layer: name of the layer where the comb is fabricated; Properties of this layer are defined in process file.

node1, node2: two terminal nodes; node names can be any letter or number.

l: length of the overlap of fingers.

oz: orientation angle around z axis; It is the direction pointed from node1 to node2;

N: number of fingers on each set;

h: optional; thickness of the layer; can overwrite the thickness in process file;

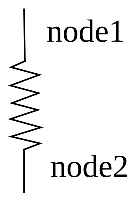
gap: gap distance between fingers;

AUTHOR

Ningning Zhou: section 1, 3, 4, 5, 6, appendix II
Jason V. Clark: section 2, appendix I.

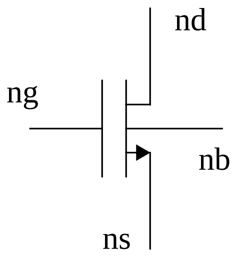
si: Optional; Magnitude of small sinusoidal current; Only required in steady state analysis;
sph: Optional; Phase of small sinusoidal current; Only required in steady state analysis;

(20) resistor



Constant resistor.
Element syntax:
name rc <*> [node1 node2] [R=val]
name: element name, can be any letter or number;
rc: model function name;
<*>: optional; leave it blank or use * since no process information associated with a current source.
node1, node2: two terminal nodes; node names can be any letter or number.
R: resistance;

(21) nmos

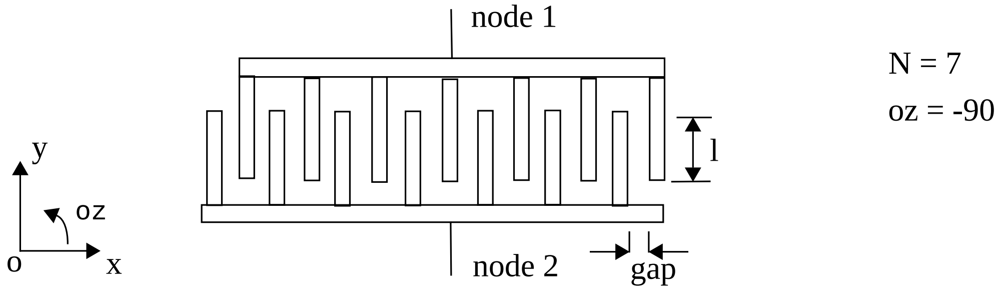


Simplest nmos transistor model as it in EE140.
Element syntax:
name nmos nmos-para [nd ng ns nb] [l=val w=val ....]
name: element name, can be any letter or number;
nmos: model function name;
nmos-para: name of the parameter block in process file which contains all necessary parameters to simulate nmos such as VTO, KP, PHI etc.
nd: two terminal nodes; node names can be any letter or number.

(22) pmos

(23) comb2d

Two dimensional electrostatic comb. Only electrostatic fringe forces are considered. Vertical electrostatic forces on each finger are assumed to be cancelled.



Element syntax:
name comb2d layer [node1 node2] [l=val oz=val N=val <h=val> gap=val ....]
name: element name, can be any letter or number;
comb2d: model function name;

l: length of the gap from node1 to node2;
w1: width of the beam1;
w2: width of the beam2;
oz: orientation angle of the direction pointed from node1 to node2, measured in degrees, counter-clockwise from the positive x axis of node1.
<h>: layer thickness of the beam; optional, not needed here if defined in process file; can overwrite the thickness in process file;
gap: Initial gap distance;
R1: resistance of the beam1;
R2: resistance of the beam2;

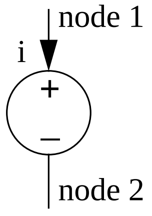
(14) g2demL

(15) g2demR

(16) g2dem_cont

(17) g2dem3D.m

(18)volc



Constant voltage sources.
Element syntax:
name volc <*> [node1 node2] [V=val <sv=val> <sph=val>]
name: element name, can be any letter or number;
volc: model function name;
<*>: optional; leave it blank or use * since no process information associated with a voltage source.
node1, node2: two terminal nodes; node names can be any letter or number. Current i is from node 1 to node 2;
V: DC voltage on this voltage source;
sv: Optional; Magnitude of small sinusoidal voltage; Only required in steady state analysis;
sph: Optional; Phase of the small sinusoidal voltage; Only required in steady state analysis;

(19) current

Constant current sources.
Element syntax:
name current <*> [node1 node2] [I=val si=val sph=val]
name: element name, can be any letter or number;
current: model function name;
<*>: optional; leave it blank or use * since no process information associated with a current source.
node1, node2: two terminal nodes; node names can be any letter or number. Current I is from node 1 to node 2;
I: DC current on this current source;

Constant two dimensional force model. Forces are applied on two nodes. They are euqal and opposite relative forces.

