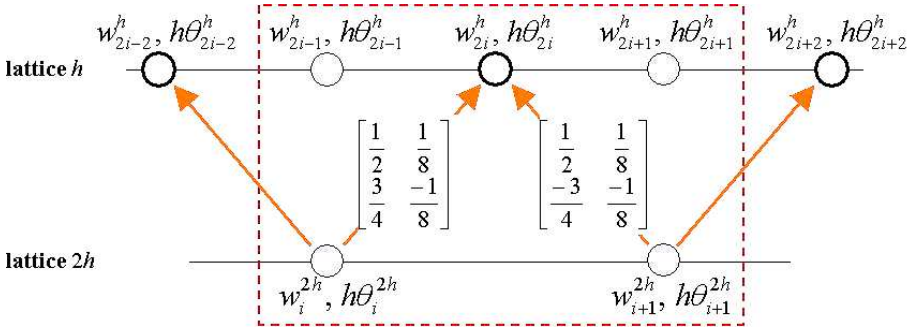


Figure A.4: V- and W-cycles (an upward arrow indicates a prolongation operation and a downward arrow indicates a restriction operation).



(a) Projection



(b) Interpolation

Figure A.5: Derivation of the prolongation operator.

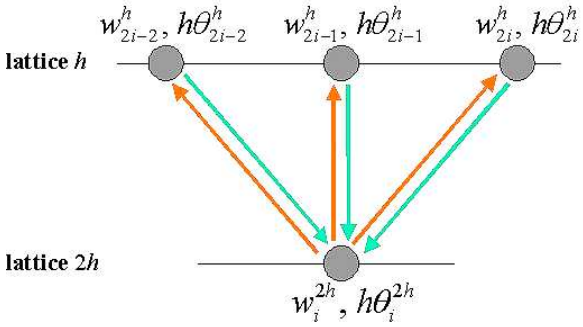


Figure A.6: Prolongation and restriction.

