SUGAR:
A MEMS Simulation Program

David Bindel

dbindel@eecs.berkeley.edu

UC Berkeley, CS Division

SUGAR contributors

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

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 conservation, 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

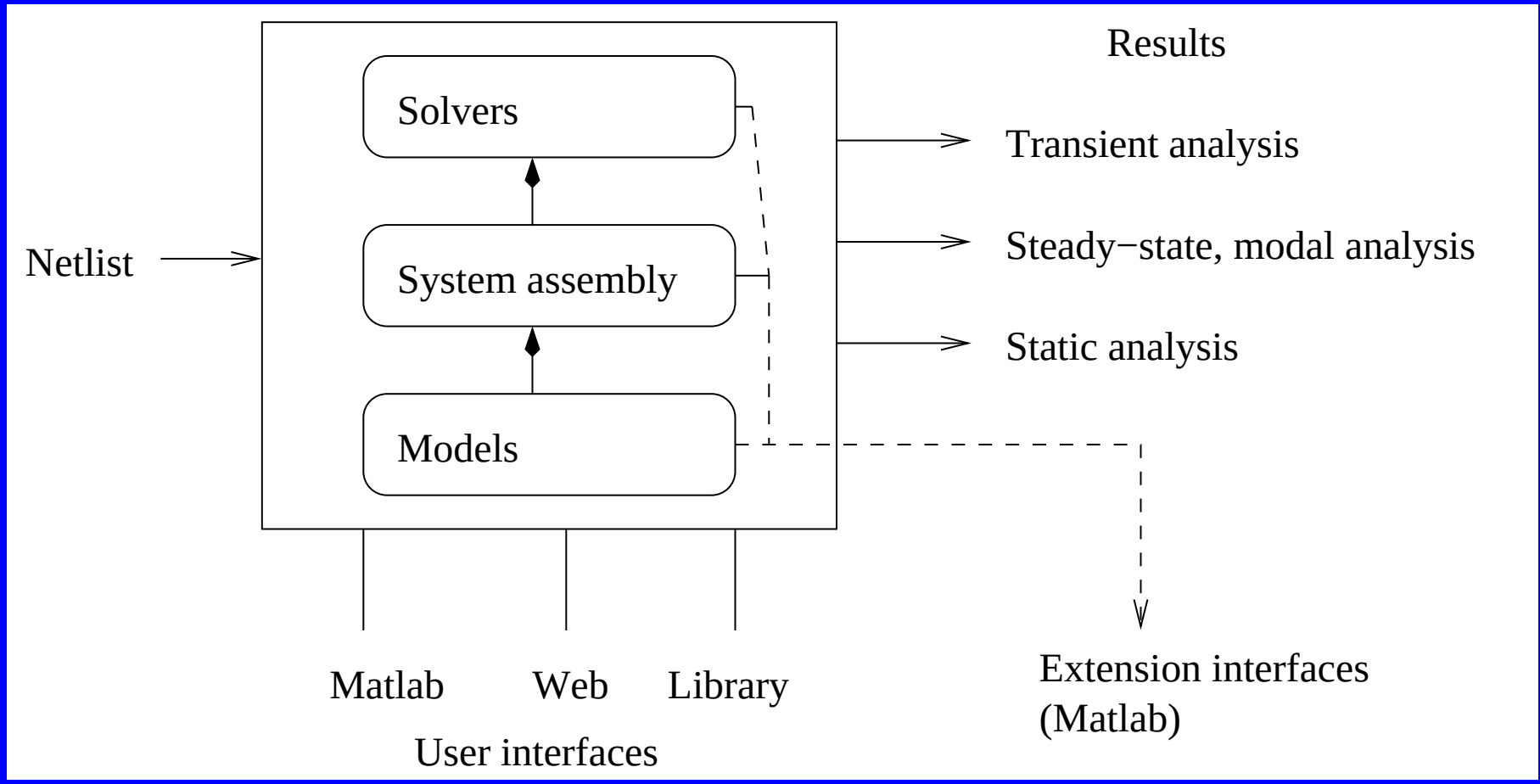
Where does SUGAR fit?

- 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

Where does SUGAR fit?

- Freely available and open source
  - `www.sourceforge.net/project/mems`
  - `sugar.millennium.berkeley.edu`
- Useful for education and prototyping
- Building block for higher-level operations
  - e.g. Design synthesis and optimization
- Part of work to “close design loop”
  - Simulation (SUGAR)
  - Measurement instruments

SUGAR architecture



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, HOMPACK, ...

Device description

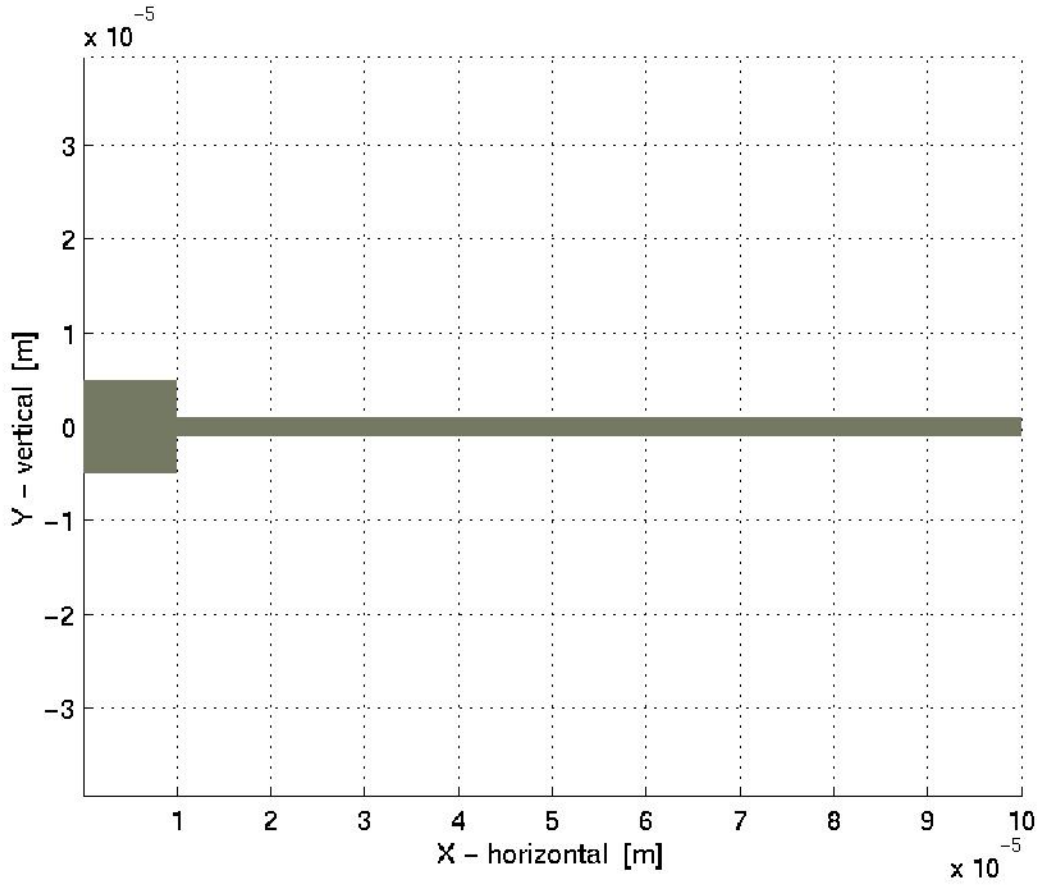
- Device descriptions are called *netlists* in analogy to SPICE
- Basic ingredients: nodes, materials, and elements
- Standard material parameter libraries available for MUMPS