Element syntax:

name f2dc2 <*> [node1 node2] [F=val oz=val M=val]

name: element name, can be any letter or number;

f2dc2: model function name;

<*>: optional; leave it blank or use * since no process information associated with a force generator.

node1 node2: nodes where forces are applied on; node names can be any letter or number.

F: magnitude of the force applied on node1 in newtons. The force on node2 is equal and opposite to the force on nodel 1. The force magnitude on node2 is -F.

M: magnitude of the moment applied on node1; counter-clock wise is positive. The moment on node2 is equal and opposite to the moment on nodel 1. The moment magnitude on node2 is -M.

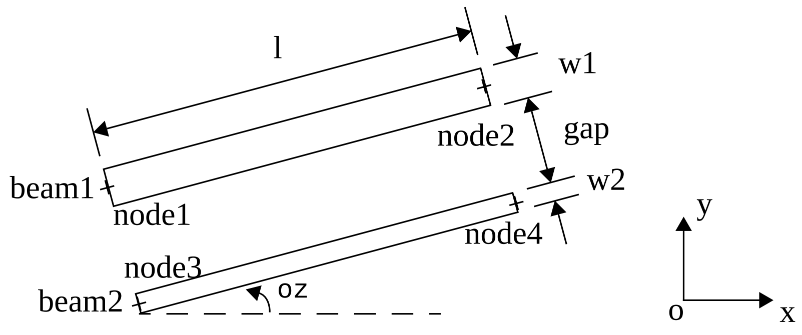
oz: orientation angle of the node1 force measured in degrees, counter-clockwise from the positive x axis of node1.

(11)f3dc1

(12) f3dc2

(13) g2dem

Electrostatic gap model; The distributed electrostatic forces and moments are equated as forces and moments on four nodes. Two beams of the gap are parallel initially. They have equal



length(l) and equal thickness(h), could have different width(w).

Element syntax:

name g2dem layer [node1 node2 node3 node4] [l=val w1=val w2=val oz=val gap=val <h=val> R1=val R2=val]

name: element name, can be any letter or number;

g2dem: model function name;

layer: name of the layer where this gap is fabricated; Properties of this layer are defined in process file.

node1, node2: two terminal nodes of the upper beam 1;

node3, node4: two terminal nodes of the lower beam 2;

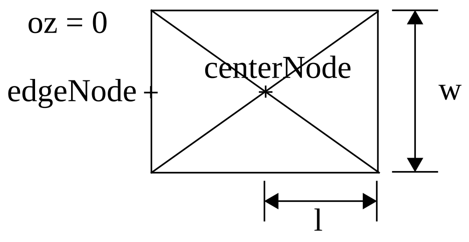
number.

l: length from edgeNode to centerNode;

w: width of the anchor;

oz: orientation angle of the direction pointed from edgeNode to centerNode, measured in degrees, counter-clockwise from the positive x axis of node1.

<h>: layer thickness of the beam; optional, not needed here if defined in process file; can overwrite the thickness in process file;



(6) a2dmef

Anchor two dimensional mechanical model. Assume both nodes are fixed mechanically. Anchor is also modeled as a constant resistor electrically

Element syntax:

name a2dmf layer [edgeNode centerNode] [l=val w=val oz=val <h=val> R=val]

R: resistance of the anchor;

See **a2dmf** for other definitions.

(7) a3dmf

(8) a3demf

(9) f2dc1

Constant two dimensional force model. Forces are applied on only one node.

Element syntax:

name f2dc1 <*> [node1] [F=val oz=val M=val]

name: element name, can be any letter or number;

f2dc1: model function name;

<*>: optional; leave it blank or use * since no process information associated with a force generator.

node1: the node where forces are applied on; node names can be any letter or number.