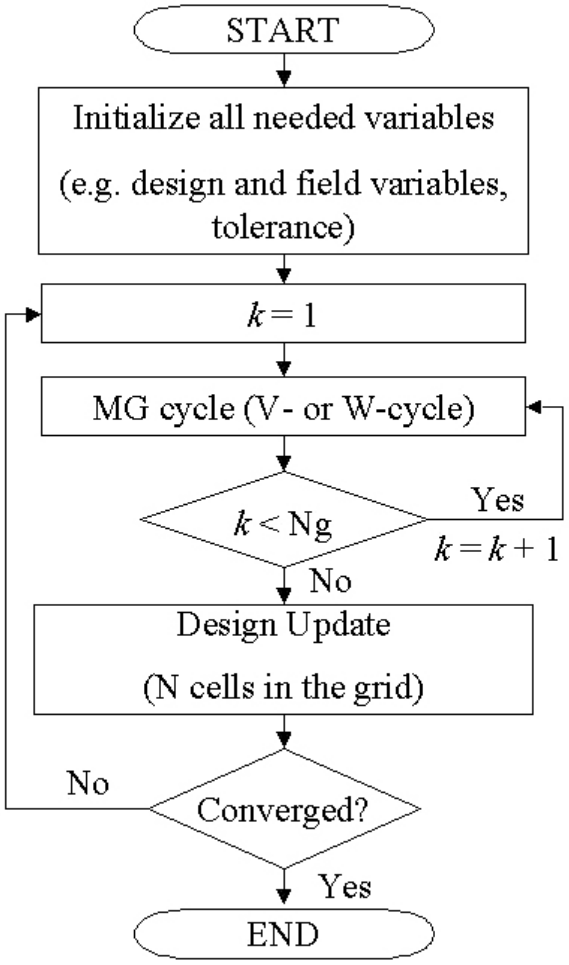


Figure A.7: MG design update scheme

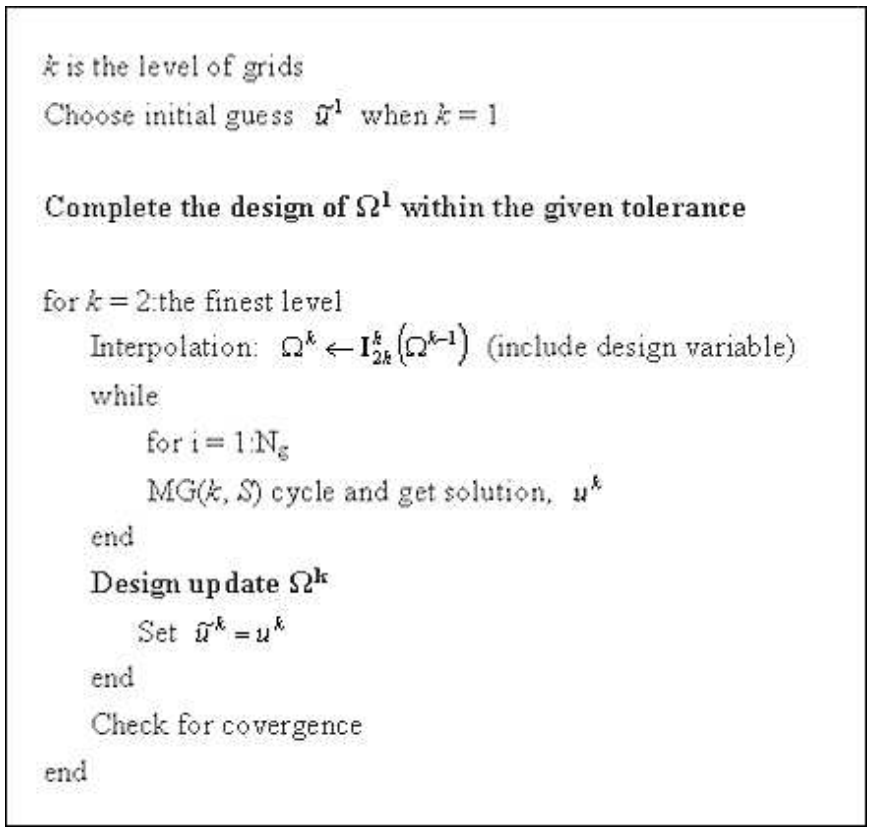


Figure A.8: Pseudo-code for the FMG design update scheme

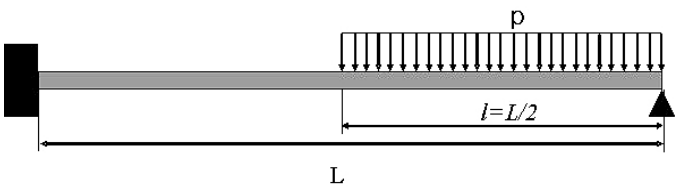
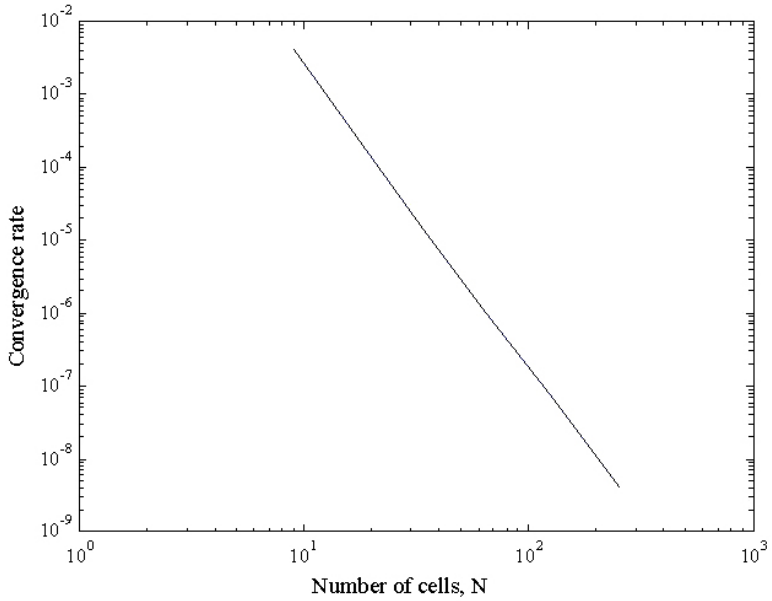
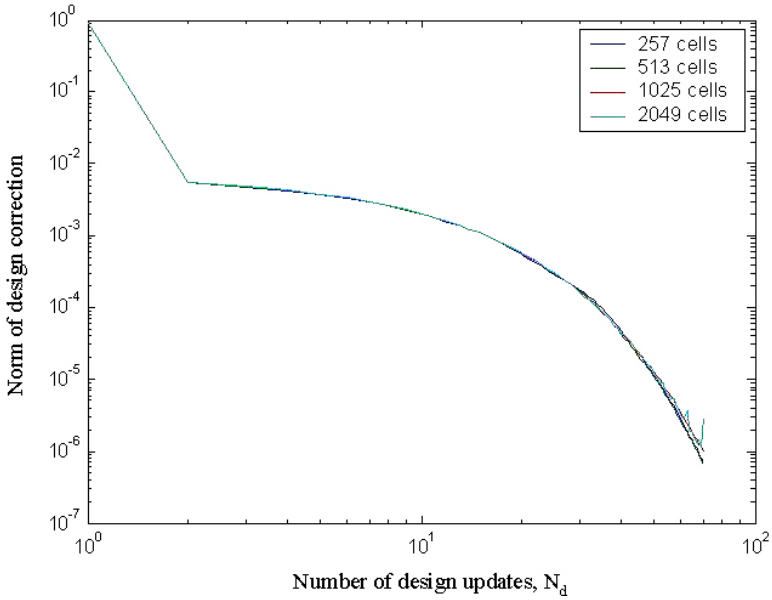


