Analysis of Realistic Large MEMS Devices

Per Ljung[1], **Martin Bächtold**[2], **Mirko Spasojevic**[2]

Abstract: This paper presents AutoMEMS®, a numerical simulation environment to efficiently analyze the behavior of large real-world MEMS designs. By automating surface-based model generation, meshing and field solver tools, it is possible to rapidly model large complex MEMS devices.

1 Introduction

Because MEMS devices utilize multiple energy domains, they are inherently difficult to design, analyze and optimize. This work addresses the verification of user-created designs by simulating the interacting 3D physical fields. Only by numerically solving the partial differential equations (PDE) describing these fields can an accurate representation of the MEMS device be obtained [Ljung (1996), Bächtold(1997)].

This premise of this work is to robustly automate the numerical solution of these fields, allowing both beginner and expert MEMS designers to rapidly design and optimize their devices. This entails that the simulator must be capable of modeling and analyzing realistic, very large systems with complex geometries. Further, the multi-physics PDE solver must be fast, accurate and fully adaptive.

Simulating large, realistic MEMS models has previously been prohibitive due to both the extensive user effort required to generate models and the computational cost of such analyses. This work robustly automates efficient and accurate MEMS analysis by eliminating substantially all user interaction required to generate accurate 3D simulations of devices described by photolithographic masks. This automatic procedure allows large, realistic MEMS devices to be accurately simulated in less than one hour on personal computers [Coyote (1999)] using the AutoMEMS computer aided design (CAD) software.

1.1 Automation

To accomplish these goals, it is necessary to automate several steps, including: (1) generating 3D model from photolithographic masks, (2) applying boundary conditions and material properties to model, (3) discretizing the model, and (4) solving the (coupled) PDEs on the model.

The steps shown in Fig. 1 have been successfully automated in AutoMEMS by using a boundary element method (BEM) discretization and solver. The ovals depict user input, while rectangles are automated steps that do not require any user interaction. Other MEMS simulators [Intellisense (1997), Gilbert (1995), ISE (1997)] use a combination of the finite element method (FEM) and BEM to simulate coupled multi-physics problems but are unable to automate each step. Further, the computational scaling of these previous simulators prevents their use in solving large, realistic MEMS devices.

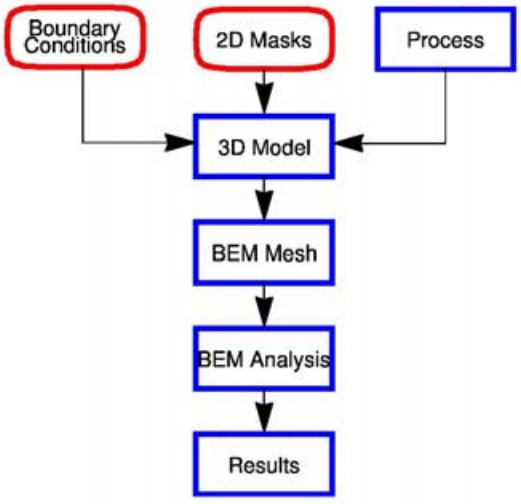

Figure 1 : Data flow to simulate a MEMS device.

1.2 Solving 3D fields with BEM

Because BEM only discretizes the boundaries or surfaces of the 2D or 3D geometry, the resulting BEM model is both very compact and easy for the user to generate. Practically any 3D solid model can be automatically robustly discretized into a valid BEM model.

BEM has the capability to solve linear, non-linear and time-varying partial differential equations (PDE). BEM solves for both the state and gradient of a PDE field. For example, in a electrostatic problem, the state is the potential and the gradient is the electrostatic flux. In a thermal problem, the state is the temperature and the gradient is the heat flux. In a elastostatic problem, the state is the displacement (x, y, z components) and the gradient is the traction (similar to stress with x, y, z components). Thus BEM allows extremely accurate results when the user is interested in capacitance or stress concentration factors.

For linear field analysis, the traditional BEM method [Brebbia

[1] Coyote Systems, Inc., pbljung@coyotesystems.com
[2] Coyote Systems, Inc.

(1994)] based on linear superpositions of Green function solutions enables solutions to Laplace, Navier, Stokes fields. The computational scaling of the traditional BEM method scales as $O(N^2)$ where N is the number of BEM nodes. For non-linear and time-varying analysis, the dual reciprocity boundary element method [Partridge (1992)] can be used.

To enable rapid solutions, the more efficient multipole accelerated Fast BEM method is used where both memory and cputime scales as $O(N \log N)$. This allows large, realistic geometric models to be rapidly simulated on standard PCs. To guarantee accurate solutions, error indicators and adaptive meshing are used to automatically refine the BEM model to a user specified solution accuracy on the state and gradients.

2 Typical application

The steps to simulate a MEMS comb-finger resonator using AutoMEMS are shown to demonstrate the functionality of the developed CAD tools.

A complete BEM model comprises a surface panel discretization of a 3D geometry, defined sets of surface panels called regions, material properties and boundary conditions applied to predefined regions.

The appropriate PDE is then solved using fast multipole accelerated BEM method. Error indicators evaluate the solved state and gradient PDE values to determine the global accuracy of the result. Where needed, the mesh is then automatically refined using p- and h-type refinement. P-type refinement increases the polynomial order of the element shape function, and h-type refinement splits an element into multiple pieces to reduce the element size.

2.1 Automatic geometry generation

By emulating MEMS process flows, it is possible to generate realistic 3D models [Elliot (1995), Scheckler (1992), Pelka (1991), Strasser (1995), Ljung (1998)] of a MEMS device given a 2D mask layout and process description.

AutoMEMS allows the user to generate planar and conformal 3D models from a standard IC or MEMS mask layout (e.g. CIF or GDSII format) in conjunction with a table-based description of the fabrication. This information includes deposition order, deposition thickness and material name. There is no limitation on the number of layers, dielectrics or boundary conditions.