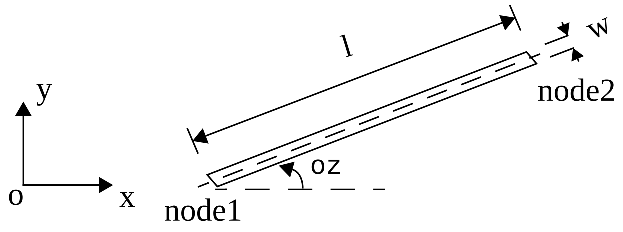
F: magnitude of the force applied on node1 in newtons.

M: magnitude of the moment applied on node1; counter-clock wise is positive.

oz: orientation angle of the force measured in degrees, counter-clockwise from the positive x axis of node1.

(10) f2dc2

Beam two dimensional mechanical model. Linear model.



Element syntax:

name b2dm layer [node1 node2] [l=val w=val oz=val <h=val>]

name: element name, can be any letter or number;

b2dm: model function name;

layer: name of the layer where this beam is fabricated; Properties of this layer are defined in process file.

node1, node2: two terminal nodes; node names can be any letter or number.

l: length of the beam from node1 to node2;

w: width of the beam;

oz: orientation angle of the direction pointed from node1 to node2, measured in degrees, counter-clockwise from the positive x axis of node1.

<h>: layer thickness of the beam; optional, not needed here if defined in process file; can overwrite the thickness in process file;

(2) b2dem

Beam two dimensional electrical mechanical model; Linear mechanical model and also models the beam as a constant resistor.

Element syntax:

name b2dem layer [node1 node2] [l=val w=val oz=val <h=val> R=val]

R: resistance of the beam;

See **b2dm** for other definitions.

(3) b3dm

(4) b3dem

(5) a2dmf

Anchor two dimensional mechanical model. Assume both nodes are fixed.

Element syntax:

name a2dmf layer [edgeNode centerNode] [l=val w=val oz=val <h=val>]

name: element name, can be any letter or number;

a2dmf: model function name;

layer: name of the layer where this anchor is fabricated; Properties of this layer are defined in process file.

edgeNode: the terminal node at the edge of anchor; node names can be any letter or number.

centerNode: the terminal node at the center of anchor; node names can be any letter or gaps are pulled in. Traditionally pull-in voltage can be found by continuously increasing the applied voltage until the program cannot find a balanced equilibrium DC solution. Usually after applied voltage is larger than pull-in voltage, you will either see some imaginary or unreasonable solutions, or get error messages like ’couldn’t find convergent solution’.

An alternative method is using the gap model with contact forces, such as ’g2dem_cont’ in 2D case. Contact forces could balance the electrostatic forces at the stage beyond pull-in and when two beams of the gap contact to each other. ’demo_pullin’ would be an example of this.

To do voltage sweeping, you can write a loop in matlab.

APPENDIX I   MODEL FUNCTION LIST

(1) b2dm.m	2D mechanical beam
(2) b2dem.m	2D electromechanical beam
(3) b3dm.m	3D mechanical beam
(4) b3dem.m	3D electromechanical beam
(5) a2dmf.m	2D mechanical anchor
(6) a2demf.m	2D electromechanical anchor
(7) a3dmf.m	3D mechanical anchor
(8) a3demf.m	3D electromechanical anchor
(9) f2dc2.m	2D constant 2-node force
(10) f2dc1.m	2D constant 1-node force
(11) f3dc1.m	3D constant 1-node force
(12) f3dc2.m	3D constant 2-node force
(13) g2dem.m	2D electromechanical gap
(14) g2demL.m	2D electromechanical gap left
(15) g2demR.m	2D electromechanical gap right
(16) g2dem_cont.m	2D electromechanical gap with contact forces;
(17) g2dem3D.m	2D(planar) electromechanical gap in 3D;
(18) volc.m	constant voltage
(19) rc.m	constant resistor
(20) current.m	constant current source
(21) nmos.m	nmos
(22) pmos.m	pmos
(23) comb2d.m	electric comb

APPENDIX II   ELEMENT MODELS

(1) b2dm

f : (rad/sec) resonant frequencies sorted in ascending order. They are the eigenvalues of system matrices.
egv: NxN matrix of eigenvectors corresponding to each frequency.
dq : solution to equilibrium position.

