COMPUTER AIDED MACROMODELING FOR MEMS

by

LYNN DANIEL GABBAY

B.S., Applied and Engineering Physics
B.S., Computer Science
Cornell University, 1993

S.M., Electrical Engineering and Computer Science
Massachusetts Institute of Technology, 1995

Submitted to the

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June, 1998

Signature of Author [illegible]
Department of Electrical Engineering and Computer Science
May 21, 1998

Certified by [illegible]
Professor Stephen D. Senturia
Barton L. Weller Professor of Electrical Engineering
Thesis Supervisor

Accepted by [illegible]
Professor Arthur C. Smith
Chair, Department Committee on Graduate Students

Computer Aided Macromodeling for MEMS

by

Lynn Daniel Gabbay

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 1998 in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

The great challenge for MEMS designers is to create low-order dynamic device models (macromodels) that accurately capture the complex behavior that is often discovered only by experiment or by full three-dimensional simulation. In this thesis, we report the successful implementation of a methodology for automatically generating analytical macromodels of non-linear, electrostatically actuated microstructures from meshed simulations and inserting them into system-level simulators. This approach is based upon representing the positional state of a device with a set of generalized coordinates that represent contributions of a set of basis shapes that we derive from the linear elastic modes of the system. Reduction of the state of the system to these generalized coordinates permits us to construct analytical models for the elastostatic energy and the electrostatic co-energy of the system, whose gradients provide us with the actuation forces expressed directly in modal coordinates. We then encapsulate the generalized equations of motion in a circuit element that can be inserted into an analog circuit simulator for dynamics simulation.

Thesis Supervisor: Stephen D. Senturia
Title: Barton L. Weller Professor of Electrical Engineering

Acknowledgments

It is a genuine pleasure for me to be able to take this opportunity to acknowledge the numerous people without whom this work would not have been possible. First and foremost, I want to express my most sincere gratitude to Professor Stephen D. Senturia. Throughout my research, he provided me endless support that I consider to be unparalleled by other research advisors. Above and beyond this, he afforded me his time, patience, and tolerance in addition to his keen, incisive insight and guidance. In short, Professor Senturia enhanced the value and experience of my graduate research immeasurably, and for this I thank him. Additionally, I would like to thank DARPA (Contract J-FBI-95-215) for supporting my research. This research was carried out using the computer facilities of the Microsystems Technology Laboratories at MIT.

The bulk of this research would not have been possible if it were not for the in-depth technical assistance of Microcosm Technologies. I must give an enormous thanks Manish Deshpande and Vladimir Rabinovich for answering my persistent questions about MEMCAD, and in particular to Manish for writing additional code into MEMCAD with the specific intent to assist the software development of my research. I must also send a tremendous thanks to Dr. John Gilbert, who provided guidance and advice as well as made the MEMCAD project possible. I would also like to thank Analogy, and in particular Darrell Teegarden, for providing the SABER tools and Analogy's technical support resources. Additionally, I would like to extend thanks to Darrell Schiebel, current maintainer of the Glish scripting language, who provided excellent technical support beyond his own responsibilities.

I would like to thank the members of the MIT MEMCAD group with whom I have had the pleasure of working. In alphabetical order, these are Professor G. K. Ananthasuresh, Professor Luis Castaner, Erik Deutsch, Gretchen "Scotti" Fuller, Dr. Robert Harris, Elmer Hung, Lily Kim, Dr. Rob Legtenberg, Carol Nicolora, Professor Peter Osterberg, Dr. Bart Romanowicz, Dr. Dan Sobeck, Mathew Varghese, Professor Jacob White, and Joseph Young.

I would be amiss to neglect to mention how much I appreciate my time sharing an office with Elmer Hung. Not only is Elmer is the ideal officemate, but he is also a good friend. We have shared a chemistry in our office like no other, and we have both had tremendous fun. I will always look back on the time we shared with a smile.

Outside of my research environment, there have been a few people whose close friendship kept my spirits up throughout my indentured servitude to MIT. I want to thank my two closest friends from Cornell, Naresh Kannan and Stephen Waite. Although we have been living in separate cities for five years, we have kept in remarkably close contact throughout that time. Their close friendship helped me face the slings and arrows of the Ph.D. program with decisive certainty. They are family to me. Thank you, Naresh. Thank you, Steve.

And finally, I want to thank the two most important women in my life: my mom, Lea Gabbay, and my sister, Sherri Gabbay. They have supported me for every step of my studies without falter. My mother has pushed me to excel academically for as long as I can remember, and I am as proud to present to her this accomplishment as she is to see it. I love them both dearly. I dedicate this thesis to the memory of my father, Professor Edmond J. Gabbay.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Computer aided design (CAD) tools enable the simulation and computational prototyping of devices that may not have been constructed yet. However, because the development of CAD tools is expensive, it requires strong driving forces to develop these tools, and even then CAD capabilities often lag behind the most advanced device technologies. This is the case for the field of microelectromechanical systems (MEMS). MEMS structures are machined on semiconductor wafers using existing VLSI technologies supplemented with specialized processing called "micromachining" [1]. The types of structures can range from gears and motors to deformable thin membranes.

An important class of MEMS devices involves electrostatic actuation of microstructures with moveable or deformable parts. Simulation of electrostatically actuated MEMS structures involves the tight coupling of electrostatic and mechanical forces. In the quasistatic limit, the distribution of charges and the effect of the electrostatic forces upon the deformation of the structure must be determined simultaneously and self-consistently with the deformation itself. This problem can be highly nonlinear because of the inherent nonlinearity of electrostatic actuation forces, and geometric nonlinearities caused by large deformation.

There have been several approaches for developing CAD tools for MEMS devices [2]-[19]. Many of these approaches share an architecture in which fast numerical algorithms are used to optimize simulation in each energy domain separately. Then, the coupled domain problem must be solved self-consistently by suitable iteration to determine the behavior of the system [5]. This technique has the advantage of accuracy, but at the cost of computation time. Commercial packages, such as MEMCAD [4], SOLIDIS [7], and IntelliCAD [9] implement full, self-consistent, three-dimensional simulation of coupled electrostatic-elastostatic devices, and can solve quasistatic or frequency domain behavior. However, it is time consuming and computationally costly to simulate the small-amplitude general dynamical behavior for these coupled non-linear systems. An effective design tool must be capable of handling system dynamics in a timely fashion.

In order to solve this problem, the MEMS design community has been converging upon the technique of solving dynamical behavior by constructing low-order analytical models to agree with full three-dimensional analysis [20]. These models, called *macromodels* or *reduced-order models*, can then be used in lieu of computationally expensive full three-dimensional analysis. It is thus possible to solve system-level dynamics by constructing reduced-order macromodels to represent each energy domain, and then inserting these models into a system-level simulator. Because macromodels are analytical, simulation can be performed without any significant computational overhead; and because they agree with full three-dimensional simulation, they can provide accuracy sufficient for design purposes. It is the purpose of this research to further develop this technique for MEMS simulation.

In order to reduce the complexity of the system, the state of the system can be constrained to be the linear superposition of a set of basis shapes [21]-[24]. These basis shapes can be generated in a variety of ways, such as singular value decomposition [21], Fourier decomposition [22], or Karhunen-Loève decomposition [23]-[24]. One commonly used technique is the normal mode summation method [25]. For example, Lees [26] presented an approximation of a Timoshenko beam that uses the mode shapes to represent some of the degrees of freedom of the structure. Furthermore, this technique has been applied to MEMS simulation specifically. Ananthasuresh [27] presented a technique for reducing the degrees of freedom of a MEMS device by representing the deformation of the structure as a linear superposition of the first few mechanical mode shapes. In this implementation, the electrostatic forces were calculated by approximating the capacitance as a chain of lumped parallel plate capacitors. Cojocaru [28] improved upon this research by executing full 3D simulation at each integration time-step in order to compute electrostatic forces accurately. In both these cases, linearized representations of the mechanical energy domain are used, which neglects any possibility of stress stiffening. The logical progression of this research would be to execute full three-dimensional simulation for every energy domain at each time-step, but this would become computationally cumbersome. Ideally, the 3D simulation should be replaced with accurate, fast-to-compute macromodels.

Macromodels that substitute 3D simulation can be constructed automatically by computer. Milzner [29] presented a good overview of the art of computer aided macromodel construction as applied to circuit simulation. Regidor [30]-[31] presented an algorithm to generate macromodels for thyristors, allowing any computationally expensive tasks to be executed and managed automatically by CAD tools rather than manually by device designers. Gabbay [32] implemented a technique for automatically constructing accurate macromodels for the electrostatic component of a particular class of MEMS devices by automating the calculation of system capacitance over a reduced set of degrees of freedom. In this case, lumped-element beam approximations were used to represent the mechanical forces. The arbitrary choice of generalized coordinates in this work, however, induced inertial inconsistencies. Divekar [33] presented a means and implementation to take n-port parameter data as a function of frequency and generate macromodels in the form of rational polynomials that can be inserted as circuit elements into the SPICE circuit simulator.

In this thesis, we report the successful implementation of a methodology for automatically generating analytical macromodels for two conductor, conservative electrostatic-elastostatic microstructures. These macromodels are exported as analog simulator circuit elements that can be repeatedly used within a circuit simulator to determine dynamical behavior. Above all, these macromodels are fast to compute, requiring only the initial investment of computation time to construct them. This document is organized as follows. Chapter 2 discusses the theory behind the process by which we construct these macromodels. Chapter 3 explains the means by which we implemented this process. Chapter 4 presents analysis of our implementation of this process, in particular the computation time and the comparison of results to full three-dimensional simulation. Chapter 5 provides some examples of this process being executed on a variety of complex devices.

Chapter 2 Theory

In this chapter, we present a theoretical explanation of what we shall refer to as *The Churn Process*. This is a means by which a full three-dimensional meshed numerical model of a two conductor electromechanical device without dissipation can be converted into a reduced-order analytical macromodel that can readily be inserted as a black-box circuit element into an analog circuit simulator. This process is based upon the energy method approach [34], in that we shall construct analytical models for each of the energy domains of the system and determine all forces as gradients of the energy. The energy method approach has the advantage of making this process modular, enabling us to incorporate other energy domains into our models in the future. Another beneficial side effect of energy methods is that the models we shall construct are guaranteed to be energy conserving, because each stored energy shall each be constructed as an analytical function, and all forces shall be computed directly from analytically computed gradients. The Churn process also has the advantage of being able to be performed almost entirely automatically, requiring the designer only to construct the model, run a few full three-dimensional numerical computations, and set a few preferences a priori. Above all, this process has the ultimate benefit of constructing models that are computationally efficient, allowing their use in a dynamical simulator.

A high level description of our approach is depicted in Figure 1. Our first task is to reduce the degrees of freedom of the system. Rather than allow each node in a finite element model to be free to move in any direction, we constrain the motion of the system to a linear superposition of a select set of deformation shapes. This set will act as our basis set of motion. The positional state of the system will hence be reduced to a set of generalized coordinates, each coordinate being the scaling factor by which its corresponding basis shape will contribute. Next, we must construct analytical macromodels of each of the energy domains of the system. In the case of conservative capacitive electromechanical systems, these consist of the electrostatic, elastostatic, and kinetic energy domains. These macromodels will be analytical functions of the generalized coordinates. (As we will see in Section 2.1.1, some of these energy domains will be determined as a byproduct of modal analysis, avoiding the need for explicit calculation.) We can then use Lagrangian mechanics in order to construct the equations of motion of the system in terms of its generalized coordinates. Finally, we can translate these equations of motion into an analog hardware description language, thereby constructing a black-box model of the electromechanical system that can be inserted into an analog circuit simulator.

We shall present this process in three parts. In Section 2.1, we present the means by which we reduce the degrees of freedom of the full system. In Section 2.2, we discuss the algorithm by which we can

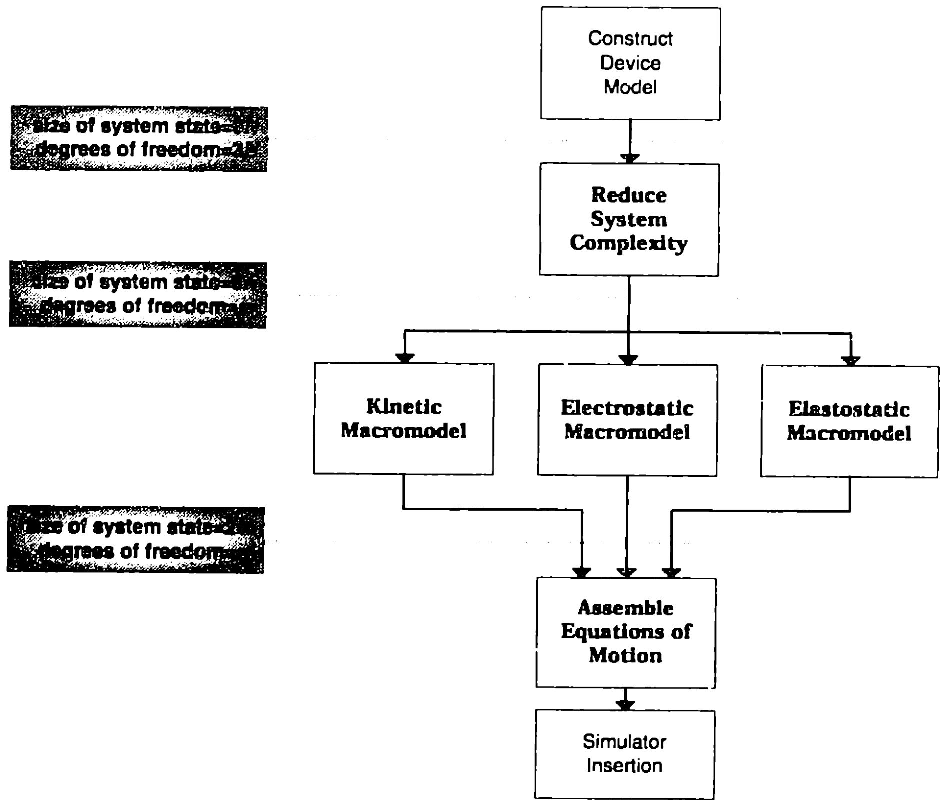


Figure 1: Overview of *The Churn Process*

construct a macromodel for an arbitrary energy domain, in particular, how we apply this to the energy domains of a conservative electromechanical system. Finally, in Section 2.3, we demonstrate how we apply Lagrangian mechanics to this problem, thereby constructing the equations of motion of the system.

2.1 Choosing the Generalized Coordinates

Before the system we wish to analyze is macromodeled, it is represented as a finite element model with N free nodes. Neglecting node rotations, this system has $3N$ degrees of freedom. To represent the dynamical state of the system, $6N$ terms are necessary; $3N$ terms record the node positions, and an additional $3N$ terms record the node velocities. Furthermore, for $6N$ state terms, $6N$ first order differential equations are needed to represent the equations of motion of the system. If N is large, it will be computationally expensive to integrate these equations in time to simulate dynamic behavior.

To solve this problem, we restrict the motion of the system. Let us define ψ to be a $3N$ element vector representing the positional state of the system. We constrain the degrees of freedom of the system by declaring that ψ is a linear superposition of m linearly independent basis shapes φ_i offset from an equilibrium state ψ_{eqm}. Note that this equilibrium state is the shape of the structure *after* any initial stress relaxations take place; naturally, if there are no initial stresses, or if the system is clamped so as to prevent any relaxation, the basic shape is already at its equilibrium position. Thus, we rewrite our representation of the state of the system as

$$\psi = \psi_{\text{eqm}} + \sum_{i=1}^{m} q_i \varphi_i \tag{1}$$

where q_i are the coefficients of the basis shapes of the linear superposition. Henceforth, we refer to the q_i as the generalized coordinates of the system. In effect, this constrains the description of the system from $3N$ to m spatial degrees of freedom. Correspondingly, this reduces the number of terms needed to represent the dynamical state of the system, and thus the number of first order differential equations in the equations of motion, to $2m$. When $m << N$, this constitutes a significant computational advantage over the full $6N$ state representation. (In this research, we have found that representing state as a linear superposition of displacements can create problems in situations where there is a sensitive dependence upon node position. We discuss this in Section 4.2.)

When choosing the basis shapes, the designer is faced with two questions. The first is how to find a good set of basis shapes φ_i. And, because any complete basis set of a $3N$ element vector can have $3N$ linearly independent basis shapes, the second question is which shapes from the complete set should be used? There are numerous ways to construct the shapes among which we will choose. For example, in [26], Ananthasuresh demonstrated the use of mechanical harmonic mode shapes for use with MEMS device-model complexity reduction. In [21], Hung presents a technique for using actual system motion to construct a set of representative basis shapes. Any or all of these methods are valid. For this research, we use mechanical harmonic mode shapes for our basis shapes, which we discuss in detail in Section 2.1.1.

Once a set of M basis shapes is chosen, we must choose the minimum number m of those shapes necessary to characterize typical motion behavior of the structure. Our approach begins with a single full three-dimensional quasistatic simulation for the system under a typical example of actuation. Let us define ψ_{ex} to be the positional state calculated by our example quasistatic simulation. In Section 2.1.2, we shall discuss how we can determine the coefficients c_i such that

$$\psi_{\text{ex}} = \psi_{\text{eqm}} + \sum_{i=1}^{M} c_i \varphi_i$$

Let us define the relative significance γ_i of a basis shape φ_i to be the maximum absolute displacement caused by that shape in the linear superposition for ψ_{ex}. In other words,

$$\gamma_i \equiv |c_i| \cdot \|\varphi_i\|_\infty \tag{2}$$

where $\|\varphi_i\|_\infty$ is the L_∞ norm of φ_i, equivalent to the absolute magnitude of the maximum element in the φ_i vector. By sorting the γ_i values in decreasing order, we construct a prioritized list of which basis shapes have the greatest significance on our example motion. The designer can now decide which and how many shapes to use in the reduced model of the system.

One final note is that when we construct analytical models for the various energy domains in our system, we will need to know a valid operating range for our system; that is, typical values for our generalized coordinates. Conveniently, the coefficients c_i we obtained for our example of typical motion can be used to understand the relative expected magnitudes of the generalized coordinates during dynamic simulation. By providing c_i, we enable the designers to make an educated selection for the system operating range.

2.1.1 Using Mode Shapes as a Basis Set

In [26], Ananthasuresh demonstrated that only a few mechanical mode shapes are necessary to accurately capture the motion of simple MEMS devices. For this research, we also choose to use mode shapes to populate our set of basis shapes. Numerical modal analysis solvers determine the eigenvalues of the equation

$$\left(-\omega_i{}^2\mathbf{M}+\mathbf{K}\right)\varphi_i = 0$$

where φ_i is the eigenvector describing the shape of the mode of vibration, ω_i is the angular frequency of that mode, and $\mathbf{M}$ and $\mathbf{K}$ are the mass and stiffness matrices, defined from the finite element model and its material properties. For each mode, we can also compute a generalized mass, given by

$$m_i = \varphi_i{}^{\mathrm{T}}\mathbf{M}\varphi_i$$

There are several advantages to using the shapes derived from modal analysis. The first and most evident is that the mode shapes constitute a linearly independent set of basis shapes; thus it is not necessary to perform a back orthogonalization to confirm the independence of each additional mode shape. Second, the modal formulation readily provides a representation for the kinetic energy of the system $T(\dot{q})$, given by

$$T(\dot{q}) = \sum_i \frac{1}{2} m_i \dot{q}_i^2 \qquad (3)$$

Third, the modal formulation also readily provides a linear representation for the elastostatic strain energy of the system $U_{\text{linear strain}}(q)$, given by

$$U_{\text{linear strain}}(q) = \sum_i \frac{1}{2} m_i \omega_i^2 q_i^2 \qquad (4)$$

We shall use the energy representations in both equations (3) and (4) in Section 2.3, where we discuss the assembly of the equations of motion of the system.

2.1.2 Determination of Mode Relevance

Now that the first few mechanical modes of the moving part of the device have been calculated, and an example of how the structure might bend due to electrostatic actuation has been computed, we can determine the relative significance of each of the mode shapes with respect to actuation. We accomplish this by projecting the deformation of our example quasistatic solution onto the space spanned by the deformations of the calculated mode shapes. This is done as follows. Let us define ψ_{ex} to be the vector representing the deformation of the sample quasistatic equilibrium solution, and let us define ϕ_i to be the vector representing the deformation calculated for the i^{th} mode. Our goal is to calculate the coefficients c_i such that:

$$\psi_{\text{ex}} = \psi_{\text{eqm}} + \sum_i c_i \phi_i$$

We can rewrite this in a matrix form as:

$$\begin{bmatrix} \\ \phi_1 & \cdots & \phi_m \\ \\ \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} \\ \psi_{\text{ex}} - \psi_{\text{eqm}} \\ \\ \end{bmatrix}$$

Because this is an overdetermined system when $m < N$, we use the QR factorization algorithm, which uses a least squares approach to solve for the coefficients c_i [35]. Using this technique, we create a non-square, $m \times N$ matrix that is effectively the inverse of the $N \times m$ matrix of the ϕ_i basis vectors.

2.2 Energy Domain Macromodeling

The second step in the churn process is to construct macromodels that will replace full three-dimensional simulation to evaluate each of the energy domains of the system. Some of the energy domains can already be represented in a simple manner, such as the kinetic energy and a linearized approximation of the elastostatic energy domain as discussed in Section 2.1.1. Here, we shall present how we construct macromodels for any function that would normally require full three-dimensional simulation and then apply this technique to create macromodels for the electrostatic and geometrically non-linear elastostatic energy domains.

The requirements for the macromodel we wish to construct are as follows.

- The macromodel must be an analytical function.
- The macromodel must compare accurately to full three-dimensional simulation.
- The macromodel must be a function of the generalized coordinates only.

The process by which we create such a macromodel is depicted in Figure 2. The overall concept is that the full three-dimensional simulation in the single energy domain of interest is run several times for values of the generalized coordinates that adequately span the pre-chosen operating range for the system. (Because these are single energy-domain simulations, they are much faster than the single coupled-simulation used to find the example deformation.) Then, we select a generalized functional form, on which we then use a non-linear function fitting scheme in order to determine the parameters that fit this generalized form to the data.

In Section 2.2.1, we discuss the process by which we choose the values of the generalized coordinates (or *sampling points*) in order to adequately span the operating range. In Section 2.2.2, we discuss the algorithm by which we fit a generalized functional form to the accumulated data. Finally, in sections 2.2.3 and 2.2.4, we present the full three-dimensional functions we must calculate for the electrostatic and non-linear elastostatic energy domains and the generalized functional forms we use to represent them.

2.2.1 Randomizing Sample Points of the Operating Range

We need to compute the data to which we shall fit our macromodel so that we minimize the number of computations while spanning the operating range sufficiently to accurately capture the physics with the data. There are numerous algorithms with which we could choose the sample points within this operating range. For example, one might be to break the range into a grid and sample at each point on the grid. Although this would sufficiently span the space, it might result in redundant information, hence needless computations. Another might be to choose the sample points completely randomly. Although this might avoid redundant information, it might also miss regions of the operating range entirely. For this research, we use a combination of these two approaches, based in part upon the Latin Hypercube method [36].

Suppose our generalized coordinates q_i for $i \in 1..m$ are constrained to the operating range defined by $q_i \in [\min_i, \max_i]$. In $\Re^m$ space, this defines an m-dimensional volume. Our first step is to divide this volume into a grid of subvolumes. For each dimension i, the designer can choose the number n_i of equal divisions to be made on that dimension of the volume. This divides the volume into

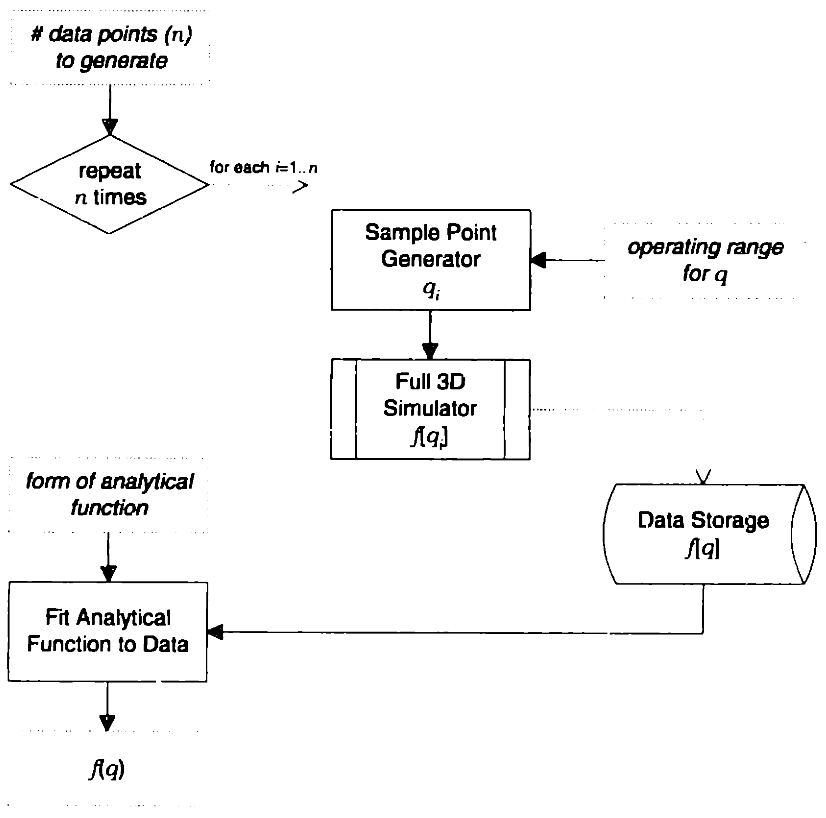
$$\prod_{i=1}^{m} n_i$$

subvolumes. For nomenclature, we shall refer to a volume divided in this manner as a $[n_1 \quad n_2 \quad \cdots \quad n_m]$ sampling volume, and we shall use $[k_1 \quad k_2 \quad \cdots \quad k_m]$ to refer to the individual subvolume within the k_ith division of the ith dimension of the volume for all $i \in 1..m$.