To illustrate the automated model building, meshing and simulation, consider a simple MEMS electrostatic comb-finger resonator is shown in Fig. 2.

2.1.1 Specify process information

The process description assumes that every step consists of (a) planar deposition of a material indicated by a mask pattern, and (b) planar deposition of a specified dielectric everywhere

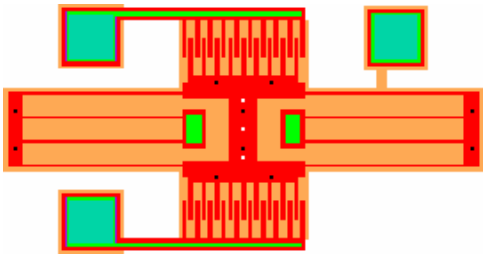

Figure 2 : Photolithographic CIF mask of a comb-drive resonator

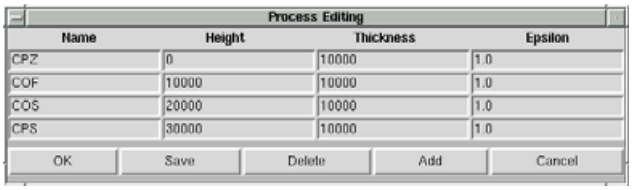

Figure 3 : Process description window

else. As a result, if no mask pattern exists on a layer, then only dielectric is deposited. The user must specify the layer names, bottom height, thickness and dielectric constants as shown in Fig. 3 to completely specify a planar VLSI fabrication process.

The process description emulates typical planar VLSI fabrication steps, it does not attempt to simulate the physics behind each fabrication step. As a result, the generated AutoMEMS model may not exactly coincide with an actual fabricated device since some effects (e.g. non-homogeneous sputtering, voids, chamfered edges, curved surfaces, non-vertical edges) are ignored.

2.1.2 Extrusion

For every photolithographic mask layer, a boolean union operation is performed to combine all the polygons in the mask layout. Each resulting n-sided polygon is then extruded and offset using the thickness and height specified in the process description. To support vias and anchors, boolean operations are also used to identify if the resulting 3D objects on different layers intersect. If a surface separates two identical materials, then the surface is eliminated, but if a surface forms a material interface between dissimilar materials, then the surface is retained.

2.2 Automatic meshing

Since AutoMEMS uses BEM, the discretization of a 3D model is substantially simplified since only the surfaces need to be identified and discretized which enables robust automatic mesh generation. The top and bottom surfaces of the extruded 3D geometry are discretized into triangular panels using Delauney meshing while the vertical surfaces are discretized into

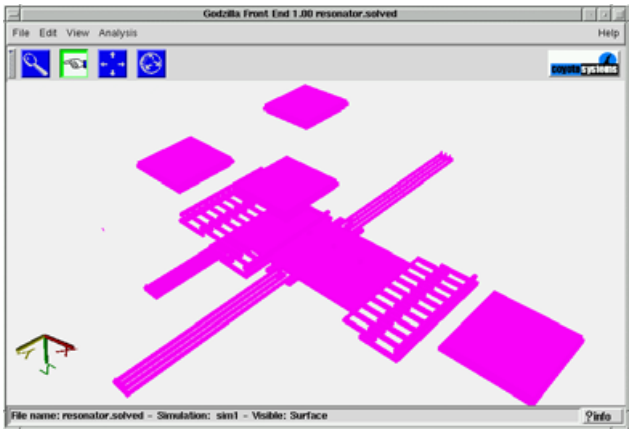

Figure 4 : AutoBEM display of generated 3D model

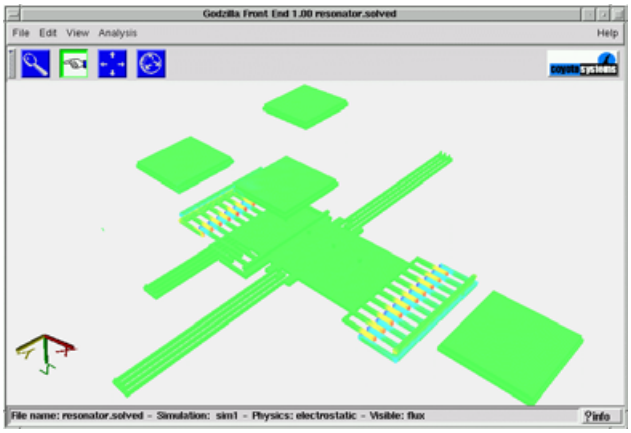

Figure 6 : Solved electrostatic flux on comb-finger resonator

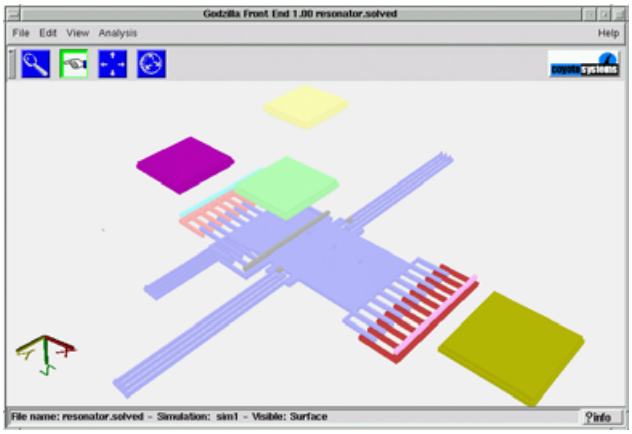

Figure 5 : User created regions on comb-finger model

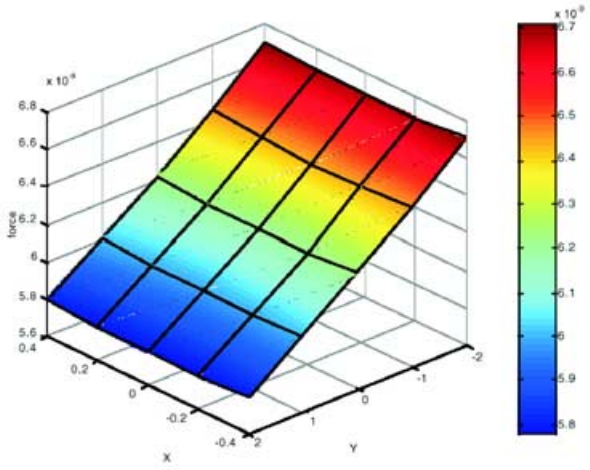

Figure 7 : Calculated charge, capacitance and forces for various resonator deflections

quadrilateral panels. This set of panels forms the initial BEM surface mesh. An example of the resulting discretized 3D model is shown in Fig. 4.

