An Integrated Simulator for Coupled Domain Problems in MEMS

Robert M. Kirby, George Em Karniadakis, Oleg Mikulchenko, *Associate Member, IEEE*, and Kartikeya Mayaram, *Senior Member, IEEE*

***Abstract*—An integrated circuit and microfluidic simulator is presented in this paper. It allows coupled simulation of flow, structure, thermal, and electrical domains. The overall architecture and various algorithms including the coupling of the circuit and microfluidic simulators are described. An application of the simulator is demonstrated for a controlled microliquid dosing system using detailed numerical models for the fluid field, a low-dimensional model for the flow sensor, and circuit elements for the electronic control. Unlike other simulators which employ mostly lumped models for the various components, we formulate here a methodology for distributed systems. [569]**

***Index Terms*—Couopled domains, distributed systems, macromodels.**

I. Introduction

A. Coupled-Domain Problems

In coupled-domain problems, such as flow-structure, structure-electric or a combination of both, there are significant disparities in temporal and spatial scales. This, in turn, implies that multiple grids and heterogeneous time-stepping algorithms may be needed for discretization, leading to very complicated and consequently computationally prohibitive simulation algorithms. Simplifications are typically made with one of the fields represented at a reduced resolution level or by low-dimensional systems or even by equivalent lumped dynamical models. For example, consider the electric activation of a cantilever microbeam made of piezoelectric material. The emphasis may be on modeling the electronic circuit and the motion, and thus a simple model for the motion-induced hydrodynamic damping may be constructed avoiding full simulation of the flow around the beam.

A possible construction of low-order dynamical models is by projecting the results of detailed numerical simulations onto spaces spanned by a very small number of degrees of freedom—the so-called *nonlinear macromodeling* approach (see [1]). To clarify the concept of a macromodel we give a specific example (taken from [1]) for a suspended membrane of thickness t and deflected at its center by an amplitude d under the action of uniform pressure force P. Let us also denote by $2a$ the length of the membrane, by E the Young's module, by ν the Poisson ratio, and by σ the residual stress. One can use analytical methods to obtain the resulting form of the pressure-deflection relation (e.g., power series assuming a circular thin membrane). This can be extended to more general shapes and nonlinear responses, for example,

$$P = \frac{C_1 t}{a^2} + \frac{C_2 f(\nu)}{a^4} \frac{E}{1-\nu} d^3 \tag{1}$$

where C_1 and C_2 are dimensionless constants that depend on the shape of the membrane, and $f(\nu)$ is a slowly varying function of the Poisson ratio. This function is determined from detailed finite element simulations over a range of length a, thickness t, and material properties ν and E. Such "best-fits" are tabulated and are used in the simulation according to the specific structure considered, without the need for solving the partial differential equations governing the dynamics of the structure. Another type of a macromodel based on neural networks training will be presented later for a flow sensor.

Unfortunately, construction of such macromodels is not always possible, and this lack of simplified models for the many and diverse components of microsystems makes system-level simulation a challenging task. On the other hand, model development for electronic components (transistors, resistors, capacitors, etc.) has reached a state of maturity. Therefore, considerable attention should be focused on models for the nonelectronic components. This is necessary for the design and verification of complete microsystems. In this paper, we describe an integrated approach for simulation of microsystems with the emphasis being on microfluidic systems. To this end, we resort to full simulation of the fluidic system, which also involves interactions with moving structures. To illustrate the formulation more clearly, we present next a target simulation problem that represents the aforementioned challenges.

B. A Prototype Problem

An example of a microfluidic system is a microliquid dosing system shown schematically in Fig. 1. This system is made up of a micropump, a microflow sensor, and an electronic control circuit. The electronic circuit adjusts the pump flow rate so that a constant flow is maintained in the microchannel. A realization of this system is shown in Fig. 2 along with the details of the control circuit. The simulation of the complete system requires models for the micropump, the microflow sensor, and the electronic components shown in Fig. 2. When models, low-order or full-physics models, are available for all components, including the fluid flow, the complete system can be simulated using a standard circuit simulator such as SPICE [2], [3].

Manuscript received May 1, 2000; revised November 8, 2000. This work is supported in part by DARPA under agreement number F30602-98-2-0178. Subject Editor R. T. Howe.

R. M. Kirby and G. E. Karniadakis are with the Division of Applied Mathematics, Brown University, Providence, RI 02912 USA.

O. Mikulchenko and K. Mayaram are with the Department of Electrical and Computer Engineering, Oregon State University, Corvallis, OR 97331 USA.

Publisher Item Identifier S 1057-7157(01)04818-1.

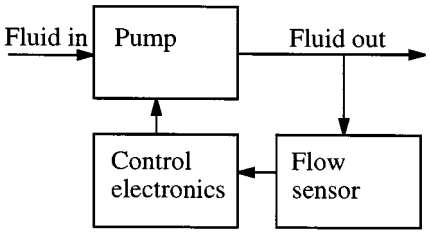

Fig. 1. Block diagram of a generic microfluidic system. The flow sensor senses the flow rate which is controlled by the electronic circuit controlling the pump.

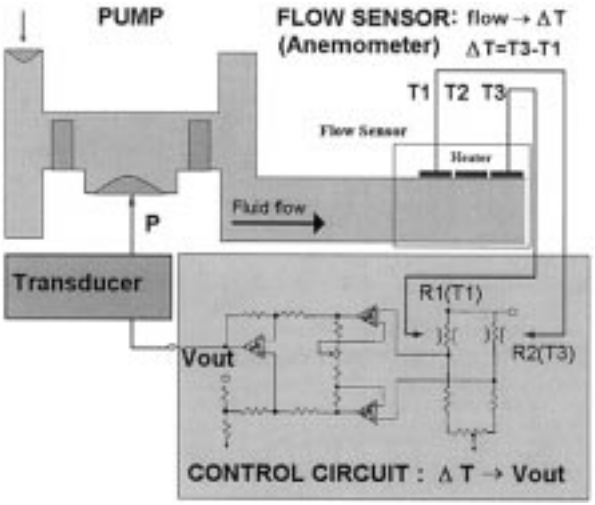

Fig. 2. Realization of the microfluidic system showing the electronic control circuit. The fluid flow determines the temperature ΔT of the flow sensor. This temperature is transformed by the control electronics into the voltage V_{out}, which in turn controls the pump pressure P by a transformation of the voltage to a proportional pressure.

In the absence of macromodels for the micropump and the microflow sensor, the typical approach for microsystem simulation makes use of lumped-element equivalent circuit descriptions for these devices [4]. However, such an approach has two main limitations:

- it is suitable only for open-loop systems, where there is no feedback from the output to the input;
- it is applicable only for small-signal conditions.

These two limitations arise in the model development process where several assumptions are made in order to construct the lumped-element equivalent circuits. Therefore, this approach would not be suitable when the large-signal behavior of a closed-loop system is of interest.

To address the above problem, we present a coupled circuit/microfluidic device simulator that efficiently couples the discretized Navier–Stokes equations describing a microfluidic device (numerical model) to the solution of circuit equations. Such a capability is unique in that it allows direct and efficient simulation of microfluidic systems without the need for mapping finite element descriptions into equivalent networks [4] or analog hardware description languages (AHDLs) [5].

The paper is organized as follows: An overview of coupled circuit and device simulation is given in Section II followed by a description of the circuit and fluidic simulators in Section III. The details of the coupled circuit/fluidic simulator are presented in Section IV and an illustrative example is described in Section V. Conclusions are provided in Section VI.

II. Coupled Circuit-Device Simulation

Coupled simulation techniques have previously been used for the simulation of a sensor system [6]. In this approach, the finite element program ANSYS [7] is coupled to an electrical simulator PSPICE [8]. Although such an approach has been demonstrated to work for system simulations, the coupling is not efficient. Special coupling algorithms and time-stepping schemes are required to enable fast simulation of microsystems. Therefore, a tight coupling between the circuit and device simulators is necessary for simulation efficiency [9], [10].

The coupled circuit-device simulator allows verification of microfluidic systems. It provides accurate large and small-signal simulation of systems even in the absence of proper macromodels for the microfluidic devices. On the other hand, the coupled simulator is important for constructing and validating macromodels. As important effects (such as highly nonlinear or distributed behavior, compressibility or slip-flow), are identified, they can be implemented in the macromodels and verified for system simulation using the coupled simulator. Furthermore, critical devices can be simulated using the full physics-based numerical models when there are stringent accuracy requirements on the simulated results.

The concept of a coupled circuit and device simulator has proved to be extremely beneficial in the domain of integrated circuits. Since the first of such simulators, MEDUSA [11], was available in the early 1980s, there has been significant work addressing coupled simulation. These activities have focused on improved algorithms, faster execution speeds, and applications. Commercial TCAD vendors also support a mixed circuit-device simulation capability [12], [13]. Since the computational costs of these simulators are high, they are not used on a routine basis. However, there are several critical applications in which these simulators are extremely valuable. These include simulation of RF circuits [14], single-event-upset simulation of memories [15], simulation of power devices [16], and validation of nonquasistatic metal–oxide–semiconductor field-effect transistor (MOSFET) models [17].