In order to choose a sampling point, we first choose a subvolume and then pick a sampling point purely randomly from within that subvolume. In order to choose subvolumes separated enough to avoid clustering sample points, we shall associate three different variables to each subvolume. These variables will act as counters, and all will be initialized to zero. The first counter, which we shall call **chosen**, will keep track of how many sample points have been selected out of that subvolume. The second counter, which we shall call **nearest neighbors**, will count the number of times a sample point was selected from within a nearest neighbor subvolume. The third, which we shall call **co-linear neighbors**, will count the number of times a sample point was selected from within a subvolume that is co-linear with it in the overall volume. To express this mathematically, we define the counting algorithm as follows:

- when we choose a sample point within subvolume $[k_1 \quad k_2 \quad \cdots \quad k_m]$:
  - the chosen variable for subvolume $[k_1 \quad k_2 \quad \cdots \quad k_m]$ is incremented by one
  - the nearest neighbors variable for all subvolumes $[l_1 \quad l_2 \quad \cdots \quad l_m]$ such that $\left\{ \exists i \in 1..m \left| \begin{matrix} l_j = k_j & j \neq i \\ l_j = k_j \pm 1 & j = i \end{matrix} \right. \right\}$ are incremented by one
  - the co-linear neighbors variable for all subvolumes $[l_1 \quad l_2 \quad \cdots \quad l_m]$ such that $\left\{ \exists i \in 1..m \left| \begin{matrix} l_j = k_j & j \neq i \\ l_j \neq k_j & j = i \end{matrix} \right. \right\}$ are incremented by one

Thus, in order to choose the next sample point, we pick a subvolume that has the lowest values for the chosen, nearest neighbors, and co-linear neighbors variables. We define the selection algorithm



# data points (n) to generate
repeat n times
for each i=1..n
Sample Point Generator q_i
operating range for q
Full 3D Simulator f[q]
Data Storage f[q]
form of analytical function
Fit Analytical Function to Data
f(q)

Figure 2: Energy Domain Macromodeling

as the method we use to choose each successive subvolume, and it is describes as follows. (Note that this algorithm is repeated for each selection of a sample point)

1. Initialize a set of subvolumes that contains all of the subvolumes in the volume.
2. Among all subvolumes in the set, determine the minimum value contained in the chosen variables.
3. Remove all subvolumes from the set whose chosen variables are greater than the minimum just calculated. This is equivalent to only keeping the subvolumes that have the minimum chosen value.
4. Among all subvolumes in the set, determine the minimum value contained in the nearest neighbors variables.
5. Remove all subvolumes from the set whose nearest neighbors variables are greater than the minimum just calculated. This is equivalent to only keeping the subvolumes that have the minimum nearest neighbors value.
6. Among all subvolumes in the set, determine the minimum value contained in the co-linear neighbors variables.
7. Remove all subvolumes from the set whose co-linear neighbors variables are greater than the minimum just calculated. This is equivalent to only keeping the subvolumes that have the minimum co-linear neighbors value.

A noteworthy byproduct of this algorithm is that if we request as many sample points as there are subvolumes, one sample point will have been chosen from each and every subvolume.

Let us recap and summarize the sample point selection algorithm:

1. Construct the operating volume and divide it into its subvolumes.
2. Create the chosen, nearest neighbors, and co-linear neighbors variables for all subvolumes, and initialize all of them to zero.
3. Pick a subvolume from which to choose a sample point using the selection algorithm defined above.
4. Choose a random point within that subvolume, use this as a sampling point.
5. Increment the appropriate variables according to the counting algorithm defined above.
6. Return to step 3.

2.2.2 Levenberg-Marquardt Non-Linear Function Fitting

After the set of simulations is complete, we will fit an analytical model to the acquired data by using a non-linear function fitting scheme. We implement the Levenberg-Marquardt method, which determines the best-fit parameters that minimize the χ^2 merit function [37]. Let us define that the model to be fitted is

$$y = y(x;\mathbf{a})$$

where the parameters $\mathbf{a}$ are what we must alter to minimize the χ^2 merit function, given by

$$\chi^2(\mathbf{a}) = \sum_{i=1}^{N}\left[\frac{y_i - y(x_i;\mathbf{a})}{\sigma_i}\right]^2$$

where σ_i is the standard deviation associated with the data point x_i. Near the minimum of χ^2, we expect the function to be well approximated by a quadratic form, which we can write as

$$\chi^2(\mathbf{a}) \approx \gamma - \mathbf{d}^{\mathrm{T}}\mathbf{a} + \frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{D}\mathbf{a} \qquad (5)$$

where **d** and **D** are determined simply from the first and second derivatives of χ^2, respectively. If this is a good approximation, we can use the inverse Hessian method to jump directly from the current trial parameters $\mathbf{a}_{\mathrm{cur}}$ to the minimizing ones $\mathbf{a}_{\mathrm{min}}$ in a single step, namely

$$\mathbf{a}_{\mathrm{min}} = \mathbf{a}_{\mathrm{cur}} + \mathbf{D}^{-1} \cdot \left[-\nabla\chi^2(\mathbf{a}_{\mathrm{cur}})\right] \qquad (6)$$

On the other hand, if this is a poor approximation, we can take a step down the gradient, as in the steepest descent method. In other words,

$$\mathbf{a}_{\mathrm{next}} = \mathbf{a}_{\mathrm{cur}} - \text{constant} \times \nabla\chi^2(\mathbf{a}_{\mathrm{cur}}) \qquad (7)$$

The advantage of the Levenberg-Marquardt method is that it varies smoothly between both of the inverse Hessian method and the steepest descent method. Both require the computation of the gradient of χ^2 with respect to the parameters **a**, which will be zero at the χ^2 minimum. This has the components

$$\frac{\partial\chi^2}{\partial a_k} = -2\sum_{i=1}^{N}\frac{\left[y_i - y(x_i;\mathbf{a})\right]}{\sigma_i^2}\frac{\partial y(x_i;\mathbf{a})}{\partial a_k} \qquad k = 1..M$$

Taking an additional partial derivative yields

$$\frac{\partial^2\chi^2}{\partial a_k \partial a_l} = 2\sum_{i=1}^{N}\frac{1}{\sigma_i^2}\left[\frac{\partial y(x_i;\mathbf{a})}{\partial a_k}\frac{\partial y(x_i;\mathbf{a})}{\partial a_l} - \left[y_i - y(x_i;\mathbf{a})\right]\frac{\partial^2 y(x_i;\mathbf{a})}{\partial a_k \partial a_l}\right]$$

We make the following substitutions

$$\alpha = \left[\alpha_{kl}\right] \text{ where } \alpha_{kl} \equiv \frac{1}{2}\frac{\partial^2\chi^2}{\partial a_k \partial a_l}$$

$$\beta = \left[\beta_k\right] \text{ where } \beta_k \equiv -\frac{1}{2}\frac{\partial\chi^2}{\partial a_k}$$

making $\alpha=½\mathbf{D}$ and $\beta=½\mathbf{d}$ from equation (5). Frequently, it can be beneficial to neglect the second derivative term within α, thus from this point on, we will always use as the definition of α the formula

$$\alpha_{kl} = \sum_{i=1}^{N}\frac{1}{\sigma_i^2}\left[\frac{\partial y(x_i;\mathbf{a})}{\partial a_k}\frac{\partial y(x_i;\mathbf{a})}{\partial a_l}\right]$$

With these substitutions, we can rewrite equations (6) and (7), that is the inverse Hessian and steepest descent methods, as

$$\alpha\,\delta\mathbf{a} = \beta \qquad (8)$$

and

$$\delta \mathbf{a} = \text{constant} \times \beta \tag{9}$$

The Levenberg-Marquardt method combines these methods by constructing a new matrix α' such that

$$\alpha' = [\alpha'_{kl}] \text{ where } \alpha'_{kl} = \alpha_{kl}(1 + \lambda \delta_{kl})$$

where δ_{kl} is the Kronecker delta. We can now combine both equations (8) and (9) into one equation

$$\alpha' \delta \mathbf{a} = \beta$$

The magnitude of the non-dimensional parameter λ allows us to smoothly traverse from using inverse Hessian to steepest descent, and vice versa.

The Levenberg-Marquardt method is summarized as follows:

1. Pick a modest value for λ, say λ=0.001.
2. Compute $\chi^2(\mathbf{a})$.
3. Compute α, β
4. Compute α'
5. Solve $\alpha' \delta \mathbf{a} = \beta$ for $\delta \mathbf{a}$.
6. Compute $\chi^2(\mathbf{a} + \delta \mathbf{a})$.
7. If $\left[\chi^2(\mathbf{a} + \delta \mathbf{a}) - \chi^2(\mathbf{a})\right] / \chi^2(\mathbf{a}) \leq$ some cutoff value, terminate.
8. If $\chi^2(\mathbf{a} + \delta \mathbf{a}) \geq \chi^2(\mathbf{a})$, increase λ by 10 (or any other substantial factor) and return to step 4.
9. Let $\mathbf{a} \leftarrow \mathbf{a} + \delta \mathbf{a}$
10. Decrease λ by 10 (or any other substantial factor)
11. Go to step 3.

The continued reduction of λ in step 10 tends toward inverse Hessian as we approach the χ^2 minimum.

2.2.3 Electrostatic Energy Domain

Electrostatic forces are produced by the charges that accumulate on the conductor surfaces of the MEMS device under an applied voltage. The force $\mathbf{F}_e$ that stores energy into the electrostatic domain of the system is given by the gradient of the electrostatic energy U_e:

$$\mathbf{F}_e = \nabla U_e \tag{10}$$

Recall that the electrostatic energy U_e is given by

$$U_e = \frac{1}{2}\frac{Q^2}{C}$$

where Q is the charge on the conductors, and C is the capacitance between the conductors. When actuating a MEMS device, however, we prefer to work in terms of voltage rather than charge. Thus, it is generally advantageous for us to consider the electrostatic co-energy U_e^*, given by

$$U_e^* = \frac{1}{2}CV^2$$

where V is the applied voltage. In this case, the gradient of the electrostatic co-energy U_e^* is the force that draws energy out of the electrostatic domain. Thus, the force $\mathbf{F}_e$ that puts energy into the electrostatic domain is given by

$$\mathbf{F}_e = -\nabla U_e^* \tag{11}$$

When comparing Equations (10) and (11), we note an often overlooked sign change issue in MEMS. Because we choose to determine our electrostatic forces from the electrostatic co-energy rather than energy, a sign change must be taken into account.

Let us return to determining the electrostatic forces. Because the applied voltage is independent of motion, the gradient need only be applied to the capacitance, thus:

$$\mathbf{F}_e = -\left(\frac{1}{2}V^2\right)\nabla C$$

In order to construct a macromodel of the electrostatic domain, we must construct an analytical model of the capacitance of the system. We shall use an arbitrary multivariate form to represent the capacitance, but it should be designed such that it can capture the geometric non-linearities that can be expected from a capacitance function. Recall that the capacitance of a large parallel plate capacitor neglecting fringe field effects is given by

$$C = \frac{\varepsilon_0 A}{d}$$

where A is the area of the plate, and d is the distance between the plates. Our generalized coordinates would most correspond to the gap d. Thus, it makes sense that our analytical form should have denominator terms. In this research, we use the form of a rational fraction of multivariate Taylor polynomials to represent the capacitance function. This form is given by:

$$\frac{\sum_{i_1=0}^{R_1}\sum_{i_2=0}^{R_2}\cdots\sum_{i_m=0}^{R_m} a_{i_1 i_2 \cdots i_m} q_1^{i_1} q_2^{i_2} \cdots q_m^{i_m}}{\sum_{i_1=0}^{S_1}\sum_{i_2=0}^{S_2}\cdots\sum_{i_m=0}^{S_m} b_{i_1 i_2 \cdots i_m} q_1^{i_1} q_2^{i_2} \cdots q_m^{i_m}}$$

Henceforth, we shall refer to this as a $[R_1 \quad R_2 \quad \cdots \quad R_m/S_1 \quad S_2 \quad \cdots \quad S_m]$ model.

2.2.4 Elastostatic Energy Domain

Elastostatic forces result from the strain energy stored in the body of the system. The strain energy is given by:

$$U_m = \int_{\text{vol}} \frac{1}{2} \sigma \varepsilon \, dV \tag{12}$$

where σ is the stress and ε is the strain in the system. The force $\mathbf{F}_m$ that stores energy into the elastostatic domain of the system is given by the gradient of the elastostatic energy U_m:

$$\mathbf{F}_m = \nabla U_m$$

Because the strain energy can be computed directly from full three-dimensional simulation, we macromodel the elastostatic energy domain by constructing an analytical function to fit the strain energy directly.

In this research, we use the form of a multivariate Taylor polynomial to represent the strain energy function. This form is given by:

$$\sum_{i_1=0}^{R_1} \sum_{i_2=0}^{R_2} \cdots \sum_{i_m=0}^{R_m} a_{i_1 i_2 \cdots i_m} q_1^{i_1} q_2^{i_2} \cdots q_m^{i_m}$$

Henceforth, we shall refer to this as a $\begin{bmatrix} R_1 & R_2 & \cdots & R_m \end{bmatrix}$ model.

2.3 Assembling the Equations of Motion

Given representations for the kinetic and potential energy domains of a system, we can use Lagrangian mechanics to construct the equations of motion [38]. Recall that the Lagrangian $L(q,\dot{q},t)$ is a function of the general coordinates q, their first time derivatives $\dot{q}$, and time t. $L(q,\dot{q},t)$ is defined by

$$L(q,\dot{q},t) = T(q,\dot{q},t) - U(q,\dot{q},t)$$

where $T(q,\dot{q},t)$ is the kinetic energy and $U(q,\dot{q},t)$ is the potential energy of the system. The equations of motion come directly from Lagrange's equations, given by

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0 \tag{13}$$

representing a set of m equations, one for each generalized coordinate q_i. Recall that in equation (3), we formulated the kinetic energy from the modal analysis, yielding

$$T(q,\dot{q},t) = \sum_{i=1}^{m} \frac{1}{2} m_i \dot{q}_i^2$$

Combining this with Lagrange's equations of (13) yields

$$m_i \ddot{q}_i = \frac{d}{dt}\left(\frac{\partial U}{\partial \dot{q}_i}\right) - \frac{\partial U}{\partial q_i}$$

In general, potential energy $U(q,\dot{q},t)$ is the sum of the energy domains of the system, which we express by

$$U(q,\dot{q},t) = \sum_{\substack{\text{each energy} \\ \text{domain } d}} U_d(q,\dot{q},t)$$

For our systems, the only energy domains we consider are the electrostatic and elastostatic domains, both of which are conservative and do not depend upon $\dot{q}$. Thus, the partial derivative with respect to $\dot{q}_i$ drops out, yielding

$$m_i\ddot{q}_i = -\frac{\partial U_{\text{electrostatic}}}{\partial q_i} - \frac{\partial U_{\text{elastostatic}}}{\partial q_i}$$

Recall that for our formulation of the electrostatic energy domain, as discussed in Section 2.2.1, we choose to represent that energy in terms of the co-energy, given by

$$U^*_{\text{electrostatic}}(q,t) = \frac{1}{2}V(t)^2\, C(q)$$

As we noted, we must take into account a sign change due to the use of the co-energy in place of the energy. Thus, for energy domains using co-energy, we shall make the substitution:

$$+\frac{\partial U^*}{\partial q_i} = -\frac{\partial U}{\partial q_i}$$

With this substitution, we can now insert the electrostatic co-energy into Lagrange's equations, yielding:

$$m_i\ddot{q}_i = +\frac{1}{2}V(t)^2\frac{\partial C(q)}{\partial q_i} - \frac{\partial U_{\text{elastostatic}}}{\partial q_i} \tag{14}$$

All that remains is to include the elastostatic energy domain in the system. Recall that we have two alternatives for representing the elastostatic energy domain. The first is to use an analytical fit of the full three-dimensionally simulated non-linear strain energy, as discussed in Section 2.2.4. This leaves Lagrange's equations relatively unchanged, yielding

$$m_i\ddot{q}_i = +\frac{1}{2}V(t)^2\frac{\partial C(q)}{\partial q_i} - \frac{\partial U_{\text{strain}}(q)}{\partial q_i} \tag{15}$$

The second is to use the linearized representation of the strain energy that can be extracted from modal analysis, as discussed in Section 2.1.1. Recall that the linearized elastostatic energy is represented by

$$U_{\text{linearized strain}}(q) = \sum_{i=1}^{m}\frac{1}{2}m_i\omega_i^2 q_i^2$$

Incorporating this into equation (14) yields

$$m_i\ddot{q}_i = +\frac{1}{2}V(t)^2\frac{\partial C(q)}{\partial q_i} - m_i\omega_i^2 q_i \tag{16}$$

The sets of equations represented by either (15) or (16), depending upon which strain energy representation we choose, become our equations of motion. Because our representations of the capacitance and the non-linear strain energy are analytical functions, we can compute the gradients of these functions analytically rather than numerically. This averts the possibility of numerical error creating hidden energy sources or sinks, thereby creating or destroying energy arbitrarily within our equations of motion.

2.4 Summary

In this chapter, we have proposed an algorithm by which a full three-dimensional model of a conservative electromechanical device can be reduced to a black-box macromodel that can be inserted easily into an analog circuit simulator. In Chapter 3, we shall present an implementation of this algorithm and apply it to a simple electromechanical structure.

Before closing, we present a more detailed flowchart describing the Churn process. As discussed in sections 2.1.1 and 2.2.4, we have two ways to macromodel the elastostatic energy domain. The first is to use the linearized elastostatic energy domain determined from modal analysis; this case is depicted in detail in Figure 3. The other is to use a non-linear elastostatic energy domain determined from full three-dimensional simulation, as depicted in detail in Figure 4.

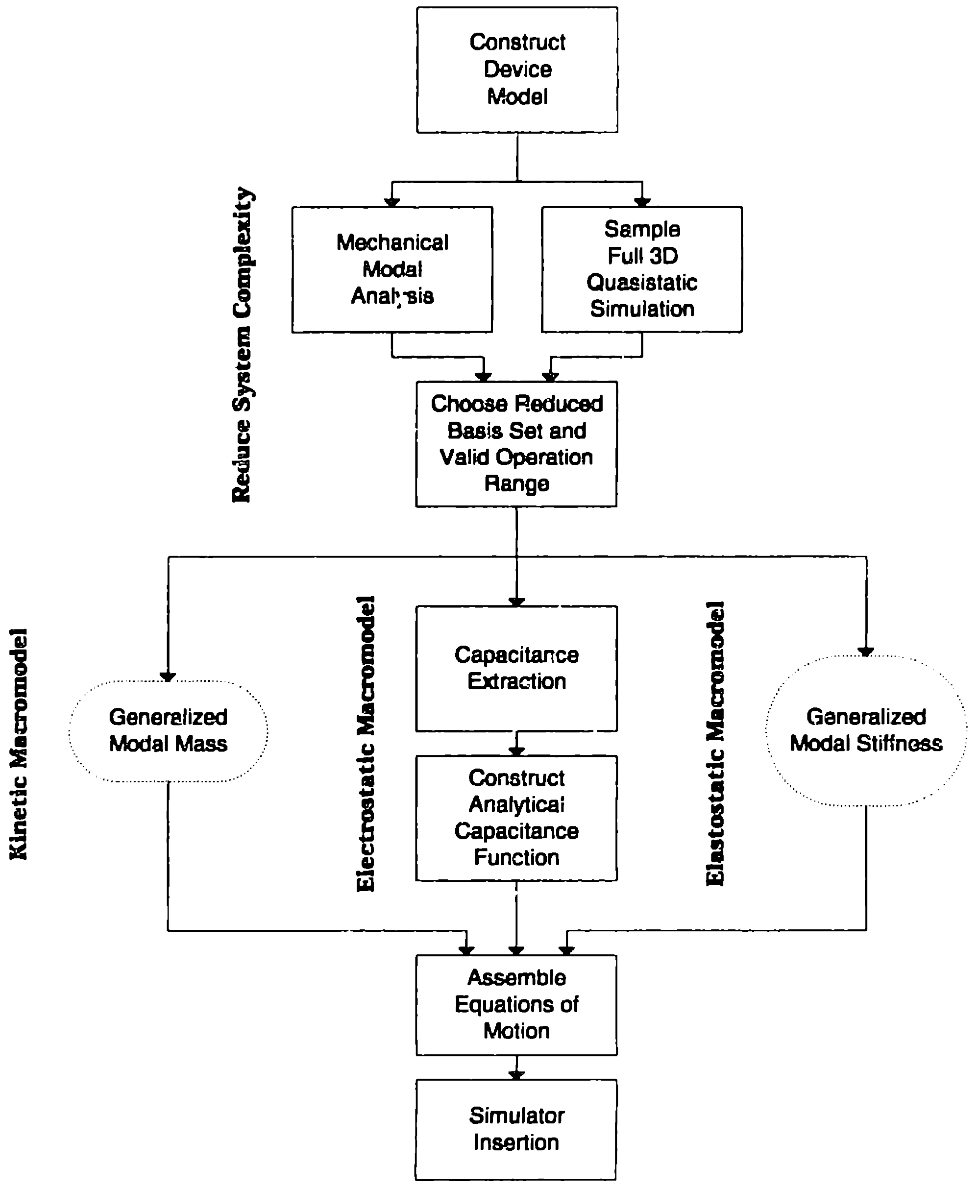

Construct Device Model
Reduce System Complexity
Mechanical Modal Analysis
Sample Full 3D Quasistatic Simulation
Choose Reduced Basis Set and Valid Operation Range
Kinetic Macromodel
Generalized Modal Mass
Electrostatic Macromodel
Capacitance Extraction
Construct Analytical Capacitance Function
Elastostatic Macromodel
Generalized Modal Stiffness
Assemble Equations of Motion
Simulator Insertion

Figure 3: Automatic MEMS macromodeling process with linearized mechanics

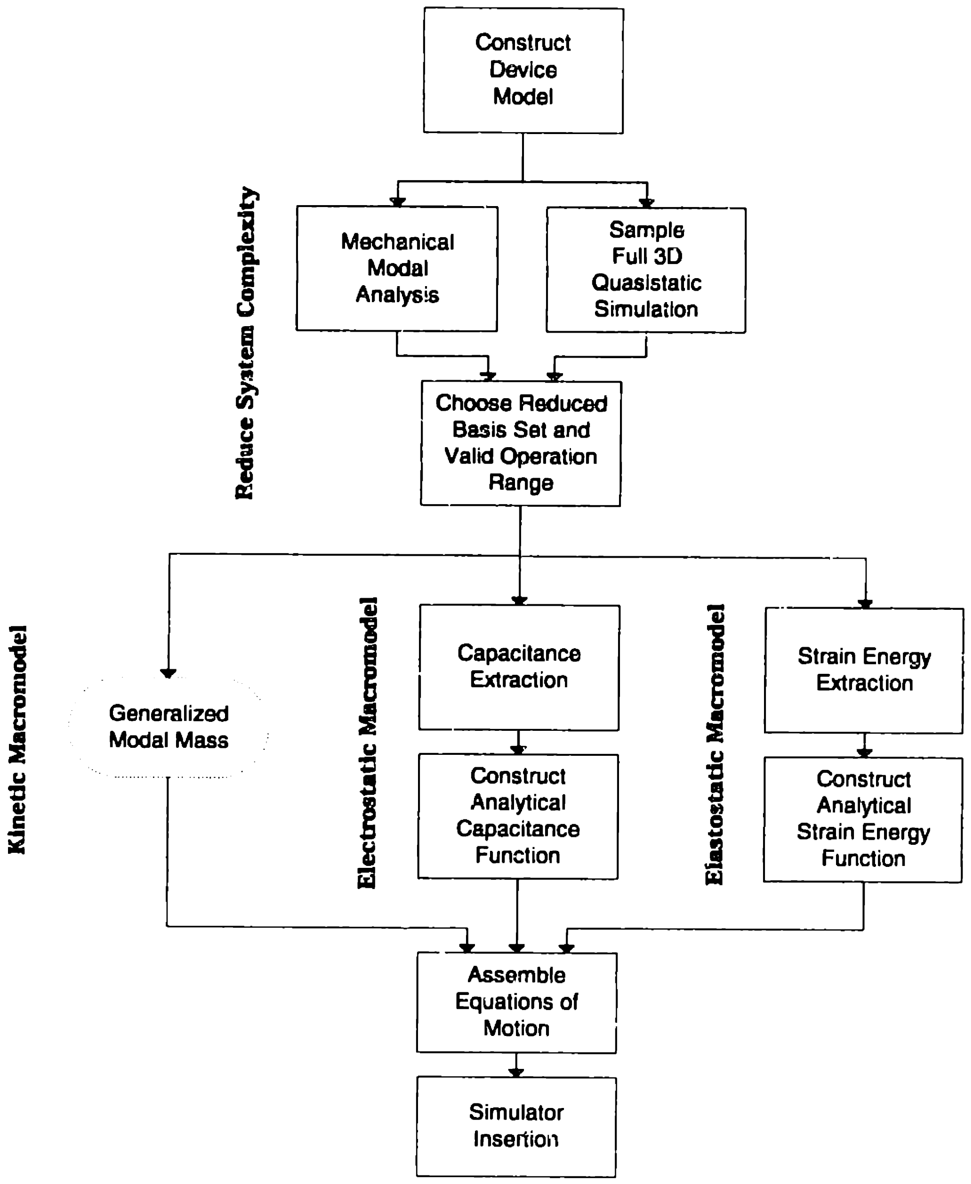

Construct Device Model
Mechanical Modal Analysis
Sample Full 3D Quasistatic Simulation
Reduce System Complexity
Choose Reduced Basis Set and Valid Operation Range
Kinetic Macromodel
Generalized Modal Mass
Electrostatic Macromodel
Capacitance Extraction
Construct Analytical Capacitance Function
Elastostatic Macromodel
Strain Energy Extraction
Construct Analytical Strain Energy Function
Assemble Equations of Motion
Simulator Insertion

Figure 4: Automatic MEMS macromodeling process with non-linear mechanics

Chapter 3 Implementation

In this chapter, we shall present the means by which we have implemented the Churn process, as depicted in Chapter 2. In Section 3.1, we discuss what software utilities we shall use and how these tools can be used in conjunction with each other. In Section 3.2, we present an overview of the implementation of the Churn process, including a brief introduction to the software utilities that were developed for the purposes of this research. In Section 3.3, we conclude with a walkthrough of the Churn process as we have implemented it using the example of a simple electrostatically actuated fixed-fixed beam device. In Chapter 5, we present the results of three additional devices that exhibit more complex behavior.

