SUGAR: A MEMS Simulation Program

David Bindel
dbindel@eecs.berkeley.edu

UC Berkeley, CS Division
# SUGAR contributors

<table>
<thead>
<tr>
<th>Faculty</th>
<th>Grad students</th>
<th>Undergrads</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Agogino (ME)</td>
<td>D. Bindel (CS)</td>
<td>W. Kao (CS)</td>
</tr>
<tr>
<td>Z. Bai (Math/CS)</td>
<td>J.V. Clark (AS&amp;T)</td>
<td>A. Kuo (EE)</td>
</tr>
<tr>
<td>J. Demmel (Math/CS)</td>
<td>D. Garmire (CS)</td>
<td>E. Zhu (CS)</td>
</tr>
<tr>
<td>S. Govindjee (CEE)</td>
<td>B. Jamshidi (CEE)</td>
<td></td>
</tr>
<tr>
<td>M. Gu (Math)</td>
<td>R. Kamalian (ME)</td>
<td></td>
</tr>
<tr>
<td>K.S.J. Pister (EE)</td>
<td>S. Lakshmin (CS)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>J. Nie (Math)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N. Zhou (ME)</td>
<td></td>
</tr>
</tbody>
</table>
Overview

- Background, target applications, grand vision
- Simple cantilever beam example
- Describing MEMS: ingredients and examples
- A bigger example: analysis of a micromirror
- Ongoing work: measurement feedback, synthesis, web-based simulation
- Q & A
Levels of simulation

• Solve continuum equations (momentum balance, Maxwell’s, etc.)
• Solve simplified equations of beam and plate theory (structural elements)
• Solve network equations (e.g. modified nodal analysis in SPICE; Simulink models)
• These approaches are not mutually exclusive!
• Share similar software structures
Where does SUGAR fit?

- Primarily simulates electromechanical systems
- Has element models at the structural and network levels
- Provides a flexible language for device description
- Performs static, frequency-response, modal, and (some) transient analysis
- Can build quick models that get high-level behavior
- Freely available and open source
SUGAR architecture

![Diagram showing the SUGAR architecture with components like Netlist, System assembly, Models, Solvers, Matlab, Web, Library, Results, Transient analysis, Steady-state, modal analysis, Static analysis, Extension interfaces (Matlab) and User interfaces.](diagram.png)
SUGAR: Recent evolution

- SUGAR 2.0 released last year
- SUGAR 3.0 is a major overhaul: a C program with Matlab interfaces
- Can still use 2.0 model functions and netlists
- Integrating more efficient solvers
  - SuperLU, SLICOT, DASSL, ...
- Web version will only require a browser
Device description

- Device descriptions are called *netlists* in analogy to SPICE
- Basic ingredients: nodes, materials, and elements
- Standard material parameter libraries available for MUMPS
Hello world: a cantilever

use 'mumps.net'
use 'stdlib.net'

anchor {node 'substrate', p1; l=10u, w=10u}
beam3d {node 'substrate', node 'tip', p1; l=100u, w=2u}
f3d {node 'tip'; F=2u, oz = 90}
Hello world: a cantilever
Running a simple analysis

```matlab
net = cho_load('cantilever.net');
cho_display(net);
dq = cho_dc(net);
cho_display(net, dq);
dy = cho_dq_view(dq, net, 'tip', 'y')
```

- Load and display device description
- Analyze and display static displacement
- Get y-displacement of tip
Deflected cantilever
Node positioning

Can position nodes *implicitly* via element geometries and connectivity

```plaintext
beam3d {node 'substrate', node 'tip', p1;
   l=100u, w=2u}
```

or *explicitly*

```plaintext
substrate = node {name='substrate';
   0, 0, 0}
tip = node {name='top';
   0, 100u, 0}
beam3d {tip, top, p1; w=2u}
```
Materials

poly = material {
    Poisson = 0.3,
    ...
}

p1 = material {
    parent = poly,
    h = 2u
}

- Specify material properties in material structures
- Materials can inherit properties from other materials
Example: Gap-closing actuator
Gap actuator netlist

use 'mumps.net'
use 'stdlib.net'
Vsrc {node 'A', node 'f'; V = Vin or 10}
eground {node 'f'}
anchor {node 'A', p1; l=5u, w=10u, oz=180}
beam2de {node 'A', node 'b', p1;
  l=100u, w=2u, h=2u, R=100}
gap2de {node 'b', node 'c', node 'D', node 'E', p1;
  l=100u, w1=10u, w2=5u, gap=2u}
anchor {node 'D', p1; l=5u, w=10u, oz=-90}
anchor {node 'E', p1; l=5u, w=10u, oz=-90}
eground {node 'D'}
eground {node 'E'}
Netlist explanation

use 'mumps.net'
use 'stdlib.net'