Rather than apply boundary conditions to the discretized mesh, AutoMEMS defines boundary conditions on sets of surfaces called regions. Using the graphic user interface (GUI), the user grouped several surfaces together to form several non-intersecting regions as shown in Fig. 5. If annotated GDSII layout masks are used to generate the model, then the region names are automatically identified from the masks.

The boundary conditions and material properties are specified in table-based GUI windows completing the AutoMEMS model generation. Arbitrary dielectric domains are supported, such as infinite surrounding space, silicon oxide and silicon nitride. Various applied boundary conditions are supported, including Neuman, Dirichlet, mixed and floating.

2.3 Field solver

By specifying the boundary conditions on the regions, the problem has been completely defined and the unknown field state or gradient can be solved using the BEM field solver. Fig. 6 shows the magnitude of the electrostatic flux after a simulation, where the color variations indicate that the electrostatic flux is concentrated in the comb-finger tips.

The electrostatic forces and charges for each panel and each region are available in the simulation results. On a typical PC (e.g. 500 MHz Pentium III processor, 128MB RAM), the entire flow from Fig. 2 to Fig. 6 takes about 2 minutes. By changing the displacement of the moving resonator, it is possible to efficiently create a table-based model of the resonator for system simulations as shown in Fig. 7.

Table 1 : Performance improvement achieved

	Sandia BEM [Womble (1994)]	Coyote AutoMEMS	Comment
Computer	massively parallel	personal computer	
CPU	1,900 Intel i860	1 Intel PentiumIII	2,000× cheaper?
Method	naive BEM	multipole accelerated BEM	improved algorithms
Combined Capacity	50E3 panels	20E6 panels	400× larger problems
Combined Speed	1.4E5 panels/hour	1.2E6 panels/hour	9× faster

2.4 Coupled electro-mechanical simulations

By creating both electrostatic and mechanical boundary conditions, it is possible to easily create coupled electromechanical MEMS simulations. Typically an electrostatic simulation is first solved, and the solved electrostatic gradient is used to calculate a mechanical traction. Using this inherited traction, the mechanical model is solved to obtain the new mechanical state (i.e. displacement). The electrostatic model typically inherits the new displacement and the cycle iterates until a suitable successful convergence criteria (e.g. small change in displacement, small change in capacitance) or failure criteria (e.g. collision, number of iterations, amount of cputime) are satisfied. The AutoMEMS GUI uses "wizards" to help create these coupling scenarios and convergence criteria.

Fig. 8 depicts an example of a series of coupled simulation steps of a torsional MEMS micromirror. The driving electrodes create an electrostatic torque deflecting the paddle. A typical use of numerical simulations is to calculate the electrostatic torque as a function of voltage and deflection as well as the pull-in voltage. Creating the model, specifying the initial boundary conditions, specifying the convergence criteria and running the coupled simulations near the pull-in voltage takes approximately 1 hour on a 500 MHz Pentium III processor.

2.5 Summary

The AutoMEMS software can efficiently generate 3D models given photolithographic masks of MEMS devices and analyze the coupled partial differential equations (usually electrostatic and elastostatic) to determine the behavior of the MEMS device. A large MEMS device such as Analog Devices' ADXL50 accelerometer in Figure 16 contains about 35,000 BEM panels including all the etch-holes. The AutoMEMS software has demonstrated peak speeds exceeding 1,250,000 panels/hour per cpu (500MHz Pentium III). The AutoMEMS software fully supports symmetric multiprocessing (SMP) machines with a near linear scaling for every added cpu. The largest model successfully solved contained over 20 million BEM panels.

Compared to a 1994 Gordon-Bell Prize winning BEM paper [Womble (1994)] using parallel supercomputers, AutoMEMS has demonstrated approximately 9 times speedup and 400 times larger model sizes using a single standard personal computer. The remainder of this paper discusses the technological developments underlying these capabilities and performance.

3 Technology

The technology underlying the surface-based AutoMEMS modeling, meshing and analysis is based on the boundary element method (BEM). BEM is a very accurate integral method in contrast to the more common differential finite element method (FEM). Both FEM and BEM methods are numerical methods to solve partial differential equations. However, the BEM approach requires a significantly smaller model, and requires smaller computer resources enabling the simulation of extremely large, complex models.

Assuming that both FEM and BEM use polynomial order p element shape functions, then the BEM numerical solution is typically more accurate. FEM calculates a polynomial order p approximation to the state and numerically differentiates this to obtain a polynomial order $p-1$ approximation to the gradient. In contrast, BEM simultaneously calculates polynomial order p approximations to both the state and gradient.

As a result, engineering values (e.g. capacitance, electrostatic force, mechanical stress concentration factors) which are based on the gradient are more accurate [Brebbia (1984)] with integral-based BEM models.

The mathematics underlying the naive BEM and multipole accelerated BEM are first presented. Refinement methods including error indicators and heuristic meshing goals are next described. To reduce the actual degrees of freedom a development called constrained BEM (CBEM) is next described. Finally tunnel acceleration, a method to reduce the geometric complexity of the model, is presented.