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

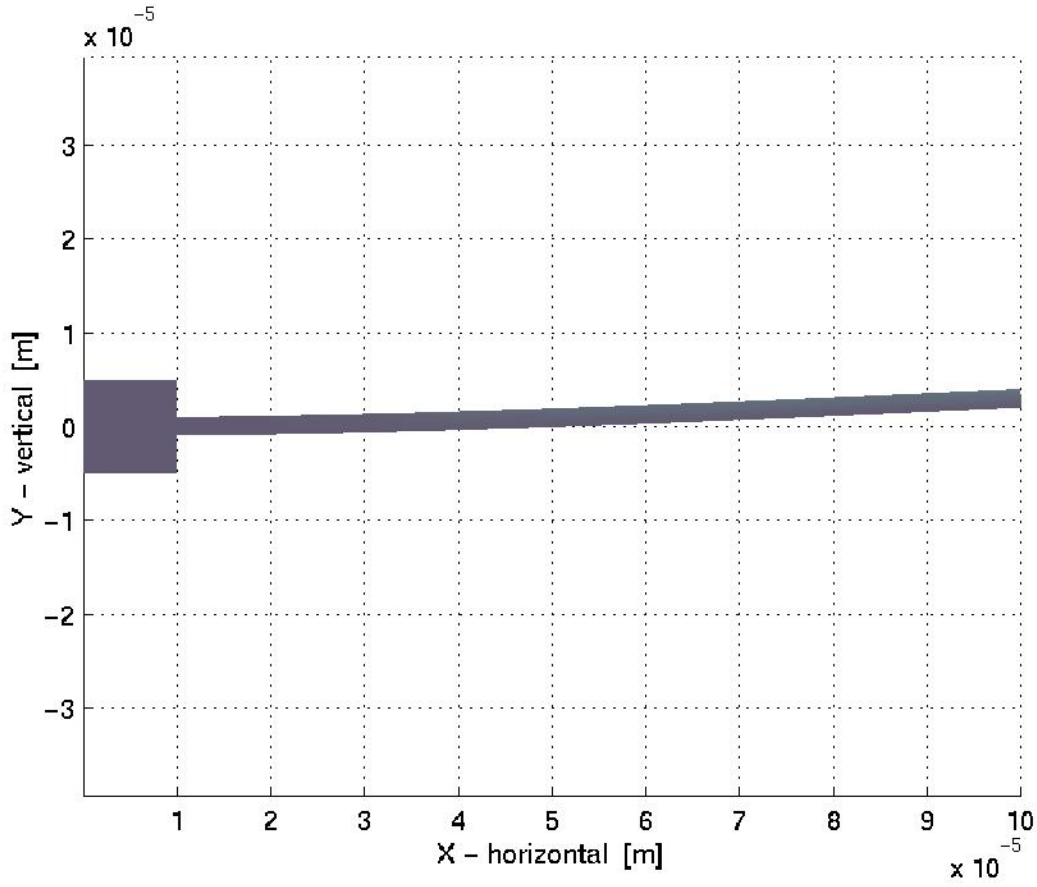


Running a simple analysis

```
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

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

or *explicitly*

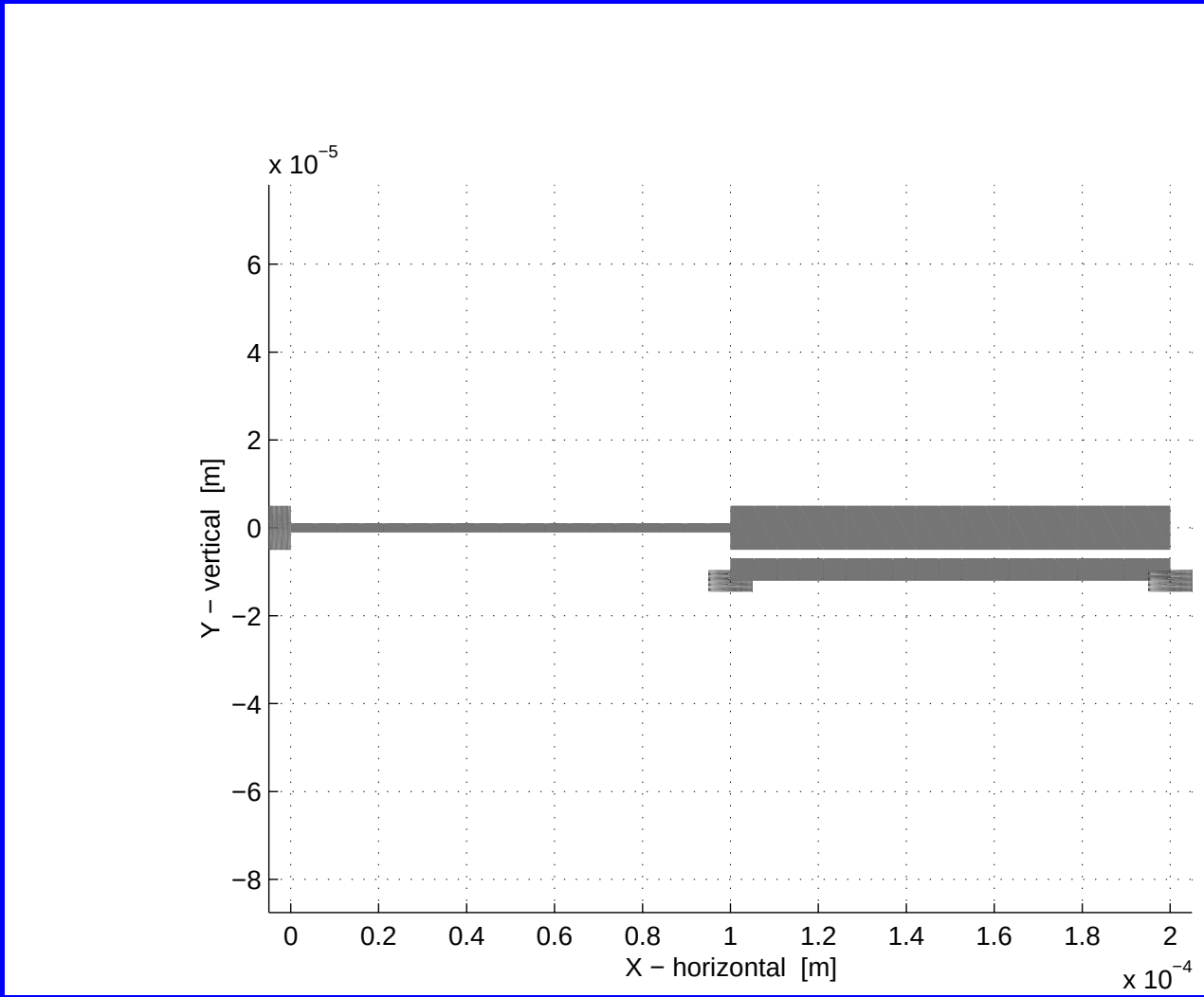
```
substrate = node {name='substrate';
                  0, 0, 0}
tip       = node {name='tip';
                  100u, 0, 0}
beam3d {substrate, top, p1; w=2u}
```

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'
if not Vin then Vin = 10 end
Vsrc    {node 'A', node 'f'; V = Vin}
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