- Include mumps.net for process info
- stdlib.net includes standard model declarations and support routines
Netlist explanation

Vsrc {node 'A', node 'f';
    V = Vin or 10}
eground {node 'f'}

- Voltage source connects base of beam at A to electrical ground at f
- If Vin defined, use that for voltage
- If Vin not defined, default to 10V
Netlist explanation

anchor  {node 'A', p1;
  l=5u, w=10u, oz=180}
beam2de {node 'A', node 'b', p1;
  l=100u, w=2u, h=2u, R=100}

- Anchored node A is where voltage is applied
- Cantilever / resistor extends from A to b
Netlist explanation

gap2de {node 'b', node 'c',
        node 'D', node 'E', p1;
        l=100u, w1=10u, w2=5u, gap=2u}
anchor {node 'D', p1; l=5u, w=10u, oz=-90}
anchor {node 'E', p1; l=5u, w=10u, oz=-90}

- Gap element consists of two initially parallel beams
- Top beam from $b$ to $c$ attaches to cantilever
- Bottom beam from $D$ to $E$ is anchored down
Netlist explanation

eground {node 'D'}
eground {node 'E'}

- Bottom plate is also grounded
Using the Matlab interface

We wrote the netlist to allow changing input voltages:

\[ V_{src} \{ \text{node 'A', node 'f'; } V = V_{in} \text{ or } 10 \} \]

Sweep the voltage to see pull-in:

\[ dq = []; \]
\[ \text{for } k=1:12 \]
\[ \quad \text{param}.V_{in} = k; \]
\[ \quad \text{net} = \text{cho_load('beamgap.net', param)}; \]
\[ \quad dq = \text{cho_dc(net, dq)}; \]
\[ \quad \text{cho_display(net, dq)}; \]
\[ \quad \text{tip}(k)=\text{cho_dq_view(dq, net, 'c', 'y')}; \]
\[ \quad \text{pause}; \]
\[ \text{end} \]
Subnets and hierarchical design

- *Subnets* are parameterized components
- Subnet calls look like built-in model calls
- Parallels functional decomposition of design
- Can put commonly-used subnets in a library
Subnet parameters

subnet meander(A, B, material, l, w, h, nmeanders)
    ...
end
meander {node 'C', node 'D', p1;
    l=100u, nmeanders=5}

- Parameters identified by position or by name
- Parser checks the material parameter for parameters undefined by caller
- Any undefined parameters are left nil
Nested coordinate systems

- Each subnet has an associated local coordinate system
- Nested subnets result in multiple nested coordinate systems
- Simplifies subnet re-use
Example: simplified ADXL-05
Building arrays

Loop structure lets us build a structure with ten or a thousand units using the same code.

```plaintext
c = {node()}
Suspension {c(1), p1, 200u}
for k = 1, nfingers do
    c[k+1] = node()
    Mass {p1, c[k], c[k+1], p1, 100u}
end
Suspension {c(1), p1, 200u; oz = 180}
```
Modes of ADXL-05 model

(Displacements are exaggerated)

```matlab
net = cho_load('adxl.net');
[f,e,dq] = cho_mode(net);
cho_modeshape(net, f,e,dq, 1);
```
A bigger example
Micromirror SEM
Simulated frequency response
Measured frequency response
First resonant mode
Second resonant mode
Third resonant mode

Mode 3, frequency 8682.451935 Hz
Matisse: Integrating measurement

- Set of optical measurement tools
- Make available on the web as a “virtual lab”
- Matisse team: R. Kant, R. Muller, C. Rembe, M. Young
Matisse: Closing the design loop

- Idea
- Rough draft
- CAD/Simulation Tool
  - Design
  - Mathematic mode
  - Simulation
  - Evaluation of model
  - Prototype

- Production tools
  - Manufacturing
  - Sample
  - Characterization tools
    - Measurement
    - Optical measurement methods
M&MEMS: Web-based MEMS simulator

- Hosted on UC Berkeley Millennium cluster
- Used in Introduction to MEMS course, Fall 2001
- Currently offline while upgrading to SUGAR 3.0
- Accounts will be available for outside users
M&MEMS: SUGAR on the Web

http://sugar.millenium.berkeley.edu/
Design synthesis and optimization

- Genetic algorithms to evolve new designs
- Specializing designs from a library
And more!

- Models: more and better
- Improved numerics and additional analyses
- Feedback from measured data
- SUGAR as a component in design optimizer