3.1 Tools

The Churn process exploits the inherent programmability of the MEMCAD suite [3]. MEMCAD is a software package that was originally developed at MIT and is now licensed by Microcosm Technologies, Inc. [6] for commercial development and distribution. It provides many useful features for MEMS designers that enable them to perform computational analysis of MEMS devices. Its MemBuilder utility [4] allows the designer to generate a solid model of a MEMS device automatically by merely specifying a process flow and mask set. MEMCAD includes the ability to construct finite element models of MEMS devices and set their material properties and boundary conditions. It can perform modal analysis of a MEMS device, thus calculating the shapes, generalized masses and frequencies for any number of modes. Most notably, MEMCAD provides CoSolve-EM [5], a utility to solve the coupled, non-linear quasistatic electrostatic-elastostatic equilibrium problem.

MEMCAD itself is a software layer that lies on top of existing modeling utilities and numerical solvers for various energy domains. MEMCAD uses I-DEAS [39] to construct solid and finite element models. It uses a hybrid of FastCap [40] to determine the capacitance matrix for a set of finite element modeled conductors. It uses ABAQUS [41] to solve a variety of mechanical analysis problems. Each of these individual packages uses a different format of input and output for its specification of the problem. MEMCAD introduces MEMBase and the MBIF standard, an application programming interface (API) and file format, respectively, that enable the interchange of MEMS models and problems between the various external utilities used within the MEMCAD suite.

The underlying control of MEMCAD lies in a hybrid of a publicly available scripting language called Glish [42]. This language was designed to enable communication between concurrently running processes. These processes are binary executables that have been specifically compiled to be Glish clients. MEMCAD provides a Glish client for each numerical solver, enabling developers to write their own MEMCAD routines. The primary component of the Churn process is written as a Glish script, automating computation according to a minor amount of user input queried a priori. Many of the utilities necessary to enable the Churn process are written and compiled as Glish clients.

This research is highly dependent upon numerical analysis, thus it is necessary to have an optimized mathematical routine library. For this implementation, we have chosen to use MATLAB [43]. MATLAB provides a C library that can be included into source code and linked into executables. Most of the Glish clients developed for the Churn process link to the MATLAB library.

3.2 Implementation Architecture Overview

In this research, we have implemented the Churn process as a two stage process; this is depicted in Figure 5. The first stage is to prepare the model for Churn analysis by using the MEMCAD interface to

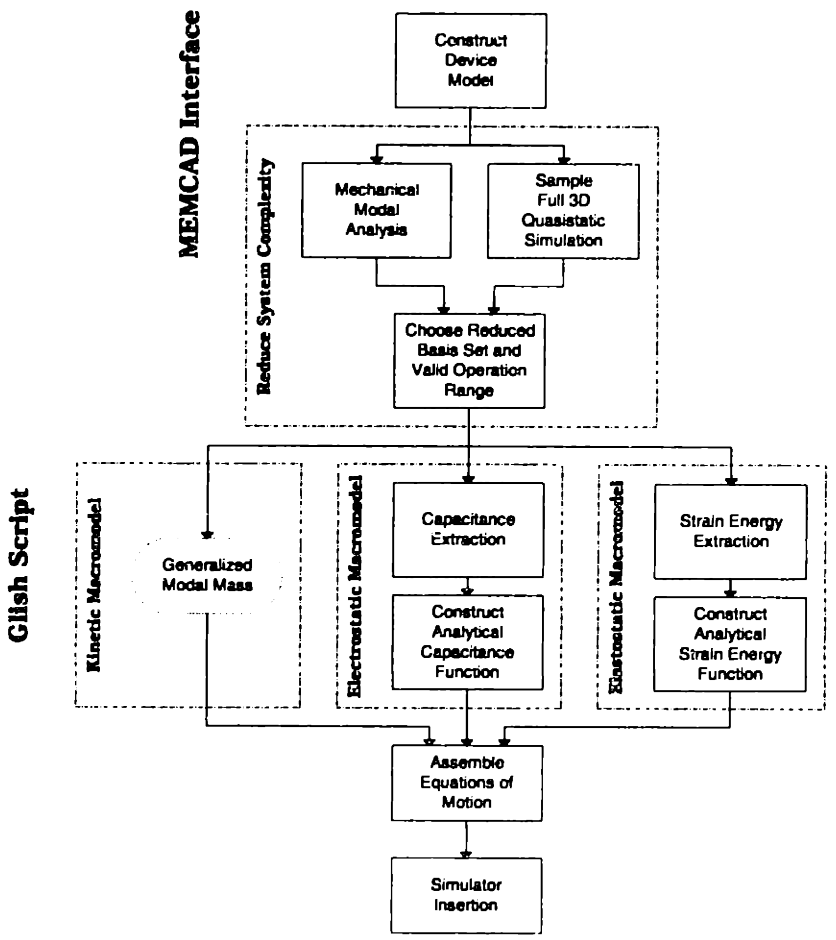


Figure 5: Two Stage Churn Implementation (for the non-linearized strain energy case)

construct the model, perform the modal analysis, and compute an example full three-dimensional simulation. The second stage is an automated Glish script that interrogates the designer up front for various preferences and settings, and then follows through the entire remainder of the Churn process, thus constructing the circuit simulator input file.

While discussing the architecture of this implementation, it is impractical to delve into the details of the code; for this kind of information, the actual code itself is far more useful and shall be publicly available at the MIT MEMCAD website [44]. Instead, we shall present a depiction of the information flow through each step of the implementation. In this chapter, we separate the flowing information into several categories of information type. The *var* information type constitutes any kind of information that is being stored in memory. Examples would include numbers and character strings, as well as arrays or records of these. The *file* information type constitutes any information that is stored persistently, such as on a hard disk. This includes FEMs stored in MBIF files, data storage files, and the like. The *user* information type constitutes any information that comes directly from the designer. This includes the manual construction of a model as well as preference settings that the designer might have set earlier in the process. The *imp* information type constitutes any information that was set within the implementation of the process. Of course, this type could be interchanged with the *user* type in other implementations.

3.2.1 Preparation via MEMCAD

From Figure 5, we note that there are three primary tasks to perform within MEMCAD. The first task is to construct the MBIF file containing the finite element model of the structure and the associated material properties, initial conditions, and boundary conditions. The next is to perform modal analysis on the structure, thereby extracting the first few mode shapes and their generalized masses and frequencies. The last is to perform a single coupled electrostatic-elastostatic simulation. All three of these steps can be performed from directly within the MEMCAD interface. In this section, we present a high-level information flow description of each step performed within MEMCAD.

3.2.1.1 Model Construction

Inputs:

user device concept

Outputs:

file MBIF file containing FEM, initial conditions, and boundary conditions

The designer first constructs a solid model of the device, either by using MemBuilder to automatically create the solid model from a mask set and a process flow, or by explicitly constructing the solid model within I-DEAS. The designer then manually meshes the solid model, thereby constructing an FEM of the device. The positions of the nodes of the constructed FEM correspond to the elements of a shape vector ψ as described in Section 2.1. The designer then sets the material properties, initial conditions (such as initial stress), and mechanical boundary conditions within MEMCAD. All of this information is then written to a single MBIF file.

3.2.1.2 Modal Analysis

Inputs:

file MBIF file containing FEM, material properties, initial conditions, and boundary conditions
user number of modes to calculate

Outputs:

```
array of {
    file MBIF file containing mode shape,
    var generalized mass,
    var generalized frequencies
}
```

MEMCAD executes ABAQUS to perform mechanical modal analysis, resulting in the creation of several MBIF files, each containing a displacement vector that depicts the shape of a mode. A side effect of this analysis is that it relaxes any initial stress conditions to the extent permissible by the boundary conditions, allowing the base structure to deform. (This is important for released micromechanical structures with residual stress.) The new shape after this deformation acts as ψ_{eqm} as described in Section 2.1.

3.2.1.3 Coupled Electrostatic-Elastostatic Analysis

Inputs:

file MBIF file containing FEM, material properties, initial conditions, and boundary conditions
user typical actuation settings (such as applied voltage)

Outputs:

file MBIF file containing solution to coupled domain analysis

MEMCAD performs a single quasistatic coupled electrostatic elastostatic simulation, storing the resulting equilibrium position in an MBIF file. This new shape acts as ψ_{ex} as described in Section 2.1. For each mode shape calculated, the vector is scaled such that each vector's largest absolute displacement magnitude $\|\varphi_i\|_\infty = 1$. According to Equation (2) in Section 2.1, prioritizing the mode shapes can be done by sorting the absolute magnitude of the projection coefficients $|c_i|$. Also, note that because MEMCAD works in units of microns, for any linear composition of these shapes, the contribution of each shape is equivalent to the maximum displacement of that shape in microns.

3.2.2 Automation via Glish

The remainder of the Churn process is implemented as a Glish script that runs entirely automatically, with the exception of querying the user for a handful of settings at the start before any time consuming computation takes place. Due to the nature of Glish, most of the numerical processing must take place within compiled Glish clients, which can be spawned, communicated with, and terminated from a Glish script. Figure 6 depicts the tasks that must be performed in this stage of the Churn process, and it illustrates the applicability of each of the Glish client modules as it will be used in each step.

Although the MEMCAD suite provides Glish client modules to access full three-dimensional domain solvers like FastCap or ABAQUS, we found it necessary to develop additional modules that are specialized to perform the numerical tasks discussed in Chapter 2. In this section, we present high-level information flow descriptions of the modules both implemented for this research and those provided by Microcosm. The source code of the modules implemented for this research shall be made available at the MIT MEMCAD web site.

3.2.2.1 Shape Manager ("modeManager")

The Shape Manager module serves several purposes related to reading, constructing, and writing legal MBIF files as a function of the new generalized coordinates. This module was written for the purposes of

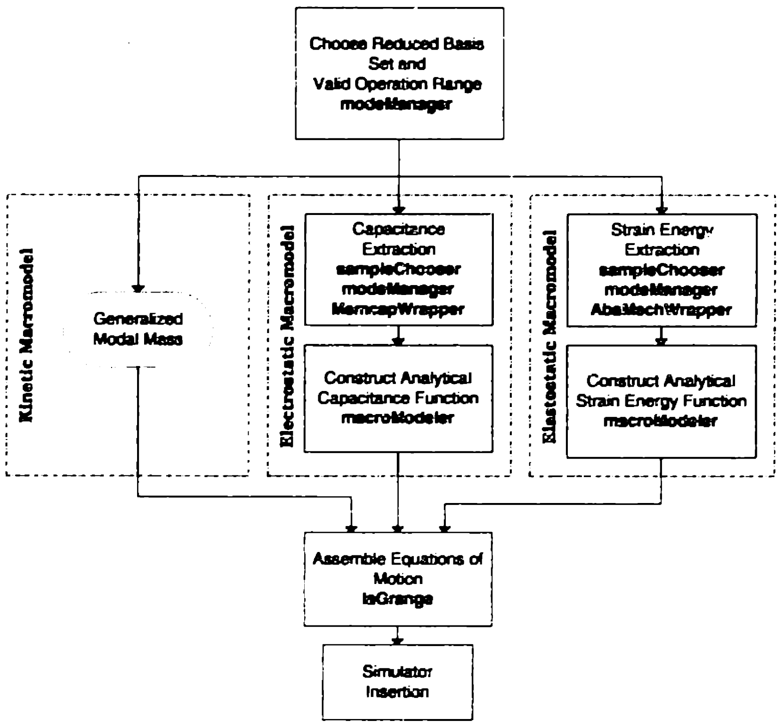


Figure 6: Breakdown of Glish Client Modules and Their Use in the Churn Process Implementation

this research.

INITIALIZE

Inputs:

file MBIF containing the device structure (after relaxation due to initial stress)

Load the shape stored in the given MBIF file into memory. This shall serve as ψ_{eqm} as described in Section 2.1.

LOAD NEW SHAPE(S)

Inputs:

```
array of {
    file MBIF containing a shape,
    var shape name
}
```

For each model file/shape name pair, load the shape stored in the model into memory, and associate the shape name with the shape. Future calls must refer to loaded shapes by using their associated names.

After all new shapes are loaded into memory, QR factorization is used to create an inverse to the matrix of the basis shapes, as discussed in Section 2.1.2. If additional shapes are loaded later, the inverse is recomputed.

CONSTRUCT MBIF FOR CAPACITANCE EXTRACTION

Inputs:

```
array of {
    var shape name
    var superposition coefficient value
}
```

Outputs:

file new MBIF

Construct an MBIF file of the original structure, displaced by the linear superposition of the given shapes with the associated magnitudes. This MBIF is designed for use with the MEMCAD FastCap wrapper client, MemcapWrapper.

CONSTRUCT MBIF FOR STRAIN ENERGY EXTRACTION

Inputs:

```
array of {
    var shape name
    var superposition coefficient value
}
var node-set to fix
```

Outputs:

file new MBIF

Construct an MBIF file of the original structure, with mechanical boundary conditions set such that nodes within the given node-set are displaced by the linear superposition of the given shapes with the associated magnitudes. Valid node-sets are all nodes, all surface nodes, or all nodes on selected faces. This MBIF is designed for use with the MEMCAD ABAQUS wrapper client, AbaMechWrapper.

Compute Shape Projection

```
Inputs:
    file MBIF

Outputs:
    array of {
        var shape name
        var superposition coefficient value
    }
```

Load the given MBIF and compute the superposition coefficients necessary to best construct this shape using the known basis shapes.

3.2.2.2 Sample Point Chooser ("sampleChooser")

The Sample Point Chooser module implements the sampling point selection strategy described in Section 2.2.1. This module was written for the purposes of this research.

Initialize

```
Inputs:
    array[m] of {
        var shape name
        var minimum
        var maximum
        var #gradations
    }
```

Construct the m-dimensional volume in $\Re^m$ space that spans the given operating range, and subdivide that volume according to the given number of gradations for each dimension. Initialize the counters for all of the subvolumes to zero.

Get Next Sampling Point

```
Outputs:
    array[m] of {
        var shape name
        var superposition coefficient value
    }
```

Generate a sampling point by iterating one step of the algorithm in Section 2.2.1.

3.2.2.3 ABAQUS Wrapper ("AbaMechWrapper")

The ABAQUS Wrapper module is a Glish client that spawns ABAQUS to perform a variety of mechanical analysis tasks upon the model represented within a given MBIF file. This module is a distributed part of Microcosm's MEMCAD suite.

Initialize

```
Inputs:
    file MBIF of original structure
```

Loads the finite element model, initial stresses, material properties, and boundary conditions stored within the given MBIF file into memory.

Perform Mechanical Analysis

Prerequisites:

Initialize

Outputs:

file MBIF containing the mechanical analysis results

Writes out a legal ABAQUS input file depicting the loaded model, material properties, and boundary conditions, then spawns ABAQUS to execute the mechanical analysis with the given conditions. The results, such as the model's deformations, stresses and strains, are then written to a new MBIF file.

Determine Elastostatic Strain Energy

Prerequisites:

Perform Mechanical Analysis

Outputs:

var strain energy

Determine the elastostatic strain energy of the given model under the given conditions. This requires that mechanical analysis has already been performed.

3.2.2.4 FastCap Wrapper ("MemcapWrapper")

The FastCap Wrapper module is a Microcosm version of FastCap that has been modified to work as a Glish client. This module is a distributed part of Microcosm's MEMCAD suite.

Initialize

Inputs:

file MBIF of original structure

Loads the finite element model stored within the given MBIF file into memory. Only the surface nodes and faces of the model are used; all mechanical properties and boundary conditions are irrelevant.

Perform Capacitance Extraction

Prerequisites:

Initialize

Outputs:

var electrostatic capacitance matrix

Perform the FastCap algorithm on the finite element model to determine the electrostatic capacitance matrix of the conductors depicted in the loaded finite element model.

3.2.2.5 Macromodel Generator ("macroModeler")

The Macromodel Generator module loads a data storage file containing full three-dimensional simulation results such as capacitance or strain energy, and to this data it fits one of several built-in analytical functional forms, as according to the algorithm discussed in Section 2.2.2. This module was written for the purposes of this research.

IMPORT DATA

Inputs:

file data storage

Load the data stored in the given file. This data must contain the domain and range of the function we shall fit, and may optionally contain a standard deviation field to individually weight each of the data points.

EXPORT DATA

Outputs:

file data storage

Write the data stored in memory to a compatible data storage file.

CREATE MACROMODEL FUNCTION

Inputs:

user functional form
imp initial λ (0.001 by default)
imp λ increasing scaling factor (10 by default)
imp λ decreasing scaling factor (10 by default)
imp improvement cutoff factor (0.2 by default)

Outputs:

file analytical macromodel

Using the Levenberg-Marquardt algorithm with the given settings, fit a function of the given functional form to the data loaded in memory. The resulting function and its derivatives are subsequently written to an expression file.

3.2.2.6 Analog Circuit Element Generator ("laGrange")

The Analog Circuit Element Generator module loads the analytical macromodels of the energy domains, created by the Macromodel Generator, and generates legal SABER input files that express the equations of motion, as described in Section 2.3. This module was written for the purposes of this research.

GENERATE ELECTROSTATIC, LINEAR-ELASTOSTATIC ELEMENT

Inputs:

array of {
  var shape name
  var generalized modal mass
  var generalized modal frequency
}
file capacitance macromodel

Outputs:

file SABER input file

With the given values for the modal masses and frequencies for each of the generalized coordinates, along with the expression file for the electrostatic capacitance, construct a SABER input file that expresses the physics of the macromodeled device with a non-linear electrostatic energy domain and a linear elastostatic energy domain, as expressed by the equations of motion in Equation (16).

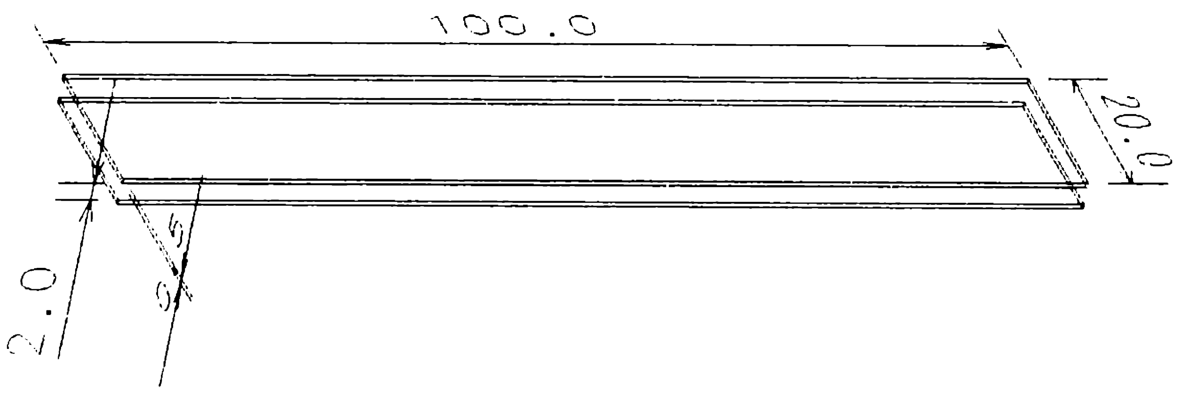


Figure 7: Simple Fixed-Fixed Beam Suspended Over a Ground Electrode (units are in microns)

```
GENERATE ELECTROSTATIC, NON-LINEAR-ELASTOSTATIC ELEMENT
  Inputs:
    array of {
        var shape name
        var generalized modal mass
    }
    file capacitance macromodel,
    file strain energy macromodel
  Outputs:
    file SABER input file
```

With the given values for the modal masses for each of the generalized coordinates, along with the expression files for both the electrostatic capacitance and the elastostatic strain energy, construct a SABER input file that expresses the physics of the macromodeled device with non-linear electrostatic and elastostatic energy domains, as expressed by the equations of motion in Equation (15).

3.3 The Churn Process Walkthrough

In this section, we shall step through the Churn process as we have implemented it for the case of a simple fixed-fixed beam suspended above a fixed electrode strip, as depicted in Figure 7. Both the beam and the electrode are assumed to be unstressed polysilicon conductors that are 100x20x0.5μm in size, separated by a 2μm gap.

3.3.1 Preparation via MEMCAD

As we explained in Section 3.2, the first stage of the Churn process requires the designer to work within the MEMCAD framework. It is inappropriate for us to delve into detail about using MEMCAD here; for information about using the MEMCAD interface, we refer the reader to the MEMCAD User's Manual [45].

3.3.1.1 Model Construction

The first step of this process is to construct the MBIF file that specifies the problem. To do this, the designer first constructs a solid model of the device and creates a finite element model from the solid model. Within MEMCAD, there are two possible ways to construct a solid model of a device. The first is to specify a mask set and process flow. MemBuilder can use this information to implement the fabrication process virtually, thereby creating a solid model in I-DEAS. The second way is to manually use the solid modeling tools built into I-DEAS to construct the solid model from scratch. Either way will create the solid model of the device within I-DEAS. We constructed our example manually from within I-DEAS. This model is shown in Figure 7. Often, we shall find it easier to visualize a device if we exaggerate the scale of the some of the axes. Henceforth for this example, we shall scale the z-axis of this device by a factor of 20, as shown in Figure 8.

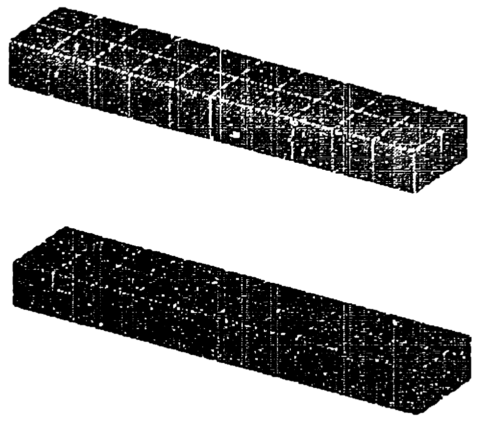


Figure 8: 20x z-axis Zoom of Simple Fixed-Fixed Beam Structure

Once the solid model exists within I-DEAS, the designer uses I-DEAS to manually mesh the solid model, thereby constructing the finite element model for the device. MEMCAD prefers the use of 20-node parabolic brick elements, and so we use that here. Figure 9 shows the finite element model we generated for our simple fixed-fixed beam example. Each conductor is 10x2x1 elements, totaling 20 volume elements and 267 nodes per conductor. Note that the mesh depicts the faces of the bricks as having been cracked into eight triangles. This is a feature of MEMCAD that enhances the precision of the capacitance extraction process. These cracked faces are used solely during capacitance extraction, and thus will not be used during any kind of mechanical analysis.