```
if not Vin then Vin = 10 end
Vsrc {node 'A', node 'f'; V = Vin}
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:

```
if not Vin then Vin = 10 end
```

Sweep the voltage to see pull-in:

```
dq = [];
for k=1:12
  param.Vin = k;
  net = cho_load('beamgap.net', param);
  dq = cho_dc(net,dq);
  cho_display(net, dq);
  tip(k)=cho_dq_view(dq,net, 'c', 'y');
  pause;
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

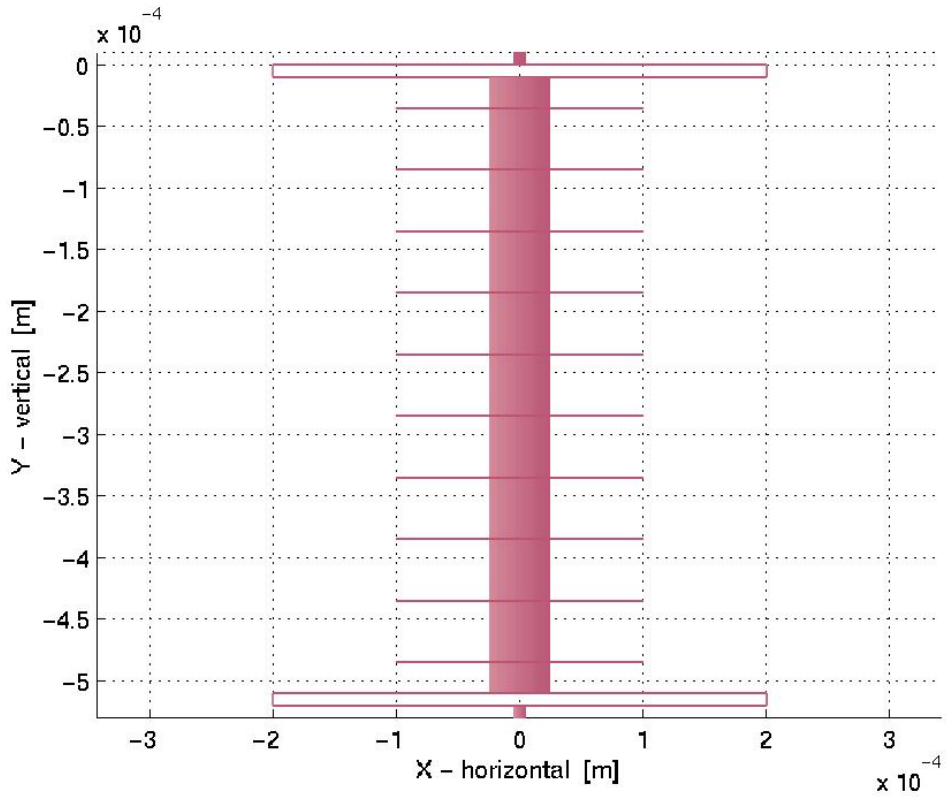
```
subnet meander(A, B, material,
               l, w, h, nmeanders)
  ...
end
meander {node 'C', node 'D', p1;
         l=100u, nmeanders=5}
```

- 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

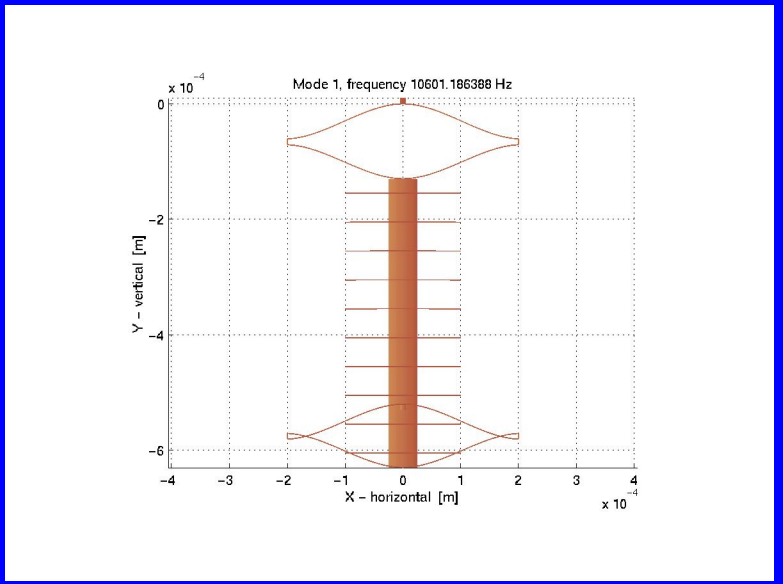


Building arrays

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