3.1 Naive boundary element method

The boundary element method [Brebbia (1984)] uses Green's Functions to describe the effects of loadings on the entire domain. Using the divergence theorem, the BEM equation is expressed in terms of integrals on the boundary of the domain

$$c \cdot u(\xi) = \int_\Gamma q(x) \cdot u^*(\xi, x) \cdot dx - \int_\Gamma u(x) \cdot q^*(\xi, x) \cdot dx \qquad (1)$$

with the known state boundary condition $u(x) = \overline{u}(x)$ on the Dirichlet boundary $x \in \Gamma_D$, and the known gradient boundary condition $q(x) = \overline{q}(x)$ on the Neumann boundary $x \in \Gamma_N$. The appropriate Green's functions are denoted u^* and q^*. A simple 2D BEM model is shown in Fig. 9.

The integral coefficients in Eq. 1 are calculated for all node pairs for each boundary segment. This coefficient genera-

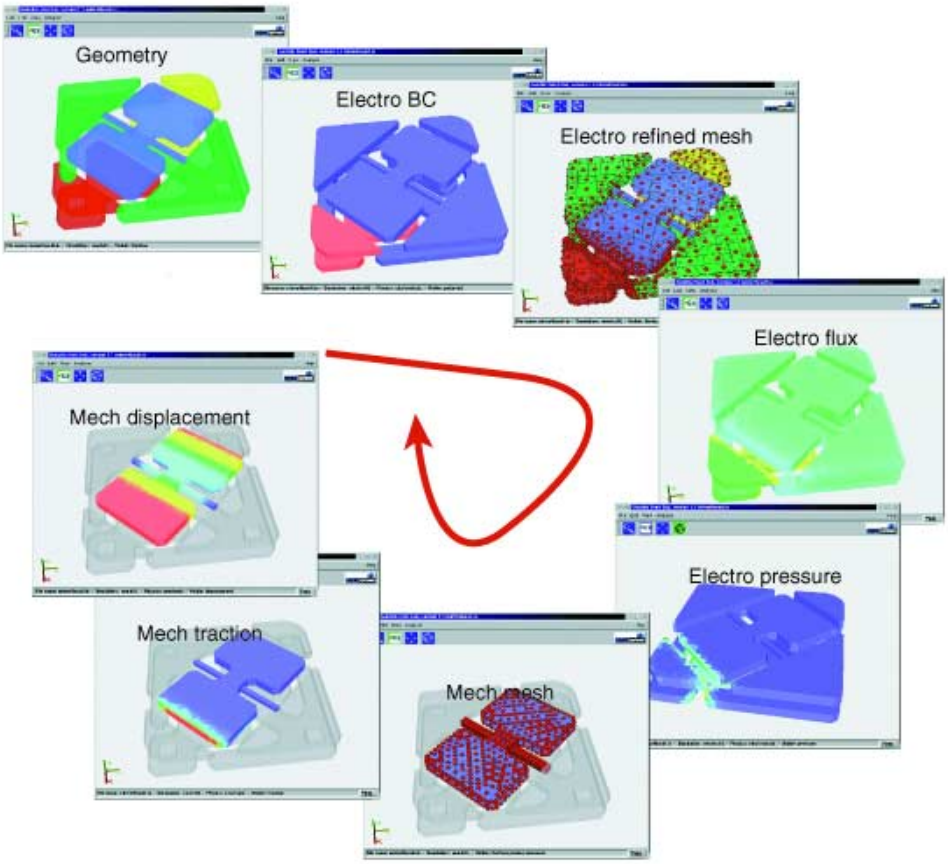

Figure 8 : Coupled MEMS micromirror simulations

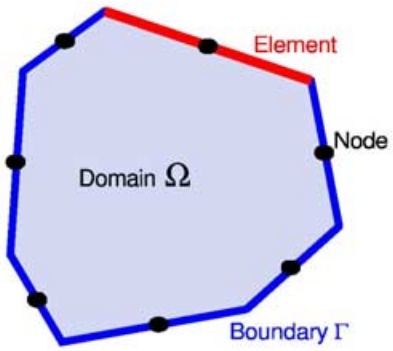

Figure 9 : 2D BEM model with constant elements

tion is very computationally expensive because the naive BEM method calculates interactions with all node pairs resulting in a $O(N^2)$ computational scaling.

All the integral coefficients are assembled into the dense matrices H and G describing the effect of the states u and gradients q, respectively.

$$H \cdot \begin{bmatrix} u_0 \\ \cdots \\ u_{N-1} \end{bmatrix} = G \cdot \begin{bmatrix} q_0 \\ \cdots \\ q_{N-1} \end{bmatrix} \tag{2}$$

The known states $\overline{u}$ and gradients $\overline{q}$ specified in the boundary conditions are inserted into the above equation resulting in a standard linear system which can be solved for the unknowns.

$$A \cdot \begin{bmatrix} x_0 \\ \cdots \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} b_0 \\ \cdots \\ b_{N-1} \end{bmatrix} \tag{3}$$

Given a BEM solution on the boundary of the domain, post-processing can be used to calculate the state or gradient on arbitrary points in the domain or on the boundary. This capability is also used to evaluate error-estimators on the boundary.

3.2 Multipole accelerated BEM

The naive BEM can be dramatically accelerated by using multipole accelerated BEM [Greengard (1988), Nabors (1991), Bächtold(1997)] which clusters distant nodes together using multipole expansions eliminating the need to calculate all the boundary integrals.

