## Assignment for Week 3 readings: (due Tuesday 2 November before class)

Anderson Nardin, Luca Arleo, Franceso Piqué and Mark Cutkosky (contact us at [#week3 on Slack](https://topicsinmulti-b5g1638.slack.com/archives/C02HQL6EVDL))

Goal: get everybody familiar with grasp stiffness analysis. Thirty years ago, it was not practical to do dynamic simulations of hands and grasps. Today with dynamics engines like [MuJoCo](https://mujoco.org/) etc. and packages like [SimGrasp](https://bitbucket.org/shiquan/sim-grasp/src/master/), it is entirely possible. These packages work well for systems with many rigid and/or compliant bodies with frictional contacts.

*So, today, what is the value of grasp stiffness analysis?* Probably the main value is to understand, theoretically, how grasp stiffness predicts grasp behavior to perturbations. For example, why do rounded fingers -- even if they are fairly hard and have small contact areas -- stabilize a grasp due to rolling? And why do compliant fingertips further stabilize the grasp? And which degrees of freedom do they affect the most? Grasp stiffness (the grasp stiffness matrix) also appears when doing [Impedance Control in the Operational Space](http://bdml.stanford.edu/oldweb/touch/publications/griffin_thesis.pdf) for a grasped object.

There is a lot of notation in this assignment. To make it easier we have built analysis scripts in Sympy for the left finger, which you can easily convert to symbolic Matlab if you prefer. In either case, we give you the left finger and you need to add the right finger. If you choose to use Python you may want to install [Anaconda](https://www.anaconda.com/) (3.x) which will come with Sympy and the Spyder IDE for a very Matlab-like experience. You want to go through the scripts a few lines at a time, checking intermediate results to confirm they make sense.

We also recommend making a couple of small XYZ coordinate frames for yourself and using them to help keep track of transformations between body, contact and fingertip coordinate systems.

* [Link to the slides](https://docs.google.com/presentation/d/1D1n_Oiv2RsspM-v06nqhCaCeuVj0nBBIOfPQuJxNBX0/edit?usp=sharing)

### Q1

Example 2 of the [Cutkosky & Kao (1989) IEEE Transactions](https://ieeexplore-ieee-org.stanford.idm.oclc.org/abstract/document/88036) paper shows an example with two 3-joint fingers holding a rectangular block. We will use this as a starting point. It is recommended to start with the scripts in the Week 3 Googledrive folder. Load the script **Kb\_left-finger.py** and modify it to add the right finger so you can compute the total “direct” grasp stiffness that arises from the joint stiffnesses (ka, kb, kc). You should get a result that matches [Kb] in eq (29) in the paper, where we use “w” for the half-width of the object instead of “r”. Along the way, you’ll get results that match some stuff in Appendix II, B Example 2; however, note thathas a typo: and are 0 instead of 1. (The python script is correct.)

### Q2

Now make a modified version of this script that accounts for “soft fingers.” We assume a small rubber block at the fingertip. The way the numbers work, it turns out that rotations about the local (l,m) axes in the tip coordinate frame are still freely possible (i.e. no bending moments transmitted), but now torsional moments about the local n axis can be transmitted through the contact. So we have Salisbury’s “soft finger” contact model. Modify the [H] matrix accordingly. (Hint, look at the [H] matrix above equation (2) in the paper).

It will also turn out that compliance (1/stiffness) in the rotational degrees of freedom will be much more significant than compliance in pure compression or shear. So we can assume a 6x6 local structural compliance matrix, Ctip, at each fingertip where *all the elements are zeros except for last 3 elements along the diagonal, which have a compliance of 1/kq.*

Compute the total direct grasp stiffness matrix for this new case. To simplify the algebra a bit more, let’s assume that kb=kc=ka for the joint stiffnesses.

### Q3

Now we consider the possibly destabilizing geometric effect, which produces the matrix [Kj]. Load the script **Kj\_left-finger.py** and modify it to add the right finger. Due to symmetry some terms will cancel out, and you should get a result for [Kj] that matches the one in equation (30) where we use “w” for the half-width of the object instead of “r”. Again, you’ll see intermediate results that match things in Appendix II, B Example 2. However, what the paper calls deltaAT and deltaRT in the Appendix are actually not the transposes, but deltaA and deltaR. Again, the python script is correct.

### Q4

Finally, let’s consider that we have hemispherical fingertips with a radius of curvature, r. You will need to modify the differential jacobian to account for translation of the contact patch on the object by amounts equal to r\*dtheta where dtheta is the appropriate *relative* rolling angle about the local *l* and *m* axes. The final result should be intuitively satisfying. (Contact us if it’s not.) You can consider that the fingertips are hard, with small contact areas so that we don’t need to use the ‘soft contact’ H matrix.

* *What is the critical value of r for which the bottom right term of [Kj] is no longer negative?*