```
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}
```

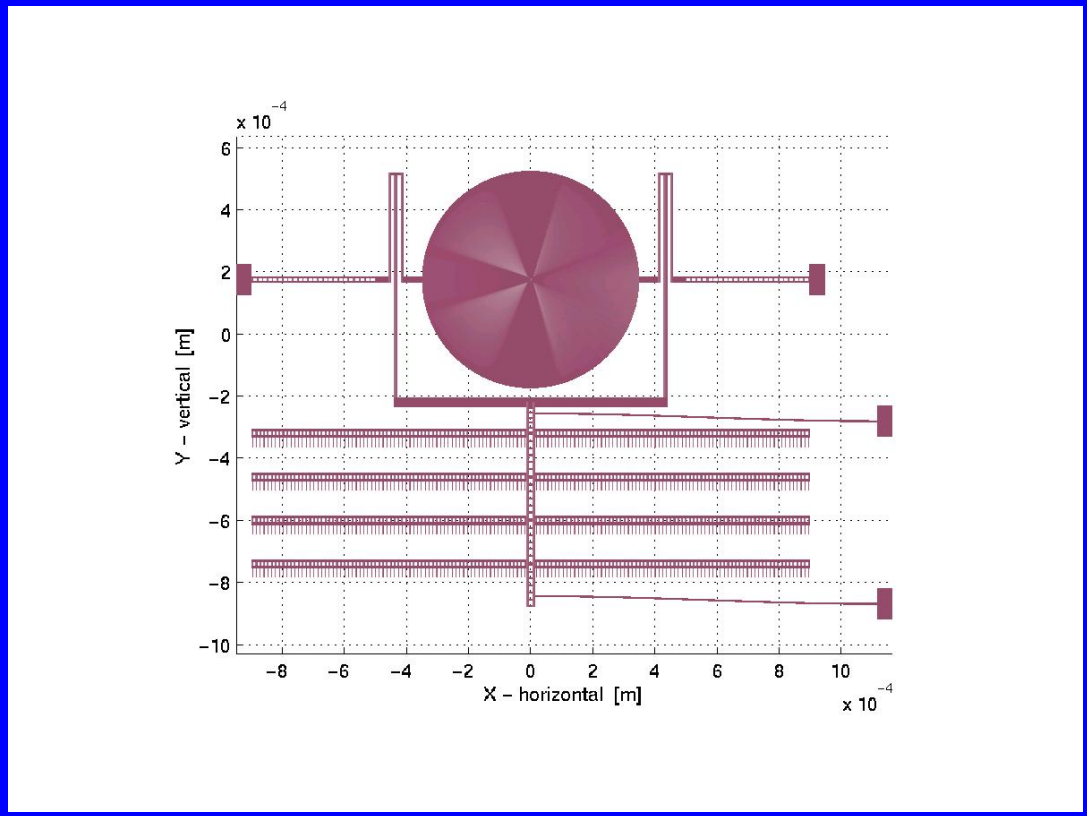
Modes of ADXL-05 model



(Displacements are exaggerated)

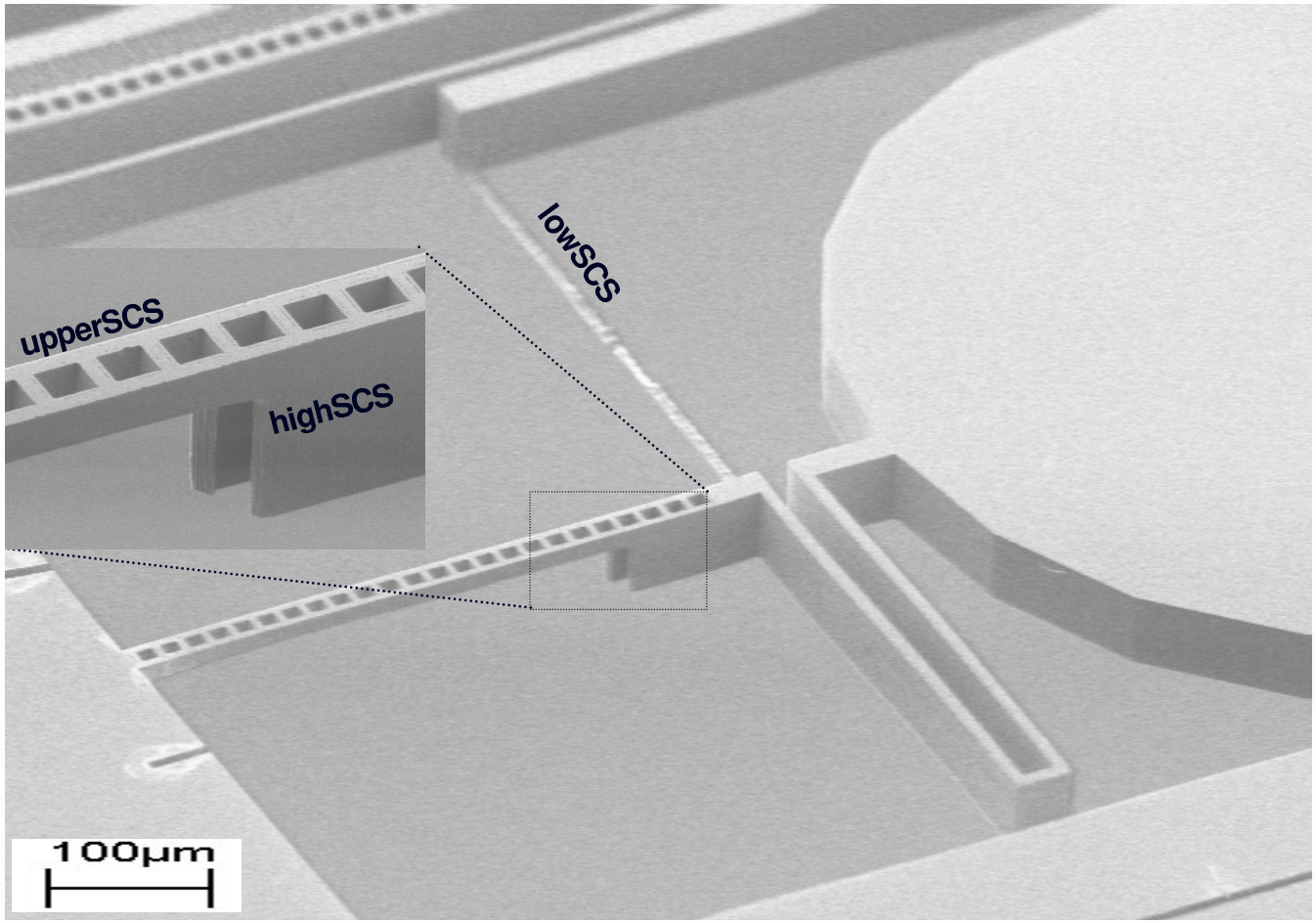
```
net = cho_load('adxl.net');
[f,e,dq] = cho_mode(net);
cho_modeshape(net, f,e,dq, 1);
```

A bigger example

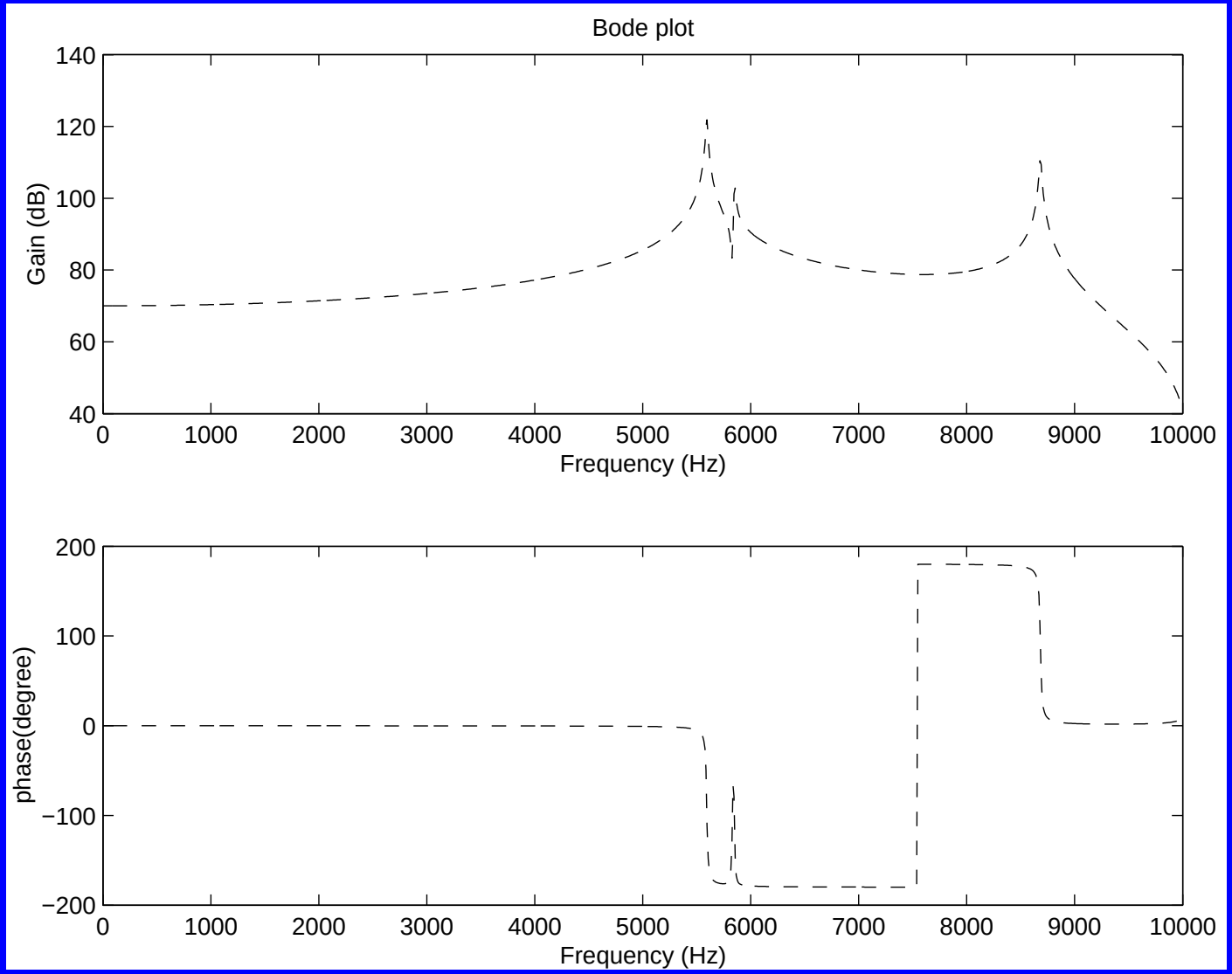


- Micromirror design due to Matt Last
- Model has roughly 11K degrees of freedom

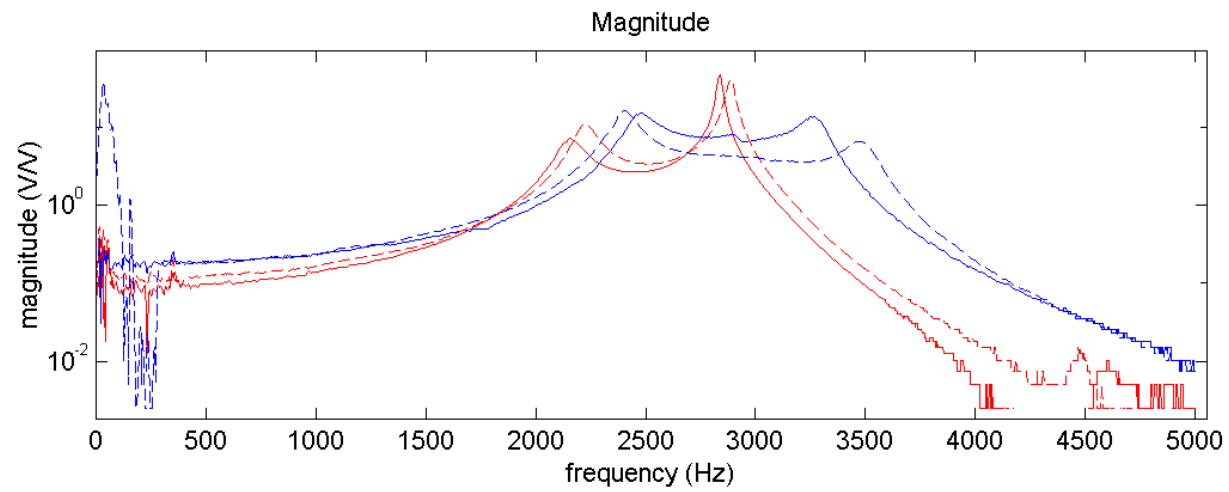
Micromirror SEM

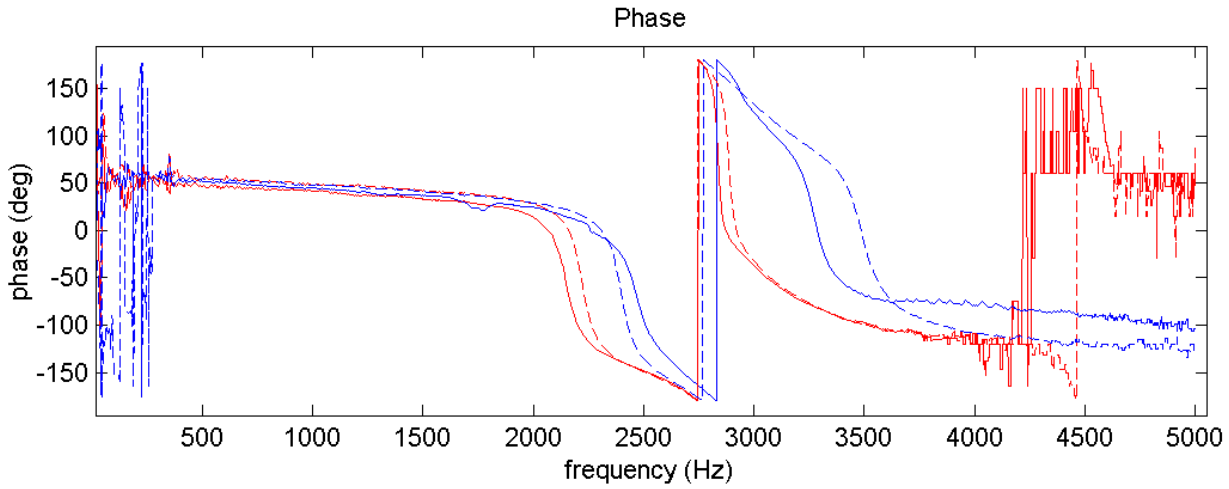


Simulated frequency response

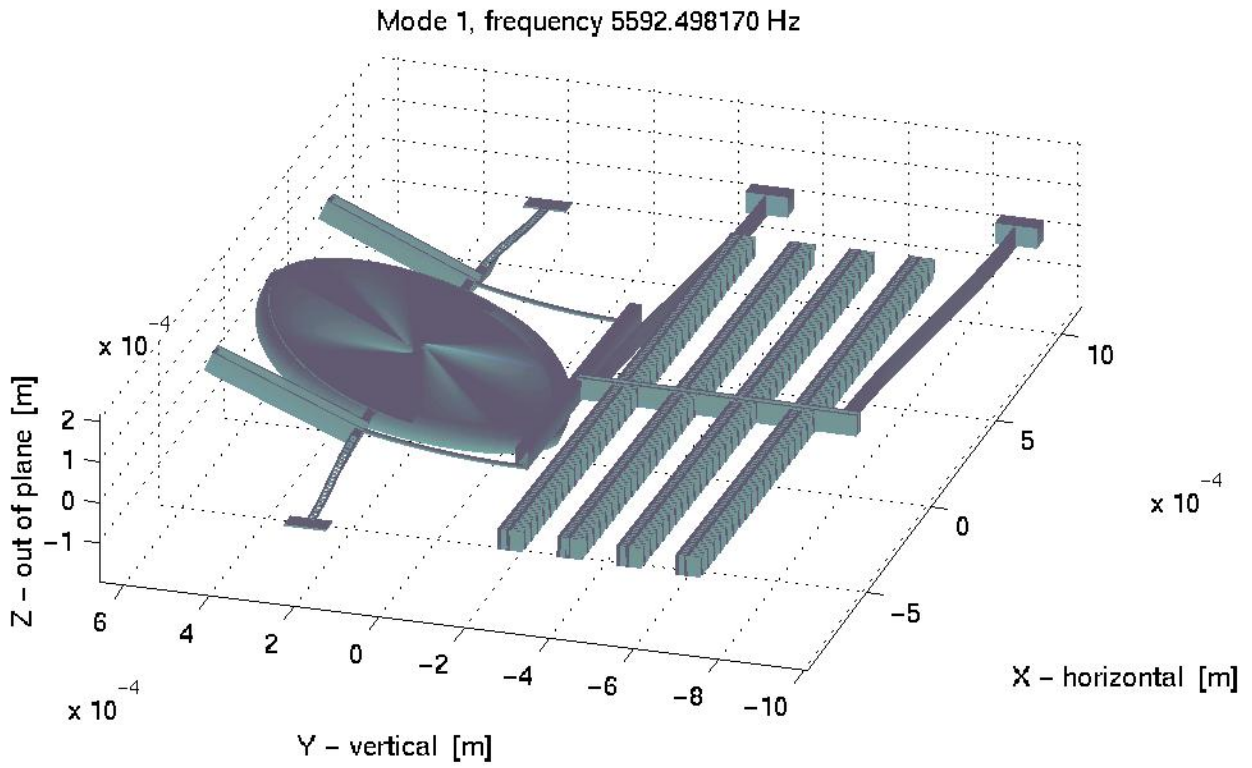


Measured frequency response

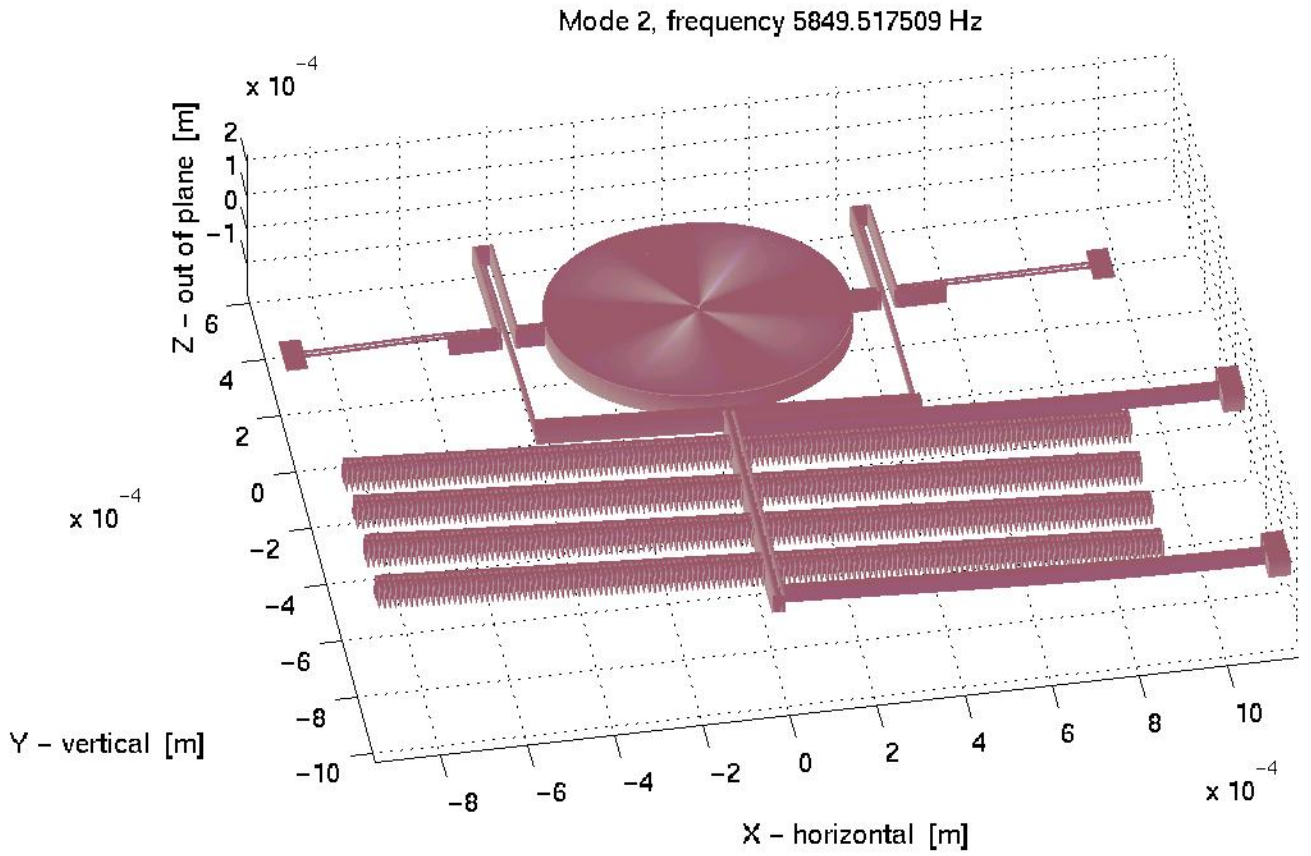