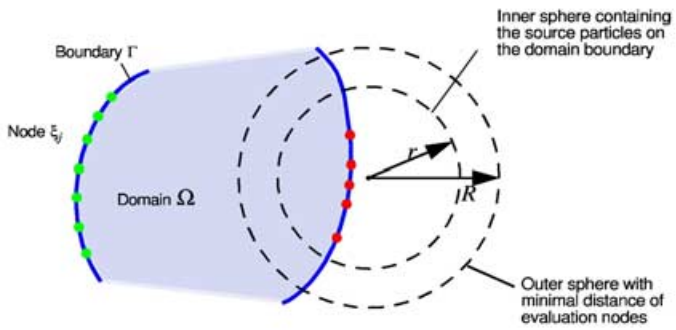

Figure 10 : Clustering of distant nodes reduces required number of computations

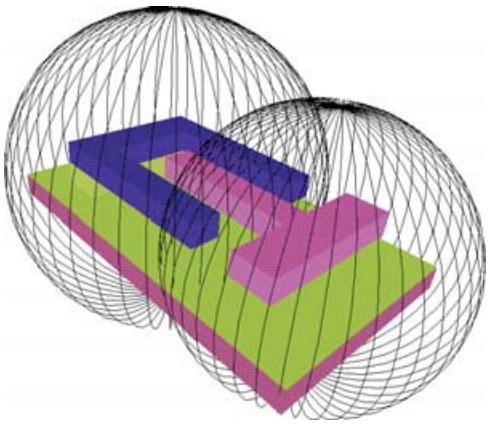

Figure 11 : Hierarchical decomposition level 1

Nodes in close physical proximity are calculated directly using Eq. 1, while distant clustered boundary integrals are now computed using multipole expansions. As an example, the gradient integral in Eq. 1 can be approximated by

$$\int_{\Gamma} q(x)u^*(\xi,x)dx \approx \sum_{n=0}^{p} a_n \cdot M_n(\xi - c) \tag{4}$$

where M_n is a multipole function, a_n is the appropriate weighting value and c is the center of expansion of the multipole.

This allows integral evaluation by clusters instead of one element at a time, introducing sparsity into BEM. Further, it is possible to decompose each cluster into hierarchical clusters, where the multipole coefficients depend only on the multipole coefficients of the lower cluster [Greengard (1988)]. Fig. 11-Fig. 13 illustrates the hierarchical decomposition and clustering of a sample comb-finger MEMS device. Several hundred decomposition levels are typically used.

Instead of the naive BEM computational scaling of $O(N^2)$, MA-BEM dramatically reduces the computational effort and cputime required for accurate analysis to a $O(N \log N)$ computational scaling, where N are the number of BEM nodes.

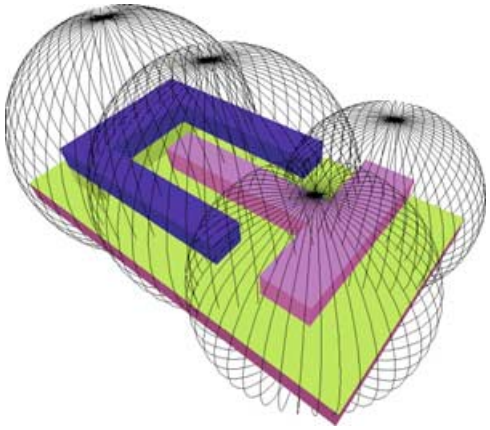

Figure 12 : Hierarchical decomposition level 2

The approximation introduced with multipole expansions have known error bounds [Greengard (1988)] dependent on the cluster radius and level of hierarchical clusters, therefore appropriate multipoles can be chosen which result in tight error bounds to allow very accurate simulation results.

3.3 Element refinement

BEM calculates very accurate solutions to PDEs. There are essentially only two sources of error in these calculations: (1) numerical integration accuracy and (2) geometric discretization accuracy. Coyote uses an automatic identification of the integration accuracy using Gaussian-Legendre integration to obtain fast and efficient numerical integration. The second item reflects that the model may have curved surfaces (approximated as a p-order polynomial) or that the state or gradient may vary nonlinearly (approximated as a p-order polynomial).

Currently this tool only supports BEM elements with straight edges or planar faces with constant, bilinear, biquadratic or bicubic shape functions. The extensions to support curved edges or surfaces are straightforward, but require considerably more complex numerical and analytical singular integrations.

To capture the effects of a nonlinearly varying solutions, the model discretization must be allow an accurate "fit" to the physical solution. For example, if the user a priori knows that the result varies logarithmically, then he shouldn't expect a good fit with a single linear element interpolation. However, a very good fit may be obtained using several linear elements or only a few quadratic elements.

To ensure consistently accurate simulations, Coyote has developed an automatic method to mesh and refine the BEM model. This allows the user to specify a desired accuracy (e.g. 2% error on the states/gradients, or a 1% error on the capacitance) and then let the software refine the mesh until this accuracy is achieved.

