Proj 3: MEMS Madness

1 Introduction

Micro-Electro-Mechanical Systems (MEMS) are tiny coupled-physics systems that are often built using the same basic processes one uses to make integrated circuits. At characteristic length scales on the order of microns (millionths of a meter), the physics of these devices are still predominantly classical, but the dominant forces might be different than what you would expect. For example, electrostatic attraction can be an incredibly strong force at these scales, and is often used to drive motion.

You are provided with a small finite element code to simulate the displacement of a cantilever beam suspended over an electrode; the geometry is shown in Figure 1. The beam and the electrode are at different voltages, and so attract each other; but elastic forces resist that attraction. The basic force balance equation is

$$Ku = f_e(u, V),$$

where K is a positive definite stiffness matrix and f_e is the electrostatic force for a given displacement and voltage.

The electrostatic forces tend to pull the beam toward the electrode, and scale like with $(g+u)^2$, where g is the initial gap and u is the displacement. The elastic restoring forces, in contrast, scale linearly with u. Hence, at some critical voltage, the electrostatic forces become so strong that the elastic forces cannot balance them, and the system becomes unstable. At this point, the beam is "pulled in" to the electrode underneath.

2 The approaches

2.1 Fixed point iteration

The `gap_solve_fpV` code solves for the deflection in the model problem using the fixed point iteration

$$Ku^{(k+1)} = f_e(u^{(k)}; V)$$

where f_e denotes the electrostatic force vector. The driver `gap_demo` shows how this code is used.

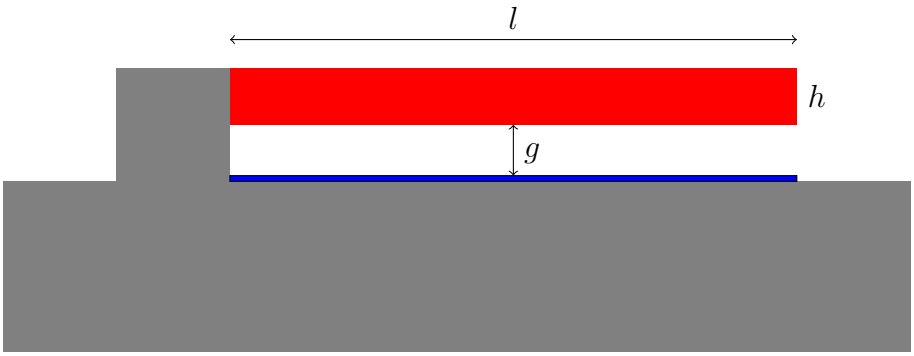

Figure 1: Cantilever beam gap-closing actuator. A potential difference between the beam (red) and an electrode along the substrate (blue) causes the beam to be pulled down.

1. Plot the residual error versus iteration count. Does this look like a linearly convergent iteration?

2. Write a new script, `gap_continuer0` that uses this fixed point iteration to solve the equation for voltages starting at $V = 0$ and going up to $V = 10$ (past the pull-in voltage). Use a continuation strategy in which the initial guess for u at each voltage is the equilbrium solution for the previous voltage (much like in PS7). Plot the tip displacement versus V for your computations. What is the largest voltage you can manage before the solver diverges?

2.2 Newton iteration: Voltage control

The `gap_solve_fpV` code converges slowly for larger voltages, and eventually diverges as we approach pull-in. However, the `gap_force` routine computes both the electrostatic force $f_e(u; V)$, but also the Jacobian $\partial f_e / \partial u$[1]. Therefore, we can code a Newton iteration.

1. Complete the routine `gap_solve_NV` to compute the displacement at a fixed voltage using a Newton iteration. Check that the rate of convergence is quadratic!

[1]Like K, this Jacobian is a symmetric matrix.

2. Write a script, `gap_continuer1` that uses Newton iteration to solve the equation for varying voltages, similar to `gap_continuer0`. What is the largest voltage you can manage before the solver fails to adequately converge?

2.3 Newton iteration: Displacement control

So far, we have considered methods in which we control V and compute the corresponding u. We now consider control of one component of u (the tip displacement) and computation of the corresponding V along with the rest of u. That is, we want to solve

$$F(u, V; d) = \begin{bmatrix} Ku - f_e(u; V) \\ e_{tip}^T u - d \end{bmatrix} = 0$$

where e_{tip} is a vector that indicates the controlled degree of freedom (the displacement at the tip).

1. Complete the routine `gap_solve_Nd` to compute the displacement and voltage corresponding to a fixed tip displacement using a Newton iteration. I recommend using the displacement vector u and the *squared* voltage V^2 as your unknowns for the purpose of Newton iteration; otherwise, you will have a singular Jacobian at zero voltage and displacement.

2. Write a script, `gap_continuer2` that computes the state for tip displacements between zero and 80% of the gap. Use Newton iteration to solve the equation for varying displacements. Your script should produce a plot of tip displacement vs voltage for the full range of tip displacements considered.

2.4 Finding pull-in

The `gap_continuer2` code should give a pretty clear notion of the critical voltage (pull-in). In particular, the pull-in state is the point along the solution curve at which the Jacobian becomes singular. For the "top" part of the curve of deflection vs voltage, the equilibrium is stable, and the Jacobian matrix is positive definite; for the "bottom" part of the curve, the Jacobian is indefinite.; and the pull-in state is the transition between these two. It is

possible to detect whether the Jacobian matrix is positive definite or not by attempting to run Cholesky factorization; that is,

```
[R,p] = chol(J);
if p
  disp('Indefinite');
else
  disp('Positive definite');
end
```

Using this test, it is possible to find the tip displacement associated with pull-in (and the associated voltage) by running bisection on the nonsingularity.

1. Complete the function `gap_pullin` to find the pull-in voltage. You may use the bisection strategy above, or you may do something different, but do make sure to sanity check your code.

2. Write a script `gap_pull_sweep` to compute the pull-in voltage for beam lengths between 20 and 100 microns, and show the results on a plot.

3 Notes

1. This project is an example of *numerical bifurcation analysis*, an area often associated with continuation methods.

2. This model should not be taken all *that* seriously. It's a good first cut, but the model we use for the electrostatic field is a "parallel plate" approximation where we assume the potential varies linearly between the substrate and the beam.

3. It is possible to directly write down a system of equations that describes the pull-in state:

$$\begin{aligned} Ku - f_e(u,V) &= 0 \\ \left(K - \frac{\partial f_e}{\partial u}(u,V)\right) w &= 0 \\ c^T w - 1 &= 0 \end{aligned}$$

   where c can be chosen as a random vector. If you want, you can `gap_pullin` based on this augmented system rather than using the bisection technique recommended above.