15. Manipulator Control

Last Lecture: Force Control

- Some tasks more naturally specified with force
  - Programming the interaction (stiffness & damping)
  - High level stiffness(impedance) controller is simple, but it is able to do something complicated like filp a box at corner. It will break the friction cone itself.
  - If you are doing learning control, you might choose these kind of representations in the output. Like you have a 100 billion parameter transformer and it only runs at a few hertz, but interaction is extremely fast. So the command coming from high level controller could be slow and simple but the resulting behavior is still beautiful and good. An okay bandwidth argument, enough to get it done.
- Assume robot is a point finger. Now in this lecture, put the whole robot back.

Today's Topic: Manipulator Control

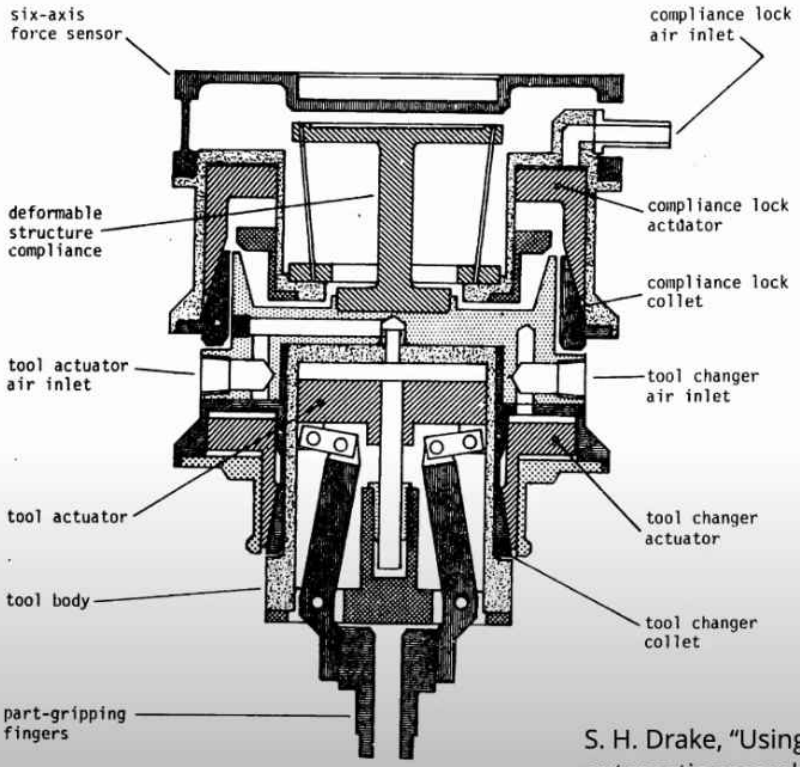
Forget about the environment, just think about how we can control robot extremely well.

- Zoo of controllers
  - PidController: $u = K_p(q_d - q) + K_d(\dot{q}_d - \dot{q})$
  - Stiffness (+Damping) Controller:
    - $u = PD - \tau_g$, gravity compensation
    - $m\ddot{q} + K_p(q - q_d) + K_d(\dot{q} - \dot{q}_d) = f_{ext}^F$
  - Force Control
    - assume $\ddot{q} = 0$, $u = -\tau_g - f_{desired}^F$
    - so $f_{ext}^F = f_{desired}^F$
  - InverseDynamicController
    - Think about trajectory tracking
    - $u = -\tau_g + m[\ddot{q}_d + K_p(q_d - q) + K_d(\dot{q}_d - q)]$
    - $\tilde{q} = q_d - q$; $\dot{\tilde{q}} = \dot{q}_d - \dot{q}$; $\ddot{\tilde{q}} = \ddot{q}_d - \ddot{q}$
    - $\ddot{\tilde{q}} + K_d\dot{\tilde{q}} + K_p\tilde{q} = 0$, error dynamics looks like spring-damper

- JointStiffnessController
- SpatialForceController
  - act like a spring at end-effector, programmable stiffness, super useful
  - Writing the dynamics of the robot in the EE coordinates, will be some projection and nullspace of course.
  - Key idea: program the interaction at a point
    - P^E ~postion of EE
    - $f_{ext}^{B_E}$:external force applied to robot body B at pos E
    - $\tau_{ext} = J^{T_E}(q) f_{ext}^{B_E}$, Principle of virtual work, d'Alembert
    - what if pushing in the middle of the robot arm? if you dont know the contact location, joint stiffness might be a more natural solution
- SpatialStiffnessController
- $P^E = f_{kin}^E(q)$
- $\dot{P}^E = V^E = J^E(q)\dot{q}$
- $\ddot{P}^E = a^E = J^E(q)\ddot{q} + \dot{J}^E\dot{q}$, ask drake or other software package
- $\ddot{q} = M^{-1}(q)[\tau_g(q) + u - C(q,\dot{q})\dot{q}]$, replacing $\ddot{q}$ above
- we get: $M_E(q)\ddot{p}^E + C(q,\dot{q})\dot{q} = f_g^{B_E}(q) + f_u^{B_E} + f_{ext}^{B_E}$, M_E end effector inertia matrix
- In most cases, external forces are unknown. Treat it as disturbance. If you can measure it, u can add it in a feedforward way
- The essential point is, the dynamics of the system written in the effector coordinates look heck a lot like the dynamics of the original system in the full multibody coordinates. We can do exactly the SAME control approaches. Like setting $f_u^{B_E}$ in PD, we get end effector PD controller. If cancel out gravity torque, we get end effector stiffness controller, etc.
- Dealing with multiple tasks, prioritized tasks and nullspace control.
- Operational space control(OSC)

- Trajectory Tracking
  - Assume $f_{ext}^F = 0$
- Super important: using soft controllers to execute desired motion plans and when things go wrong it has a softer response. Cancel more terms, you can get softer.
- Core idea: make complex robot acting simple, virtual spring at end effector.

- Hardware approaches, remote centered compliance. Peg insertion, peg & hole etc. Mechanical complicance. Instead of programming the force and moment the stiffness kind at the end effector, they put force and rotation to be at the end of the tool of the tip of the insertion.



S. H. Drake, "Using compliance in lieu of sensory feedback for automatic assembly.," Massachusetts Institute of Technology, 1978.

Student Questions

1. In the point robot case, everything is nice and convex. In manipulator case, if q is fixed, everything else is convex. So if you have a path you can do like time parameters, time optimization in a convex way. Many things can be formulated in that way. But in general case we tend to manipulate the nonlinear equations, in order to do gravity compensation.
2. If you were to have a desired path, like linearizing the equations, making a convex approximation in the vicinity of the trajectory can be a useful thing to do. We actually in more complicated systems where we donot have closed form controllers we fall back to doing that. In this case, we do not need to do that because the equations are so structured and I have enough control authority that I can take exactly the right non convex corrective terms.
3. How to eliminate the steady state error? adding integral term or gravity compensation which one is better?
    - adding I term is dangerous
    - adding gravity compensation to reduce KP gains, soft controller but gets good tracking

- In simulation, inverse dynamics controller is preferred over stiffness cuz tuning lower PD gains and set longer timestep, because my equations are stiff

4. About multiple tasks
    - same as diff-IK, if you dont specify completely, it will pick one for you(depending on pinv). You are not going to satisfy the solution most of the time(elbow swing around). So give yourself at least the lowest priority do sth like joint centering and sth guaranteed to be full rank. Don't leave it empty.