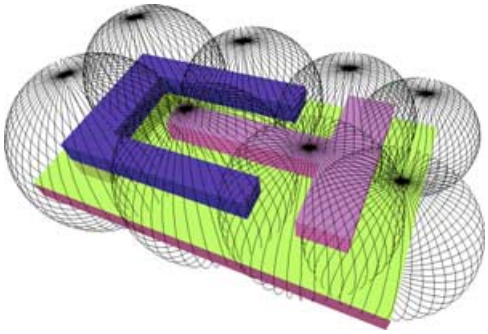

Figure 13 : Hierarchical decomposition level 3

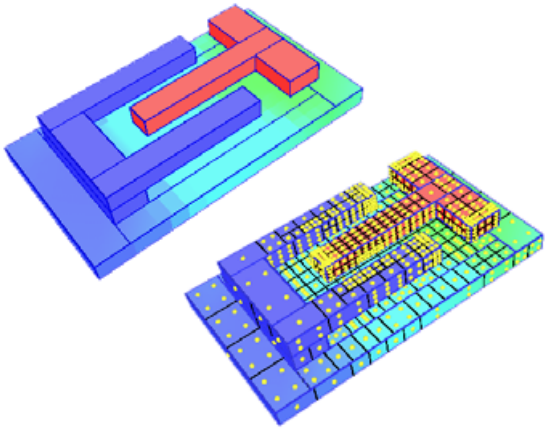

Figure 14 : Automatic mesh refinement based on error indicators

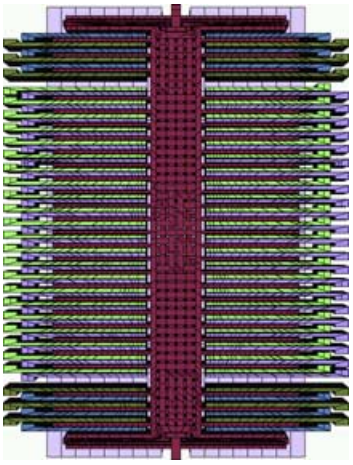

Figure 15 : ADXL50 model generated from layout

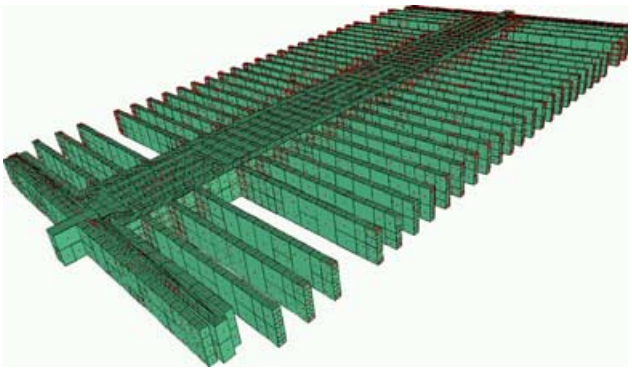

Figure 16 : Detail of the adaptively refined ADXL50 moving proof mass. Note the automatic concentration of nodes at the comb-finger tips and anchors

3.3.1 Error indicators

To achieve this automatic mesh refinement, the initial mesh is solved and then post-processing evaluates the states and gradients at points inside each element. For example, if a particular element is a triangle with a bilinear shape function, then the state/gradient error indicator evaluations [Bächtold, Korvink (1997)] on the surface should map closely to a plane face. If they do not, then this indicates that the size of the element should be reduced (h-refinement) or that the polynomial shape function of the element should be increased (p-refinement). If all error indicators show that the elements are well-formed, then the adaptive refinement is finished and the results are reported to the user. If some elements have too large errors, then the model is again refined until all errors are below the user-specified limits.

To illustrate this automatic adaptive mesh refinement, consider the initial BEM mesh of a MEMS comb-finger shown in Fig. 14. The number of dots on each element indicate the order of the shape function. A single dot indicates that the element is a constant element, 4 dots indicate a bilinear shape function, 9 dots indicate a biquadratic shape function and 16 dots indicate a bicubic shape function.

After 6 iteration cycles, the refined mesh is visible in Figure 14. Note that intuitively the electrostatic flux will vary greatly along the comb-finger tips, and the automatic mesh refinement has captured this behavior by refining these elements. Elements with small errors are not refined. This error indicator driven refinement approach is substantially faster than refining all elements for large models.

Adjacent elements with mixed orders are also valid CBEM meshes, as can be seen in Fig. 18 which demonstrates a large bicubic element adjacent to several smaller bilinear elements. Only the unconstrained nodes are degrees of freedom and solved using BEM. In contrast, the constrained nodes are automatically identified from the geometry and interpolated from the BEM solution.

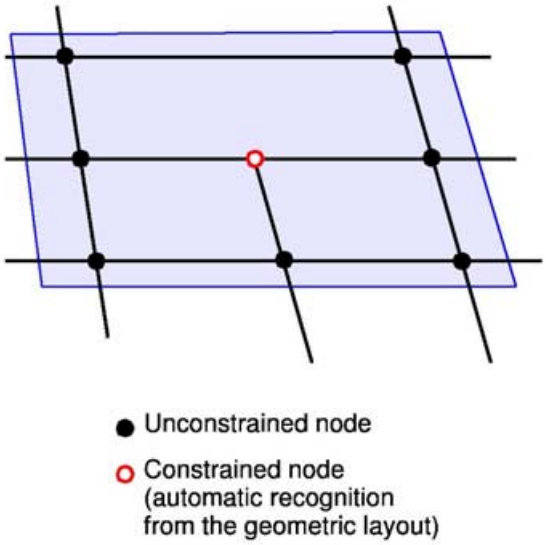

Figure 17 : CBEM model allows discontinuous meshes using constrained nodes.

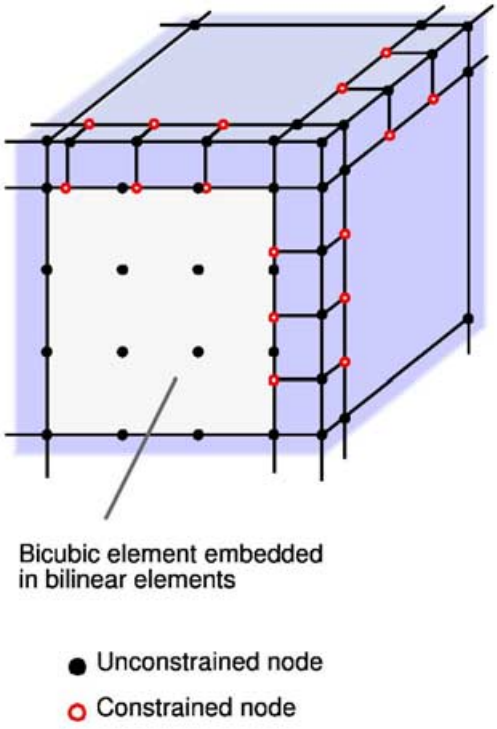

Figure 18 : 3D CBEM model showing how mixed order and mixed size elements can legally be combined.

3.3.2 Heuristic meshing

The meshed model of the ADXL50 accelerometer [Analog (1999)] is easily generated using the procedure outlined in "Automatic Geometry Generation". Note the high fidelity of the model, including the etch-holes.

Similar to the comb-drive resonator, it is expected that the electrostatic flux will be concentrated along the edges and corners since physics dictates that there is a flux singularity along sharp edges. Since the capacitance of the fingers is obtained by integrating the electrostatic charge, it is important to efficiently model the comb-fingers.