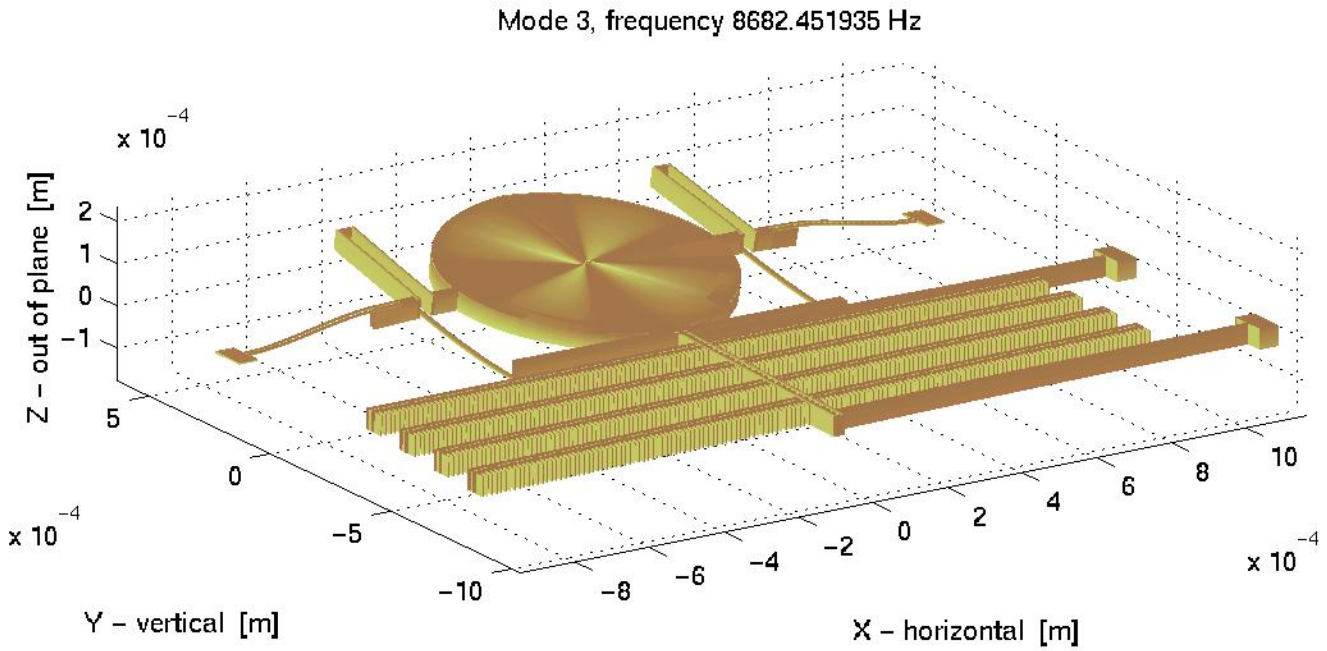
First resonant mode



Second resonant mode



Third resonant mode



Existing models

- Mechanical: anchor, beam2d, beam3d, f2d, f3d, rigid, constraint
- Electrical: L, R, C, Isrc, Vsrc, opamp, vcvs
- Coupled: comb2d, gap2dforce, gap3dforce
- Subnets: beam2de, beam3de, gap2de, gap3de

Models under construction

- Plates
- Simple hinges and sliders
- Anisotropic beams
- Nonlinear beams
- Thermal circuit analogues
- Electrothermal and thermomechanical

Future models

- Contact models
- Improved damping
- Wrappers around FEAP models
- Controllers
- Any requests?
- Feel free to add your own!

Analyses

- Current
  - Static equilibrium
  - Steady-state frequency-response analysis