The coupled circuit-device simulator for microfluidic applications is illustrated in Fig. 3. This simulator supports compact models for the electronic components and available macromodels for microfluidic devices. In addition, numerical models are available for the microfluidic components which can be utilized when detailed and accurate modeling is required. As an example, specific components such as microvalves, pumps, and flow sensors are shown in Fig. 3. The coupling of the circuit and microfluidic components is handled by imposing suitable boundary conditions on the fluid solver. This simulator allows the simulation of a complete microfluidic system including the associated control electronics. The details of the various simulators and coupling methods are described in the sections below.

One of the biggest disadvantages of such an approach is the high computational cost involved. The main cost comes from solving the 3-D time-dependent Navier–Stokes equations in complex geometric domains. Thus, efficient flow solvers are

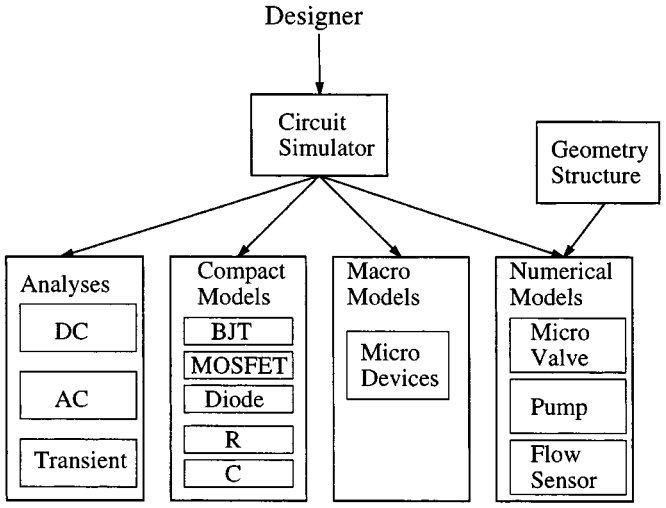

Fig. 3. The coupled circuit-fluidic device simulator. Microfluidic systems, including the control electronics, can be simulated using a hierarchy of numerical models for all microcomponents.

critical to the success of a coupled circuit-microfluidic device simulator. Any performance improvements in the solution of the Navier–Stokes equations directly translate into a significant performance gain for the coupled simulator.

III. Overview of Simulators

The circuit simulator employed here is based on the circuit simulator SPICE3f5 [3] and the microfluidic simulator on the code $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ [18], [19]. A brief description of the algorithms and software structure of each of these simulators is provided in this section.

A. The Circuit Simulator: SPICE3

Electrical circuits consist of many components (resistors, capacitors, inductors, transistors, diodes, and independent sources) which are described by algebraic and/or differential relations between the component's currents and voltages. These relationships are called the *branch constitutive relations* [20]. The circuits also satisfy conservation laws known as the Kirchhoff's laws; these laws result in algebraic equations. Therefore, a circuit is described by a set of coupled nonlinear differential algebraic equations, which are both highly nonlinear and stiff, and this imposes certain limitations on the solution methods. One of the most commonly used analysis is the *time-domain transient analysis*. It involves *time discretization* using a linear multistep method of the backward-differentiation type suitable for stiff ODEs is used [20]; *linearization* via a Newton–Raphson method, and *algebraic system solution* using sparse matrix techniques [21]. Specifically, the time-domain simulation algorithm consists of the following steps [20]:

1) read circuit description and initialize data structures;
2) increment time $t_n = t_{n-1} + h$;
3) update values of independent sources at t_n;
4) predict values of unknown variables at t_n;
5) apply time-integration to capacitors and inductors;
6) apply linearization to nonlinear circuit elements;
7) assemble linear circuit equations;
8) solve linear circuit equations;
9) check convergence; if not converged, go to step 6);
10) estimate local truncation error;
11) select new time step h; rollback time if truncation error is unacceptable;
12) if $t_n < t_{\text{stop}}$ go to step 3).

B. The Fluid Simulator: $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$

The flow solver corresponds to a particular version of the code $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$, which is a general purpose CFD code for simulating incompressible, compressible and plasma flows in unsteady three-dimensional geometries. The major algorithmic developments are described in [22] and [23] and the capabilities are summarized in Fig. 4. The code uses meshes similar to standard finite element and finite volume meshes, consisting of structured or unstructured grids or a combination of both. The formulation is also similar to those methods, corresponding to Galerkin and discontinuous Galerkin projections for the incompressible and compressible Navier–Stokes equations, respectively. Field variables, data and geometry are represented in terms of hierarchical (Jacobi) polynomial expansions [18]; both iso-parametric and super-parametric representations are employed. These expansions are ordered in vertex, edge, face and interior (or bubble) modes. For the Galerkin formulation, the required C^0 continuity across elements is imposed by choosing appropriately the edge (and face in 3-D) modes; at low order expansions this formulation reduces to the standard finite element formulation. The discontinuous Galerkin is a flux-based formulation, and all field variables have L^2 continuity; at low order this formulation reduces to the standard finite volume formulation. This new generation of Galerkin and discontinuous Galerkin spectral/hp element methods implemented in the code $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ do not replace but rather extend the classical finite element and finite volumes that the CFD practitioners are familiar with [18]. The additional advantage is that convergence of the discretization and thus solution verification can be obtained without remeshing (h-refinement) and that the quality of the solution does not depend on the quality of the original discretization. In Fig. 4 we summarize the major current capabilities of the general $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ code for incompressible, compressible and even plasma flows.

In particular, for microflows both the compressible and incompressible versions are used. For gas microflows we account for rarefaction by using *velocity-slip* and *temperature-jump* boundary conditions as described in [24], [25]. An extension of the classical Maxwell's boundary condition is employed in the code in the form

$$U_g - U_w = \frac{Kn}{1 - bKn}(\nabla U)_w \cdot \hat{n}. \tag{2}$$

Here we define the Knudsen number $Kn = \lambda/L$ with λ the mean free path of the gas molecules and L the characteristic length scale in the flow. Also, U_g is the velocity (tangential component) of the gas at the wall, U_w is the wall velocity, and $\hat{n}$ is the unit normal vector. The constant b is adjusted to reflect the physics of the problem as we go from the slightly rarefied regime (*slip flow*) to the transition regime ($Kn \approx 1$) or free

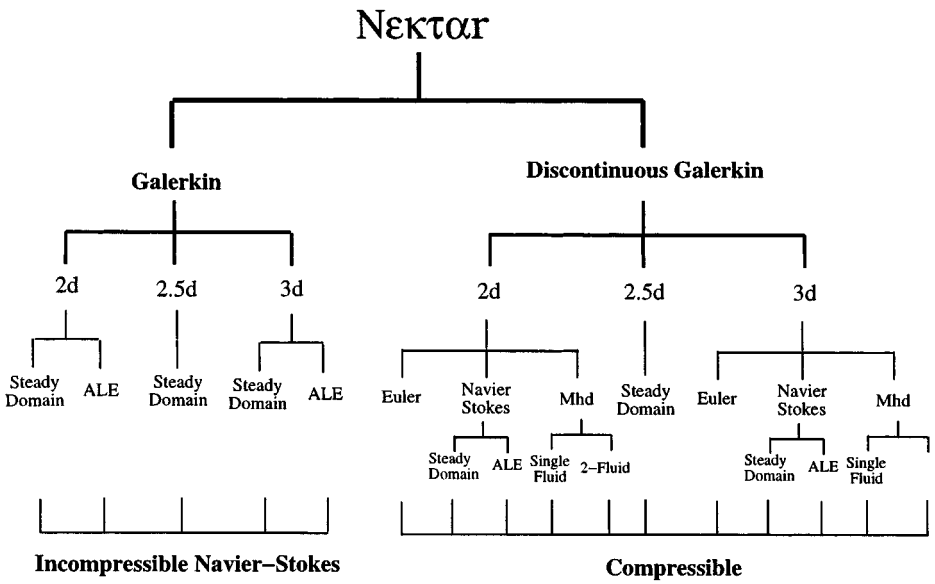

Fig. 4. Hierarchy of the $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ code for transient nonlinear flow analysis. Note that "2d" refers to two-dimensions, similarly "3d" to three-dimensions, and "2.5d" refers to a three-dimensional capability but with one of the geometric directions being homogeneous in the geometry. Also, ALE refers to moving computational domains required in dynamic flow-structure interactions. Gaseous microflows can be simulated by either the compressible or incompressible version depending on the pressure/density variations.

molecular regime ($Kn > 5$–10). For $b = 0$, we recover the classical linear relationship between velocity-slip and shear stress, first proposed by Maxwell. However, for $b = -1$ we obtain a second-order accuracy ([25]), and in general for $b \neq 0$ (2) leads to finite slip at the wall unlike the linear boundary condition (for $b = 0$) used in most codes. The boundary condition in (2) has been used with success in the entire Knudsen number regime, i.e., $Kn \approx 0$–200, (see [25]).

One of the key points in obtaining **efficiency in** simulations of moving domains is the type of discretization employed in the flow solver. In $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$, we employ the so-called **h-p** version of the finite element method with spectral Jacobi polynomials as basis functions. Convergence is obtained via a dual path in this approach, either by increasing the number of elements (h *-refinement*) or by increasing the order of the spectral polynomial (p*-refinement*). In the latter case a faster convergence is obtained without the need for remeshing. Instead, the number of degrees of freedom is increased in the *modal space* by increasing the polynomial order (p) while keeping the mesh unchanged. It is of course the cost of constructing the mesh that is orders of magnitude higher in time-dependent simulations both in terms of computer and human time.