Using the automatic error indicators described in "Element Refinement", the software identifies which elements have large relative errors and only refines those identified elements. This effect can also be cheaply replicated using heuristics based on the variation in normals of adjacent BEM elements. This can efficiently be used to identify panels on edges or corners for refinement. This results in a very efficient method to generate well meshed model as can be seen in Fig. 16 which will ensure high-accuracy capacitance simulations.

3.4 Constrained boundary element method

Constrained mesh refinement provides the benefits of both continuous and discontinuous BEM meshes. This allows for simple discontinuous subdivision of existing elements (See Fig. 17) while still resulting in continuous nodal solutions. The Constrained Boundary Element Method (CBEM) [Bächtold, Ljung (1998)] identifies and solves for unconstrained nodes in a manner similar to BEM, while the identified constrained nodes are assigned nodal values based on interpolation of the solved constrained nodes. Using CBEM enables simple mesh refinement while still allowing for continuous states and gradients. As a result, the degrees of freedom of the system are reduced to the number of unconstrained nodes, typically reducing the DOF by a factor of 4-6. This implies that the cputime and memory requirements are also reduced by 4-6$\times$ with no accuracy loss.

CBEM supports general mesh geometries and allows anisotropic mixed-order element refinement. Further, unlike multipole acceleration, the reduction in DOF is Greens's function independent and valid for all BEM simulations, including electrostatic, thermal, elastostatic, thermoelastic, poisson, stokes and coupled fields. CBEM is fully automatic, and no user intervention is required. Experimental results show that minimal additional computation cost is incurred to interpolate the constrained nodal values. The CBEM model of the accelerometer in Fig. 19 has approximately one quarter the number of degrees of freedom compared to a standard BEM model, and as a result, the CBEM model is solved in approximately one-quarter the time compared to the fast BEM approach.

3.5 Tunnel acceleration

While fast constrained BEM has a very good computational scaling of $O(N \log N)$, the number of BEM panels N can still

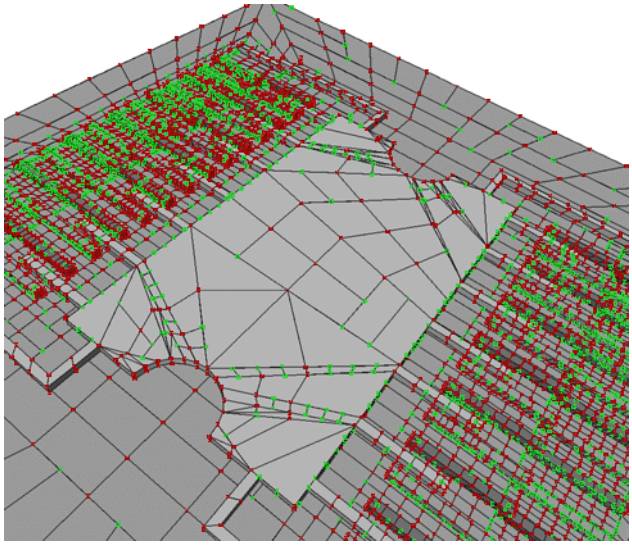

Figure 19 : CBEM model of accelerometer reduces degrees of freedom by 6x, which reduces the cputime by 4x compared to fast multipole accelerated BEM

be very large for large, complex models. To achieve even faster computations, it is possible to automatically selectively eliminate geometry which does not significantly contribute to the solution. This method is called "tunnel acceleration".

BEM solves for a 3D field and enforces that the field value is zero at infinity. For many problems, the field's region of interest is much closer to the geometry than infinity, perhaps limited to within 20 microns of the region of interest. After heuristically determining a tunnel radius, the field solution on the tunnel surface should be zero (to approximate infinity). If the result is not negligible, then the tunnel radius was too small and the method is repeated with a larger tunnel radius. Fig. 20 shows a portion of a VLSI interconnect where an aggressor net is surrounded by a tunnel (dark gray) and the remaining geometry is discarded (light gray). The self and cross capacitances of the aggressor net to all the victim nets can be extracted in about 10 seconds on a typical PC using tunneling acceleration.

Depending on the geometry of the problem, tunnel acceleration can provide memory and cputime reductions of more than 40x while other problems are not accelerated at all. By comparing the charge results for various tunnel radii, we have verified that the tunnel algorithms can result in less than 1% errors for electrostatic simulations.

Similarly to VLSI interconnects, MEMS can also be accelerated using tunnel acceleration. Fig. 21 depicts the elements in the tunnel around the driver finger region, and Fig. 22 shows the calculated electrostatic flux in this area. The sense comb-

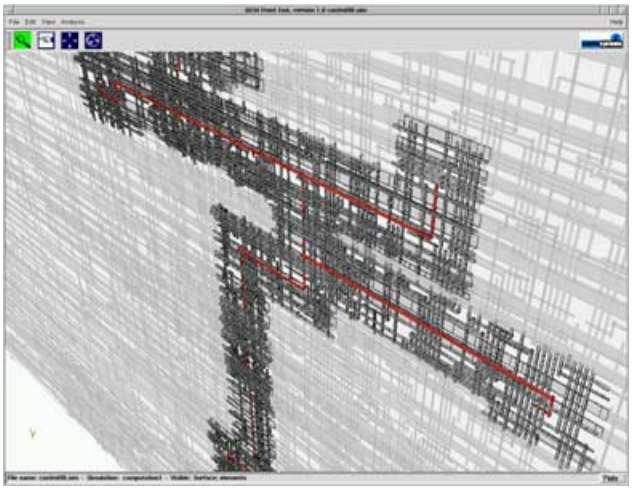

Figure 20 : Geometric truncation using a "tunnel" around the field of interest.

fingers are not included in the tunnel making the BEM model significantly smaller. This electrostatic field simulation for the drive interdigitated comb-fingers allows rapid capacitance coupling and electrostatic force calculations to be calculated in 20 seconds on a typical PC.

4 Summary

By automating the model generation, mesh refinement and analysis, this work enables the rapid and accurate analysis of large, realistic MEMS devices to be performed by users without extensive numerical simulation training. This work can similarly be used to automatically calculate cross-capacitances in large 3D submicron integrated circuits.