  - Modal analysis
  - Transient analysis (2.0)
- Future
  - Sensitivity (various flavors)
  - Bifurcation analysis

Ongoing numerical work

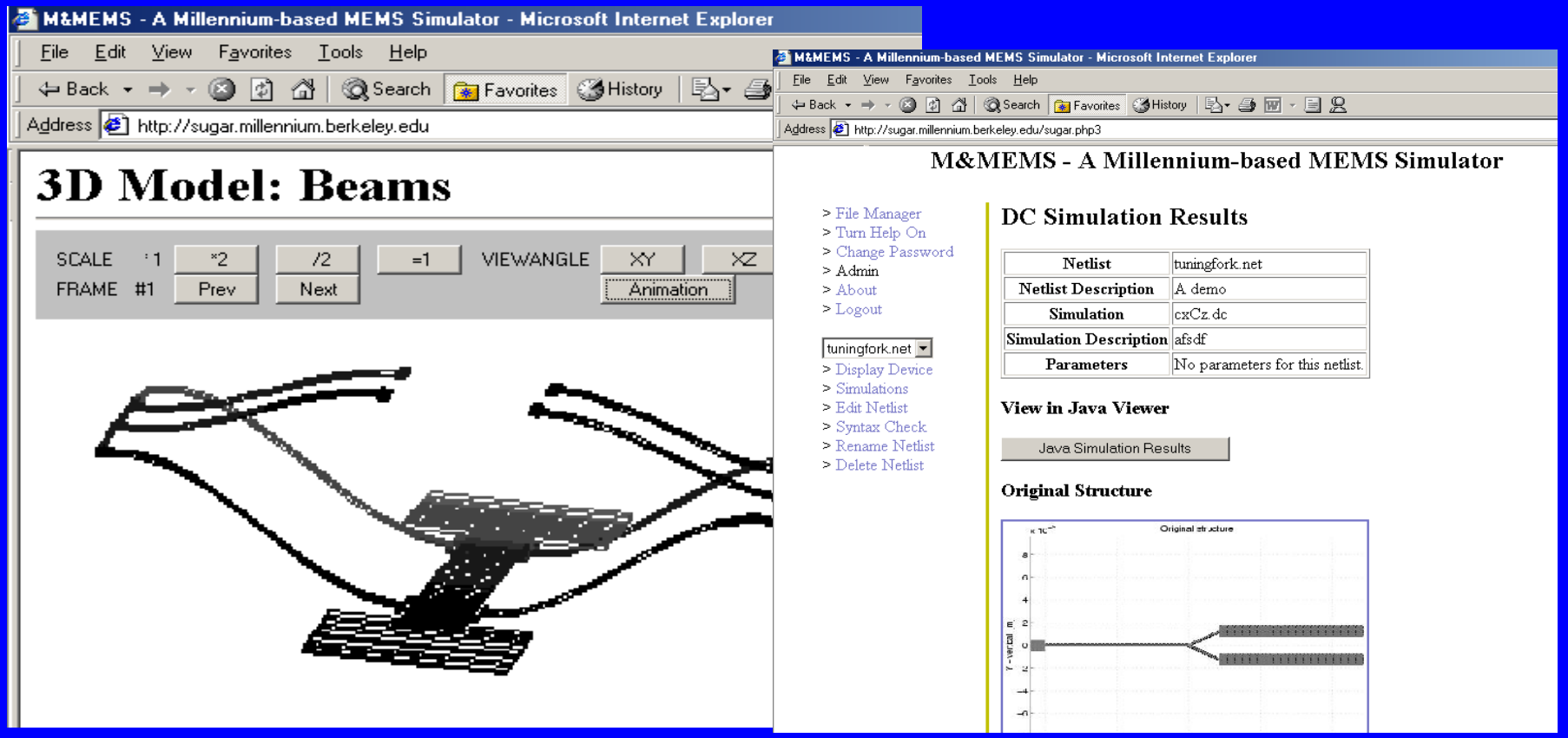
- Have adopted standard sparse solver packages for linear solves and modal analysis
- Reduced order modeling techniques (used for mirror steady-state response analysis)
- Incorporating newest DAE solvers (IDA); parameter sensitivity for DAEs
- Bifurcation analysis of DAE systems
- CIS algorithm for large-scale bifurcation analysis
- Dealing with multi-scale problems

Closing the design loop

- Integrate measurement and simulation facilities
  - R. Muller, R. Kant, C. Rembe, M. Young working on measurements at UCB