(2) cho_modeshape(net, egv, dq, scale_factor, mode_number)
Graphics displaying of the mode shape.
egv, dq: outputs from cho_mode;
scale_factor: scales the deflection in mode shape displaying. usually between 0 and 1; Make it larger if the mode deflection is too small.
mode_number: number of modes.

4) Steady State Analysis

Steady state analysis is the system response to small sinusoidal inputs around a DC equilibrium point. Both DC inputs and small sinusoidal inputs need to be specified in the input netlist file. DC inputs could be set as zero.

1) [Xw, dq] = cho_ss(net, W)
Computes the steady state response Xw.
W : is an Mx1 vector of frequencies.
net: variable which contain elements and system information.
Xw: is a NxM matrix. Each column corresponds to the solution for mth frequency in W.
dq : solution to equilibrium position.

(2)W = cho_w(f_start, f_stop, steps)
Generate a frequency vector W(with steps+1 frequencies) from f-start to f_stop. The frequencies in between are evenly distributed in log scale. Users can also create their own functions to generate frequency vectors other than this.

(3) [magn, phase] = cho_ss_plot(’node’, ’var’, W, Xw, dq)
Plots the log of magnitude vs. log frequency, and linear phase versus log frequency. Calculates the magnitude and phase for the input node and variable.
'node’: the node name or element name;
’var’ : the node or branch variable name;
W : Mx1 vector of frequencies;
Xw, dq: the first and second output parameter from cho_ss;
magn: magnitude response of the node or branch variable;
phase: phase response of the node or branch variable;

4) Transient Analysis

6. FREQUENTLY ASKED QUESTIONS

1) How to find pull-in voltage?
Pull-in point is the critical point beyond which system mechanical springs can not sustain the increases of electrostatic forces, structures can not reach equilibrium position, electrostatic

displaying and analysis.

netlist file: file name of input netlist.

net: cell array which contain elements and system information used in analysis and displaying program. net{1} is a structure array which contains element information. For example, net{1}(1) includes ’name’, ’model’, ’parameter’,’node’ information for the first element. net{2} is a structure contains system information.

2) DC equilibrium position analysis

(1) dq = cho_dc(net)

DC analysis function.

net: variable which contain elements and system information used in analysis and displaying program.

dq : a Nx3 cell structure. N is the total number of dynamic node nd branch variables. Each row of dq is given by: [node or branch name   variable name   solution].

Node or branch variable names are specified in model functions.

x, y, z: (m) translational displacement in x, y and z direction;

rx, ry, rz: rotational displacement around x, y and z axes;

i : (A) electrical current;

e: (V) electrical potential;

(2) cho_display(net, dq)

Display the original or deflected structures in three dimensional space. dq is the DC solution from cho_dc. Note that dq can be 0, in which case the original design is displayed.

(3) row = cho_locate(net, ’node’, ’var’)

Locate the row number of node or branch variable in DC solution ’dq’. The column number of any variable values in ’dq’ is always 3. Therefore, dq{row,3} is the DC soultion for this node variable. Note that dq is a cell structure, we should use dq{row,3} instead of dq(row,3).

row: row number of 'node' 'variable' in dq;

net: variable contains elements and system information;

'node': node name or branch name;

'var': variable name;

For example, run ’row = cho_locate(net, ’node5’, ’y’)’ first to find the row number, then ’dq{row,3}’ would be the DC solution for ’y’ displacement of node 5.

(4) num = cho_elem(net, ’element’)

Locate the index ’num’ of an element in 'net{1}'.

’element’:   element name; corresponding the first entry of each line in input netlist;

For example, to find the index of an anchor element ’BottomAnchor’ in example netlist in section 2, run

num = cho_elem(net, ’BottomAnchor’)

then, net{1}(num) contains all information of this anchor element.

3) Mode analysis

(1) [f, egv, dq] = cho_mode(net)

Calculate resonant frequencies and mode shapes.

4. DEMONSTRATIONS:

1) demo_canti

This is a pure mechanical domain analysis of cantilever beam in 2D and 3D. Element models are 2D and 3D mechanical anchors, beams and forces. DC analysis and displaying functions are called and displacements are calculated.

2) demo_dc

This is a 2D coupled electrical and mechanical domain analysis. It contains electrical voltage sources, electrical ground, electro-mechanical anchors, beams and gaps.