Figure A.9: A cantilevered-pinned beam with a distributed load



(a) Analysis



(b) Design

Figure A.10: CA convergence.

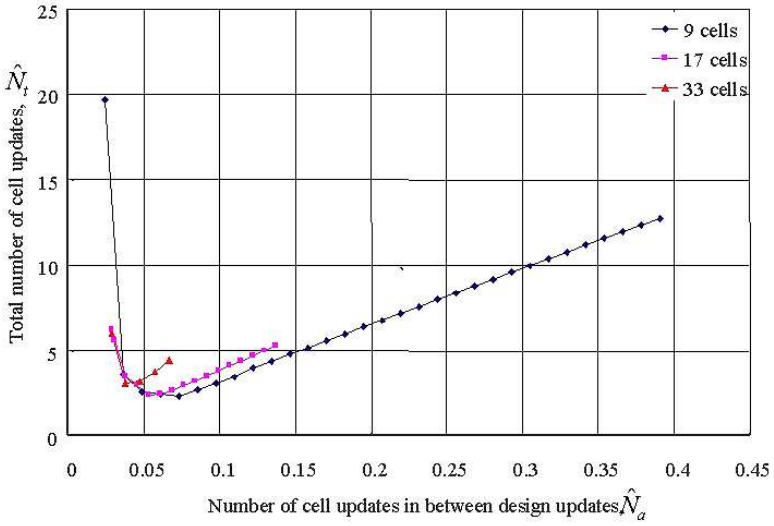


Figure A.11: CA convergence combining analysis and design

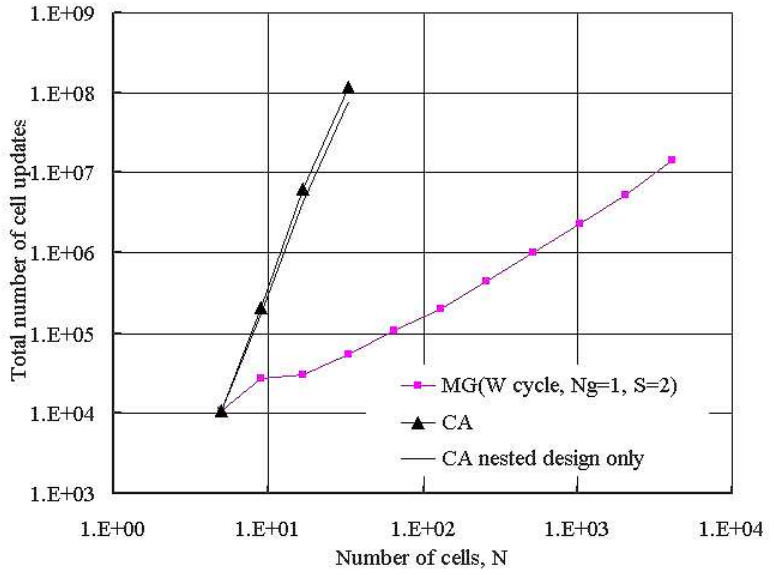


Figure A.12: Comparison of MG accelerated CA and CA results.