Regarding the type of elements (subdomains), $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ uses hybrid meshes, i.e., both structured and unstructured meshes. For example, in three-dimensional simulations a hybrid grid may consist of tetrahedra, hexahedra, triangular prisms, and even pyramids. In Fig. 5 we plot the mesh used in the simulation of the pump, and in Fig. 6 we plot the flow field at three different time instances. This geometry has been studied extensively by Beskok & Warburton in [26] using the same code $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ both in two- and three-dimensions where an optimization of the pump was sought. Here we have modified the function of the pump in order to demonstrate the features of the integrated simulator rather than the flow response as was done successfully in [26].

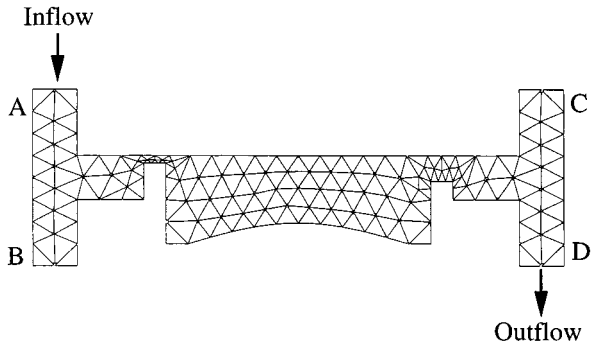

Fig. 5. Mesh of the pump used in the flow simulator $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ (see Section V-C). This device was first introduced by Beskok and Warburton (1998) as a mixing device between two microchannels. Here B and C are blocked so the device is operating as a pump from A to D.

In the following, we briefly describe how we formulate the algorithm for a compatible and efficient flow-structure coupling.

1) Formulation for Flow-Structure Interactions: We consider the incompressible Navier–Stokes equations in a time-dependent domain $\Omega(t)$

$$u_{i,t} + u_j u_{i,j} = -(p\delta_{ij})_j + \nu u_{i,jj} + f_i \text{ in } \Omega(t) \quad (3)$$

$$u_{j,j} = 0 \text{ in } \Omega(t) \quad (4)$$

where ν is the viscosity and f_i is a body force. We assume for clarity homogeneous boundary conditions; velocity-slip boundary conditions can be included relatively easily in the Galerkin framework as mixed (Robin) boundary conditions. For details of the derivation of the variational form we refer the reader to [27]. The final variational statement is

$$\frac{d}{dt}\int_{\Omega(t)} v_i u_i d\boldsymbol{x} + \int_{\Omega(t)} [v_i(u_j - w_j)u_{i,j} - v_i u_i w_{j,j}] d\boldsymbol{x} = \int_{\Omega(t)} [v_{i,j} p\delta_{ij} - \nu v_{i,j} u_{i,j} + v_i f_i] d\boldsymbol{x}. \quad (5)$$

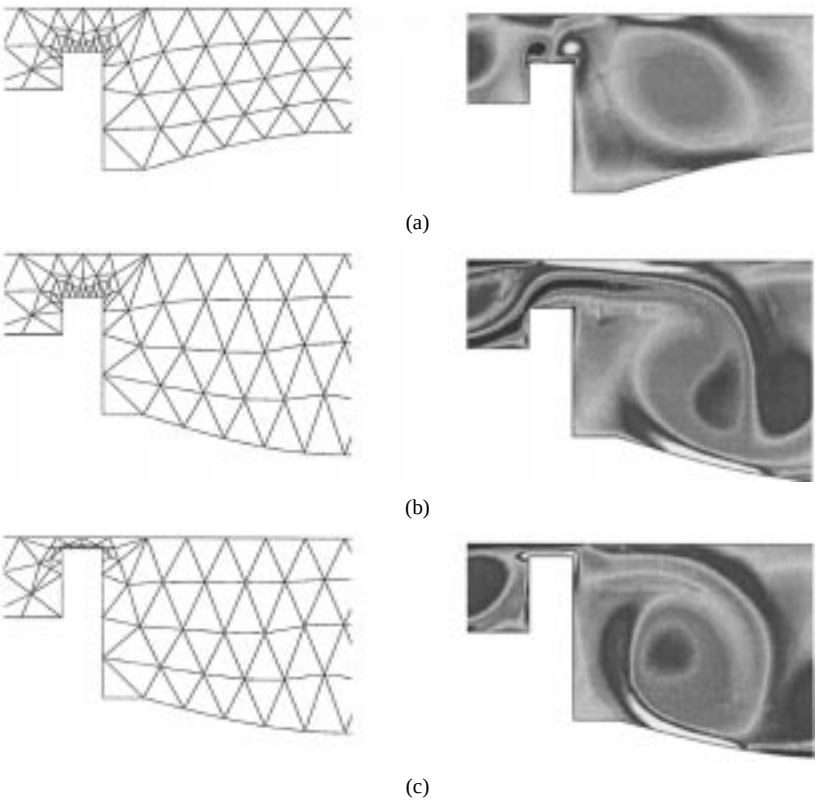

Fig. 6. Close up of the vorticity contours for $Re = 30$ simulation at the left valve (meshes shown on left side). (a) $\tau\omega = 0.28$, corresponds to the beginning of the suction stage. Start-up vortices due to the motion of the inlet valve can be identified. (b) $\tau\omega = 0.72$, corresponding to the end of the suction stage. A vortex jet pair is visible in the pump cavity. (c) $\tau\omega = 84$, corresponding to early ejection stage. Further evolution of the vortex jet and the start-up vortex of the exit valve can be identified. Note that $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$, due to its high-order, allows for very large deformations and thus h-remeshing is avoided unlike other low-order flow solvers. However, h-remeshing is required when elements are squeezed to zero volume. (Courtesy of Beskok and Warburton [26]).

This is the Arbitrary Lagrangian Eulerian (ALE) formulation of the momentum equation. The arbitrary velocity is w_j. This form reduces to the familiar Eulerian and Lagrangian forms by simply setting $w_j = 0$ and $w_j = u_j$, respectively. However, w_j can be chosen arbitrarily to minimize the mesh deformation. We discuss the grid velociy algorithm next.

There are different ways of handling moving meshes but the ALE formulation is the most general one, and it is particularly suitable for MEMS applications that involve multiple moving and stationary domains. The grid velocity is arbitrary in the ALE formulation and therefore great latitude exists in the choice of technique for updating it. Mesh constraints such as smoothness, consistency, and lack of edge crossover, combined with computational constraints such as memory use and efficiency dictate the update algorithm used. In the current work, we address the problem of solving for the mesh velocity in terms of its graph theory equivalent problem. Mesh positions are obtained using methods based on a graph theory analogy to the spring problem. Vertices are treated as *nodes*, while edges are treated as *springs* of varying length and tension. At each time step, the mesh coordinate positions are updated by equilibration of the spring network. Once the new vertex positions are calculated, the mesh velocity is obtained through differences between the original and equilibrated mesh vertex positions. Typical speed ups in MEMS applications show at least a 100-fold improvement over the classical approach involving accurate solutions of the Laplace's equation.

Specifically, we incorporate the idea of variable diffusivity while maintaining computational efficiency by avoiding solving full Laplacian equations. The method we use for updating the mesh velocity is a variation of the barycenter method [28] and relies on graph theory. Given the graph $G = (V, E)$ of element vertices V and connecting edges E, we define a partition $V = V_0 \cup V_1 \cup V_2$ of V such that V_0 contains all vertices affixed to the moving boundary, V_1 contains all vertices on the outer boundary of the computational domain, and V_2 contains all remaining interior vertices. To create the effect of variable diffusivity, we use the *concept of layers*. As is pointed in [29], it is desirable for the vertices very close to the moving boundary to have a grid velocity almost equivalent to that of the boundary. Hence, locally the mesh appears to move with solid movement, whereas far away from the moving boundary the velocity must gradually go to zero. To accomplish this in our formulation, we use the concept of *local tension* within layers to allow us to prescribe the rigidity of our system. Each vertex is assigned to a layer value which heuristically denotes its distance from the moving boundary. Weights are chosen such that vertices closer to the moving boundary have a higher influence on the updated velocity value. To find the updated grid velocity u^{g}, at a vertex $v \in V_2$, we use a force-directed method. Given a configuration as in Fig. 7, the grid velocity at the center vertex is given by

$$u^{\mathrm{g}} = \sum_{i=1}^{\deg(v)} \alpha_i^l u_i, \qquad \sum_{i=1}^{\deg(v)} \alpha_i^l = 1$$

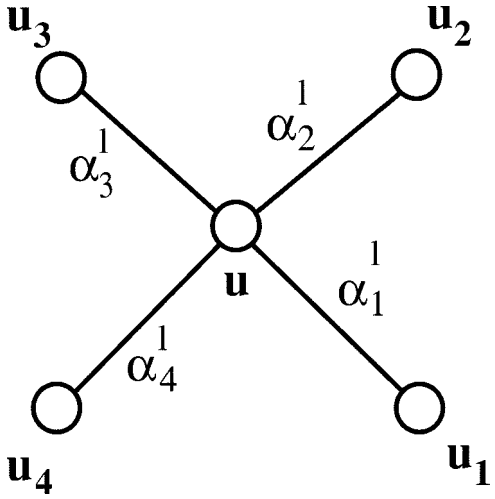

Fig. 7. Graph showing vertices with associated velocities and edges with associated weights.