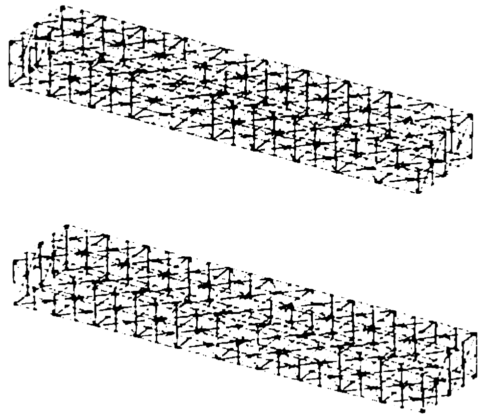


Figure 9: FEM of Simple Fixed-Fixed Beam

Finally, the designer returns to the MEMCAD interface to assign the device's material properties, initial stresses, and mechanical boundary conditions. Via the MEMCAD interface, we manually set the material properties for our example to be that of polysilicon according to MEMCAD's material property database, that is to say a Young's modulus of 165 GPa and a Poisson ratio of 0.23. The designer must also set boundary conditions within MEMCAD. For our example, we fix all the nodes on each of the ends of the deformable beam. The designer also has the option to specify initial stresses for the structure. We do not set any initial stresses in this example. Once all settings have been made, the MBIF file is written, containing the device's finite element model, the material properties, the boundary conditions, and any initial stress conditions that may have been set.

3.3.1.2 Modal Analysis

MEMCAD provides a simple method to perform mechanical modal analysis. The designer uses the MemMech component within MEMCAD to call ABAQUS to compute any user-chosen number of mechanical modes. For each mode that gets computed, MemMech creates a new MBIF file that contains that mode's shape in the form of nodal displacements; it also determines each mode's generalized mass and frequency. In Figure 10, we present the first three mechanical mode shapes for our example. In Table 1, we present the calculated values for the generalized mass and frequency for each of the modes.

The mode shape vectors that it creates are scaled such that each vector's largest absolute displacement magnitude equals 1. This brings up two points. First, recall Equation (2) from within the discussion of prioritizing basis shapes as discussed in Section 2.1. If the largest absolute displacement $\|\varphi_i\|_\infty = 1$ for all shape vectors, then prioritizing the mode shapes is as simple as sorting the absolute magnitude of the projection coefficients $|c_i|$. Second, because MEMCAD works in units of microns, for any linear composition of these shapes, the contribution of each shape is equivalent to the maximum displacement of that shape in microns. This allows the designer to have a tangible understanding of the modal coordinates

Table 1: Generalized Mass and Frequency Values for Example Fixed-Fixed Beam Structure

	Generalized Mass (kg)	Generalized Frequency (Hz)
Mode 1	8.55146×10^{-13}	459253
Mode 2	9.13315×10^{-13}	1.2983×10^{6}
Mode 3	3.33213×10^{-13}	1.5555×10^{6}

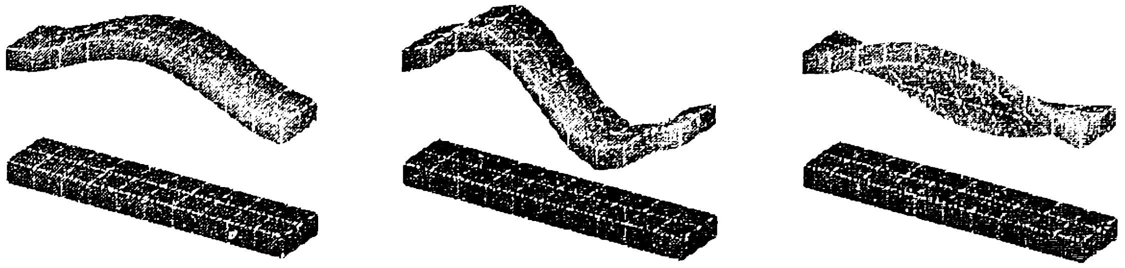


Figure 10: First Three Mode Shapes for Example Fixed-Fixed Beam Structure

and their effect on overall motion.

One additional point to note is that the modal analysis updates the existing MBIF file to exhibit any mechanical relaxation that would occur due to any preset initial stress conditions; thus, it allows the base structure to deform. This new relaxed, deformed shape acts as ψ_{eqm} as described in Section 2.1.

3.3.1.3 Coupled Electrostatic-Elastostatic Analysis

In order to perform the single typical full three-dimensional coupled solution, the designer uses CoSolve-EM from within MEMCAD to determine the quasistatic equilibrium of the system at any given applied voltage. For our example, we apply a voltage of 80 volts; the equilibrium deformation for this actuation is shown in Figure 11.

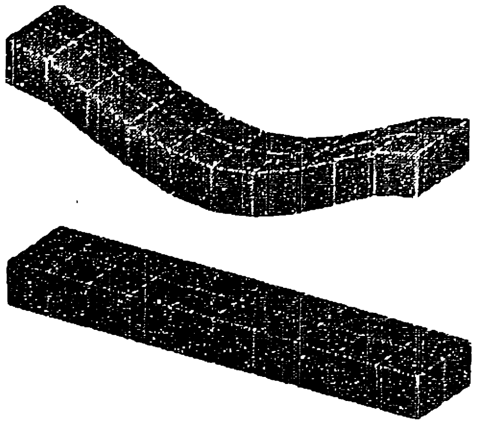


Figure 11: Equilibrium at an Applied Voltage of 80 Volts

3.3.2 Automation via Glish

We shall discuss the remaining part of the Churn process according to the steps outlined in Section 3.2.2 while simultaneously discussing the use of each of the implemented modules at each step. To best follow this, we suggest referencing Figure 6 once again.

The Glish script begins by querying the designer for all settings in advance, thus allowing the script to extract all data, fit the macromodel functions, and export the macromodels to an analog circuit simulator input file. The designer answers such questions as:

- How many generalized coordinates should be used to represent the state of the system (m)?
- Given the projections of the sample coupled solution onto the mode shape basis set, what should be the valid operating range of the device; that is, what are the limits for values of the generalized coordinates?
- Should a linear or non-linear representation of the strain energy be used?
- For capacitance, or for strain energy if a non-linear representation is requested:
  - How many data points should be extracted?
  - How finely should the operating range be subdivided when choosing where to sample?
  - How many terms should be used in the macromodel function series expansions?

To start, the Glish script creates a **modeManager** process that is then instructed to load the mode shapes into memory. For our example, these are the three mode shapes as depicted in Figure 10. Note that this **modeManager** remains in memory throughout capacitance and strain energy extraction in order to create the input files for **MemcapWrapper** and **AbaMechWrapper**, respectively.

The modeManager process is then instructed to load the results of the full three-dimensional coupled solution and determine the contributions of each of the mode shapes towards overall motion. For our example, the solution shown in Figure 11 is loaded, and the resulting calculated mode shape projections are shown in sorted order in Table 2. We choose to use only one mode shape for this example, and by default, the script understands that it is to be mode 1, because of its dominance over the other modes' contributions. In this implementation, the designer chooses the valid operating range by giving a minimum and maximum scaling factor that when multiplied by the contribution of each of the modes yields the bounds to the valid operating range for that mode. For our example, we chose a minimum scaling factor of -2 and a maximum scaling factor of 2, resulting in the valid operating range for mode 1 to be [-1.6194,+1.6194].

Table 2: Contribution of Mode Shapes to Overall Motion

	Contribution
Mode 1	-0.809704
Mode 3	-0.0122652
Mode 2	4.32353×10^{-5}

At this point, the designer is asked the aforementioned remaining questions so that the script can run automatically without pausing for user input. For the data extraction for capacitance, we chose to obtain 20 data points with a sampling grid resolution of [5]. For our example, we chose to use the non-linear strain energy representation, so for the strain energy extraction, we also chose to obtain 20 data points but with a sampling grid resolution of [4]. We selected the capacitance macromodel to be a [4 / 4] multivariate polynomial rational fraction and the strain energy macromodel to be a [4] multivariate polynomial.

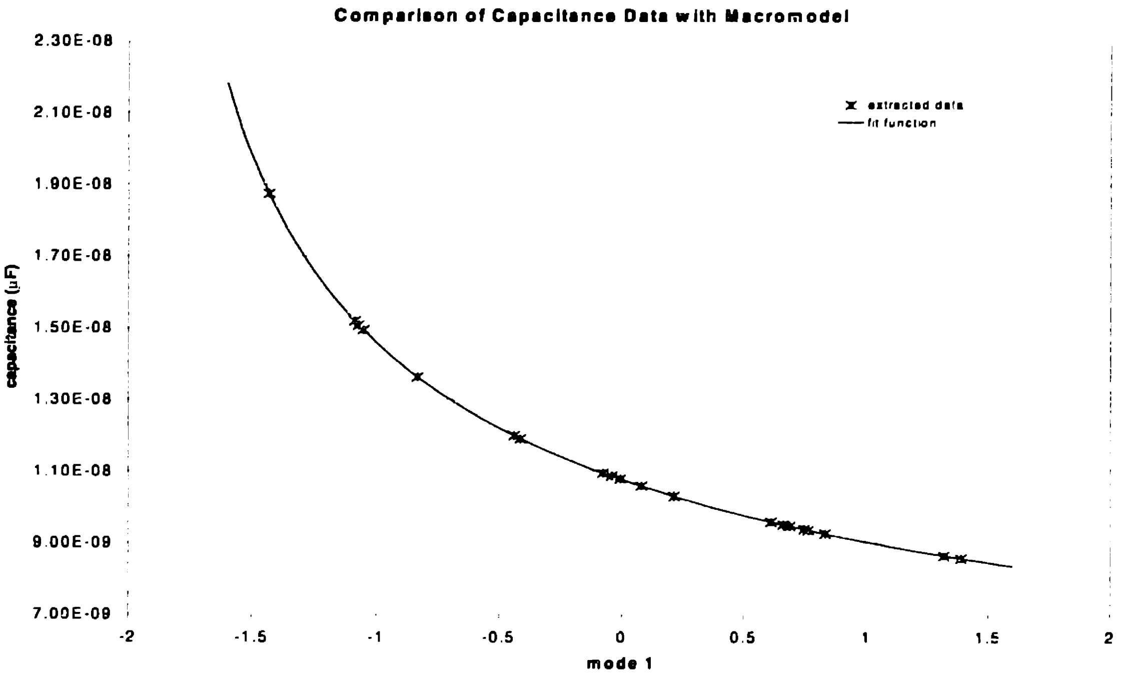


Figure 12: Comparison of Capacitance Extraction Data with Function Fit

The Glish script handles data extraction and function fitting the same way, regardless of whether it is for capacitance or for strain energy, so we shall discuss both simultaneously. Data extraction begins by constructing a sampleChooser process and initializing it to span the valid operating range of the device. A loop over the number of data points to extract begins. The sampleChooser process is instructed to provide a new data point at which to sample. This point is then passed to the resident modeManager process to write out an MBIF file that depicts the device at that state. The appropriate domain solver (MemcapWrapper or AbaMechWrapper) is then called to perform the full three-dimensional analysis at the chosen data point. The results are then stored in a data file.

When all data has been calculated, a macroModeler process is created and instructed to load the data stored in the log file. It is then told what function type to use, whether it be a multivariate polynomial or a multivariate polynomial rational fraction, and how many terms to include in the polynomials. The macroModeler process is then instructed to fit the function to the data, and upon completion writes the function to a file as an analytical expression in a binary, platform independent format that we developed for the purposes of this research. Each expression file contains not only the expression in question, such as the capacitance or the strain energy, but also the gradient of the expression with respect to each of the generalized coordinates.

The results of the capacitance data extraction and macromodeling are shown in Figure 12, and those for the strain energy are shown in Figure 13. With only one dependent variable, it is trivial to graphically

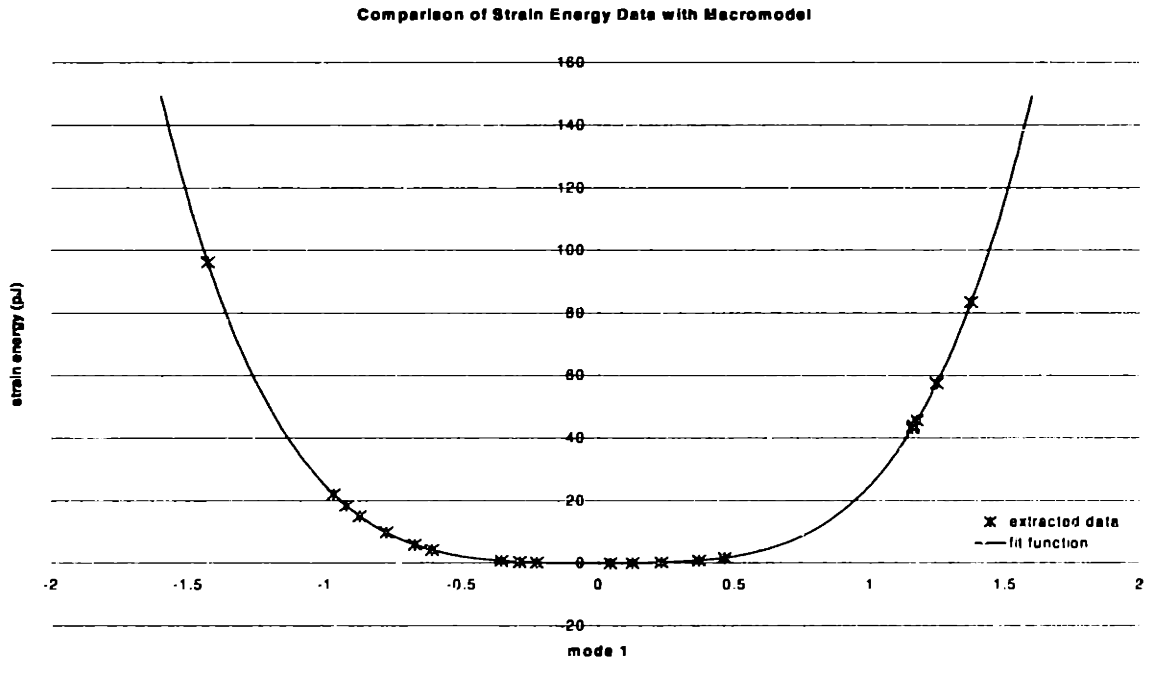


Figure 13: Comparison of Strain Energy Extraction Data with Function Fit

show how well a function can be fit to data. However, although we have only one generalized coordinate in this example, other models may have more than one, and it will become increasingly difficult to graphically depict the quality of a function fit. One way to quantify this is to note the χ^2 of the function fit to the data. For our capacitance model the χ^2 is 3.5857×10^{-24}, and for our strain energy model it is 1.09011×10^{-13}. Although these numbers give a good estimate of close the data is to the function, it is lacking in many respects. For example, it does not give a good measure of whether enough data has been taken to fully capture all the interesting physics in the operating range. Also, its order of magnitude is directly related to that of the data itself, which can change among varied devices. Moreover, it does not insure how accurate the gradient of the function will be, because it is the gradient of the function that will enter the equations of motion. Nevertheless, it has been our experience that capacitance and strain energy are relatively smooth functions and that if the χ^2 is small, the macromodels shall be good fits.

The Glish script then creates a laGrange process to assemble the equations of motion. It is provided the generalized masses and loads the capacitance expression file, and depending upon whether the designer chose to use a linear or non-linear strain energy representation, it is provided the generalized frequencies or loads the non-linear strain energy expression file. The laGrange process then assembles the equations of

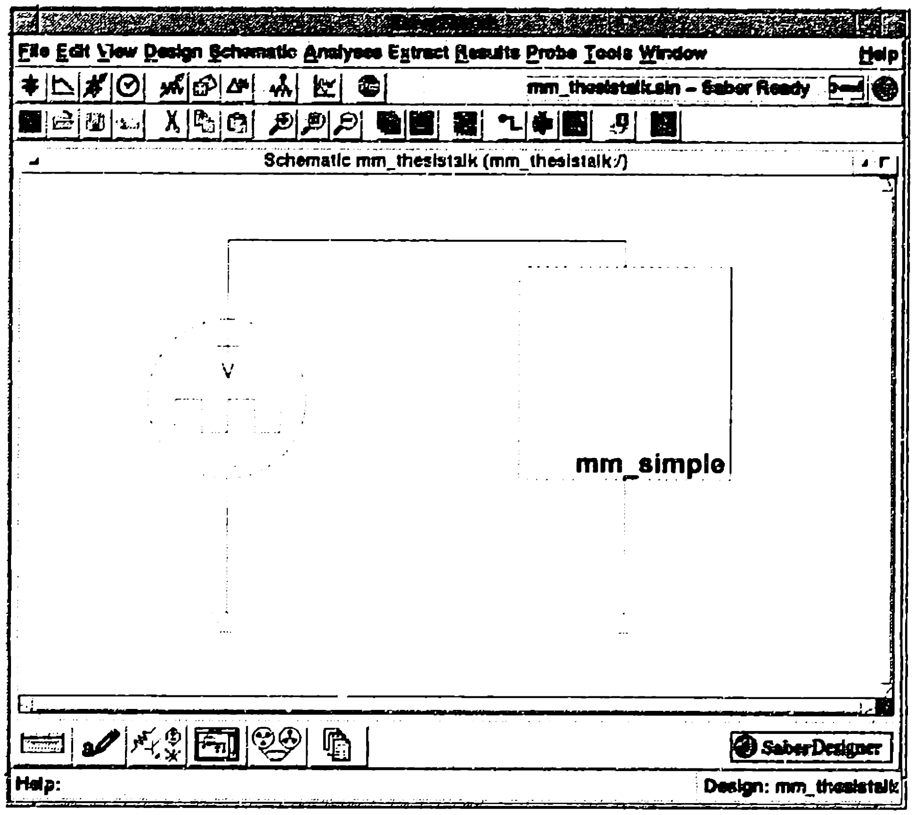

File Edit View Design Schematic Analyses Extract Results Probe Tools Window Help
mm_thesistalk.sin - Saber Ready
Schematic mm_thesistalk (mm_thesistalk:/)
V
mm_simple
SaberDesigner
Help:
Design: mm_thesistalk

Figure 14: Example Structure Incorporated into SABER as a Circuit Element

motion and writes them to file in the form of a legal SABER input file. This input file can then be loaded as a circuit element in SABER and incorporated into any circuit. This concludes the Churn process. The designer is now free to use the SABER model of the device in place of full three-dimensional simulation. Figure 14 shows the SABER model for our example as it appears when loaded into a SABER circuit schematic.

3.3.3 Sample Results

We conclude this chapter with some examples of the model we have constructed being driven by a variety of voltage waveforms. Figure 15 depicts the response of our example device when actuated by an 80 volt square wave with a hold time of 10μs and a total period of 40μs. Figure 16 depicts the response of our example device when actuated by a 100 volt saw wave with a rise time of 15μs, a hold time of 10μs, and a total period of 40μs. Figure 17 depicts the response of our example device when actuated by an 80 volt saw wave with a rise time of 5μs, a hold time of 10μs, a fall time of 10μs, and a total period of 40μs.

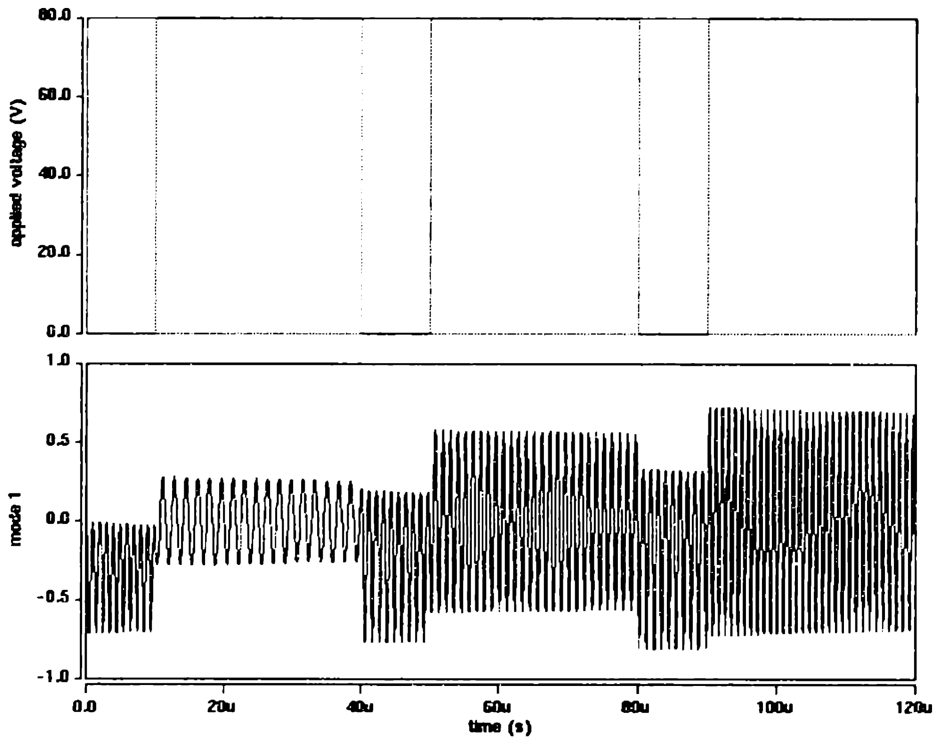


Figure 15: Response of Example Structure to an 80v Square Wave with 10μs Hold and 40μs Period

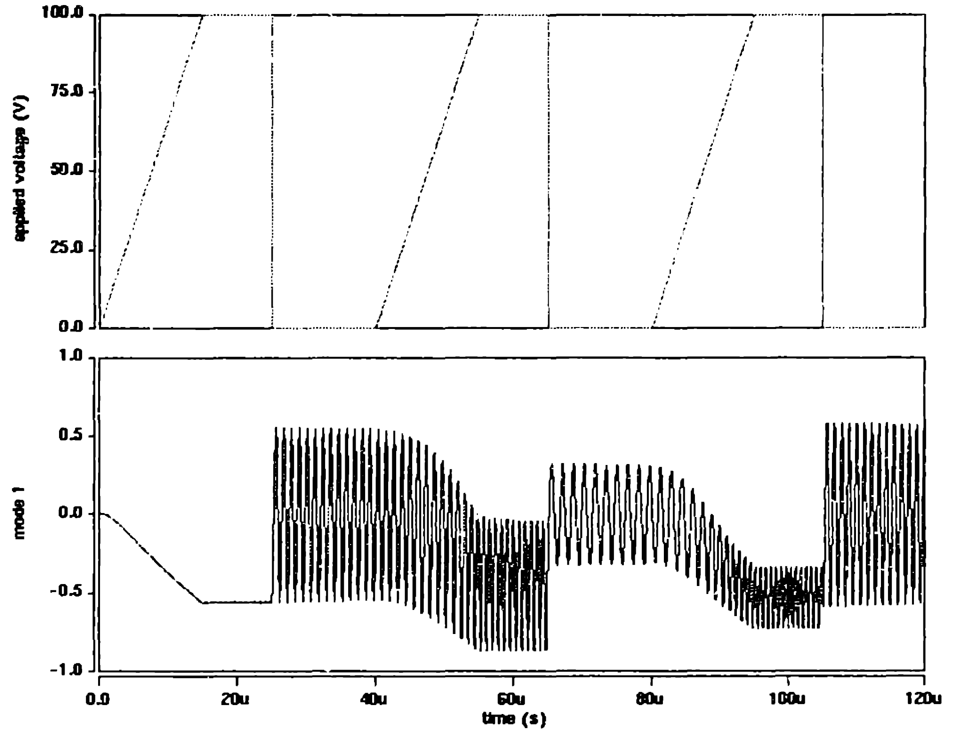


Figure 16: Response of Example Structure to a 100v Saw Wave with 15μs Rise, 10μs Hold, and 40μs Period

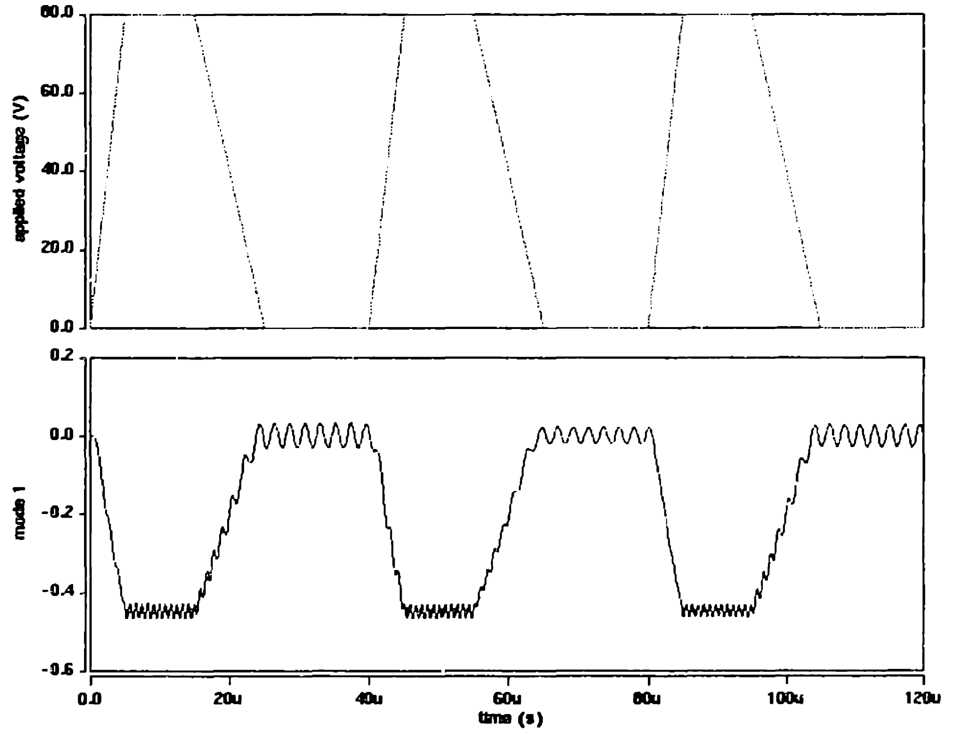


Figure 17: Response of Example Structure to an 80v Saw Wave with 5μs Rise, 10μs Hold, 10μs Fall, and 40μs Period

One of the motivations for simulating system dynamics is to determine how a system will be affected by feedback. Thus, let us demonstrate the value of the circuit element macromodel we have created by

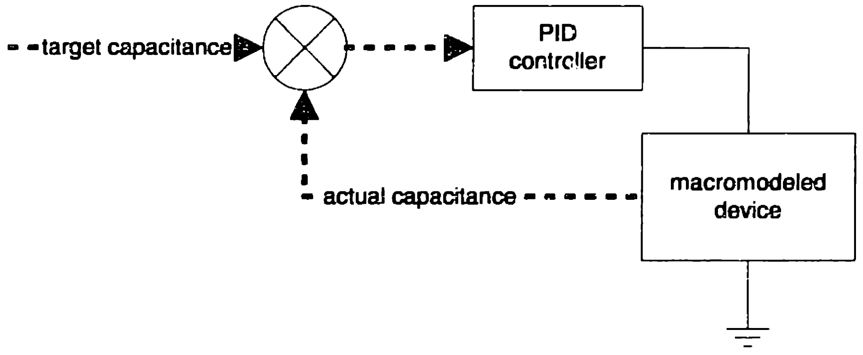


Figure 18: Macromodeled Device Controlled in a Feedback Loop

closing a feedback loop around it and attempting to control its motion. Before we can begin, we must decide upon what property of the macromodel will be used for the feedback signal. Recall that the macromodel circuit element stores information about the modal coordinates of the device. It is unreasonable to use modal coordinates for feedback control because these quantities are generally not measurable. A more plausible quantity would be the capacitance of the device, as it is an experimentally measurable quantity. Furthermore, due to the nature of the macromodel we construct, the capacitance of the device is a readily available quantity that can be easily extracted from the circuit element.