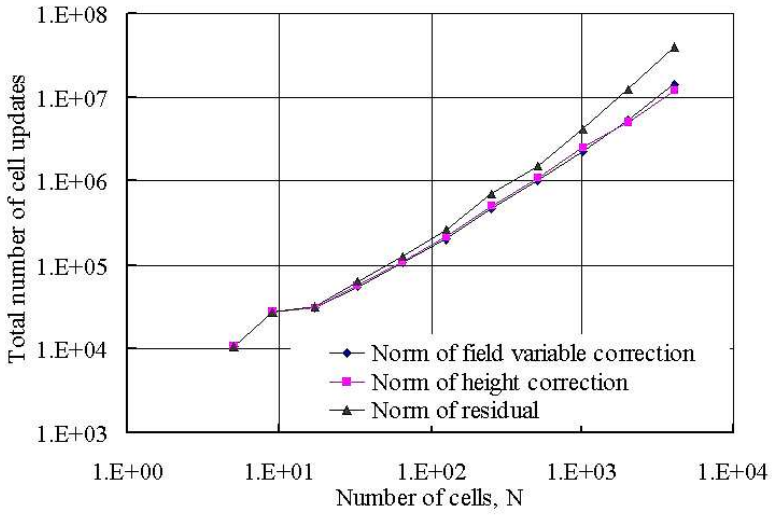


Figure A.13: Comparison of variations of stopping criteria for MG.

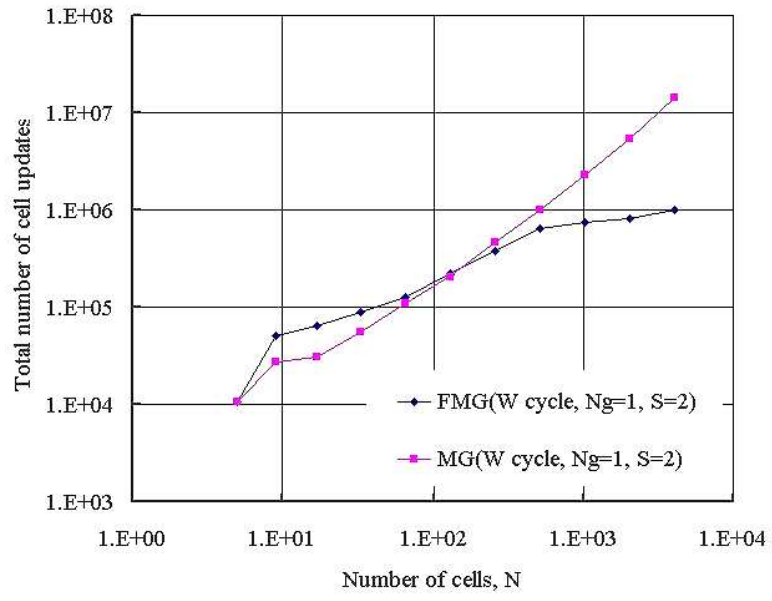


Figure A.14: Comparison of FMG and MG accelerated CA results.

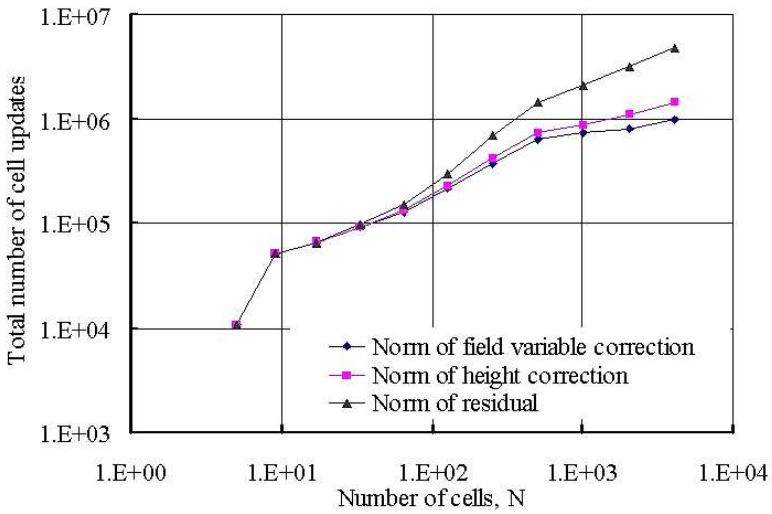
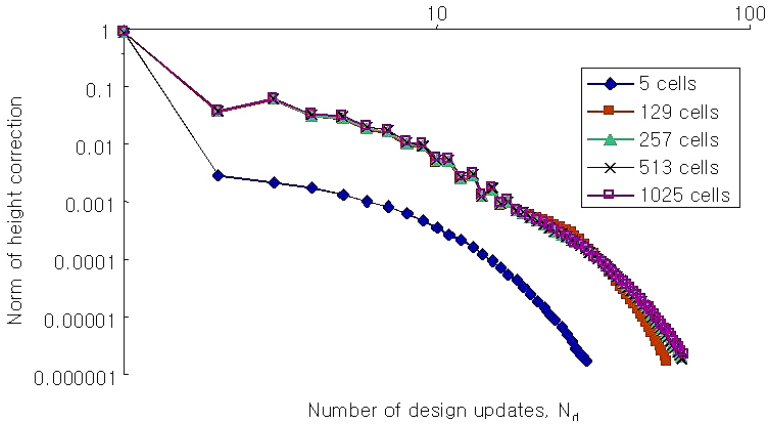
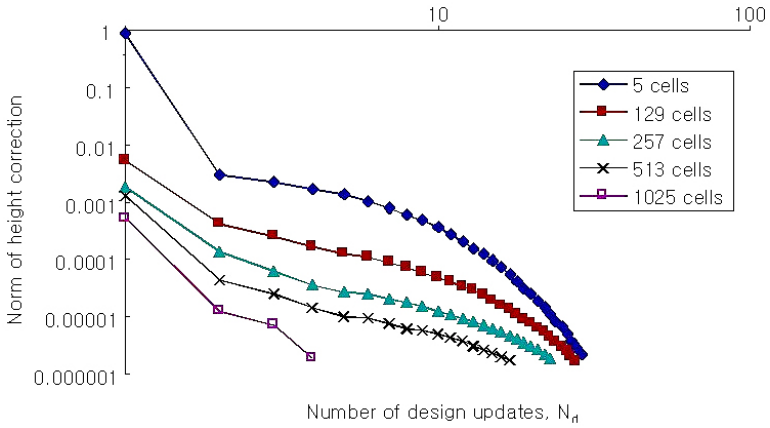