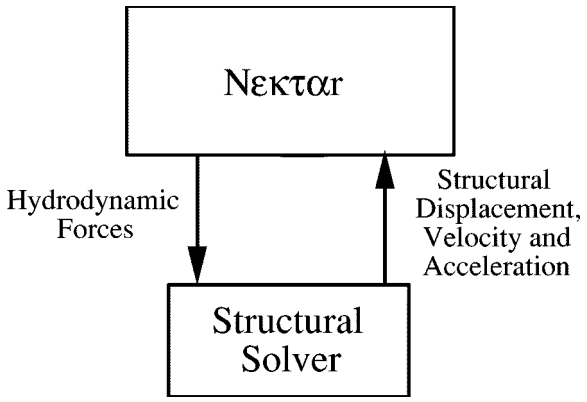

Fig. 8. Coupling between $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ and the structural solver. $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ provides the hydrodynamic force information on the membrane. With this information the structural solver calculates the membrane's response. Structural displacement, velocity and acceleration are then returned to $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ for determining the influence of the structure's motion on the fluid.

where $\deg(v)$ is the number of edges meeting at the vertex v and α_i^l is the lth layer weight associated with the ith edge. This is subjected to the following constraints: $u^{\mathrm{g}} = 0 \quad (\forall\, v \in V_1)$, and $u^{\mathrm{g}} \quad (\forall\, v \in V_0)$ is prescribed to be the wall velocity. This procedure is repeated for a few cycles following an *incomplete* iteration algorithm, over all $v \in V_2$. (Here by incomplete we mean that only a few sweeps are performed and not full convergence is sought.) Once the grid velocity is known at every vertex, the updated vertex positions are determined using explicit time-integration of the newly found grid velocities.

C. The Structural Simulator

The membrane of the micropump is modeled using the linear string-beam equation as given by the following equation:

$$\frac{d^2y}{dt^2} + \frac{R}{m}\frac{dy}{dt} + \frac{EI}{m}\frac{d^4y}{dx^4} - \frac{T}{m}\frac{d^2y}{dx^2} = \frac{F}{m} \tag{6}$$

where

E	Young's modulus of elasticity;
I	second moment of inertia;
T	axial tension;
F	hydrodynamic forcing;
R	coefficient of structural damping;
m	structural mass per unit length.

In this model, the coefficients are given by the physical parameters of the membrane used within the pump, and the hydrodynamic forcing on the membrane is provided by $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$.

Assume that the membrane lies in the interval $[0, L]$. For the micropump configuration, we have chosen the boundary conditions $y(0) = y(L) = 0,\ y''(0) = y''(L) = 0$ which correspond to a fixed-hinged membrane. Equation (6) combined with these boundary conditions lends itself to the use of eigenfunction decomposition for the efficient solution of the membrane motion. We begin by transforming the problem to lie on the interval $[0, 1]$ using the linear mapping $x = L\xi, \xi \in [0, 1]$. The eigenfunctions of this system are given by $\phi_n = 1/2(\sin\sqrt{\lambda_n}\xi);\ \sqrt{\lambda_n} = (n-1)\pi, n = 1, 2, \ldots, \infty$. If we assume a solution of the form $y(\xi, t) = \sum_{n=1}^{N} A_n(t)\phi_n(\xi)$, then by employing the Galerkin method we obtain the following evolution equation for the coefficients $A_n(t)$:

$$\frac{d^2A_n}{dt^2} + \frac{R}{m}\frac{dA_n}{dt} + \left(\frac{EI}{mL^4}\lambda_n - \frac{T}{mL^2}\right)\lambda_n A_n = \frac{1}{m}\int_0^1 F d\xi. \tag{7}$$

We then solve this evolution equation using the Newmark scheme [30], which returns the coefficients for the displacement, velocity and acceleration of the membrane. This information is then returned to $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ as demonstrated in Fig. 8.

D. Differences Between Circuit, Fluid, and Solid Simulators

The above descriptions suggest some differences between the various simulators. The key distinguishing features are as follows.

- The fluid simulator is computationally more expensive than the structure and circuit simulators.
- SPICE3 has a reliable error estimation for time discretization. Therefore, a rollback in time can be done if the truncation error is unacceptable. As a result, SPICE3 automatically controls the simulation timestep to ensure an acceptable user specified error. $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ does not have an automatic timestep control scheme for coupled fluid-structure simulation.
- SPICE3 uses implicit numerical integration methods for time domain simulation. These methods are efficient for circuit simulation, because the circuit equations are stiff. For the fluid solver, however, explicit methods are simpler to implement and reasonably efficient. For this reason, $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ uses semi-implicit methods for the time domain integration (explicit for the advection terms and implicit for the diffusion terms of the Navier–Stokes equations), which suffer from the standard CFL (Courant–Friedrichs–Levy condition for the time step) restrictions. However, the flow time step is much higher than the electronics time step due to the relevant physical time scales.

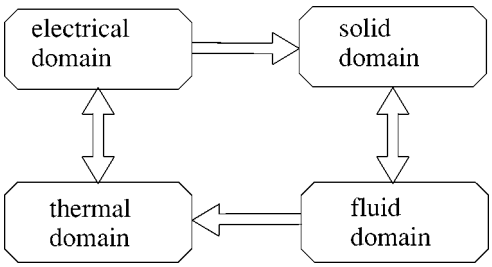

Fig. 9. Coupling between the various physical domains.

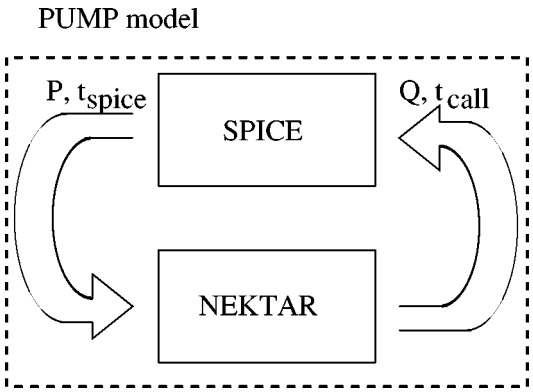

Fig. 10. The SPICE3-$\mathcal{NEKTAR}$ interaction for the pump microsystem of Fig. 2. SPICE3 provides the time $t_{\rm spice}$ and pressure P for the membrane actuation to $\mathcal{NEKTAR}$. $\mathcal{NEKTAR}$ transfers the flowrate Q and the time $t_{\rm call}$ for the next call of $\mathcal{NEKTAR}$ to SPICE3.

IV. Circuit-Microfluidic Device Simulation

For coupled circuit-microfluidic device simulation four different physical domains (electrical, structure mechanical, fluid mechanical, and thermal) must be considered, as shown in Fig. 9. These domains are coupled to one another as described below.

In Fig. 2, four types of coupling can be identified. These are as follows:

- electromechanical coupling for a piezoelectric actuation of the pump membrane;
- fluid-structure coupling due to volume displacement of the pump membrane;
- fluid-thermal coupling because of the thermoresistor cooling in the fluid when an anemometer type of microflow sensor is used;
- electrothermal thermoresistor heating due to current flow in the microflow sensor.

The overall system can be simulated using different approaches. One could use detailed physical simulation for each coupled domain. Another way is the use of lumped element equivalent circuits, compact or macromodels, and/or analog hardware description languages. A third possibility is to use a combination of coupled solvers, compact models and lumped elements. In this work, we will demonstrate this third approach.

A. Software Integration

The interaction of the full system is based on different abstraction levels, using lumped circuit elements, compact/macromodels and a direct interconnection of solvers for various domains. The circuit simulator SPICE3 is chosen as the controlling solver for the following reasons:

- SPICE3 has advanced time-step control;
- models for different abstraction levels can be easily implemented in SPICE3;
- lumped-element equivalent circuits can be readily simulated.

Relatively simple elements are implemented as lumped elements or compact models. These elements are electro-mechanical transducers (piezoelectric actuator) and thermo-resistors. Flow sensors are much more complicated but often the fluid flow around sensors is relatively simple. For example, if the fluid flow in a channel is fully-developed then it has a parabolic profile for the velocity, and thus this profile can be used (compact model) for the flow sensors as well. It is important to note that these compact models are parameterized and can be

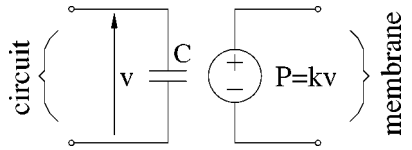

Fig. 11. Lumped model for piezoelectric actuation. The voltage V is transformed into a pressure P that is used to activate the membrane of the pump.