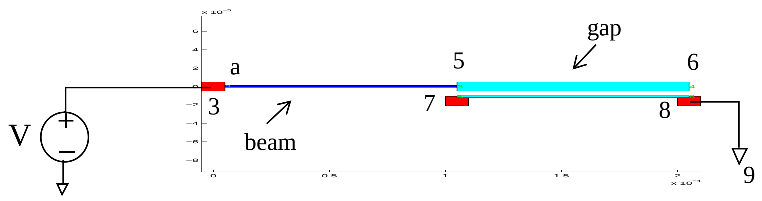


Figure 2 Beam gap demo

Equilibrium displacements have been calculated at an input voltage.

3) demo_mirror

This is a 3D mechanical mode analysis for a mirror structure. 3D mechanical anchors and beams are included. Resonant frequencies have been calculated and the first to fourth mode shapes are displayed.

4) demo_ss

This is a 2D steady state analysis for a resonator as following:

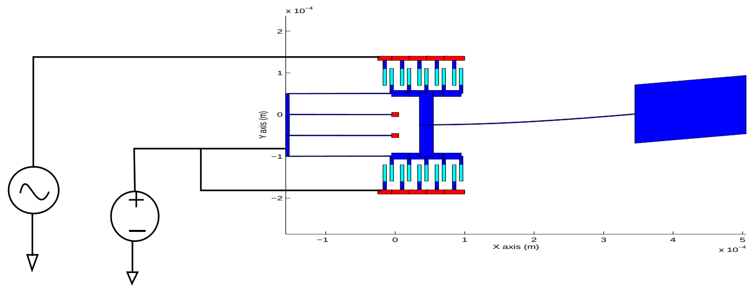


Figure 3 Steady state analysis demo

5. FUNCTION DEFINITIONS

Type ’help function-name’ in Matlab to get on-line on each function. For example, type ’help cho_dc’ in Matlab, It displays detailed information about ’cho_dc’ function.

1) Load netlist

net = cho_load(’netlist file’)

This function reads the netlist input file and process file, set up the data structure for

The COMMENT %

The % symbol may be used in the netlist for remarks and to comment-out elements.

To conclude this section, let's combine all of the above components into a complete netlist. We'll use 3 beams to form a Y-like structure. An anchor will fix the bottom node to the substrate, and a force will act on the top-left node.

```
%netlist example.
BottomAnchor   a3dmf   p1   [bottom A0]   [l=10u  w=10u  h=10u  oy=0 oz=-90 ox=0]
VerticalBeam   b3dm   p1   [bottom middle]   [l=100u  w=4u  h=2u  oy=0 oz=90 ox=0]
LeftBranch   b3dm   p1   [middle topleft]   [l=100u  w=4u  h=2u  oy=0 oz=90+45 ox=0]
RightBranch   b3dm   p1   [middle topright]   [l=100u  w=4u  h=2u  oy=0 oz=45 ox=0]
HorizontalForce   f3dc1   [topright]   [F=50u oy=0 oz=0 ox=0]
```

Once the netlist is created deflection analysis may be performed. See demos.

3. RUNNING THE SOFTWARE

0) For BSAC domain users, from outside Matlab, set MATLAB path as:

```
% setenv MATLABPATH ~cfm/cfm
```

For Non-BSAC domain users, visit the SUGAR webpage, download the free SUGAR program.
URL: http://www-bsac.eecs.berkeley.edu/~cfm

1) Create your netlist input text file with the editor of your choice. See INPUT FORMAT on how to construct netlist file)

2) Run matlab.

3) Before you run the analysis. Try to run demonstrations first.
Type 'demo_canti' to run mechanical analysis demo of cantilever beam in 2d and 3D.
Type ’demo_dc’ to run a DC analysis demo of a beam gap structure.
Type ’demo_pullin’ to run pull-in analysis for this beam gap structure.
Type 'demo_ss' to run steady state analysis demo of a resonator.
Type 'demo_mirror' to run 3D mode analysis demo of a mirror structure.

4) Load your own netlist. Netlist files and process files should be loaded first before you run any other analysis and displaying program.