Acknowledgement: This work is supported by DARPA under contract F30602-96-2-0305.

References

Bächtold, M. (1997): *Efficient 3D computation of electrostatic fields and forces in microsystems*. Ph.D. thesis, ETH Zurich.

Bächtold, M. (1997): *Efficient 3D Computation of Electrostatic Fields and Forces in Microsystems*. ETH Zurich, Physical Electronics Lab., ISBN 3-907574-17-6, Zurich, Switzerland.

Bächtold, M.; Korvink, J.; H.Baltes (1997): An error indicator and automatic adaptive meshing for 3D electrostatic boundary element simulations. In *Boundary Elements XIX*, pp. 709–718, Southampton, UK. Comput. Mechanics Publ.

Bächtold, M.; Ljung, P. (1998): The constrained boundary element method, a technique allowing general surface meshes

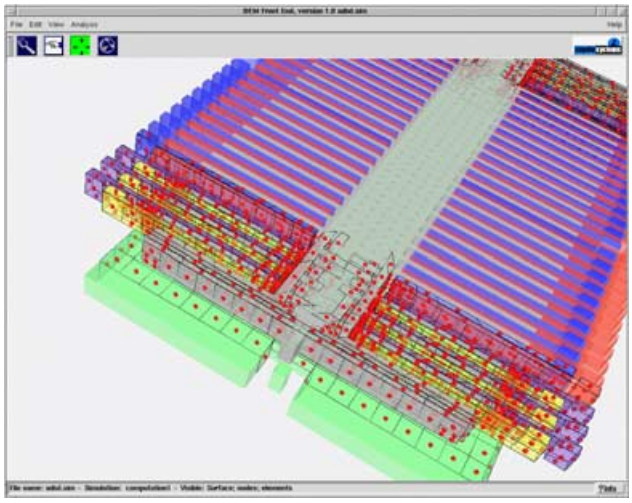

Figure 21 : Tunnel around magenta/yellow driving comb-fingers in ADXL50.

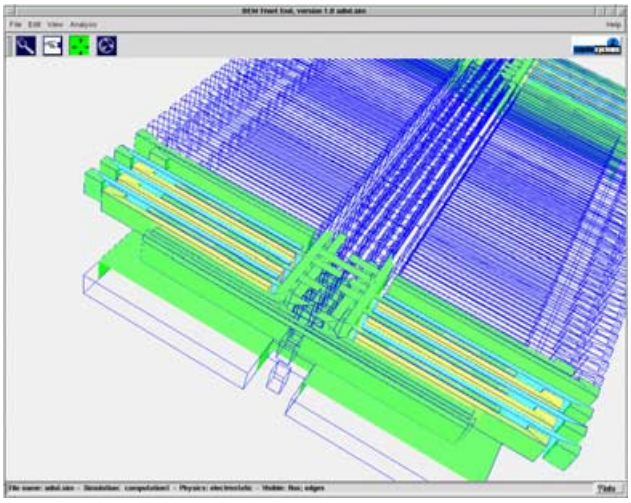

Figure 22 : The calculated charge on the drive comb-fingers is displayed.

in MEMS simulations. In *Tech. Digest Solid State Sensor and Actuator Workshop*, pp. 201–204, Hilton Head Is.

Brebbia, C.; Telles, J.; Wrobel, L. (1984): *Boundary Element Techniques, Theory and Application in Engineering*. Springer Verlag, Berlin.

Coyote Systems. *Users Guide reference manuals (http://www.coyotesystems.com/pdf/autobem.pdf)*, 1999.

Elliot, J.; Allan, G.; Walton, A. (1995): The automatic generation of conformal 3D data for interconnect capacitance simulation. In *Proc. VLSI Multi-Level Interconnect Conf.*, pp. 664–666.

Gilbert, J.; Legtenberg, R.; Senturia, S. (1995): 3D coupled electro-mechanics for MEMS: Applications of CoSolve-EM. In *Proc. MEMS 95*, pp. 122–127, Amsterdam, NL.

Greengard, L. (1988): *The Rapid Evaluation of Potential Fields in Particle Systems*. MIT Press, Cambridge MA.

Integrated Systems Engineering AG, Zurich. *Modeling of Semiconductor Technology, Devices and Systems, reference manuals*, 1997a.

Intellisense Corp., Wilmington, MA. *IntelliCad and IntelliFab reference manuals*, 1997b.

Ljung, P. (1996): Sequential solutions of field equations using a single BEM model. In *Solid State Sensor and Actuator Workshop*, pp. 117–122, Hilton Head Island, SC.

Ljung, P.; Bächtold, M. (1998): Automatic model generation and analysis for MEMS. In *Proc. ASME'98*.

Nabors, K.; White, J. (1991): FastCap: A multipole-accelerated 3-D capacitance extraction program. In *IEEE Trans. on CAD of Integrated Circuits and Systems*, volume 10, pp. 1447–1459.

Partridge, P. (1992): *The Dual Reciprocity Boundary Element Method*. Elsevier.

Pelka, J. (1991): Three-dimensional simulation of ion-enhanced dry-etch processes. *Microelectronic Engng.*, vol. 14, pp. 269–281.

Saad, Y.; Schulz, M. (1986): GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, vol. 3.

Scheckler, E.; Neureuther, A. (1992): Coupling model and algorithms for 3D topography simulation: Plasma etching, ion milling and deposition in SAMPLE-3D. In *Proc. Workshop on Num. Mod. of Processes and Devices for Integrated Circuits*, pp. 9–14.

Strasser, E.; Selberherr, S. (1995): Algorithms and models for cellular based topography simulation. *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 14, no. 9, pp. 1104–1114.

Womble, D.; Greenberg, D.; Wheat, S.; Benner, R.; Ingber, M.; Henry, G.; Gupta, S. (1994): Applications of boundary element methods on the intel paragon. In *Proceedings of Supercomputing 1994*, pp. 680–684, Washington DC.