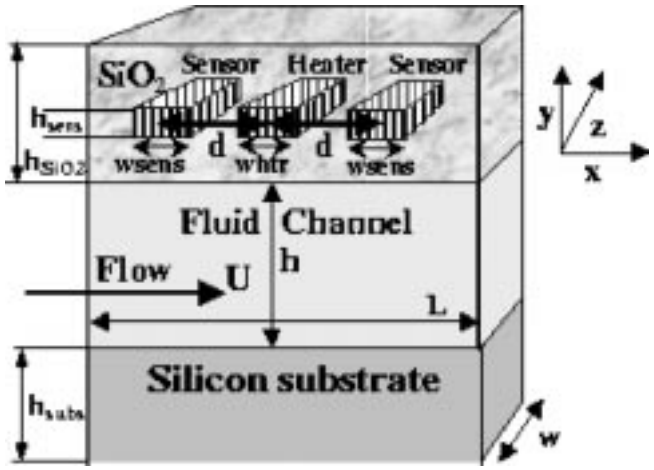

Fig. 12. Structure of an anemometer type flow sensor (thermocouple). This sensor is made up of a heating element and two sensing elements. The temperature difference between the sensors is used to measure the flowrate.

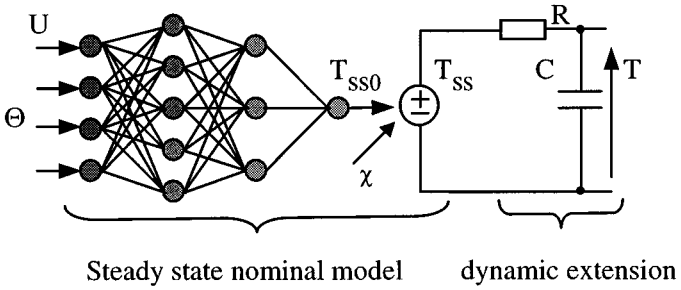

Fig. 13. Dynamic macromodel for the flow sensor. The steady-state solution $T_{\rm SS0}$ corresponds to a nominal power for the heat source χ. The neural network output $T_{\rm SS0}$ is a multivariate function of the flow velocity U and the vector of geometrical and physical parameters Θ. $T_{\rm SS}$ is a linear function of the heat source χ and $T_{\rm SS0}$.

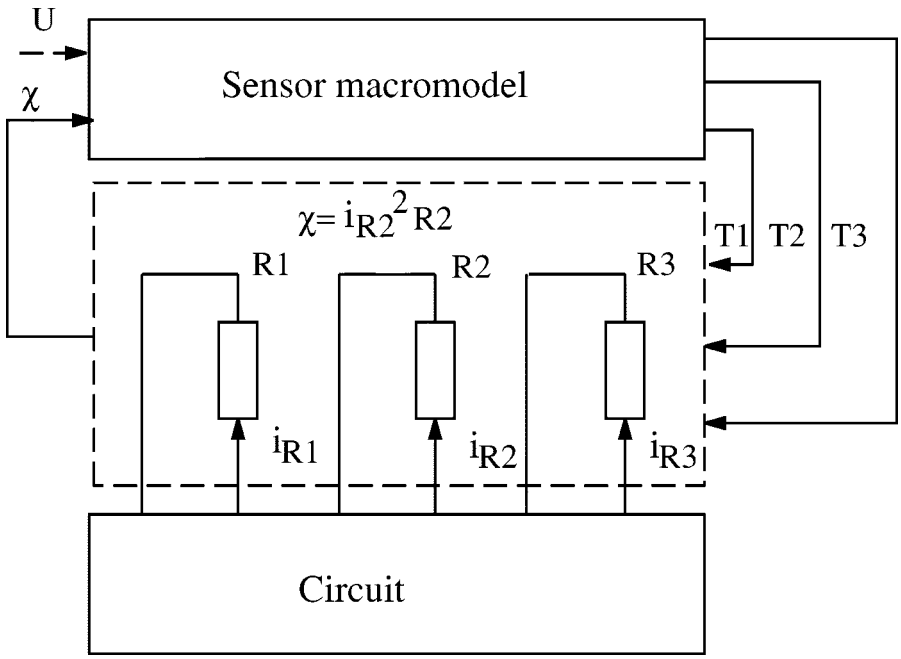

Fig. 14. Macromodel implementation in SPICE3. Based on the fluid flow rate the thermoresistor temperatures, $T1$, $T2$ and $T3$, change which in turn alters the resistance values and the sensing circuit currents and voltages.

highly nonlinear. These models are obtained by insight gained from detailed physical level simulations, such as Navier–Stokes simulations, DSMC, and linearized solutions of the Boltzmann equation [25]. The pump can also be described as a lumped element [31]. However, these lumped element descriptions are applicable only for small variations in the fluid flow. Usually pumps operate in a nonlinear and nonsmooth mode of fluid flow with a strong fluid-structure interaction. Therefore, a detailed physical level simulation of the pump is required. A simplification can be made by employing a macromodel of the form described in (1) but here we employ full Navier–Stokes simulations with full dynamics.

For this reason, the following options are used. Electro-mechanical actuators, thermo-resistors and flow sensors are described as lumped elements and/or compact models. The pump is modeled at the detailed physical level. All lumped elements and models are implemented in SPICE3. The pump is implemented as a direct SPICE3-$\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ interconnection (Fig. 10). SPICE3 transfers the time t_{spice} and pressure P for the membrane activation to $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ and receives the flowrate Q and the time t_{call} for the next call to $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$. A detailed description of this coupling is provided later.

B. Lumped Element and Compact Models for Devices

1) Model for Piezoelectric Transducers: The model for electro-mechanical coupling with a piezoelectric actuation of the membrane is shown in Fig. 11. This model forms the interface between the electrical and mechanical networks. The electrical characteristics of the piezoelectric actuator are described by the capacitor C. The input voltage V translates into an output pressure P by virtue of the piezoelectric effect with coefficient k. This pressure is an input argument to $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$. The mechanical characteristics of the piezoelectric actuator are coupled with the mechanical characteristics of the substrate [32], [33].

2) Compact Model for Flow Sensor: For an anemometer type flow sensor [34] shown in Fig. 12, a macromodel has been developed in [35]. This macromodel (Fig. 13) is based on neural networks trained using data from detailed physical simulations.

The inputs to the neural network are the flow velocity U and the vector of geometrical and physical parameters Θ. The results from this model are in good agreement with the simulated data for a large range of parameters [35].

The dynamic macromodel is incorporated in SPICE3 by coupling it with a sensor circuit and a model for thermoresistors for the heater and sensors as shown in Fig. 14. Based on the fluid flow rate the thermo-resistor temperatures, $T1$, $T2$ and $T3$, change, which in turn alters the resistance values and the sensing circuit currents and voltages.

C. Effective Time-Stepping Algorithms

In general, the flow solver $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ can be implemented as one big model in SPICE3. This is accomplished by calling $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ from SPICE3 for each Newton iteration. However, such a coupling is extremely inefficient, because a call to $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ is computationally very expensive. Furthermore, the time scales and nonlinearities are extremely different for the circuit and fluidic devices. If one considers only the circuit element, then a SPICE3 simulation results in nonuniform time steps and several Newton iterations for each time step. Typical time constants for circuits are of the order of $10^{-12}\ldots10^{-6}$ s. On the other hand, fluidic devices have a typical time constant of the order $10^{-4}\ldots10^{-1}$ s.

This property can be exploited to improve simulation performance by calling $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ only at some of the circuit time points following a *subcycling type algorithm*. Between these