Figure A.15: Comparison of variations of stopping criteria for FMG.



(a) MG



(b) FMG

Figure A.16: Norm of height correction using MG and FMG acceleration.

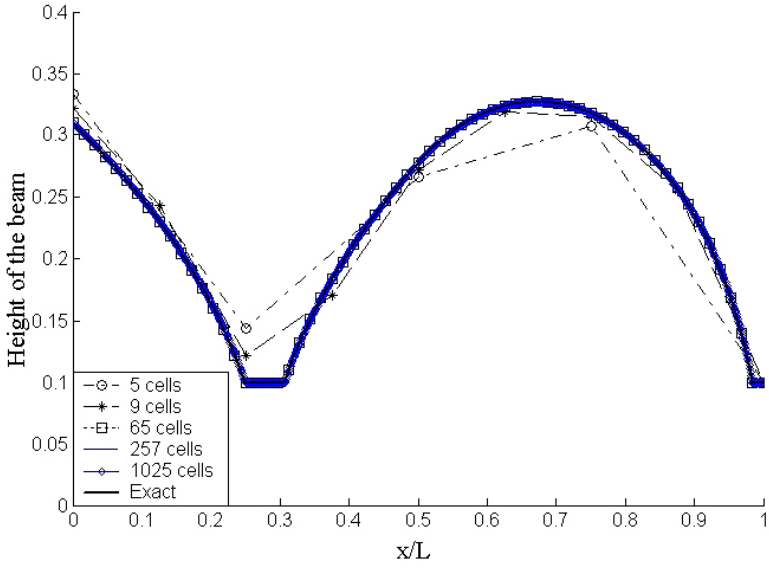
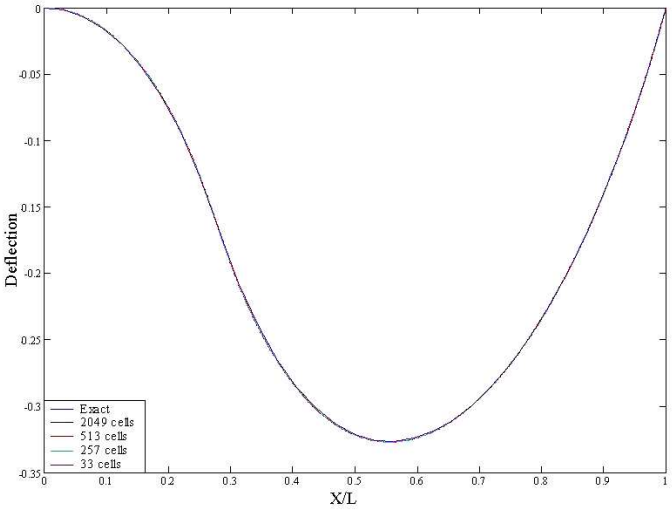
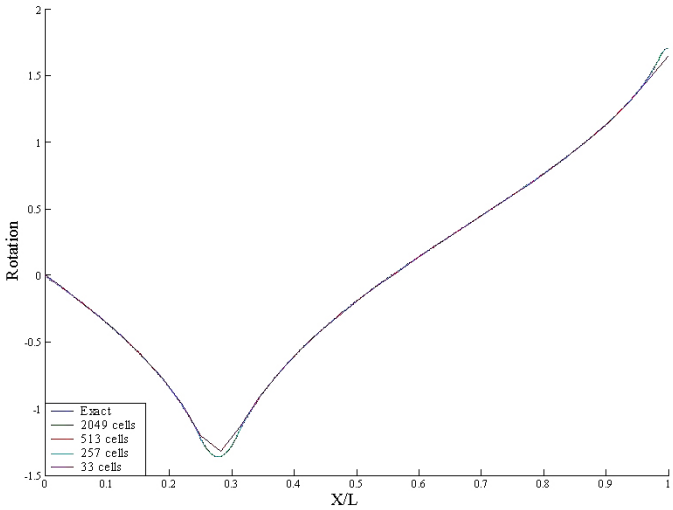