5) See the FUNCTION DEFINITIONS to run your own analysis.
For example, to run a DC simulation:

```
Load netlist first:   net = cho_load(’netlist’);
DC analysis:          dq = cho_dc(net);
```

where ‘netlist’ is the file name for your netlist file.

model function gets info from process p1 (e.g. poly1 density, Young's modulus, etc). The length of the beam goes from node A to node B. The parameter section provides geometry and orientation in meters and degrees respectively. For geometry, the beam's length,
width, and height (layer thickness) are given by l, w, and h. Postfix u implies e-6. For orientation, rotations oy, oz, and ox are rotations about the y-, z-, and x-axes respectively.

Orientation methodology:
For 3-dimensional rotations, think in terms of the right hand rule where the substrate (computer screen) is the x-y plane. Positive x, y, & z directions are to the left, up, and out respectively. The initial position of the beam is with its first node at the origin and its second node somewhere along positive-x. At this stage, the beam's local and global axes coincide. The beam's width and height dimensions are in-plane and out-of-plane respectively. When rotating the beam into position there are 2 things to remember: (1) For each rotation, the local x-, y- & z- axes rotate with the beam. (2) Sugar does rotations in the order of {y, z, then x}.

To better understand Sugar rotations, mull over the following 3 ordered sets. They're all equivalent rotations, resulting in a beam oriented along the vertical (+)y direction.
(1) oy=0 oz=90 ox=-90,   (2) oy=-90 oz=90 ox=0, and (3) oy=90 oz=90 ox=180.
To see why rotation-order is important, consider the sequence to be {ox,oy,oz}. This would cause the beam to orient along (1) negative-z, (2) positive-y, and (3) negative-y.

The ANCHOR element:

Mechanical systems in Sugar require at least one anchor. An anchor is represented by a short, cantilever beam where node 1 is free to mode and node 2 is fixed to the substrate. WARNING: only use node 1 to connect other beams to. (First-time users often confuse these two nodes).

```
Anchor  a3dmf p1 [UseableNode  SubstrateNode][l=10u w=10u  h=10u  oy=0 oz=180 ox=0 ]
```

This instruction creates a 10 micron-cubed anchor. Since oz = 180, the usable-node ends up on the right side of the structure.

The FORCE / MOMENT element:

The orientation of the force (moment) vector is analogous to the beam example above, where the beam is replaced by a vector with magnitude F (M). Note that ox, twisting the vector along its own axis, has no effect.

```
Force f3dc1 [Jason] [F=6u oy=0 oz=45 ox=0]
```

This says that a force of 6 micro-Newtons is applied, diagonally, at node Jason.

```
Moment f3dc1 [Ningning] [M=10n oy=-90 oz=0 ox=0]
```

This creates a moment of 10e-9 Newton-meters, applied clockwise about the z-axis at node Ningning.

Updated on 10/15/1999

SUGAR VERSION 1.0 MANUAL

(still under construction)

1. INTRODUCTION

SUGAR v1.0 is a collection of MATLAB functions which use nodal analysis to simulate the performance of some types of planar MEMS devices. The user provides a text input file called a netlist, which describes the geometry and connectivity of the system, then calls one or more analysis routines. The numerical results are returned as output parameters for additional MATLAB processing if desired. The graphical results can also be displayed. Currently the DC analysis, steady state analysis, mode analysis and transient analysis have been implemented.

2. NETLIST PRIMER

Basically, the netlist is your MEMS device. It tells Sugar how it's constructed and how it's to be excited. In the netlist text file, each line of instruction (called an element) has the following format:

NAME   MODELFUNCTION   LAYER   [NODES]   [PARAMETERS]

NAME
This optional field allows you to associate a name to the element.

MODELFUNCTION
This is the name of the function that models the element. See the appendix for a list of available ModelFunctions.

LAYER
This field is only required if the ModelFunction needs any ProcessFile info.

[NODES]
Connectivity is given by an ordered set of node names. E.g. [node1 node2].

[PARAMETERS]
This is a set of model parameters & values. E.g. [param=value]. Any parameter specified here can overwrite the same parameter specified in process file.

Let’s examine a few elements and then form a complete neltist.

The BEAM element:

Cantilever b3dm p1 [A B] [l=100u w=2u h=2u oy=0 oz=0 ox=0]

This says that an element (optionally named Cantilever) is modeled by the b3dm function. The