We implement a PID controller and construct a feedback loop circuit as depicted in Figure 18. The PID controller has a transfer function of $K + as + b/s$. For this example, we set $K = -5$, $a = 10^{-6}$, and $b = 10^{7}$. The results for a 16fF target capacitance are presented in Figure 19. The constructed macromodel for this device uses a non-linear strain energy representation and exhibits pull-in at a 1.3μm maximum displacement (with 0.7μm gap remaining) with an applied voltage of 200V. (These values are higher than full coupled simulation results for reasons discussed in Section 4.2.) Note that this results in a maximum structure deformation very close to pull-in, and that the only damping in this system is implemented by the PID controller.

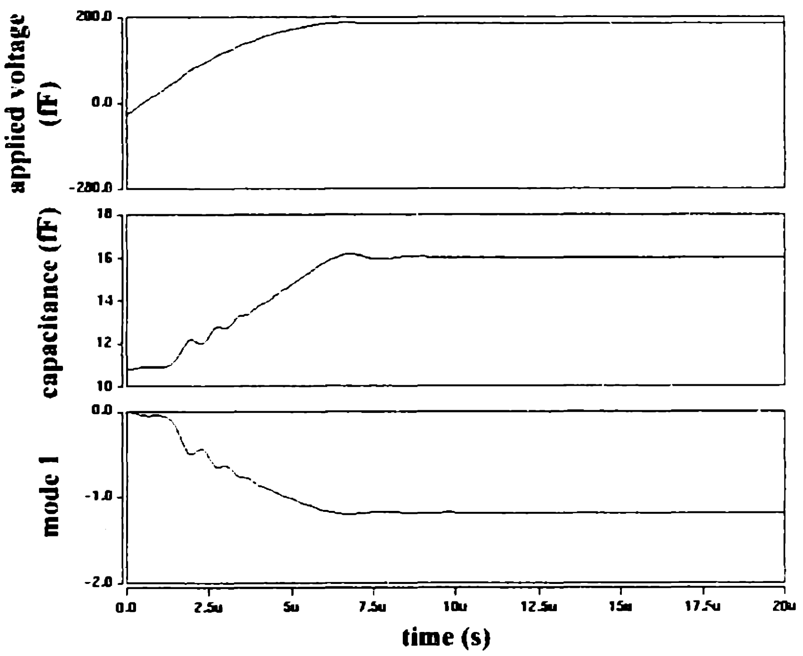


Figure 19: Simulation of Feedback Controlled Macromodel

Chapter 4 Analysis

4.1 Computation Time

In this section, we discuss the relative time scale for each stage of the Churn process and provide a measure of O(n) growth with various model complexity parameters. Although it would be desirable to be able to present accurate CPU-time measurements and theoretical O(n) bounds for each of the software modules involved, there are several factors that make this impossible. One reason is that the architecture of Glish and the MEMCAD suite do not permit precisely extracting the CPU-time consumed by the processes they spawn. Thus, any real-time measurements include the CPU-time of all other concurrently running processes as well as kernel and operating system activities. Another obstacle is that the computation time for any model will not only vary with the number of nodes *N* and basis shapes *m*, but also will vary based upon the nature of the model. In fact, neither ABAQUS, MATLAB, nor MEMCAD publish O(n) information for their algorithms for this very reason. Thus, the results we present in this section provide merely a qualitative measure of the computation time and its growth, and should by no means be used as a precise predictive measure. We shall present qualitative measures of computation time that are based upon our observations from several examples we have computed, which we present in Chapter 5. These examples were computed upon a Sun Ultra-1 Model 170 with 196Mb RAM running SunOS 5.5.

Let us begin by discussing the steps in the Churn process that could require non-trivial computation time. These are:

- Perform modal analysis
- Perform single quasistatic coupled simulation
- Determine basis set by determining the contribution of the known shapes to the coupled solution
- Extract domain data, such as capacitance or strain energy
- Construct domain analytical macromodels, such as for capacitance or strain energy

Many of these steps reuse common pieces of the MEMCAD suite, so it will be to our advantage to report the timing statistics for these pieces individually, then report how they contribute to the timing of the steps in the Churn process.

4.1.1 MEMCAD Procedures

Because several of the steps in the Churn process rely on MEMCAD to perform a small set of tasks, it is best if we begin by presenting the computation time of the various MEMCAD components. These are:

- doMech(), which performs mechanical analysis through ABAQUS

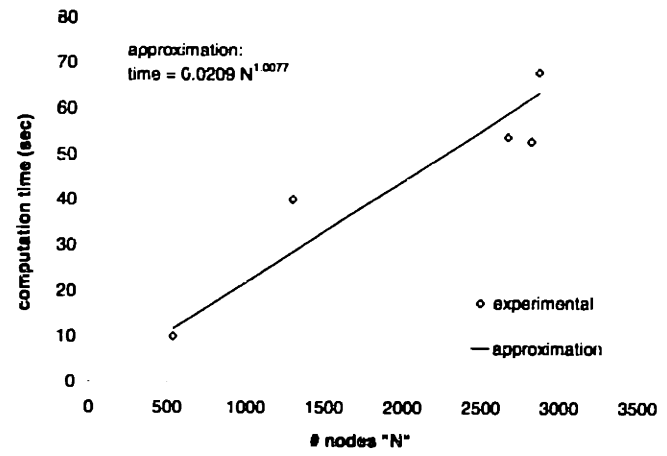


Figure 20: Computation Time for doMech()

- doMode(), which performs modal analysis through ABAQUS
- doElectro(), which performs electrostatic analysis through FastCap

Note that with all MEMCAD procedures, it is not possible to extract the consumed CPU time for the process directly. Therefore, the presented results erroneously include the CPU time for all concurrent processes, which we have attempted to minimize. In order to bound the computational growth of these procedures, we shall fit results to power functions of the form $a N^b$, where the power term b will reveal whether the procedure is linear, quadratic, cubic, etc. with N.

4.1.1.1 doMech()

The MEMCAD doMech() procedure translates an MBIF file into a legal ABAQUS input file that contains the structural finite element model, the mechanical boundary conditions, initial stresses, and any imposed loads, and then instructs ABAQUS to solve the posed problem. This procedure typically behaves in two ways, one upon initial calls, and one for subsequent calls. During initial calls, ABAQUS generates a "restart file", in which the results of common preprocessing steps are stored. For subsequent calls, this restart file circumvents the preprocessing. The difference in computation time between the two is generally negligible compared to the variation with imposed actuation conditions. However, it worth noting that the restart file accounts for any initial stress conditions, in which case there is a noticeable time savings.

Figure 20 presents the measured computation time for doMech() for a variety of models with varied numbers of nodes N. In fitting these results to a power function, we note that growth with N is roughly linear. In general for finite element solutions, computation growth is highly dependent upon the nature of the structure [46]. For example, for beam structures, where N nodes are spread along the length of the structure, computation time tends to grow as O(N). On the other hand, for plate structures, where $\sqrt{N}$ nodes line each side of the plate, the computation time tends to grow as O($N^{3/2}$). Finally, for structures where the nodes are spread evenly through a block, the computation time can grow as O(N^2). Thus for the purposes of this discussion, we declare that doMech() is the worst case order O(N^2). For the examples we have computed, this step typically takes one minute of computation time.

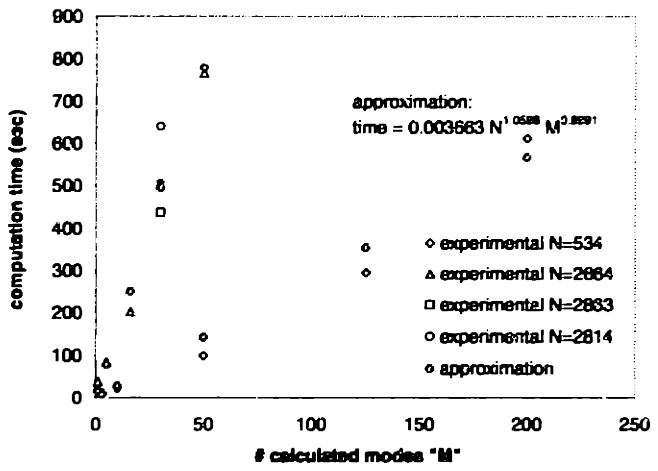


Figure 21: Computation Time for doMode()

4.1.1.2 doMode()

The MEMCAD doMode() procedure uses the restart file from an initial doMech() call and instructs ABAQUS to calculate the first *M* mechanical modes, their shapes, generalized masses and frequencies. Figure 21 presents the measured computation time for doMode() for several values of *M* for a variety of models with constant *N*. In fitting these results to a power function, we note that the computation time is roughly linear with both *M* and *N*. As we stated for doMech(), dependence on *N* can grow as high as N^2, thus for the purposes of this discussion, we declare that doMode() is order O(N^2 *M*). For the examples we have computed, this step typically takes a few minutes of computation time.

4.1.1.3 doElectro()

The MEMCAD doElectro() procedure submits an MBIF file to MemCap, the MEMCAD hybrid of FastCap, which reads an MBIF file and determines the capacitance matrix for the system and the magnitude of charges on the surface panels to satisfy LaPlace's equation. Figure 22 presents the measured computation time for doElectro() for several models with different numbers of nodes *N*. In fitting these results to a power function, we note that the growth in computation time is roughly linear with *N*. This agrees with the published computation growth for FastCap [40], thus we too declare that this step is O(*N*). For the examples we have computed, this step typically takes one minute of computation time.

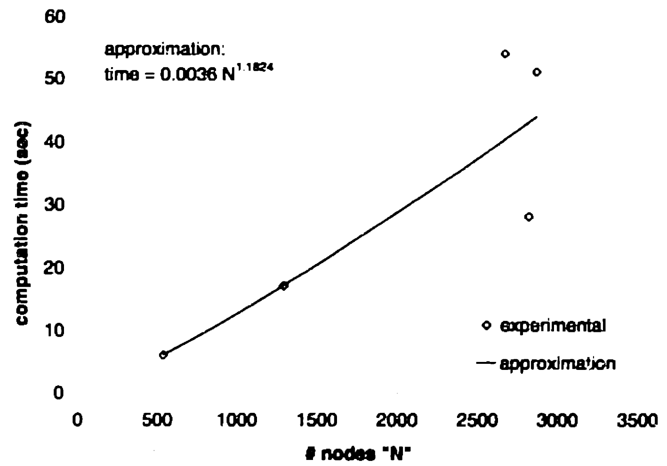


Figure 22: Computation Time for doElectro()

4.1.2 Perform Modal Analysis

Modal analysis is comprised of one initial doMech() step and one doMode() step, therefore the total time necessary to perform this step is

doMech + doMode

and computation growth, which goes as the worst case scenario, is order O(N^2 M). For the examples we have computed, this step typically takes a few minutes of computation time.

4.1.3 Perform Single Quasistatic Coupled Simulation

MEMCAD CoSolve performs our coupled simulation first by performing one initial doMech() call to create the restart file. It then alternates between doMech() and doElectro(), starting and ending with doMech(), until quasistatic equilibrium is reached. Supposing that i iterations are necessary to reach equilibrium, the total time necessary to perform this step is

2 × doMech+ i × (doMech+ doElectro)

and computation growth is order O(i N^2). Regardless of the model, typical values for i range from 2 to 15, based upon how close to pull-in the applied voltage is. For the examples we have computed, the computation time for this step is on the order of a half hour.

4.1.4 Basis Set Determination

As discussed in Section 2.1, the choice of modes to be adopted as generalized coordinates depends upon the projection of the quasistatic coupled solution onto the known mode shapes. Recall that this is accomplished by using QR factorization to create an inverse to the matrix of the basis shapes, as discussed in Section 2.1.2, and then by using that inverse to project the solution. We found that via MATLAB it was possible to determine the exact c‹ ›umed CPU time to perform the matrix manipulation portion of these tasks. We determined that on average the time to execute the QR factorization on a set of M known basis vectors of length N is

$$3 \times 10^{-6} M^{1.3} N^{1.0} \text{ seconds}$$

and that the time to compute the projection of a shape onto the known basis set is

$$5 \times 10^{-11} M^{1.3} N^{1.8} \text{ seconds.}$$

For the examples we performed, these results were three orders of magnitude less than the actual time consumed by other process overheads. Thus, although technically this step is order $O(N^{1.8} M^{1.3})$, we find that unless M or N are significantly large, the computation time is overwhelmed by order $O(1)$ overhead costs. Typically, the actual time for the matrix computations is on the order of milliseconds, yet the actual time for the step takes a few seconds.

4.1.5 Domain Data Extraction

The Churn process uses MEMCAD to determine capacitance and strain energy, thus the computation time for n capacitance calculations is

$$n \times \text{doElectro}$$

and the time for n strain energy calculations is

$$n \times \text{doMech}$$

Note that as the number of generalized coordinates m increases, the size of the sample space increases exponentially. In order to maintain a constant fractional coverage of the sample space, the number n should also grow exponentially, as order $O(e^m)$. Thus, the overall computation time for domain data extraction is order $O(N e^m)$ for capacitance and order $O(N^2 e^m)$ for strain energy. In general, the computation time for complete data extraction could take hours or even days, depending upon how much coverage of the sample space the designer desired. However, as with most of these steps, remember that this computation time is a one-time investment, because once the data is extracted, it can be reused repeatedly to generate varying complexities of domain macromodels.

4.1.6 Macromodel Construction

The primary computation component of macromodel construction is the execution of the Levenberg-Marquardt process as described in Section 2.2.2. The computation time for this algorithm depends upon the number of iterations necessary to satisfy whatever convergence conditions were specified. Therefore, let us discuss the computation time necessary for a single iteration. All iterations require solving $\alpha' \delta\mathbf{a} = \beta$ for $\delta\mathbf{a}$, then computing $\chi^2(\mathbf{a} + \delta\mathbf{a})$. However, note that an iteration that succeeds in lowering the χ^2 merit function must additionally compute the derivative matrices α and β. Therefore progressive and regressive iterations must be measured separately.

Two factors can affect the computation time of the progressive and regressive iteration steps. One is the number of data points n to be used for fitting the function, and the other is the number of fitting parameters f for which the algorithm must solve. In Figure 23, we present the computation time for progressive and regressive steps as functions of f with n held constant. We find that the results for both fit best to third order polynomials, implying that computation time is cubic order in f for both these steps. In Figure 24, we present the computation time for progressive and regressive steps as functions n with f held constant. For progressive steps, after an initial overhead, computation time grows linearly with n. Interestingly, regressive steps seem to exhibit constant time behavior, completely independent of n. This is odd because regressive steps must compute $\chi^2(\mathbf{a} + \delta\mathbf{a})$, which should be of order O(n). Upon closer inspection we find that the time consumed by solving $\alpha' \delta\mathbf{a} = \beta$, which is independent of n, is orders of magnitude above the time to compute $\chi^2(\mathbf{a} + \delta\mathbf{a})$. Thus, the linear dependence upon n is unnoticeable.

Combining these results, we find that macromodel generation is order O($i\ n\ f^3$), where i is the number of iterations necessary to converge. Recall from Section 4.1.5 that the number of calculated data points n should increase exponentially with the number of generalized coordinates m. In this case, macromodel generation computation time is order O($i f^3$ e^m). Typical values for i range from 30 to 50, with about half of these being progressive steps, half being regressive. Variation in i depends primarily upon what convergence conditions must be satisfied. Typical computation times overall for the macromodel process is sensitive to f and m, and can vary anywhere from seconds to an hour or more, but for our examples, this was typically a few minutes.

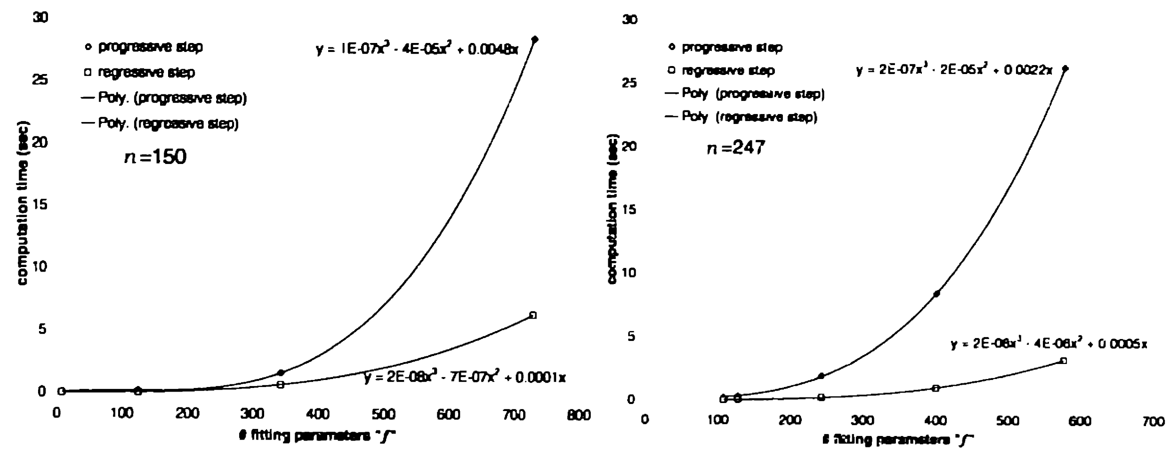


Figure 23: Macromodel Generation Computation Time as Function of Number of Fitting Parameters f at Constant Amount of Data n

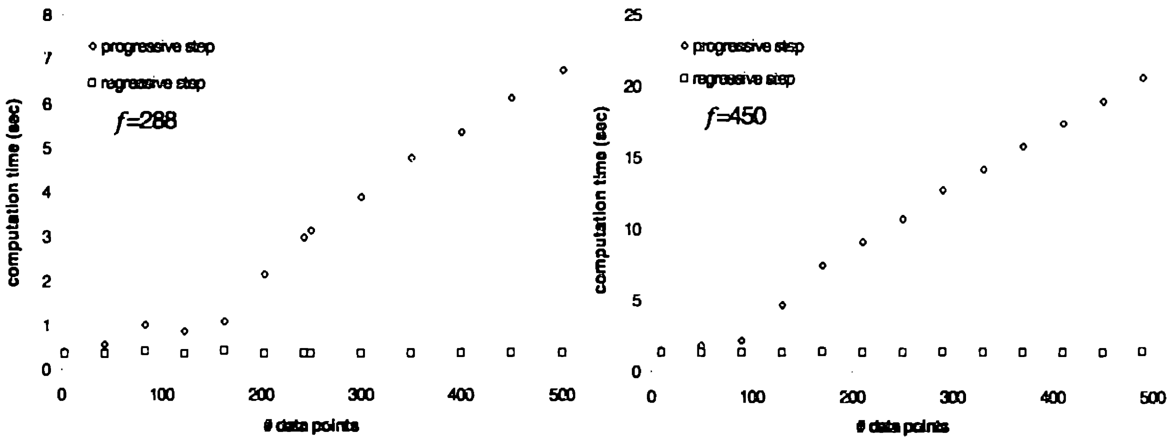


Figure 24: Macromodel Generation Computation Time as Function of Amount of Data n at Constant Number of Fitting Parameters f

4.1.7 Summary

We present a summary of timing results in Table 3. It is crucial to note that the computation time consumed by the Churn process needs to be performed only once, after which the SABER input file can be reused limitlessly in any circuit simulation. From our experiments, dynamics simulations take on the order of minutes, depending upon the circuit. In comparison, we note that the Churn process offers the designer a fast-to-compute dynamical circuit element for the trade-off of an initial, one-time, computational overhead.

Step	O(n) Growth	Typical Unit of Time
Modal Analysis	$O(N^2 M)$	Seconds
Coupled Simulation	$O(N^2)$	Hours
Basis Set Determination	$O(N^{1.8} M^{1.3})$	Seconds
Capacitance Extraction	$O(N e^m)$	Hours
Strain Energy Extraction	$O(N^2 e^m)$	Hours
Macromodel Construction	$O(f^3 e^m)$	Minutes

Table 3: Summary of Timing Results

4.2 Comparison to Quasistatic Analysis

In this section, we attempt to measure the accuracy of the dynamical macromodels generated by the Churn process by comparing them with full three-dimensional coupled simulation. Because MEMCAD does not support dynamic simulation at this time, we must compare quasistatic simulations. We do this by actuating the system with a constant applied voltage and then comparing the resulting generalized coordinates for the dynamical macromodel with those for the full three-dimensional coupled simulation. In order to obtain the equilibrium generalized coordinates from dynamical simulation, we must artificially damp the device macromodel to eliminate oscillatory motion. We do this by manually adding damping terms to the equations of motion embedded in the generated SABER input file. When the damped model is subjected to an applied voltage, it will converge upon its equilibrium, as demonstrated in Figure 25. In order to obtain the equilibrium generalized coordinate values from full three-dimensional coupled simulation, we project the calculated displacement vector onto the generalized coordinate basis set, as described in Section 2.1.2.

We shall simulate the models over a range of applied voltages. The behavior we expect is that as the applied voltage is increased, the device shall bend, resulting in more (albeit negative) contribution from mode_1. Ultimately, a critical voltage will be reached that causes the attracting electrostatic force to supercede the restoring elastostatic force, such that the device will collapse. This behavior is known as "pull-in", and the critical voltage at which it occurs is referred to as the "pull-in voltage". Figure 27 presents the quasistatic simulations of a variety of simulation models for the simple fixed-fixed beam example presented in Section 3.3. The first point to note is that at small deformations, all the simulation models agree with the full three-dimensional solution from CoSolve-EM. However, beyond a certain point, the macromodels diverge from the coupled solution.

First, let us discuss the discrepancy for the linearized strain energy macromodel. As applied voltage increases, this model begins to pull in at too low a voltage. Recall that this device is a 0.5µm thick beam, and that for generalized coordinates, a value of 1 corresponds to 1µm of maximum displacement. The device begins to deviate from three-dimensional simulation when it has displaced roughly half its thickness, at which point the structure becomes stress dominated, thus we would expect to see non-linear stress stiffening effects starting to take place. Thus, we believe that this lower voltage pull-in is due to the use of a

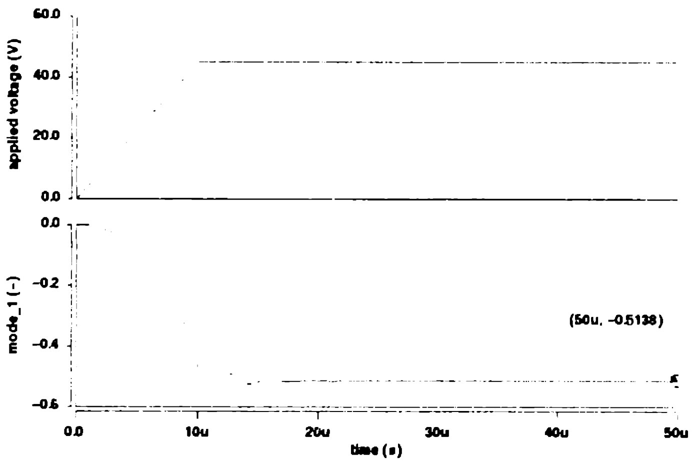


Figure 25: Response of Damped Simple Fixed-Fixed Beam to a 10µs Ramp to 50 Volts

linearized strain energy, which inherently neglects the non-linear stress stiffening effects we expect.