Figure A.17: Comparison of FMG design progress and analytic solution.



(a) Deflection progress and analytic solution.



(b) Rotation progress and analytic solution.

Figure A.18: Comparison of FMG displacement profiles and analytic solution.

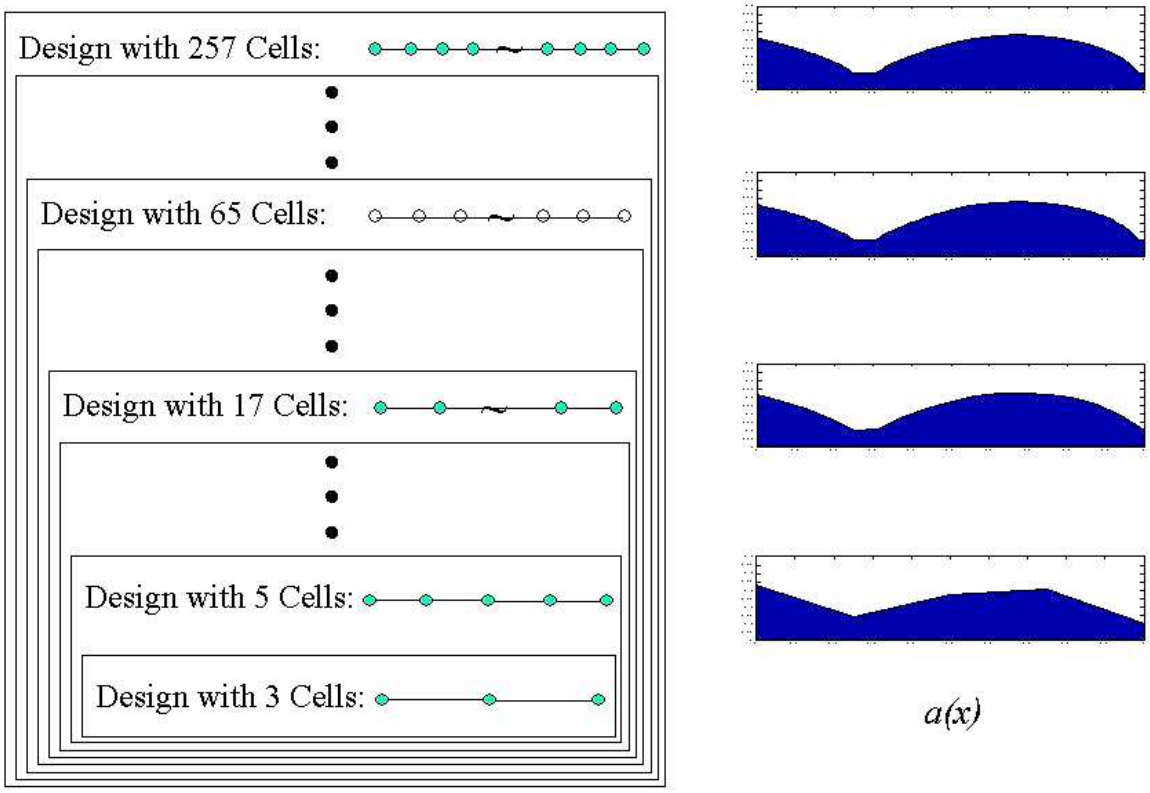
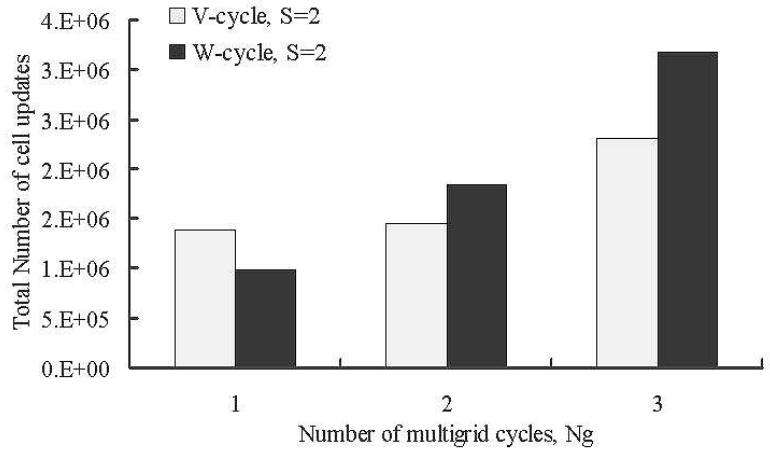
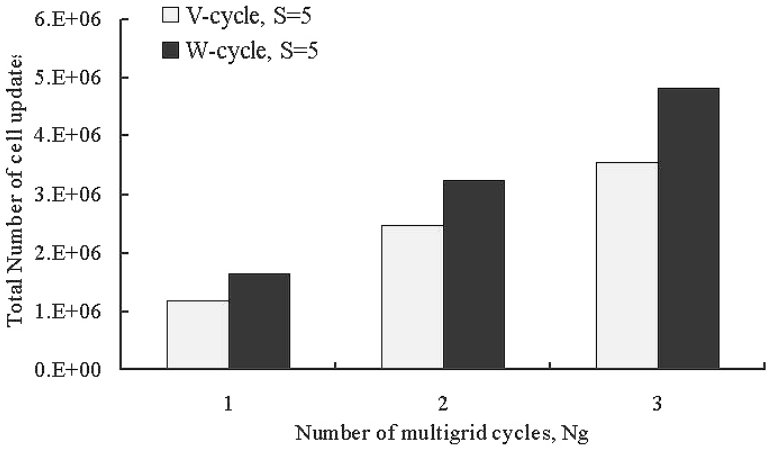