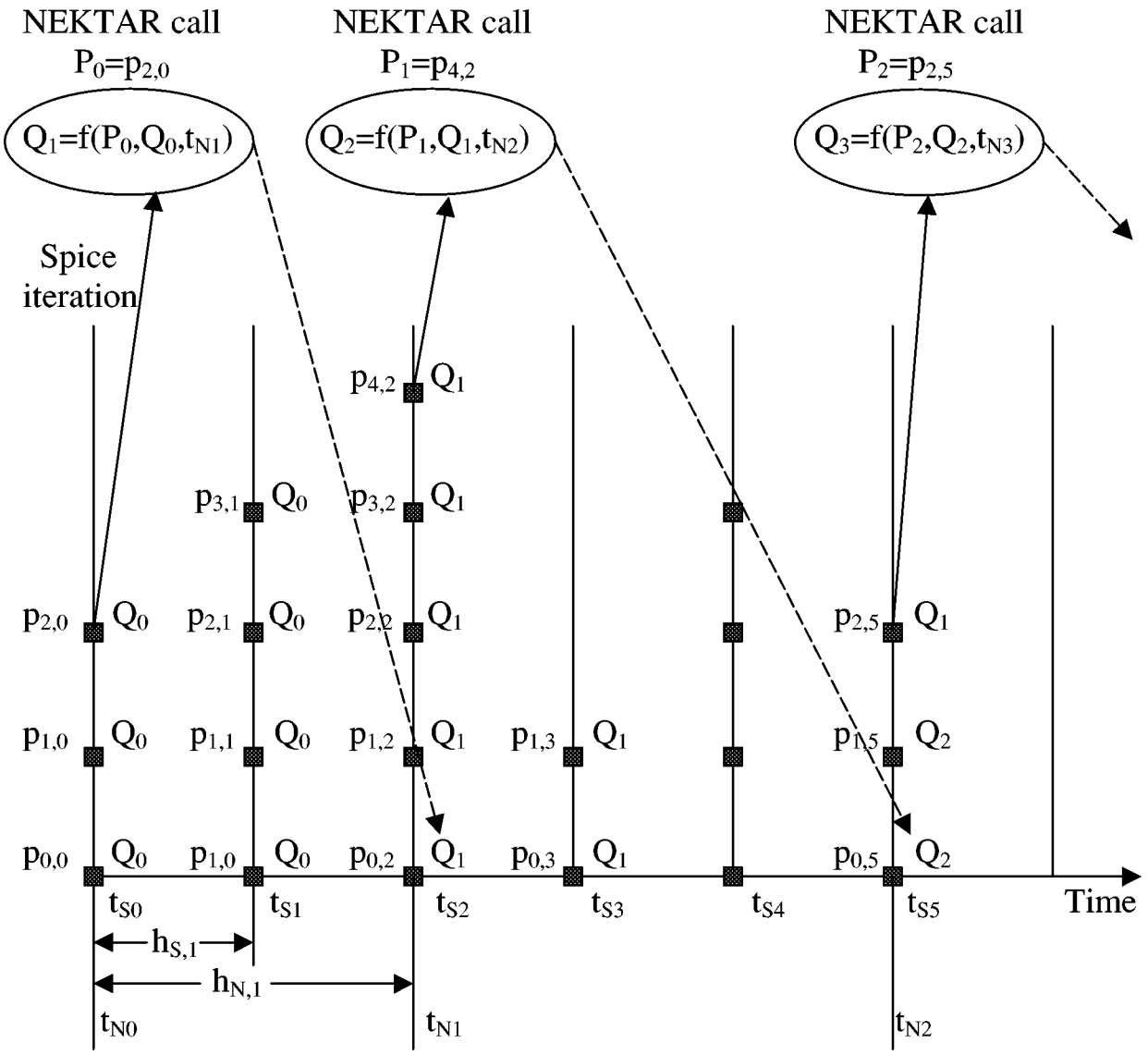

Fig. 15. The time stepping scheme for SPICE3-$\mathcal{NEKTAR}$ coupling. $t_{S,k}$ and $t_{N,k}$ are the SPICE3 and $\mathcal{NEKTAR}$ time points, respectively. Q_i is a constant value for each SPICE3 iteration and at each SPICE3 time point between the $\mathcal{NEKTAR}$ time points $t_{N,i}$ and $t_{N,i+1}$. The membrane pressure $p_{j,k}$ is calculated as a function of the circuit behavior for each SPICE3 call at time $t_{S,k}$ and iteration j. SPICE3 selects time points based on a local truncation error estimate and synchronizes with $\mathcal{NEKTAR}$ at all $\mathcal{NEKTAR}$ time points. The pressure P_i for the final SPICE3 iteration at the synchronization time point $t_{S,k} = t_{N,i}$ is used as an input to $\mathcal{NEKTAR}$. A $\mathcal{NEKTAR}$ call is made at $t_{N,i}$ and a new value of Q is computed for the next $\mathcal{NEKTAR}$ time point.

time points, the $\mathcal{NEKTAR}$ outputs can be modeled as constant values. Further improvement in performance is possible by taking into account the usage of semi-explicit methods for fluid simulation. In this case, the flowrate Q_n for time point t_n is calculated by the explicit scheme: $Q_n = F(\mathbf{P}_{n-1}, \mathbf{V}_{n-1}, t_n)$, where $\mathbf{P}$ is the vector of the pressure at mesh points, and $\mathbf{V}$ is the vector of velocities at mesh points. For the SPICE3-$\mathcal{NEKTAR}$ interaction described earlier, the important quantities are the distributed pressure P for the pump membrane and the flowrate Q_n. This functional relationship can be expressed as follows: $Q_n = f(P_{n-1}, Q_{n-1}, t_n)$. Based on this observation an efficient time stepping scheme is obtained as shown in Fig. 15.

In Fig. 15, time is plotted on the horizontal axis and the SPICE3 iterations are plotted on the vertical axis. $t_{S,k}$ and $t_{N,k}$ are the SPICE3 and $\mathcal{NEKTAR}$ time points, respectively. $\mathcal{NEKTAR}$ chooses a time step $h_{N,i} = t_{N,i} - t_{N,i-1}$ independent of SPICE3, based on the Courant number (CFL) constraint for convection. The $\mathcal{NEKTAR}$ time points $t_{N,i}$ are used as synchronization time points with SPICE3, whereby $t_{N,i} = t_{S,k}$. The flowrate Q has a constant value between these synchronization time points. The membrane pressure $p_{j,k}$ is calculated as a function of the circuit behavior for each SPICE3 call at time $t_{S,k}$ and iteration j. The pressure $P_i = p_{M,k}$ at the final SPICE3 iteration M, for a synchronization time point $t_{S,k} = t_{N,i}$, is an input to $\mathcal{NEKTAR}$. A $\mathcal{NEKTAR}$ call is made at $t_{N,i}$ and a new value of Q is computed using the relation $Q_{i+1} = f(P_i, Q_i, t_{N,i+1})$. This value is then used for the next $\mathcal{NEKTAR}$ time point, $t_{N,i+1}$.

The main features of this time stepping scheme can be summarized as follows:

- $\mathcal{NEKTAR}$ is called from SPICE3;
- the timestep for SPICE3 $\ll$ timestep for $\mathcal{NEKTAR}$;
- $\mathcal{NEKTAR}$ specifies the next synchronization time point.

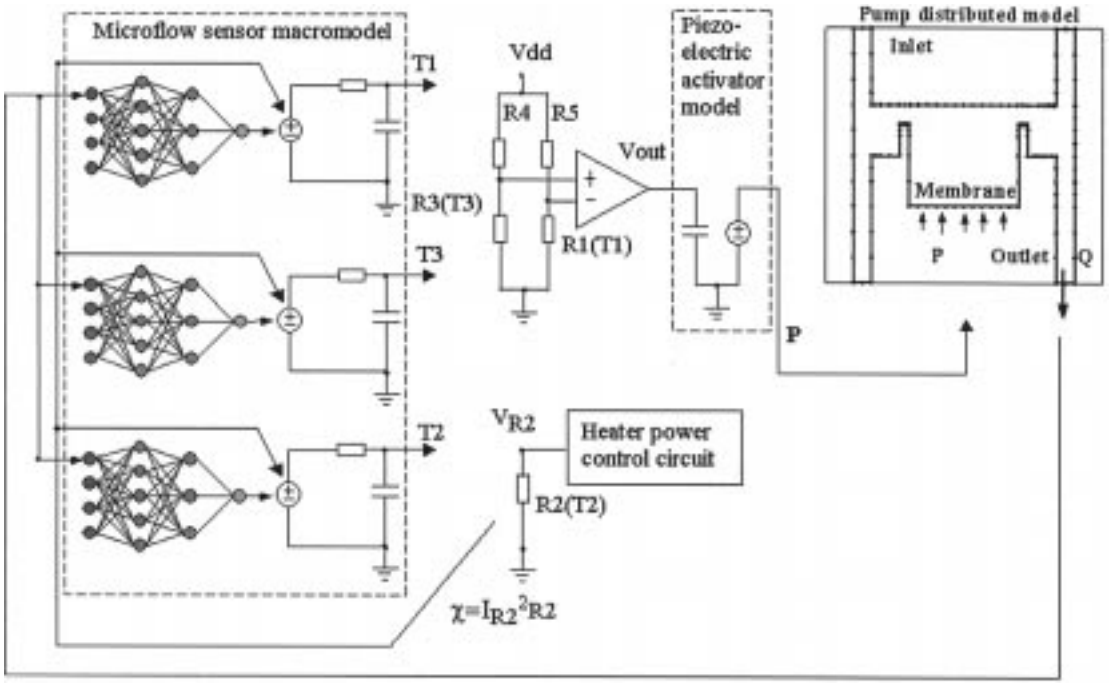

Fig. 16. Description of the complete system for simulation. The pump flow rate Q determines the flow sensor velocity U. This yields the temperatures for the sensor thermoresistors. The difference between the resistance values $R1(T1)$ and $R3(T3)$ is transformed into the voltage $V_{\rm out}$ by the control electronics, which is used to control the pressure P for the pump membrane. This, in turn, determines the flow rate Q.

From this, it can be concluded that the number of $\mathcal{NEKTAR}$ calls are the same as that of stand-alone $\mathcal{NEKTAR}$. This is the best possible situation in terms of efficiency for the coupled SPICE3-$\mathcal{NEKTAR}$ simulation.

V. Demonstrations of the Integrated Simulation Approach

A. Microfluidic System Description

A microliquid dosing system is used as an illustrative example. This system is made up of a micropump, a flow sensor and an electronic control circuit. The electronic circuit adjusts the pump flow rate. A simplified simulation circuit is shown in Fig. 16.

In this system, the flow rate Q determines the flow sensor velocity U for a given set of geometry parameters (h, d, wsens). Based on the fluid flow rate the thermoresistor temperatures, $T1$, $T2$, and $T3$ change, which, in turn, alters the resistance values $R1(T1)$, $R2(T2)$, and $R3(T3)$. The resistance $R1(T1)$ and $R3(T3)$ are included in a Wheatstone-bridge arrangement with two fixed resistors $R4$ and $R5$. The voltage difference $V_{R3(T3)} - V_{R1(T1)}$ is directly proportional to the temperature difference $T3 - T1$. This voltage difference is linearly transformed to the output voltage $V_{\rm out}$ by an operational amplifier with a controlled gain. This output voltage determines the pressure$P$, which activates the pump membrane and changes the flow rate Q. The thermoresistor of the heater $(R2)$ is activated by the control electronics that maintains a constant heater temperature.