However, when we simulate the model that incorporated the non-linear strain energy representation (labeled "Non-Linear S.E." in Figure 27), we note that the solution has significantly erred in the other direction. Effectively, this model portrays the device as being excessively stiff. We believe that this comes from the method by which we calculate the strain energy before generating the analytical function that represents it. Recall from Equation (12) that strain energy is given by:

$$U_m = \int_{\text{vol}} \frac{1}{2} \sigma \varepsilon \, dV$$

where σ is the stress and ε is the strain in the system. When performed in finite element code, the integration over volume equates to summing the individual strain energies stored in each of the elements. Thus, if there is an error in the element strain energies, then that error will be magnified across the overall strain energy.

Now, recall that we map generalized coordinates to the system state vector by:

$$\psi = \psi_{\text{eqm}} + \sum_{i=1}^{m} q_i \varphi_i$$

The position of each node in the finite element model is explicitly calculated given the generalized coordinates. When all the coordinates are zero, the system state reverts to the equilibrium state, which we know to be the relaxed, minimum energy state for the structure with no applied load. When a load is applied, the structure deforms as the nodes relax to find their minimum energy state. However, in our method to calculate strain energy, we *fix* each node in the finite element model to be the explicitly calculated position determined from the above linear superposition. Now recall that by definition, the strain for each element is related to the ratio of the change in length of the element to its original length. If the fixed node positions we generate are slightly off from the relaxed equilibrium, as demonstrated in Figure 26 with two similarly shaped beams, the strain, and thus the strain energy, will be calculated higher than necessary. This is because the positions of all the nodes are fixed, and the structure will not be allowed to relax to the minimum energy state of the structure given the deformation we expect.

To confirm that this is indeed the case, we generate two new strain energy macromodels with restrictions on fewer nodes. First, we create a strain energy macromodel for when only the nodes on the surface of the deformable beam are fixed at each strain energy calculation (marked "surface nodes" in Figure 27). Second, we create one for when only the nodes on the bottom surface of the deformable beam are fixed (marked "bottom face nodes" in Figure 27). As we can see, as we reduce the number of constrained nodes, the macromodels start to approach the results of full three-dimensional simulation.

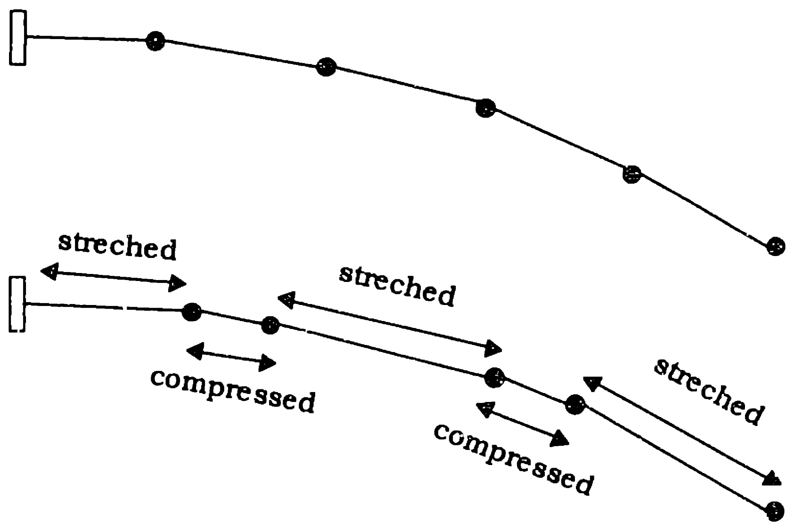


Figure 26: Example of Node Positioning Error

Although this shows us a direction to follow to approach large displacement accuracy, we have not solved this problem.

One approach to rectify this situation might be to change the way we represent system state altogether. As we have just shown, using *displacements* to represent system state inherently can result in non-relaxed states that would most likely not occur under system actuation. Suppose however that we choose to represent state by a basis set of *mechanical loads*. Any combination of these loads would result in a unique equilibrium state for the structure, thus we can map loads to displacements uniquely. This approach has the advantage of generating displacements that are guaranteed to be at the minimum energy state of the system. The disadvantage, however, is that we lose the clean representation of the kinetic energy that comes out of the mode shape displacement basis set approach.

In conclusion, we note that the operating range of the macromodels we can generate through the Churn process are limited to small displacements. In this case, it is worth noting that using the linearized version of the strain energy now becomes a valid (and faster) means to creating a device macromodel. However, a fully macromodeled non-linear strain energy representation alongside the linearized version can offer a good measure of bounds upon system behavior. For the short term, we believe that this exaggerated strain energy problem can be alleviated by constraining the nodes to a surface rather than constraining each node's position. In the long term, we believe a new paradigm for state representation may be needed to solve the issue entirely.

One final note worth mentioning is that even though we expect a lower strain energy, the high calculated value is actually correct for the problem that is posed. Given the fixing of nodes, the strain energy truly is as high as we calculate. The discrepancy lies in that when we reduced our degrees of

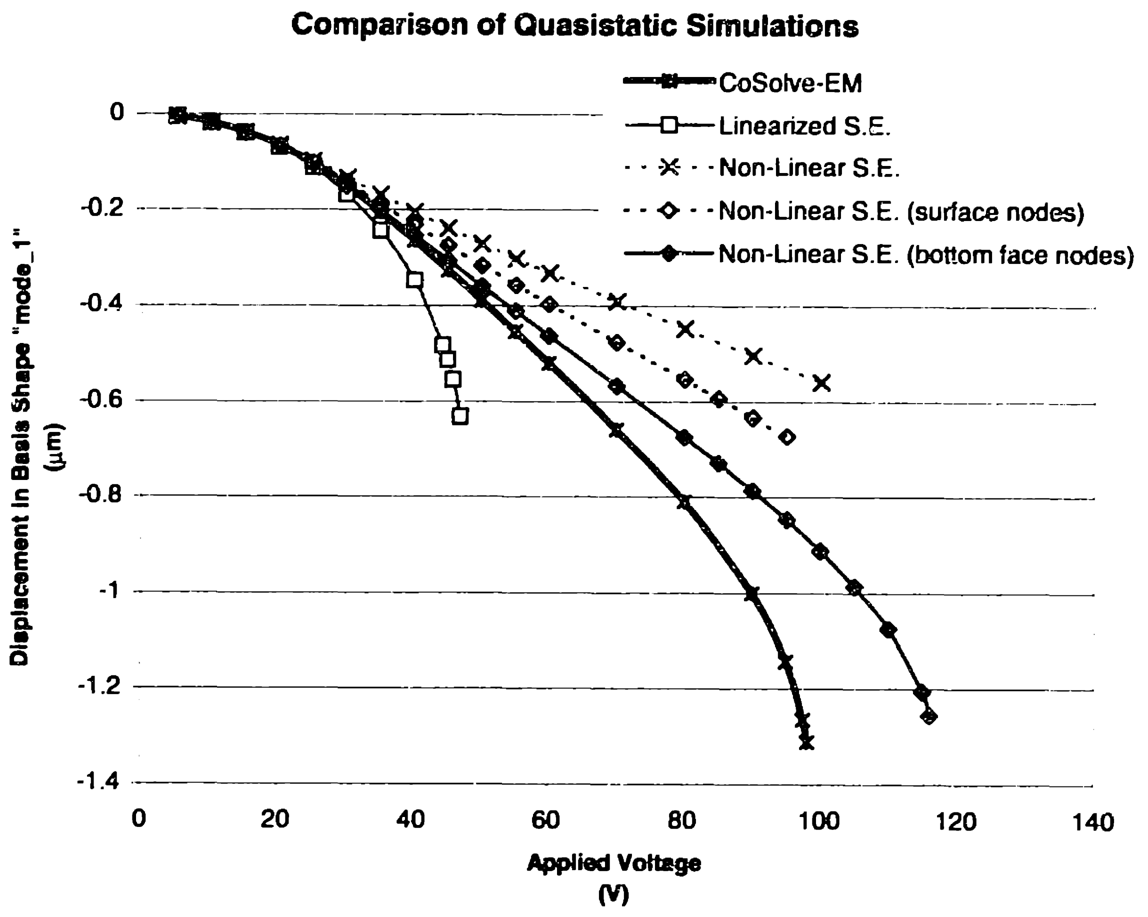


Figure 27: Comparison of Quasistatic Simulations for Simple Fixed-Fixed Beam Device

freedom to the first few mode shapes, we eliminated some crucial motion that allows the nodes to move to significantly lower strain energy states. Reducing the amount of nodes we fix does approach the strain energy we intended, but it also has the effect of asking a different problem, in that we are no longer constrained to the generalized coordinates we have chosen. Thus, there is a trade off between adherence to the generalized coordinates and accuracy with our approximation.

Chapter 5 Examples

In this chapter, we present the results for a variety of more complex structures than the one presented in Section 3.3. In Sections 5.1, 5.2, and 5.3, we present three different structures and discuss their computation time relative to what we would expect given the analysis presented in Section 4.1. In Section 5.4, we conclude with a brief comparison of timing statistics for these models and present a detailed table of information about all four structures presented in this thesis.

5.1 Fixed-Fixed Beam with Compressive Stress

5.1.1 Structure Description

This device, depicted in Figure 28, is a 600μm long, 40μm wide, and 2μm thick fixed-fixed beam with compliant supports that is suspended 2μm above a ground electrode. This device is made out of polysilicon with a Young's modulus of 165GPa and a Poisson ratio of 0.23, and has 4MPa of initial compressive stress along the axis of the beam.

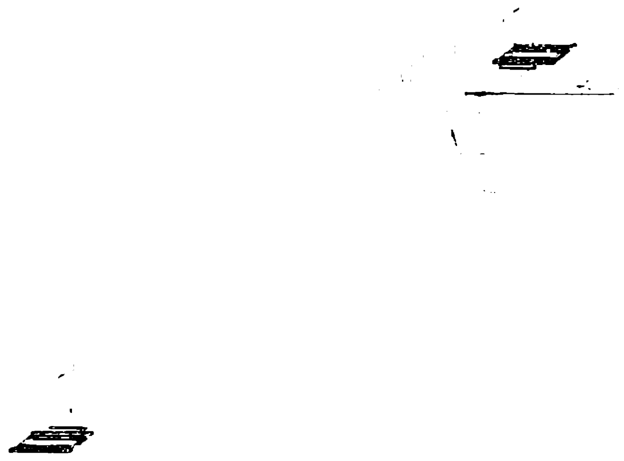


Figure 28: Schematic of Pre-Stressed Fixed-Fixed Beam

5.1.2 Solid Model

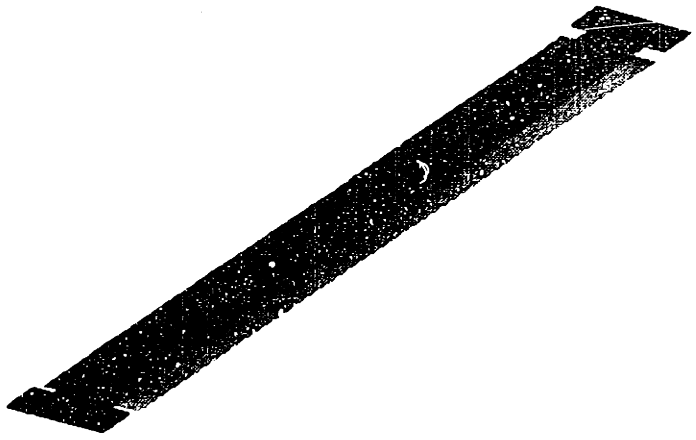


Figure 29: Solid Model of Pre-Stressed Fixed-Fixed Beam

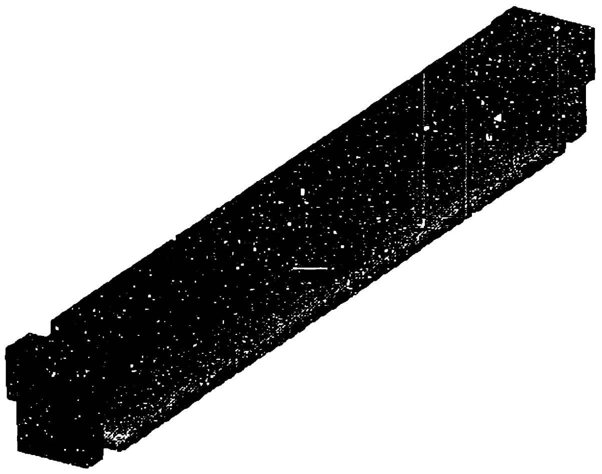


Figure 30: Solid Model of Pre-Stressed Fixed-Fixed Beam (20x Zoom)

5.1.3 Finite Element Model

The constructed finite element model, depicted in Figure 31, is comprised of 244 20-node brick elements, with a total of 2664 nodes.

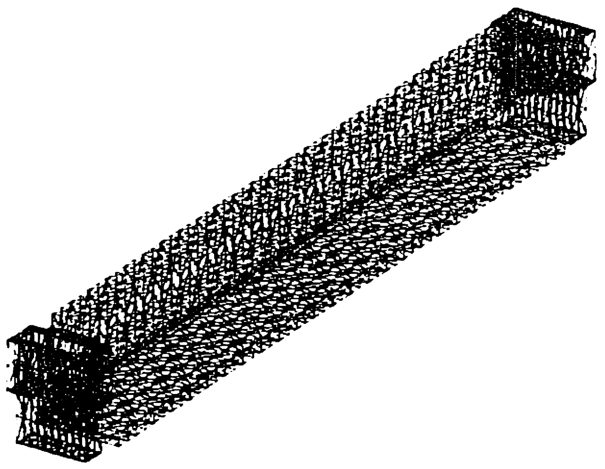


Figure 31: Finite Element Model for Pre-Stressed Fixed-Fixed Beam

5.1.4 Single Quasistatic Simulation

For the quasistatic coupled simulation, we actuate the device with 8 volts, which is 99% of the pull-in voltage for this structure. This required 13 CoSolve iterations to converge, requiring 25.5 minutes to compute. The resulting deformation is depicted in Figure 32. Table 4 presents the contribution of each of the calculated modes for this deformation.

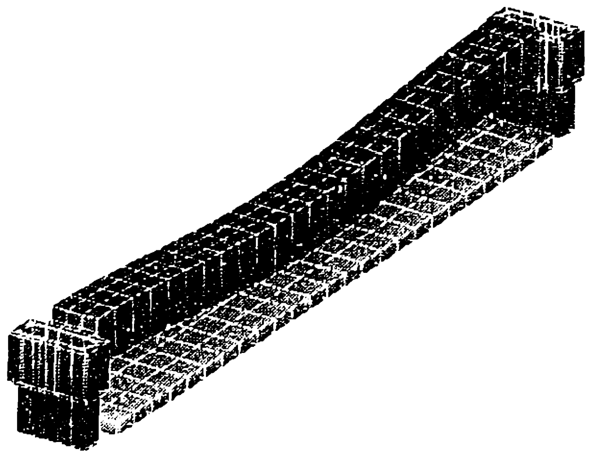


Figure 32: Quasistatic Response of Pre-Stressed Fixed-Fixed Beam to 8V Actuation

mode #	contribution
1	-1.175520000000
3	-0.011371800000
6	-0.002919480000
10	-0.001089550000
14	-0.000344592000
13	-0.000064906100
2	-0.000058467200
9	-0.000040157300
11	0.000029622700
4	0.000025428600
8	-0.000024580000
16	-0.000023693400
15	0.000003141490
12	0.000002898520
5	-0.000002151450
7	-0.000001673510

Table 4: Projection of Quasistatic Response onto Mode Shapes

5.1.5 Modal Analysis

We calculated 16 mode shapes for this device. Figure 33 shows the three mode shapes that will ultimately be used for the generalized coordinates. Table 5 presents the calculated information about each of these modes. This required 4.6 minutes to compute.

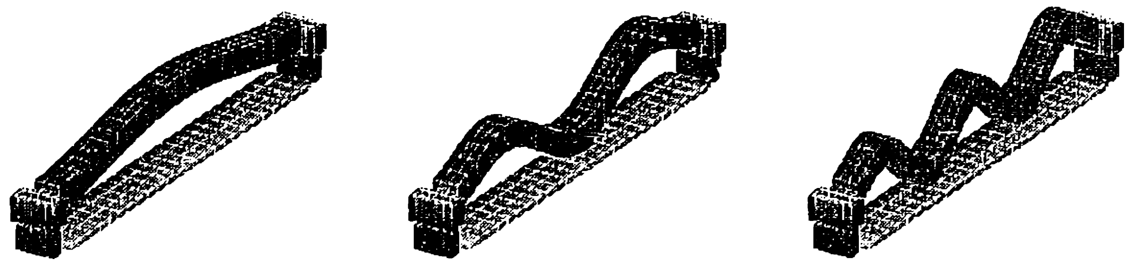


Figure 33: Mode Shapes Used to Comprise Basis Set

mode #	generalized mass (kg)	generalized frequency (Hz)	generalized stiffness	period (μs)
1	4.3011E-11	29971.5	1.53	33.37
3	4.7923E-11	235609.0	105.02	4.24
6	4.8310E-11	625115.0	745.28	1.60

Table 5: Calculated Mode Information

5.1.6 Capacitance Macromodel

We extract 100 capacitance data points and fit a [5 4 4 / 4 3 3] multivariate polynomial fraction, which has 229 fitting parameters. The resulting $\chi^2 = 4.22 \times 10^{-44}$. This required 1.5 hours to compute the capacitance data and 13.3 seconds to compute the macromodel.

5.1.7 Strain Energy Macromodel

We extract 96 strain energy data points and fit a [6 4 4] multivariate polynomial, which has 175 fitting parameters. The resulting $\chi^2 = 2.68 \times 10^{-28}$. This required 2.4 hours to compute the strain energy data and 9.2 seconds to compute the macromodel.

5.1.8 Dynamics Simulation

In Figure 34 and Figure 35, we present a couple of simulations of our macromodel for this structure. These took 50 seconds to compute on average.

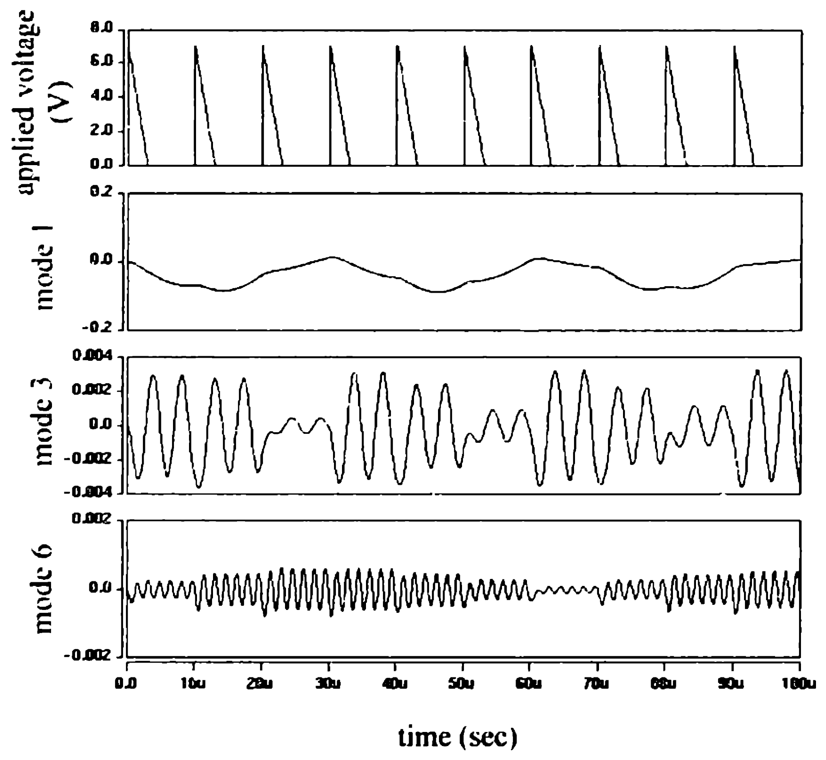


Figure 34: Response to a 7V Sawtooth Wave with 3μs Fall and 10μs Period

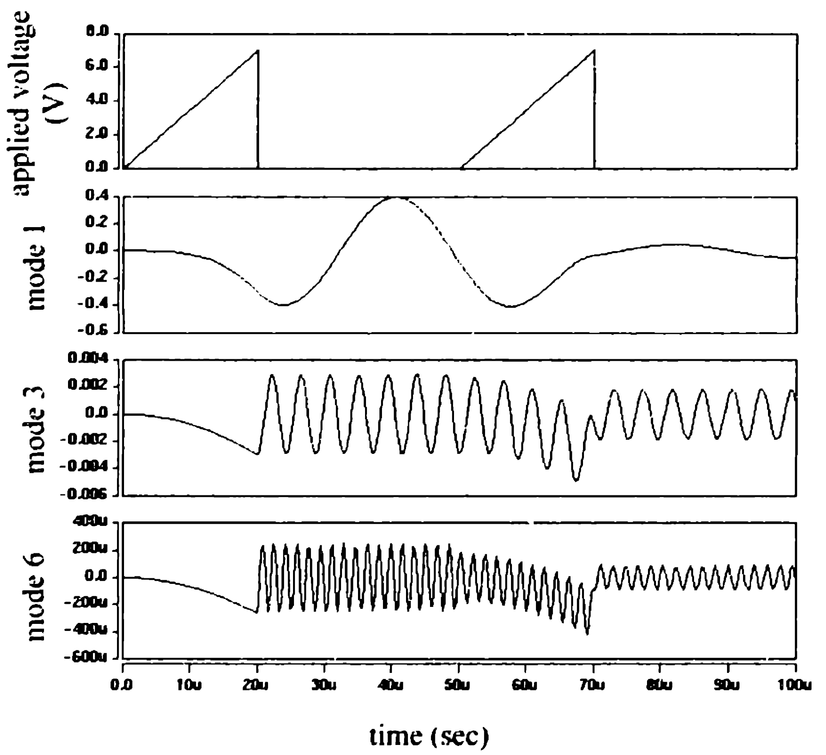


Figure 35: Response to a 7V Sawtooth Wave with 20μs Rise and 50μs Period

5.2 Fixed-Fixed Beam with Asymmetric Actuation Electrode

5.2.1 Structure Description

This device, depicted in Figure 36, is similar to before, a 600μm long, 40μm wide, and 2μm thick fixed-fixed beam that is suspended 2μm above an eccentrically placed electrode. The electrode is placed 80μm away from one of the supports. This device is made out of polysilicon with a Young's modulus of 165GPa and a Poisson ratio of 0.23.

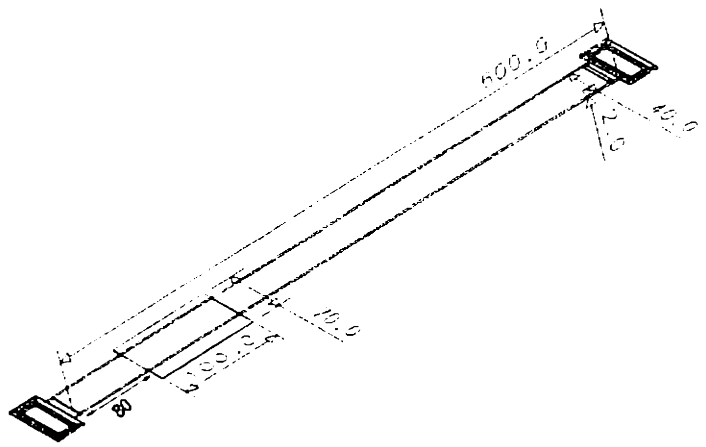


Figure 36: Schematic of Asymmetric Fixed-Fixed Beam

5.2.2 Solid Model

Figure 37: Solid Model of Asymmetric Fixed-Fixed Beam

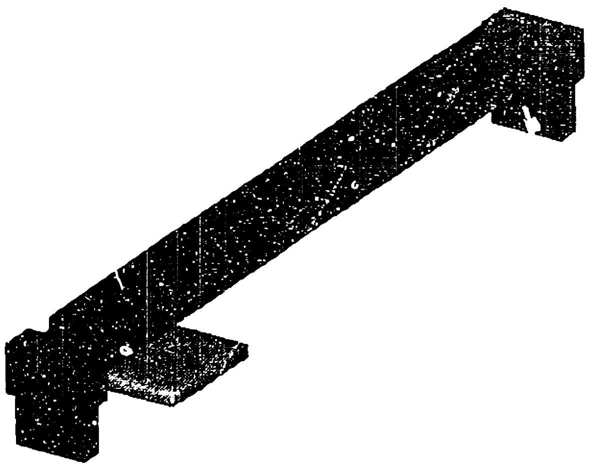


Figure 38: Solid Model of Asymmetric Fixed-Fixed Beam (20x Zoom)

5.2.3 Finite Element Model

The constructed finite element model, depicted in Figure 39, is comprised of 270 20-node brick elements, with a total of 2863 nodes.