Figure A.19: The design progress of FMG acceleration.





(b) with the smoothing number, $S = 5$

Figure A.20: The total number of cell updates N_t for FMG with N_g variations.

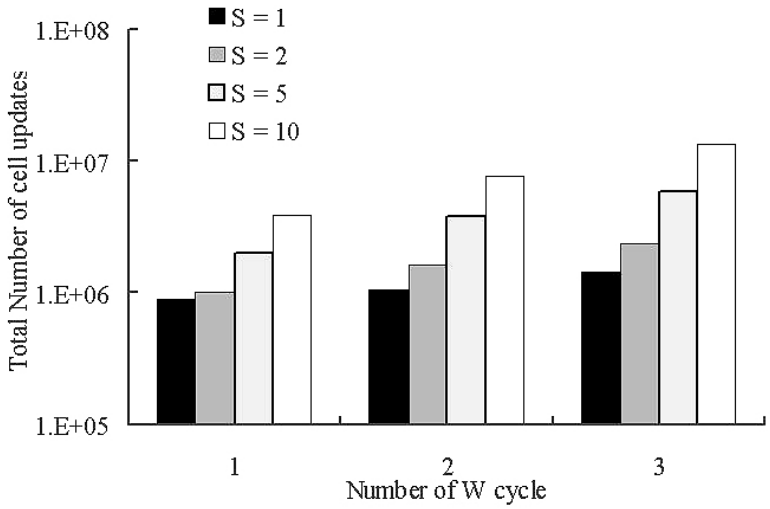


Figure A.21: The total number of cell updates N_t for FMG with S variations.

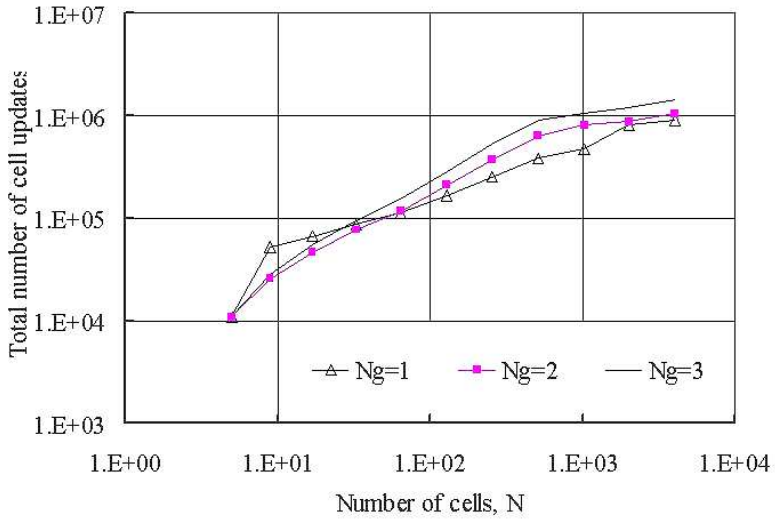


Figure A.22: The total number of cell updates N_t for FMG when $S = 1$ and N_g variations.