B. SPICE3-$\mathcal{NEKTAR}$ Integration

As mentioned earlier $\mathcal{NEKTAR}$ is embedded as a subroutine in SPICE3. The interaction with SPICE3 is by means of the model code and the simulation engine. Synchronization time points are determined by $\mathcal{NEKTAR}$ and used by the SPICE3 transient analysis engine. The pump is modeled as a SPICE3 element with $\mathcal{NEKTAR}$ being the underlying simulation engine. The other elements in the circuit are described by lumped element descriptions and/or compact models.

C. Simulation Results

The simulation results from the coupled simulator are presented in Fig. 17. In this simulation, one can determine the pressure on the pump membrane, the flow velocity, and the output control voltage as a function of time for various component parameters. As an example, consider the microflow sensor the characteristics for which are shown in Fig. 18. It is seen that for the given range of flow velocity the temperature difference between the upstream and downstream sensor temperatures is in the range 12–17 K. Design variations can be simulated for different geometry parameters. For example, with a different set of microflow sensor parameters the temperature difference characteristic changes are small as shown in Fig. 19. More specifically, the differences between the two plots are due to the different flow sensor geometry and characteristics. For the results in Fig. 18 the flow channel height is 100 μm and the channel width is 800 μm. This geometry determines the lower velocity of the flow in the flow sensor. For the results in Fig. 19 the flow channel height is 50 μm and the channel width is 200 μm. Therefore, for the same flow rate from a micropump, the velocity is 8 times greater.

With regards to computational cost, this simulation required approximately 5 min of CPU time on a 300-MHz Pentium II processor. Thus, the coupled simulator is reasonably efficient and provides valuable information to the system or device developer.

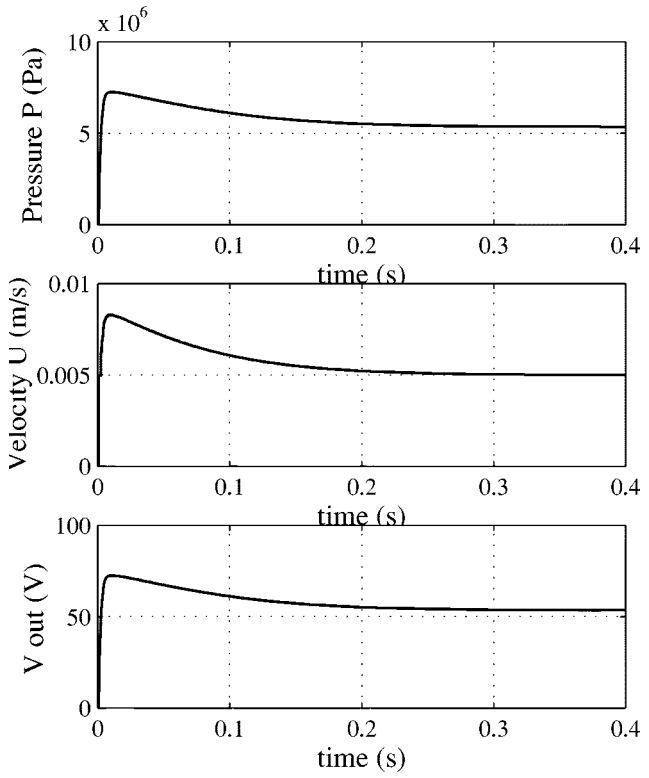

Fig. 17. The external pressure for the pump membrane, the inlet velocity for the microflow sensor, and the amplifier output voltage for the simulation of the microfluidic system as a function of time.

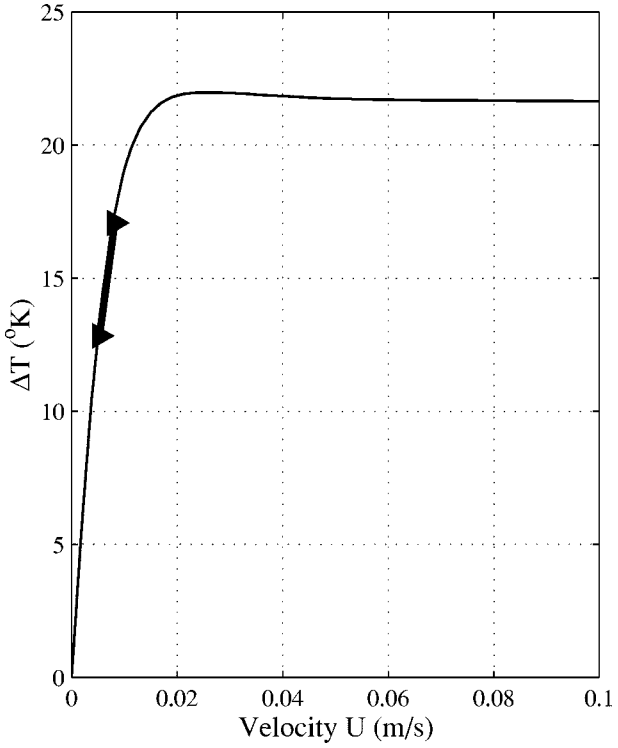

Fig. 18. Flow sensor characteristics and its region of operation. A small change in velocity results in a large change in ΔT, the difference of the upstream and downstream sensor temperatures.

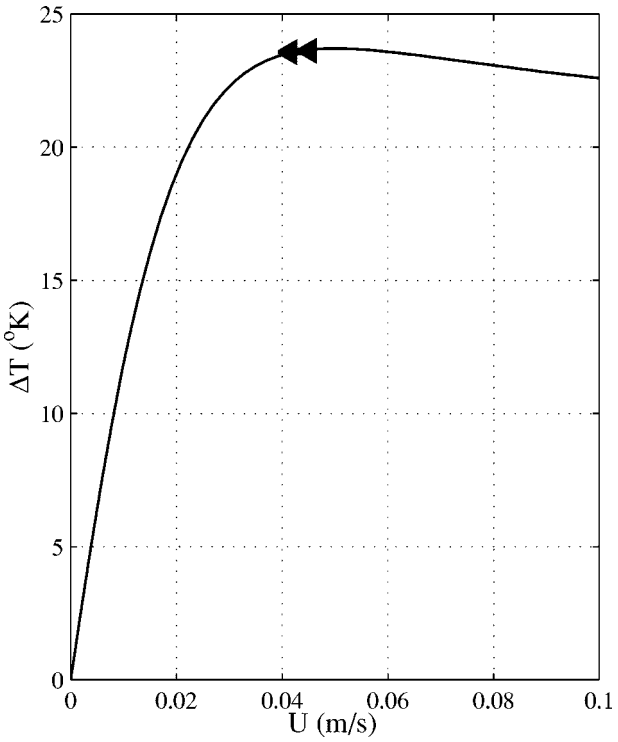

Fig. 19. Design variation in flow sensor. A small change in velocity results in a very small change in ΔT, the difference of the upstream and downstream sensor temperatures, unlike the previous case.

VI. Conclusion

Coupled-domain simulation is necessary in MEMS applications as many different physical phenomena are present, and different processes are taking place simultaneously. Depending on the specific application, e.g., a microsensor versus a microactuator or a more complex system, some aspects of the device need to be simulated in detail at high resolution while others need to be accounted for by a low-dimensional description. Nonlinear macromodels are a possibility, but this is insufficient for the microfluidic system which is typically highly unsteady and nonlinear. In addition, in the microdomain certain nonstandard flow features have to be modeled accurately such as velocity-slip or temperature-jump in gas flows, electro-kinetic effects in liquid flows, and particle trajectories in particulate flows. To this end, we have developed the code $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ that can simulate flows in the micro- and macrodomains both for liquids as well as gases. In addition, it includes a library of linear and nonlinear structures, such as beams, membranes, cables, etc.

For the coupled-domain simulation, the main driver program is SPICE3, a popular code for circuit simulation. In this paper, a coupled circuit and microfluidic device simulator was presented. The resulted simulator allows simulation of a complete microfluidic system in which thermal, flow, structural, and electrical domains are integrated. The coupling of these simulators was described and demonstrated for a microliquid dosing system. The integrated simulator can be utilized for parametric studies and optimal design of microfluidic systems.

The integration of different simulators required for complete MEMS simulations is a difficult problem with challenges well beyond software integration. It involves disparate temporal

and spatial scales leading to great stiffness and inefficiencies, new physical assumptions and approximations for some of the components, issues of numerical stability, staggered time-marching procedures, new fast solvers for coupled problems, and optimization and control algorithms. Most of the mature algorithms from one discipline are inefficient in this context so new methods are required in order to produce a new generation of simulation algorithms for MEMS devices. In this paper, we have demonstrated that this is possible by coupling two accurate codes and resolving at least at some level some of these coupling issues. However, significant improvements can be made for specific devices. For example, for the membrane-driven micropump presented here, convergence of the coupling algorithm could be accelerated by inspecting the time-dependent mass-conservation equation every SPICE time step and obtain a new estimate of flow rate

$$Q_{\text{new}} = Q_{\text{old}} + \frac{\Delta V}{\Delta t}$$

where ΔV is the change in volume due to the change in the membrane position, and Δt is the time between two consecutive SPICE calls. This requires solving for the structure only but not necessarily for the entire flow field, which is the most computationally intensive part. The structure solver is very fast and can be called as often as it is necessary without a serious computational overhead. Similar ideas can be explored for specific problems to exploit known conservation laws or other principles applied over a control volume.