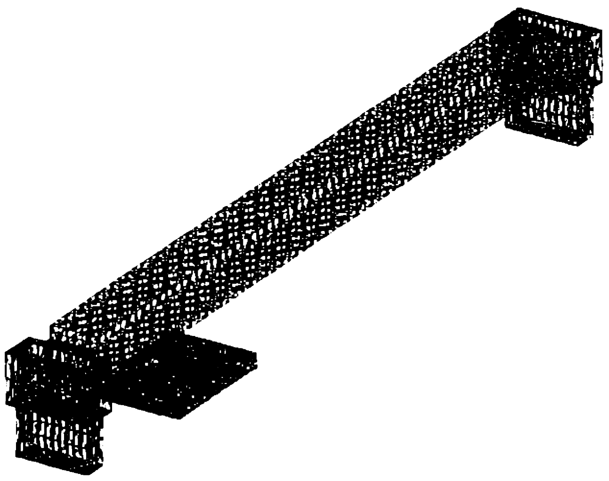


Figure 39: Finite Element Model for Asymmetric Fixed-Fixed Beam

5.2.4 Single Quasistatic Simulation

For the quasistatic coupled simulation, we actuate the device with 30 volts, which is 97% of the pull-in voltage for this structure. This required 13 CoSolve iterations to converge, requiring 27.8 minutes to compute. The resulting deformation is depicted in Figure 40. Table 6 presents the contribution of each of the calculated modes for this deformation.

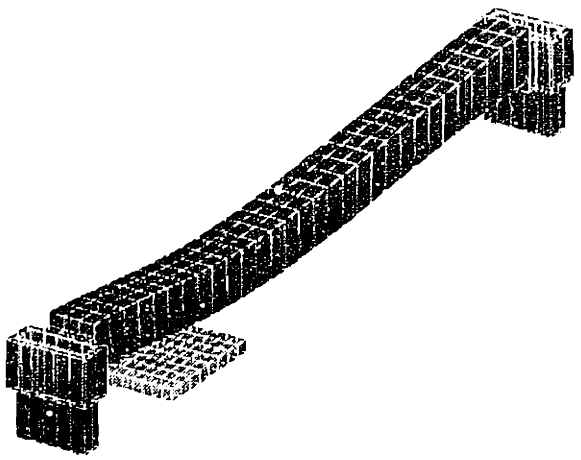


Figure 40: Quasistatic Response of Asymmetric Fixed-Fixed Beam to 30V Actuation

mode #	contribution	mode #	contribution
1	-0.973969000000	21	0.000032345700
2	0.207666000000	30	0.000031309500
3	-0.059565500000	18	0.000026917100
4	0.010083000000	27	0.000023499000
8	0.002301600000	11	0.000021445700
6	-0.000987739000	9	-0.000016901600
14	-0.000558761000	5	0.000015256500
10	0.000388974000	16	0.000014344500
19	-0.000380441000	22	0.000012232600
12	0.000377216000	25	-0.000007568970
23	-0.000191354000	24	-0.000006266890
17	-0.000136831000	26	-0.000005788590
28	0.000092457600	13	0.000003843730
20	0.000042833100	7	-0.000001377670
29	-0.000002151450	15	-0.000001221280

Table 6: Projection of Quasistatic Response onto Mode Shapes

5.2.5 Modal Analysis

We calculated 30 mode shapes for this device. Figure 41 shows the three mode shapes that will ultimately be used for the generalized coordinates. Table 7 presents the calculated information about each of these modes. This required 8.1 minutes to compute.

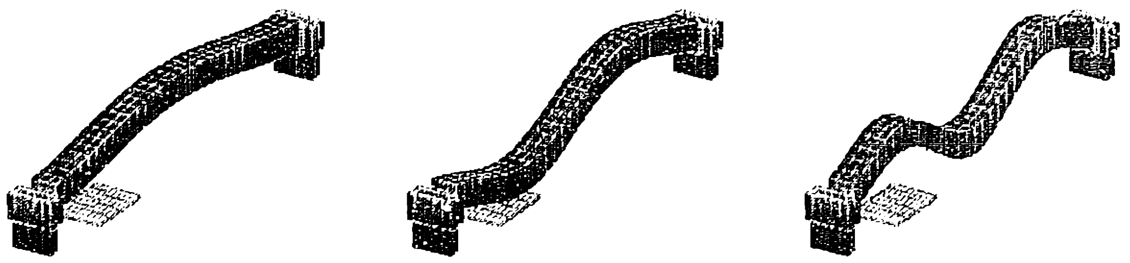


Figure 41: Mode Shapes Used to Comprise Basis Set

mode #	generalized mass (kg)	generalized frequency (Hz)	generalized stiffness	period (μs)
1	4.3201E-11	48500.0	4.01	20.62
2	4.7679E-11	133649.0	33.62	7.48
3	4.7229E-11	264117.0	130.06	3.79

Table 7: Calculated Mode Information

5.2.6 Capacitance Macromodel

We extract 200 capacitance data points and fit a [6 5 5 / 5 4 4] multivariate polynomial fraction, which has 401 fitting parameters. The resulting $\chi^2 = 2.74 \times 10^{-5}$. This required 2.8 hours to compute the capacitance data and 2.3 minutes to compute the macromodel.

5.2.7 Strain Energy Macromodel

We extract 150 strain energy data points and fit a [6 6 4] multivariate polynomial, which has 245 fitting parameters. The resulting $\chi^2 = 5.39 \times 10^{-28}$. This required 3.5 hours to compute the strain energy data and 27.1 seconds to compute the macromodel.

5.2.8 Dynamics Simulation

In Figure 42 and Figure 43, we present a couple of simulations of our macromodel for this structure. These took 1.5 minutes to compute on average.

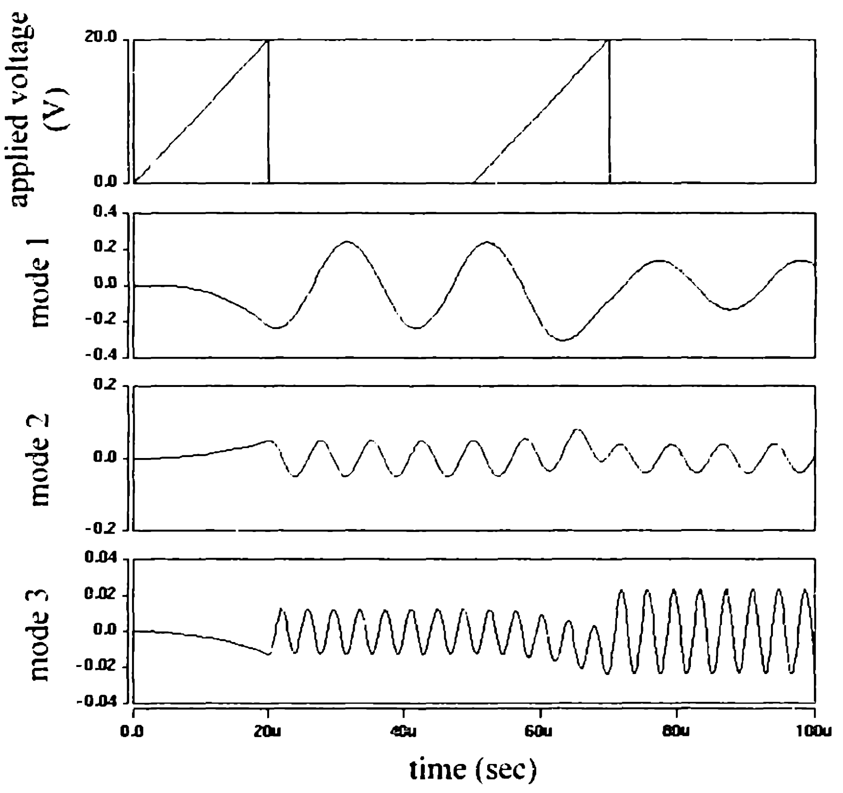


Figure 42: Response to a 20V Sawtooth Wave with 20μs Rise and 50μs Period

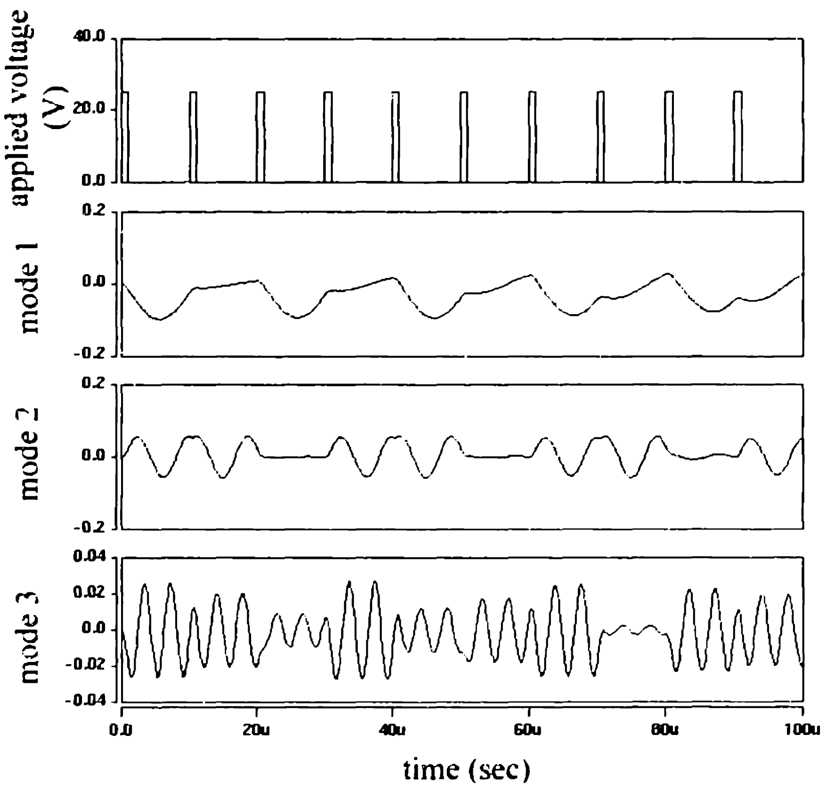


Figure 43: Response to a 25V Square Wave with 1μs Hold and 10μs Period

5.3 Suspended Plate with Unequal Support Beams

5.3.1 Structure Description

This device, depicted in Figure 44, is a 125x155x3μm plate suspended by four 85x15μm beams. Two of the beams are 3μm thick, one is 2μm thick, and one is 1μm thick. At the corner with the weakest support beam, a 62.5x70μm electrode is placed underneath, separated from the plate by a 4μm gap. This device is made out of polysilicon with a Young's modulus of 165GPa and a Poisson ratio of 0.23.

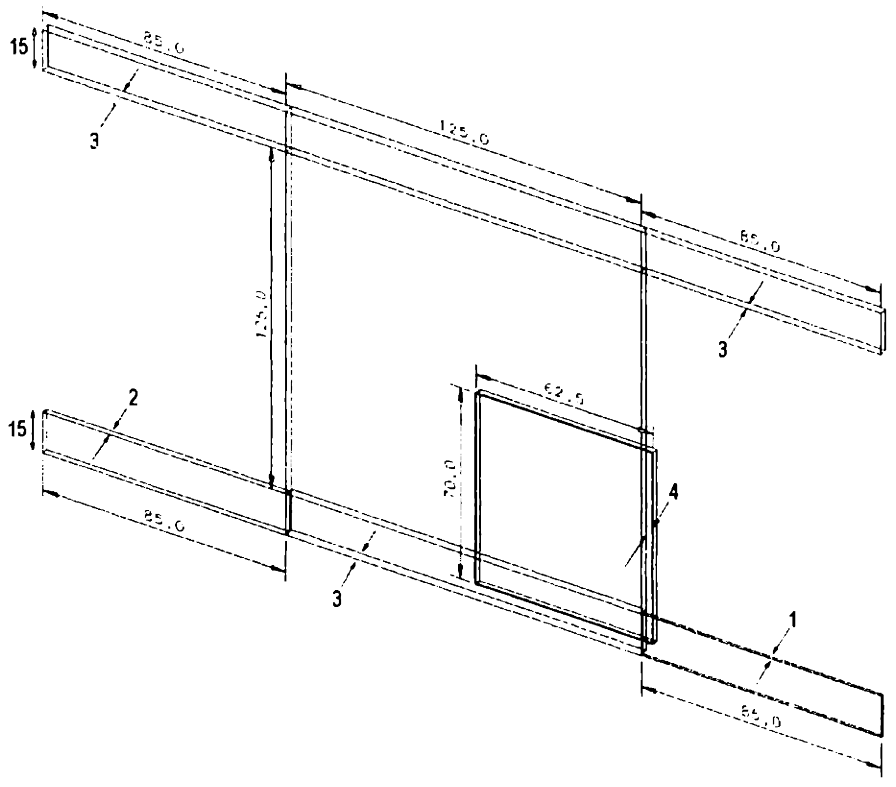


Figure 44: Schematic of Asymmetric Supported Plate

5.3.2 Solid Model

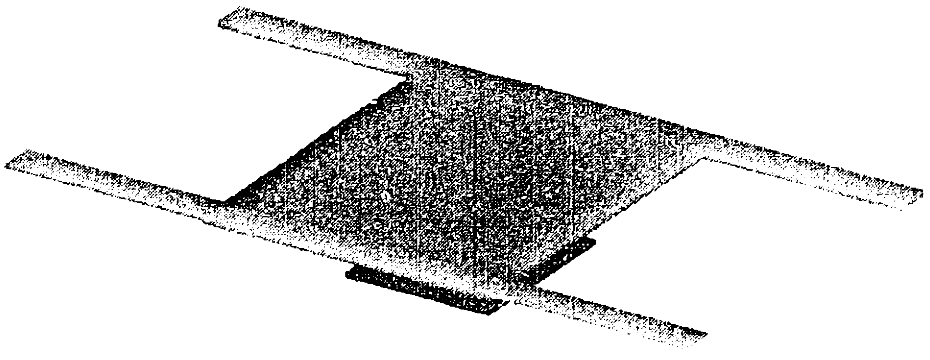


Figure 45: Solid Model of Asymmetric Supported Plate

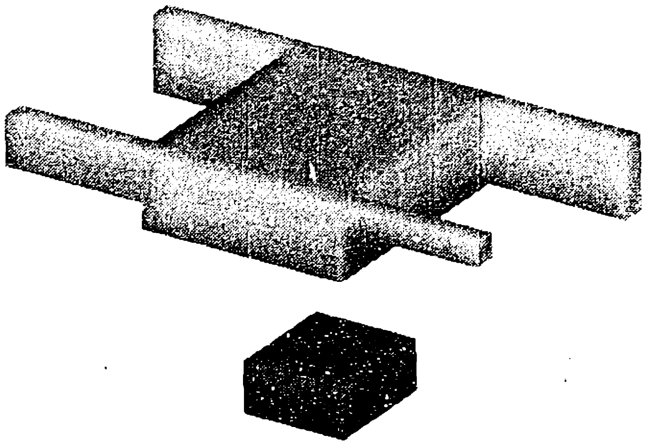


Figure 46: Solid Model of Asymmetric Supported Plate (20x Zoom)

5.3.3 Finite Element Model

The constructed finite element model, depicted in Figure 47, is comprised of 318 20-node brick elements, with a total of 2814 nodes.

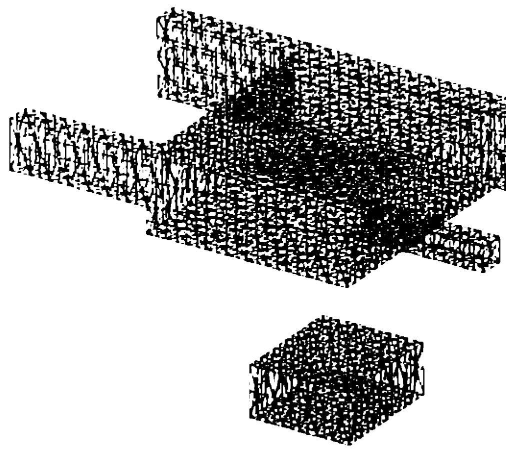


Figure 47: Finite Element Model for Asymmetric Supported Plate

5.3.4 Single Quasistatic Simulation

For the quasistatic coupled simulation, we actuate the device with 100 volts, which is 61% of the pull-in voltage for this structure. This required 4 CoSolve iterations to converge, requiring 7.5 minutes to compute. The resulting deformation is depicted in Figure 48. Table 6 presents the contribution of each of the calculated modes for this deformation.

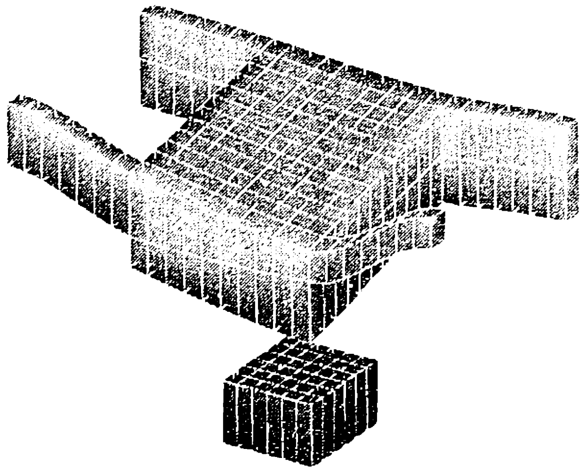


Figure 48: Quasistatic Response of Asymmetric Fixed-Fixed Beam to 100V Actuation

mode #	contribution	mode #	contribution
1	-0.457815000000	19	0.000216527000
2	0.042302700000	18	-0.000155325000
3	0.024837900000	20	-0.000142372000
7	0.005658390000	17	0.000135101000
4	-0.005222780000	23	0.000089974100
8	-0.001209240000	28	0.000077837800
6	0.000901406000	9	0.000073742200
10	-0.000700110000	30	0.000034852500
14	0.000504038000	24	-0.000032702100
15	-0.000486705000	26	-0.000032052700
27	-0.000390917000	12	0.000030375000
11	-0.000371473000	16	0.000020321100
13	0.000322518000	25	0.000018090900
5	0.000272911000	22	-0.000015363300
21	0.000237359000	29	-0.000002917070

Table 8: Projection of Quasistatic Response onto Mode Shapes

5.3.5 Modal Analysis

We calculated 30 mode shapes for this device. Figure 49 shows the three mode shapes that will ultimately be used for the generalized coordinates. Table 9 presents the calculated information about each of these modes. This required 11.6 minutes to compute.

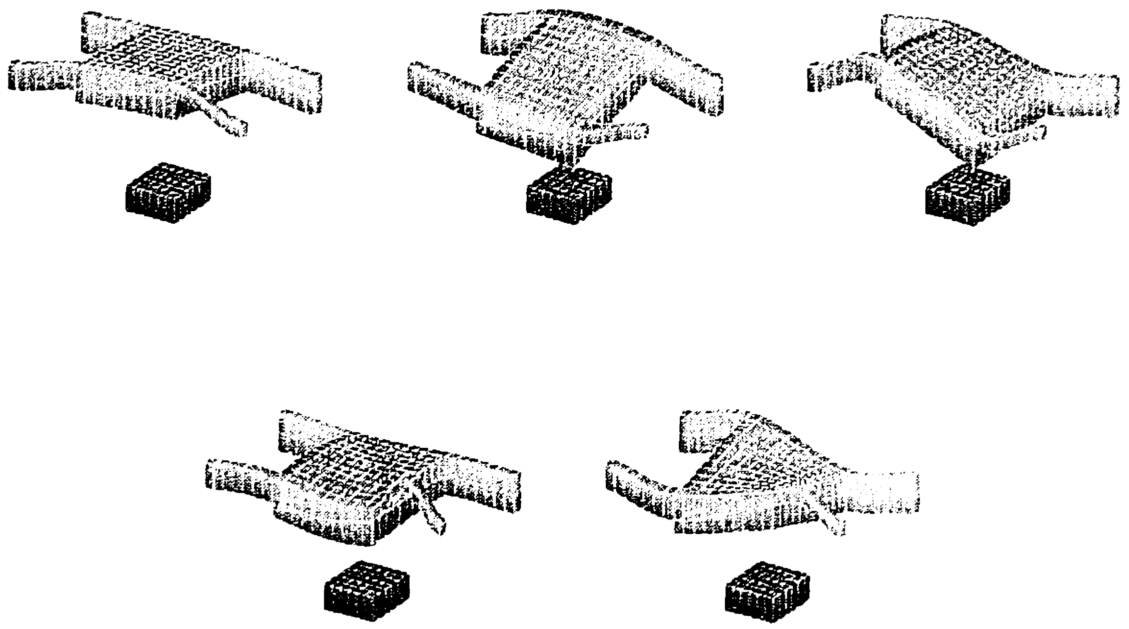


Figure 49: Mode Shapes Used to Comprise Basis Set

mode #	generalized mass (kg)	generalized frequency (Hz)	generalized stiffness	period (μs)
1	4.3397E-11	121063	25.11	8.26
2	3.5400E-11	272995	104.15	3.66
3	3.1356E-11	424306	222.86	2.36
7	1.5351E-12	1267430	97.35	0.79
4	1.1905E-11	930945	407.32	1.07

Table 9: Calculated Mode Information

5.3.6 Capacitance Macromodel

We extract 250 capacitance data points and fit a [4 3 3 2 2 / 3 2 2 1 1] multivariate polynomial fraction, which has 863 fitting parameters. The resulting $\chi^2 = 4.30 \times 10^{-5}$. This required 1.9 hours to compute the capacitance data and 30.7 minutes to compute the macromodel.

5.3.7 Strain Energy Macromodel

We extract 247 strain energy data points and fit a [4 2 2 2 2] multivariate polynomial, which has 405 fitting parameters. The resulting $\chi^2 = 1.39 \times 10^{-24}$. This required 7.5 hours to compute the strain energy data and 4.0 minutes to compute the macromodel.

5.3.8 Dynamics Simulation

In Figure 50 and Figure 51, we present a couple of simulations of our macromodel for this structure. These took 2.0 minutes to compute on average.

