2. Simulation and Optimization of MEMS

- 2.1 Faster algorithms for exterior problems
- 2.2 Coupled-domain simulation
- 2.3 Nonlinear macromodeling
The performance of most micromachined devices depends critically on the interaction between forces generated by a variety of mechanisms.

For example, in microresonator-based gyroscopes, such as in Fig. 1 and as described elsewhere, the dynamic performance is due to the coupling between electrostatic, mechanical, and fluidic forces.
Simulation of these coupled-domain problems can be accomplished using generic finite-element techniques.

The efficiency of coupled-domain simulation can be substantially improved by using domain-specific solvers, provided the coupling between domains can be handled effectively.
Three basic approaches to coupled-domain simulation, and computational comparisons between relaxation, multilevel Newton, and full-Newton methods for 3D electromechanical analysis:

- Domain solver
- Solution alternative
- An electromechanical example
Consider the problem of coupled simulation between $m$ domain, where in each domain a discretization with $n_i$ degrees of freedom has been introduced.

Let $x_i \in \mathbb{R}^{n_i}$ denote the vector of unknowns associated with domain $I$, where $x_i$ is computed by solving a possibly nonlinear residual equation

$$R_i(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_m) = 0. \quad (11)$$
Here $x_j, j \neq i$, are the unknowns associated with other domains, and usually treated as inputs when solving the $R_i$ residual equation for $x_i$.

Most domain-specific programs solve their associated residual equations using variants of Newton's method, as in

$$
\frac{\partial R_i}{\partial x_i}(x_1^{k+1}, \ldots, x_i^{k})
= -R_i(x_1, \ldots, x_{i-1}, x_i^k, x_{i+1}, \ldots, x_m), \quad (12)
$$

where $k$ is the Newton iteration index and
\[
\frac{\partial R_i}{\partial x_i} \in \mathbb{R}^{n_i \times n_i}
\]

- is the Jacobian with respect to \( x_i \).
- Of course, if the residual equation for domain \( i \) is linear, only one Newton iteration is required.
One approach to solving the coupled system is to use the simple nonlinear Gauss-Seidel relaxation scheme diagrammed in Fig. 5.

\[ R_1(x_1, \ldots, x_m) = 0 \]
\[ R_2(x_1, \ldots, x_m) = 0 \]
\[ \vdots \]
\[ R_m(x_1, \ldots, x_m) = 0 \quad (13) \]
The relaxation algorithm does not always converge, particularly when the different domains are tightly coupled.

Sufficient conditions for nonlinear relaxation convergence have been extensively analyzed.
A more robust approach than nonlinear relaxation is to use a full-Newton method, as diagrammed in Fig.6, possibly combined with a continuation or a homotopy scheme.
The difficulty with the full-Newton approach is that the off-diagonal, or coupling, derivatives $\partial R_i/\partial x_j, i \neq j$, may not be available explicitly.

If an iterative method like GMRES is used to solve the linear system, then only matrix-vector products are required.

Therefore, the off-diagonal derivatives need not be explicitly computed, and can be approximated by finite differences.
For example, if $\Delta u_j$ is a part of the vector generated by GMRES, then

$$
\frac{\partial R_i}{\partial x_j} \Delta u_j \approx R_i(x_j + \Delta u_j) - R_i(x_j),
$$

where we have assumed that the $\Delta u_j$ vector has been scaled to be small in magnitude.

If Krylov-subspace methods like GMRES are used to solve the system in Fig. 6, then the diagonal blocks should be explicitly factored and used as a preconditioner.
Programs that perform domain-specific analysis are not usually organized so that one can efficiently determine the residual equation difference required in Expression 14.

In addition, the explicitly computed parts of $\frac{\partial R_i}{\partial x_i}$, which are needed for preconditioning the full-Newton iteration equations are often unavailable.