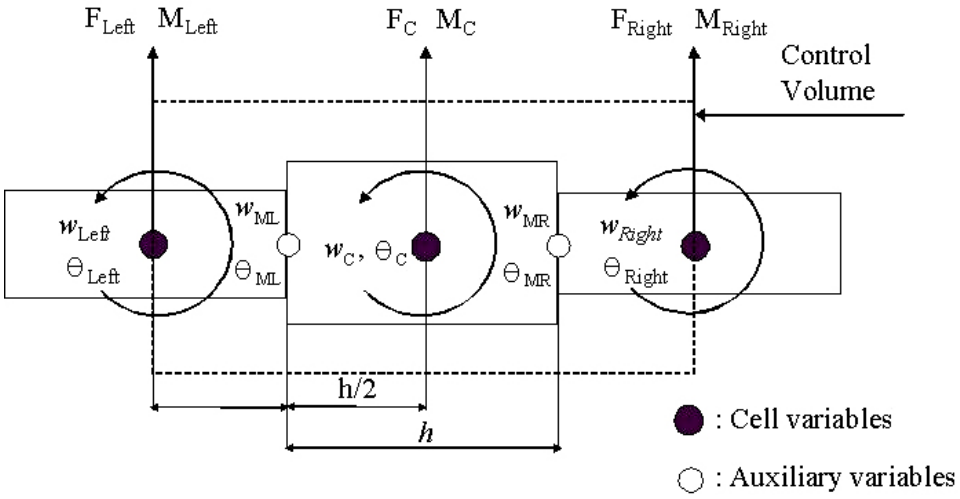


Figure A.23: 1-D Cell neighborhood (cell-centered grid).

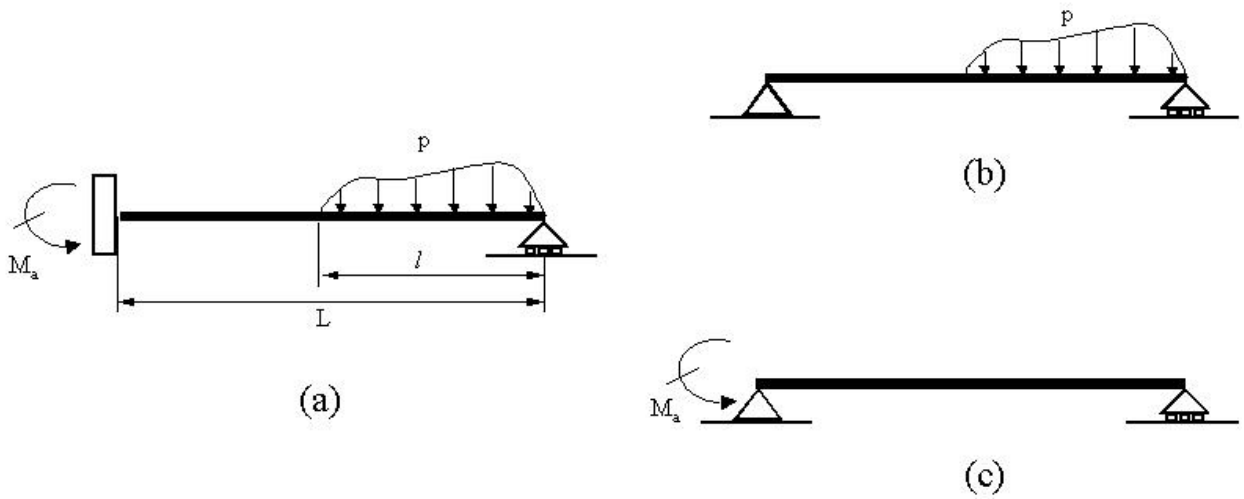


Figure A.24: Method of superposition

Vita

Kim, Sunwook was born in Dae-jun, the Republic of Korea, 1976. He entered Aerospace engineering at In-ha University in 1995 and completed his undergraduate study in 2002. In August 2002, he enrolled in the graduate program in Aerospace and Ocean engineering at Virginia Polytechnic Institute and State University in Blacksburg.