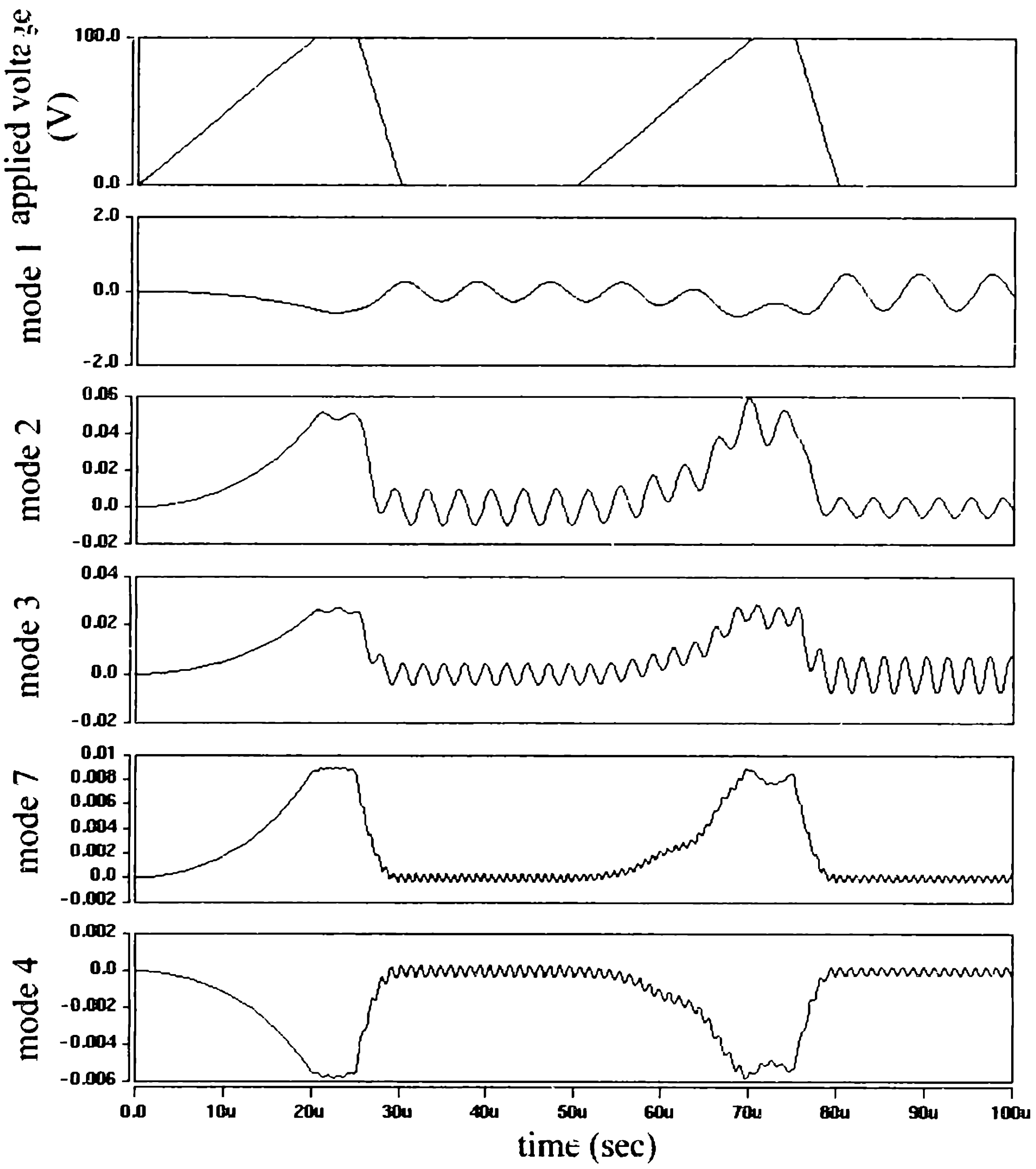


Figure 50: Response to a 100V Sawtooth Wave with 20μs Rise, 5μs Hold, and 50μs Period

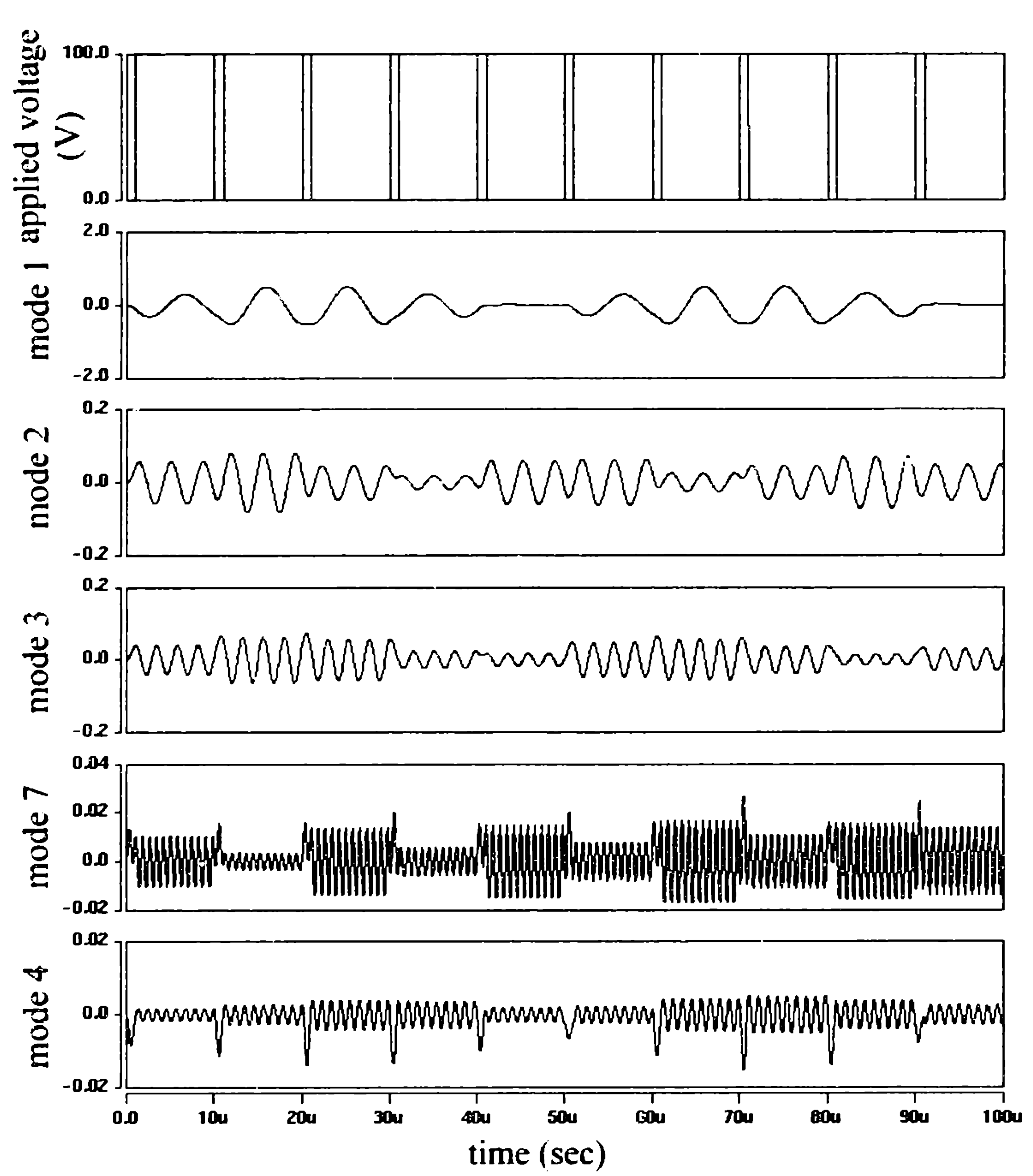


Figure 51: Response to a 100V Square Wave with 1μs Hold and 10μs Period

5.4 Summary

In this thesis, we have presented four different structures for which we have constructed macromodels using the Churn process. A detailed summary of these examples is presented in Table 10. Although computation time generally agrees with the analysis we presented in Section 4.1, we note a few exceptions. First, note that the doElectro() time for the asymmetric suspended plate is half that for the complex fixed-fixed beam cases. This is because FastCap takes advantage of co-located surface panels, and that the suspended plate model is very regional, dedicating many nodes to the off-center electrode area. Also, note that the number of iterations required for CoSolve to converge varies between 2 and 13. This depends upon the proximity of the actuating voltage to the pull-in voltage of the device, being that the closer the structure is driven to pull-in, the more difficult it is for CoSolve to converge. Finally, the dynamics simulation computation time presented in Table 10 were generated for "similar" circuits, and thus computation time will vary based upon the nature of the circuit in which the macromodel is embedded.

In order to better visualize the dependence of model complexity upon computation time, we present an abridged summary of the information presented in Table 10. Table 11 presents a list of the key parameters of a model that can affect computation time. First, there is the number of nodes N in the finite element model of the structure. This affects the computation time for all full three-dimensional mechanical and electrostatic analysis. Next, there is the number of calculated modes M. This mainly affects the modal analysis step. Then there is the number of basis shapes m, which is equivalent to the number of generalized coordinates to which the system is to be reduced. This has a tremendous (exponential) impact upon the number of data points and fitting parameters that are necessary to construct accurate macromodels for the different energy domains. Then, there is the proximity of the applied voltage to the quasistatic pull-in voltage of the structure. This will increase the number of iterations necessary during the coupled solution. Finally, the number of data points and fitting parameters necessary to construct the macromodels, which depend upon the desired macromodel accuracy of the designer, affect the data extraction and macromodel construction computation time, which comprise the bulk of the computation time for the Churn process. We conclude with Figure 52, which depicts a graphical representation of the itemized computation times for the four structures presented in this thesis.

	simple fixed-fixed beam	pre-stressed fixed-fixed beam	asymmetric fixed-fixed beam	asymmetric suspended plate
Model Information				
# nodes "N"	534	2664	2883	2814
# basis shapes "m"	1	3	3	5
Young's modulus (MPa)	165000	165000	165000	165000
Poisson ratio	0.23	0.23	0.23	0.23
initial stress (MPa)	0	-4	0	0
MEMCAD Function Calls				
doMech() initial step (sec)	10	73	53	63
doMech() subsequent step (sec)	10	54	68	53
doMode() step (sec)	11	202	437	640
doElectro() step (sec)	6	54	51	28
getStrainEnergy() step (sec)	2	7	7	15
Model Analysis				
# calculated modes "M"	3	16	30	30
doMech() initial step (sec)	11	72	50	56
doMode() step (sec)	11	202	437	640
total time (sec)	**22**	**274**	**487**	**696**
Coupled Solution				
pull-in voltage	98.4	8.1	31	165
applied voltage	80	8	30	100
applied voltage as % pull-in	81%	99%	97%	61%
doMech() initial step (sec)	8	74	55	70
doMech() subsequent step (sec)	10	54	68	53
doElectro() step (sec)	6	54	51	28
total time for 5 iterations (sec)	**98**	**662**	**718**	**528**
total time for 10 iterations (sec)	**178**	**1208**	**1313**	**933**
total time for 15 iterations (sec)	**258**	**1748**	**1908**	**1338**
actual coupled solve # iters	7	13	13	4
total coupled solve time (s)	**130**	**1532**	**1670**	**447**
Mode Shape Relevance Determination				
load mode shapes (sec)	1	12	23	24
construct contravariant basis set (sec)	0	3	11	12
load and project shape onto basis set (sec)	1	1	1	1
total time (sec)	**2**	**16**	**35**	**37**
Capacitance Extraction				
# capacitance data points	20	100	200	250
doElectro() step (sec)	6	54	51	28
total time (sec)	**120**	**5400**	**10200**	**7000**
Strain Energy Extraction				
# strain energy data points	20	96	150	247
doMech() step (sec)	12	82	78	95
getStrainEnergy() step (sec)	2	7	7	15
total time (sec)	**280**	**8544**	**12750**	**27170**
Capacitance Macromodel Construction				
model type	[4/4]	[544/433]	[655/544]	[43322/32211]
# fitting parameters	9	229	401	853
Chi squared error	3.59E-24	4.22E-44	2.74E-45	4.30E-45
# total steps	20	41	34	50
# progressive steps	18	18	18	23
# regressive steps	2	23	16	27
time per progressive step (sec)	0.001081872	0.501333	6.68515	68.4794
time per regressive step (sec)	0.000526316	0.186	1.01515	10.309
total time (sec)	**0.05**	**13.26**	**135.85**	**1844.01**
Strain Energy Macromodel Construction				
model type	[6]	[644]	[664]	[42222]
# fitting parameters	7	175	245	405
Chi squared error	4.08E-13	2.68E-28	5.39E-28	1.39E-24
# total steps	27	41	59	55
# progressive steps	12	18	27	25
# regressive steps	15	23	32	30
time per progressive step (sec)	0.001217948	0.2600302	0.729323	6.433422
time per regressive step (sec)	0.000384615	0.0928302	0.234138	0.982222
total time (sec)	**0.03**	**9.21**	**27.1**	**239.77**
total time of Churn process (hours)	**0.2**	**4.4**	**7.1**	**10.6**
Dynamics Simulation				
linear S.E. computation time (sec)	1.97	48.42	86.40	100.96
non-linear S.E. computation time (sec)	1.63	51.98	90.98	134.54

Table 10: Summary of Model Information

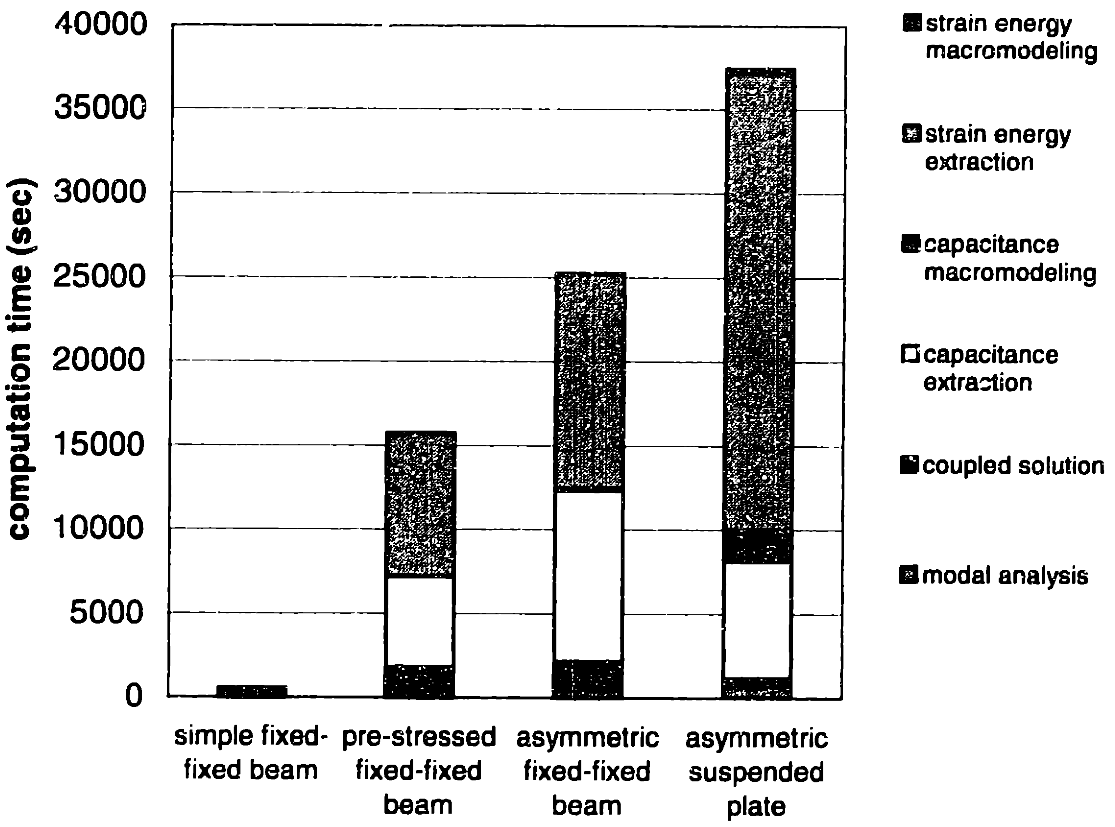


Figure 52: Summary of Computation Time
(Mode Relevance Determination is small and has been omitted.)

	simple fixed-fixed beam	pre-stressed fixed-fixed beam	asymmetric fixed-fixed beam	asymmetric suspended plate
Model Information				
# nodes "N"	534	2664	2863	2814
# calculated modes "M"	3	16	30	30
# basis shapes "m"	1	3	3	5
applied voltage as % pull-in	81%	99%	97%	61%
# capacitance data points	20	100	200	250
# capacitance fitting parameters	9	229	401	863
# strain energy data points	20	96	150	247
# strain energy fitting parameters	7	175	245	405
Computation Time (sec)				
modal analysis	22	274	487	696
coupled solution	130	1532	1670	447
mode relevance determination	2	16	35	37
capacitance extraction	120	5400	10200	7000
capacitance macromodel construction	0.05	13.26	135.85	1844.01
strain energy extraction	280	8544	12750	27170
strain energy macromodel construction	0.03	9.21	27.1	239.77
total time of Churn process (hours)	**0.2**	**4.4**	**7.1**	**10.6**

Table 11: Abbreviated Summary of Example Results

Chapter 6 Conclusion

In this thesis, we have reported the successful implementation of a methodology for automatically generating analytical macromodels for two conductor, conservative electrostatic-elastostatic microstructures, and for exporting these macromodels as analog simulator circuit elements that can be repeatedly used within a circuit simulator to determine dynamical behavior. These macromodels are fast to compute, requiring only the initial investment of computation time to construct them. Unfortunately, we have found that this methodology is limited to small displacements due to the sensitivity of full three-dimensional mechanical solvers to the representation of system node position by a linear superposition of basis vectors. Regardless, we have demonstrated that this methodology remains accurate for MEMS devices in their small-displacement, linear regime.

The concepts of macromodeling and automatic macromodel generation were well known and have been investigated in the past, but this research makes several advances. First, this thesis introduces the concept of fitting large multivariate polynomials and polynomial fractions in order to accurately replace computationally cumbersome full three-dimensional simulation. Next, this research introduced and implemented the automation of the Churn process from device concept to circuit simulation, enabling automated MEMS macromodel generation requiring only minimal interaction with a designer. Above all, this process enabled macromodel construction for a wide variety of complex MEMS devices that would otherwise be too difficult for designers to macromodel.

There are some clear directions for continued work on this research. First and foremost, a method must be devised to rectify the high strain energy calculations as discussed in Section 4.2. This might include allowing the structure to relax by reducing the number of fixed node constraints, increasing the number of generalized coordinates, choosing different basis shapes, or changing the state representation paradigm to use a basis set of mechanical loads to generate displacement states. For other work, the energy domain macromodeling approach should be extended to other conservative energy domains, such as magnetics. Finally, research must begin to macromodel dissipative energy domains and incorporate them into the equations of motion.

The prospect for computer aided macromodeling for MEMS will have tremendous impact on the field of MEMS design. Once it is possible to incorporate these models into existing simulators, designers will be able to explore the wide variety of uses these devices may have. It may be possible to close feedback loops around these devices and control their motion, enabling everything from voltage controlled flow-rates through microvalves to spectral analysis of gaseous materials. To date, the MEMS design process is still slow, but this research shows promise to accelerate the development and implementation of MEMS devices for real-world applications.

References

[1] J. Bryzek; K. Petersen, W. McCulley, "Micromachines on the March," *IEEE Spectrum*, May 1994, vol. 31, no. 5, pp. 20-31.

[2] S. D. Senturia, "Microfabricated structures for the measurement of mechanical properties and adhesion of thin films", *Proc. 4th Int'l Conf. Solid-State Sensors and Actuators (Transducers '87)*, Tokyo, 1987, pp. 11-16.

[3] S. D. Senturia, R. M. Harris, B. P. Johnson, S. Kim, K. Nabors, M. A. Shulman, J. K. White, "A computer-aided design system for microelectromechanical systems (MEMCAD)", *J. Microelectromechanical Systems*, Mar. 1992, vol. 1, pp. 3-13.

[4] J. R. Gilbert, P. M. Osterberg, R. M. Harris, D. O. Ouma, X. Cai, A. Pfajfer, J. White, S. D. Senturia, "Implementation of a MEMCAD system for electrostatic and mechanical analysis of complex structures from mask descriptions", *Proc. IEEE Workshop on Microelectromechanical Systems, MEMS '93*, Ft. Lauderdale, FL, Feb. 1993, pp. 207-212.

[5] J. R. Gilbert, R. Legtenberg, and S. D. Senturia, "3D coupled electro-mechanics for MEMS: applications of CoSolve-EM", *Proc. MEMS '95*, Ft. Lauderdale, FL, Feb. 1993, pp. 122-127.

[6] Microcosm Technologies, http://www.memcad.com/.

[7] ISE Integrated Systems Engineering AG, Technopark, Technoparkstrasse 1, Ch-8005 Zürich, Switzerland.

[8] J. Korvink, J. Funk, M. Roos, G. Wachutka, H. Baltes, "SESES: a comprehensive MEMS modelling system", *Proc. IEEE Workshop on Microelectromechanical Systems, MEMS '94*, Oiso, Japan, Jan 1994, pp. 173-176.

[9] IntelliSense, http://www.intellis.com/.

[10] S. B. Crary, Y. Zhang, "CAEMEMS: An integrated computer-aided engineering workbench for micro-electro-mechanical systems", *Proc. IEEE Micro Electro Mechanical Systems*, Napa Valley, CA, Feb. 1990, pp. 113-114.

[11] S. B. Crary, O. Juma, Y. Zhang, "Software tools for designers of sensor and actuator CAE systems", *Proc. Transducers '91: Int'l Conf. On Solid-State Sensors and Actuators*, San Francisco, CA. IEEE, Piscataway, NJ, 1991, pp. 498-501.

[12] G. M. Koppelman, "OYSTER, a three-dimensional structural simulator for microelectromechanical design", *Sensors and Actuators*, Nov. 1989, vol. 20, pp. 179-185.

[13] J. G. Korvink, J. Funk, and H. Baltes, "IMEMS Modelling", *Sensors and Materials*, 1994, vol. 6, pp. 235-239.

[14] F. Maseeh, R. M. Harris, S. D. Senturia, "A CAD architecture for microelectromechanical systems", *Proc. IEEE Micro Electro Mechanical Systems*, Napa Valley, CA, 1990, pp. 44-49.

[15] R. M. Harris, F. Maseeh, and S. D. Senturia, "Automatic generation of a 3-D solid model of a microfabricated structure", *Tech. Digest IEEE Solid-State Sensor and Actuator Workshop*, Hilton Head, SC, June 1990, pp. 36-41.

[16] Y. Zhang, S. B. Crary, and K. D. Wise, "Pressure sensor design and simulation using the CAEMEMS-D module", *Tech. Digest IEEE Solid-State Sensor and Actuator Workshop*, Hilton Head, SC, June 1990, pp. 32-35.

[17] R. A. Buser, N. F. de Rooij, "CAD for silicon anisotropic etching", *Proc. IEEE Micro Electro Mechanical Systems*, Napa Valley, CA, Feb. 1990, pp. 111-112.

[18] A. Koide, K. Sato, S. Tanaka, "Simulation of two-dimensional etch profile of silicon during orientation-dependent anisotropic etching", *Proc. IEEE Micro Electro Mechanical Systems*, Nara, Japan, Feb. 1991, pp. 216-220.

[19] C. H. Séquin, "Computer simulation of anisotropic crystal etching", *Proc. 6th Int'l Conf. Solid-State Sensors and Actuators (Transducers '91)*, San Francisco, CA, June 1991, pp. 801-806.

[20] S. D. Senturia, "CAD for microelectromechanical systems", *Transducers '95*, Stockholm, Sweden, 1995, vol. 2, pp. 5-8.

[21] E. S. Hung, Y.-J. Yang, and S. D. Senturia, "Low-Order Models For Fast Dynamical Simulation of MEMS Microstructures," *Transducers '97*, pp. 1101-1104.

[22] H. S. Cheng, C. H. T. Pan, *J. Basic Engineering*, March 1965, pp. 185-192.

[23] H. M. Park, D. H. Cho, *Chemical Engineering Science*, 1996, vol. 51, no. 1, pp. 81-98.

[24] S. R. Sipcic, A. Pecore, *Mathematical Modeling and Scientific Computation*, 1996, vol. 6.

[25] W. T. Thomson, *Theory of Vibrations with Applications*, CBS Publishers, New Delhi, 1988, pp. 117-178.

[26] A. W. Lees, D. L. Thomas, "Modal hierarchical Timoshenko beam finite elements", J. Sound Vibration, 1985, vol. 99, no. 4, pp. 455-461.

[27] G. K. Ananthasuresh, R. K. Gupta, and S. D. Senturia, "An Approach to Macromodeling of MEMS for Nonlinear Dynamic Simulation", in *Microelectromechanical Systems (MEMS)*, ASME Dynamic Systems & Control (DSC) ser. Vol. 59, part of *Proc. 1996 ASM Int'l Mechanical Engineering Congress and Exposition*, Am. Soc. Of Mechanical Engineers, New York, 1996.

[28] C. D. Cojocaru, P. Lerch, B. Kloeck, C. Bourgeois, and P. Renaud, "Complete transient simulations of electrostatic actuators", *Proc. MSM '98*, T 3.4.6.

[29] K. Milzner, "Computer aided macromodel generation (circuit simulation)", *Proc. 1993 Summer Computer Simulation Conf.*, Boston, MA, pp. 283-8.

[30] J. C. Regidor, M. I. G. de Guzman, V. M. Guzman, "GeMaT: a computer aided thyristor macromodel generator", *PEMC '96. 7th Int'l Power Electronics and Motion Control Conf.*, vol. 1, pp. 180-4.

[31] J. C. Regidor, V. M. Guzman, M. I. G. de Guzman, and S. Jimenez, "GeMaT/W: a computer aided thyristor macromodel generator for Windows," *32nd Universities Power Engineering Conference UPEC '97*, vol. 2, pp. 791-4.

[32] L. D. Gabbay, *Computer Aided Macro-Modeling of the Electromechanics of the Tilting Elastically Supported Plate*, M. S. Thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, May 1995, unpublished.

[33] D. Divekar, R. Raghuram, P. Wang, "Automatic generation of spice macromodels from N-port parameters", *Proc. 37th Midwest Symposium on Circuits and Systems*, 1994, vol. 1, pp. 670-673.

[34] H. A. Haus and J. R. Melcher, *Electromagnetic Fields and Energy*, Prentice Hall, 1989.

[35] J. Stoer, R. Bulirsch, et al., *Introduction to Numerical Analysis*, 2nd edition, Springer Verlag, Dec. 1992.

[36] V. A. Emeliv, A. M. Kononenko, "The number of plans for a multi-index selection problem", Dokl. Akad. Nauk BSSR, 1974, vol. 18, pp. 677-680.

[37] W. H. Press, S. A. Teukolsky, et al., *Numerical Recipes in C: The Art of Scientific Computing*, 2nd edition, Cambridge University Press, 1995, pp. 681-688.

[38] H. Goldstein, *Classical Mechanics*, 2nd edition, Addison-Wesley Publishing Company, 1981, pp. 16-21.

[39] I-DEAS, Structural Dynamics Research Corp., http://www.sdrc.com/

[40] K. Nabors, J. White, "FastCap: a multipole-accelerated 3D capacitance extraction program", *IEEE Trans. on Computer Aided Design*, Nov. 1991, vol. 10, no. 10, pp. 1447-1459.

[41] ABAQUS, Hibbitt, Karlsson, & Sorensen, Inc., http://www.hks.com/

[42] Glish, maintained by Darrell Schiebel, drs@nrao.edu

[43] MATLAB, The Mathworks, Inc., http://www.mathworks.com/

[44] MIT MEMCAD, http://www-mtl.mit.edu/MEMCAD/home.html

[45] MEMCAD User's Manual, Microcosm Technologies, Inc., http://www.memcad.com/

[46] I. S. Duff, A. M. Erisman, J. K. Reid, *Direct Methods for Sparse Matrices (Monographs on Numerical Analysis)*, Oxford University Press, September 1989.