So, to create a robust full-Newton method, the individual solvers must be modified somewhat, and this means the coupled method is not really a “black box” approach in which domain-specific solvers can be easily interchanged.
Consider instead that a program which solves a domain-specific residual equation, as Eq. 11, can be thought of as producing \( x_i \) given \( x_j, j \neq i \).

We denote the input/output description as

\[
x_i = H_i(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_m),
\]

With Eq. 15 in mind, consider a third approach, the multilevel Newton method given in Fig. 7, which can be used to determine the solution to the coupled system.
Figure 7. A multilevel Newton scheme for coupled-domain simulation.
We refer to this as a multilevel Newton method because application of the $H_i$ operator implies solving residual equation $R_i$, and this is typically done with an inner Newton’s method.

Note that in the multilevel Newton method, the Jacobian block diagonals are already identity matrices and need not be preconditioned. Also, application of

$$
\begin{pmatrix}
\frac{\partial H_i}{\partial x_i} & \frac{\partial H_i}{\partial x_j}
\end{pmatrix}^{-1}
$$

can be performed using finite differences and multiple calls to the domain-specific solver without modification.
An electromechanical example

- We use a MEMS comb drive as an example.
- The drive consists of a deformable comb structure, a drive structure, and a ground plane.
- As shown in Fig. 8, the F-shaped finger structure is the comb, the E-shaped finger structure is the drive, and the rectangular-shaped structure is the ground plane.
When a positive potential is applied on the drive structure, and zero potential on the comb and the ground plane, the comb structure deforms out of plane.

Figure 8. Comb drive example.
The deformation of the comb structure for an applied bias of 85 volts is shown in Fig. 9.

Note that only the comb structure deforms and the drive and the ground plane do not move.
This coupled electromechanical problem can be simulated by self-consistently solving for the electrostatic surface charges given the structure deformation, and solving for the elastostatic deformation given by the electrostatic pressure.

For this example, the exterior electrostatic analysis was performed using the accelerated boundary-element method discussed earlier, and the interior elastostatic analysis was performed with a standard parabolic-brick finite-element method.

The comb was discretized into 172 parabolic elements, the drive into 144 linear bricks, and the ground plane into 2,688 fournode elements.
A comparison of the relaxation, multilevel Newton, and full-Newton algorithms for the comb example is summarized in Table 1.

<table>
<thead>
<tr>
<th>Voltage applied</th>
<th>Number of iterations</th>
<th>CPU time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Relaxation</td>
<td>Multilevel Newton</td>
</tr>
<tr>
<td>25</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>50</td>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>75</td>
<td>70</td>
<td>4</td>
</tr>
<tr>
<td>80</td>
<td>142</td>
<td>3</td>
</tr>
<tr>
<td>85</td>
<td>*</td>
<td>3</td>
</tr>
</tbody>
</table>

At low voltages the deflection of the comb is small, the coupling between the electrical and mechanical system is weak, and the relaxation algorithms works well.
For high voltages, the Newton algorithms converge much faster than the relaxation algorithm.

The convergence of the relaxation, multilevel Newton, and full-Newton algorithms at 80 V bias is shown in Fig. 10.

For an application of 85 V on the drive, the relaxation algorithm fails to converge, while the multilevel Newton and full-Newton algorithms converge very rapidly and take 3 and 10 iterations, respectively. This is illustrated in Fig. 11.
Figure 10. Comparison of convergence of relaxation, multilevel Newton, and full-Newton (shown as “coupled”) algorithms for the MEMS comb example at an applied bias of 80 V.

Figure 11. Convergence of the same three algorithms on the comb example at an applied bias of 85 V. The relaxation algorithm does not converge; the others converge quickly.
Fast domain-specific solver and efficient approaches for coupled-domain simulation can make it possible to perform realistic analysis of single micromachined devices in a few hours on a scientific workstation.

However, such simulation times are still far too long to be used for design exploration or as part of a system-level simulation.

Instead, these tools can be used to help generate macromodels.