  - Other groups at CMU, MIT, Sarnoff
- Feedback measured data into simulation, design
  - Compare simulation and reality
  - Parameter extraction, sensitivity studies
- Make facilities available as a “virtual lab”

M&MEMS: SUGAR on the Web

http://sugar.millennium.berkeley.edu/



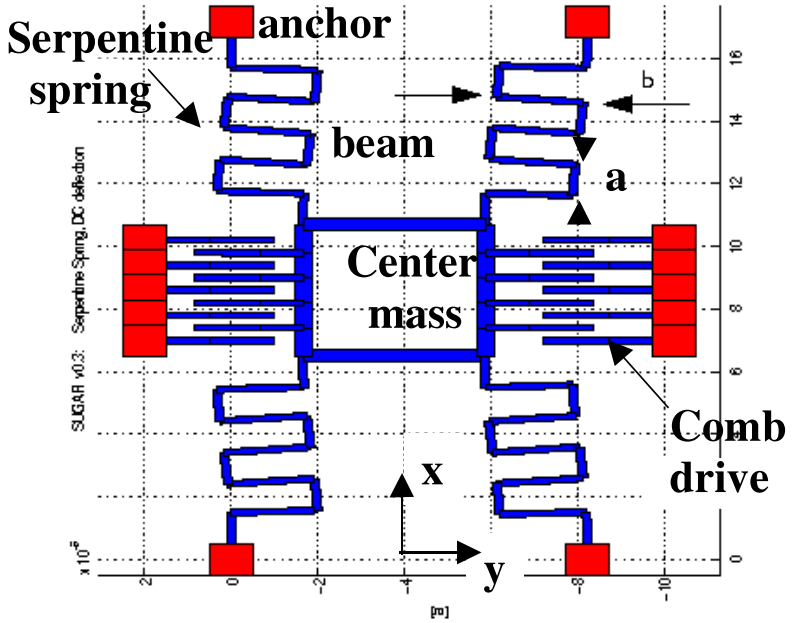
M&MEMS

- Hosted on UCB Millennium cluster
- Used in Introduction to MEMS course, Fall 2001
- Accounts available for outside users
- Currently offline while upgrading to SUGAR 3.0

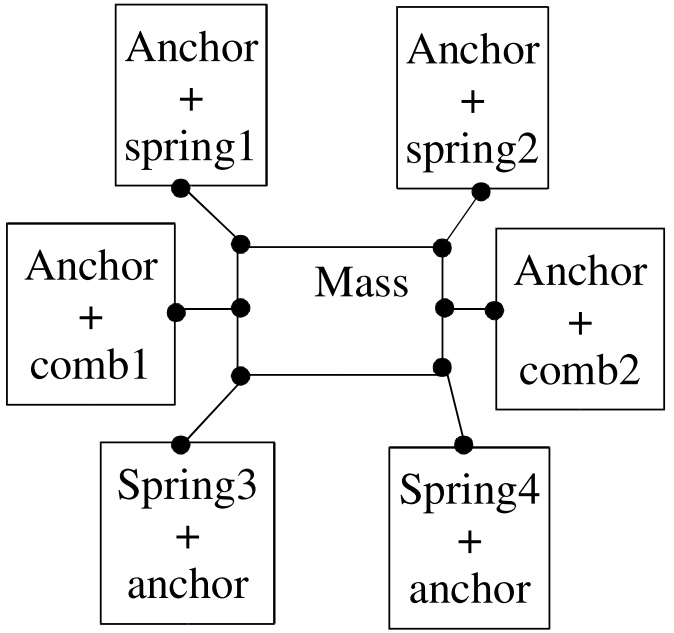
Design synthesis and optimization

- Genetic algorithms to evolve new designs
- Also simulated annealing approach
- Specializing designs from a library
- N. Zhou, B. Zhu, A. Agogino, and K. Pister: “Evolutionary Synthesis of MEMS (Microelectronic Mechanical Systems) Design” (ANNIE 2001). First Runner-up for Novel Smart Engineering System Design Award.

Functional decomposition and GA



(a) **MEMS resonator with four meandering springs**



(b) GA Building blocks and their connectivity

Conclusion

- Web links
  - `bsac.eecs.berkeley.edu/~cfm`
  - `www.sourceforge.net/project/mems`
  - `sugar.millennium.berkeley.edu`
- SUGAR is actively used
  - Educationally
  - For prototyping and exploring
  - As a testbed for larger projects
- We would like more users and contributors!
- Questions? What would you like to see?