ACKNOWLEDGMENT

The authors would like to thank Prof. A. Beskok for many useful suggestions regarding this work.

REFERENCES

[1] S. Senturia, N. Aluru, and J. White, "Simulating the behavior of MEMS devices: Computational methods and needs," *IEEE Computational Sci. Eng.*, vol. Jan.–Mar., pp. 30–43, 1997.

[2] L. W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," Electronics Research Lab., Univ. of California, Berkeley, UCB/ERL M520, 1975.

[3] T. L. Quarles, "The SPICE3 implementation guide," Electronics Research Lab., Univ. of California, Berkeley, UCB/ERL M89/44, 1989.

[4] H. A. C. Tilmans, "Equivalent circuit representation of electromechanical transducers: I. Lumped-parameter systems," *J. Micromech. Microeng.*, vol. 6, pp. 157–176, 1996.

[5] J. Bielefeld, G. Pelz, and G. Zimmer, "AHDL-model of a 2D mechanical finite-element usable for microelectro-mechanical systems," in *BMAS'97*, 1997, pp. 177–181.

[6] A. Schroth, T. Blochwitz, and G. Gerlach, "Simulation of a complex sensor system using coupled simulation programs," in *Proc. Transducers'95*, 1995, pp. 33–35.

[7] S. Moaveni, *Finite Element Analysis: Theory and Application with ANSYS*. Upper Saddle River, NJ: Prentice Hall, 1999.

[8] J. Keown, *Microsim Pspice and Circuit Analysis*, 3 ed. Upper Saddle River, NJ: Prentice Hall, 1997.

[9] K. Mayaram and D. O. Pederson, "Coupling algorithms for mixed-level circuit and device simulation," *IEEE Trans. Computer-Aided Design*, vol. 11, pp. 1003–1012, Aug. 1992.

[10] K. Mayaram, J. Chern, and P. Yang, "Algorithms for transient three-dimensional mixed-level circuit and device simulation," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 1726–1733, Nov. 1993.

[11] W. L. Engl, R. Laur, and H. K. Dirks, "MEDUSA-A simulator for modular circuits," *IEEE Trans. Computer-Aided Design*, vol. 1, pp. 85–93, Apr. 1982.

[12] *MEDICI User's Manual: Circuit Analysis*, Technology Modeling Associates, 1997.

[13] *ATLAS User's Manual: MixedMode*, Silvaco International, 1995.

[14] F. M. Rotella, B. Troyanovsky, Z. Yu, R. Dutton, and G. Ma, "Harmonic balance device analysis of an LDMOS RF power amplifier with parasitics and matching network," in *SISPAD-97*, 1997, pp. 157–159.

[15] R. L. Woodruff and P. J. Rudeck, "Three-dimensional numerical simulation of single event upset of an SRAM cell," *IEEE Trans. Nucl. Sci.*, vol. 40, pp. 1795–1803, June 1993.

[16] E. Ravanelli and C. Hu, "Device-circuit mixed simulation of VDMOS charge transients," *Solid State Electron.*, vol. 34, pp. 1353–1360, Dec. 1991.

[17] H. J. Park, P. K. Ko, and C. Hu, "A charge conserving nonquasi-static (NQS) model for SPICE transient analysis," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 629–642, May 1991.

[18] G. E. Karniadakis and S. J. Sherwin, *Spectral/hp Element Methods for CFD*: Oxford University Press, 1999.

[19] R. M. Kirby, T. C. Warburton, S. J. Sherwin, A. Beskok, and G. E. Karniadakis, "The $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ code: Dynamic simulations without remeshing," in *Proc. 2nd International Conference on Computational Technologies for Fluid/Thermal/Chemical Systems with Industrial Applications*, 1999.

[20] A. L. Sangiovanni-Vincentelli, "Circuit simulation," in *Computer Design Aids for VLSI Circuits*: Sijthoff and Noordhoff, 1981, pp. 19–113.

[21] K. S. Kundert, "Sparse-matrix techniques and their application to circuit simulation," in *Circuit Analysis, Simulation and Design*. Amsterdam, The Netherlands: North-Holland, 1990.

[22] S. J. Sherwin, "Triangular and Tetrahedral spectral/hp finite element methods for fluid dynamics," Ph.D. dissertation, Princeton University, Princeton, NJ, 1995.

[23] T. C. Warburton, "Spectral/hp Methods on Polymorphic Multi-Domains: Algorithms and Applications," Ph.D. dissertation, Brown University, Division of Applied Mathematics, 1999.

[24] A. Beskok, G. E. Karniadakis, and W. Trimmer, "Rarefaction and compressibility effects in gas microflows," *J. Fluids Eng.*, vol. 118, p. 448, 1996.

[25] A. Beskok and G. E. Karniadakis, "A model for flows in channels, pipes and ducts at micro- and nano-scales," *J. Microscale Thermophys. Eng.*, vol. 3, pp. 43–77, 1999.

[26] A. Beskok and T. C. Warburton, "Micro-fluidic design and fluid-structure interaction analysis of a micro-pump," in *Proc. ASME IMECE meeting*, Nov. 15–20, 1998, pp. 77–84.

[27] I. Lomtev, R. Kirby, and G. Karniadakis, "A discontinuous Galerkin ALE method for compressible viscous flows in moving domains," *J. Comp. Phys.*, vol. 155, pp. 128–159, 1999.

[28] G. D. Battista, P. Eades, R. Tamassia, and I. Tollis, *Graph Drawing*. Upper Saddle River, NJ: Prentice Hall, 1998.

[29] R. Lohner and C. Yang, "Improved ALE Mesh Velocities for Moving Bodies," *Comm. Num. Meth. Eng. Phys.*, vol. 12, pp. 599–608, 1996.

[30] T. J. R. Hughes, *The Finite Element Method*. Englewood Cliffs, NJ: Prentice-Hall, Inc, 1987.

[31] A. Klein, S. Matsumoto, and G. Gerlach, "Modeling and design optimization of a novel micropump," in *Proc. First International Conference on Modeling and Simulation of Microsystems, Semiconductors, Sensors and Actuators*, Santa Clara, CA, Apr. 6–8, 1998.

[32] G. G. A. Klein, "Modeling of piezoelectric bimorph structures using an analog hardware description language," in *Proc. Second International Conference on the Simulation and Design of Microsystems and Microstructures (MICROSIM'97), Lausanne (Switzerland)*, Sept. 17–19, 1997.

[33] S. P. Timoshenko and S. Woinowsky-Krieger, *Theory of plates and shells*, 2nd ed. New York: McGraw-Hill, 1970.

[34] A. Rasmussen and M. E. Zaghloul, "The design and fabrication of microfluidic flow sensors," in *Proc. ISCAS-99*, 1999, pp. 136–139.

[35] O. Mikulchenko, A. Rasmussen, and K. Mayaram, "A neural network based macromodel of microflow sensors," in *Proc. MSM2000*, 2000.

Robert M. Kirby received the B.Sc. degree in applied mathematics and computer and information sciences from The Florida State University, Tallahassee, in 1997, the Sc.M. degree in applied mathematics from Brown University, Providence, RI, in 1999, and is currently working on the Sc.M. degree in computer science and the Ph.D. degree in applied mathematics at Brown University.

His research interests are in software design, parallel computing, and direct numerical simulation of flow-structure interactions.

George Em Karniadakis received the M.Sc. and Ph.D. degrees from Massachusetts Institute of Technology (MIT), Cambridge, in 1984 and 1987, respectively, both in mechanical engineering.

He completed his postdoctoral studies at Stanford University, Stanford, CA, and he previously taught at Princeton University, Princeton, NJ. He is currently Professor of Applied Mathematics at Brown University, Providence, RI. He has pioneered spectral methods on unstructured grids, parallel simulations of turbulence in complex geometries, and microfluidics simulations.

Oleg Mikulchenko (A'97) received the Diploma of Engineer and Ph.D. degrees in electrical engineering from National Technical University of Ukraine in 1985 and 1993, respectively.

He was a Researcher, Lecturer, and since 1998, Associate Professor at National Technical University of Ukraine. He joined Washington State University, Pullman, in 1999 and Oregon State University, Corvallis, in 2000.

His research interests include circuit simulation, behavioral modeling, optimization techniques, and artificial neural network development.

Kartikeya Mayaram (S'82–M'88–SM'99) received the M.Sc. degree in electrical engineering from the State University of New York, Stony Brook, in 1982 and the Ph.D. degree in electrical engineering from the University of California, Berkeley, in 1988.

After working in industry at Texas Instruments and Bell Labs, he joined the faculty of Washington State University, Pullman, from 1996 to 1999. Since January 2000, he has been an Associate Professor in the ECE Department of Oregon State University, Corvallis. His research interests are in the areas of circuit simulation, device simulation and modeling, MEMS, integrated simulation environments, and analog/RF design.