16-299 Lecture 8:
Part 1: Controlling in different coordinate systems
Part 2: Optimal control

Independent Joint Control

In this approach each joint is controlled independently, typically with some form of PID control. This approach is common in heavily geared robots. With a gear ratio of R, mass on the other side of the transmission is effectively reduced by R^2 relative to the motor inertia. Force is amplified by R, and then acceleration is amplified by R. On current electric humanoids R is 100-200. This means the rest of the robot, and whatever load it is carrying or force it is applying, does not matter much relative to the motor inertia on that joint.

Inverse dynamics can be used to generate feedforward commands in advance of a movement.

$$\tau_{\text{ff}}(t) = \mathbf{ID}(\mathbf{q}_d(t), \dot{\mathbf{q}}_d(t), \ddot{\mathbf{q}}_d(t)) \tag{1}$$

The d subscript indicates this is a desired value. Often the inverse dynamics model is simplified to only compensate for gravitational forces (gravity compensation).

The rest of this writeup focuses on the control of robots with rotary joints. Robots with all prismatic joints typically have less dynamic coupling between joints.

Computed Torque Control

In computed torque control an inverse dynamics model

$$\tau = \mathbf{ID}(\mathbf{q}_d(t), \dot{\mathbf{q}}_d(t), \ddot{\mathbf{q}}_d(t) + \ddot{\mathbf{q}}_{\mathrm{fb}}) \tag{2}$$

generates the expected torques necessary to drive the robot along a desired trajectory. Corrective accelerations $\ddot{\mathbf{q}}_{\mathrm{fb}}$ are generated by some controller, typically independent joint PID controllers that output accelerations rather than torques.

$$\ddot{\mathbf{q}}_{\mathrm{fb}} = \mathbf{PID}(\mathbf{q}, \mathbf{q}_d, \dot{\mathbf{q}}, \dot{\mathbf{q}}_d) \tag{3}$$

Each of the planned positions $\mathbf{q}_d(t)$ and velocities $\dot{\mathbf{q}}_d(t)$ can be replaced by the actual positions $\mathbf{q}$ and velocities $\dot{\mathbf{q}}$ in Equation 2. Reasons not to do this include avoiding the complexity of taking into account feedback through the inverse dynamics model, and noise added to the desired velocities when the actual positions are numerically differentiated.

The Jacobian

The Jacobian of the robot's forward kinematics plays an important role in controlling the robot in different coordinate systems. The robot's forward kinematics predicts end effector location based on joint angles:

$$\mathbf{x} = \mathbf{FK}(\mathbf{q}) \tag{4}$$

The derivative of the forward kinematics is the Jacobian matrix for the robot:

$$\dot{\mathbf{x}} = \frac{\partial \mathbf{FK}(\mathbf{q})}{\partial \mathbf{q}} \dot{\mathbf{q}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{5}$$

Multiplying both sides by a small amount of time leads to a relationship between displacements in end effector and joint coordinates:

$$\Delta \mathbf{x} = \mathbf{J}(\mathbf{q})\Delta \mathbf{q} \tag{6}$$

Power (the product of force and velocity) is the same no matter what coordinate system it is calculated in, so:

$$\begin{aligned} \dot{\mathbf{x}}^{\mathrm{T}}\mathbf{f} &= \dot{\mathbf{q}}^{\mathrm{T}}\tau \\ (\mathbf{J}(\mathbf{q})\dot{\mathbf{q}})^{\mathrm{T}}\mathbf{f} &= \dot{\mathbf{q}}^{\mathrm{T}}\tau \\ \dot{\mathbf{q}}^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}(\mathbf{q})\mathbf{f} &= \dot{\mathbf{q}}^{\mathrm{T}}\tau \\ \mathbf{J}^{\mathrm{T}}(\mathbf{q})\mathbf{f} &= \tau \end{aligned} \tag{7}$$

At singularities $\mathbf{J}(\mathbf{q})$ is not full rank. This means that joint motion in some direction results in no motion in some direction in end effector space. For a two joint arm, what postures are singular and what directions can the robot not move in at those postures? Singularities also mean that force in some direction does not result in joint torques (or motion).

What happens if the robot has more joints than end effector degrees of freedom (usually this is when a robot has more than six joints)? In this situation the robot has redundant kinematics, in that more than one set of joint angles attains the same end effector position.

The relationship for accelerations is:

$$\begin{aligned}
\frac{d\dot{\mathbf{x}}}{dt} &= \frac{d\mathbf{J}(\mathbf{q})\dot{\mathbf{q}}}{dt} \\
\ddot{\mathbf{x}} &= \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} \\
\ddot{\mathbf{x}} &= \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^{\mathrm{T}}\frac{\partial \mathbf{J}(\mathbf{q})}{\partial \mathbf{q}}\dot{\mathbf{q}}
\end{aligned} \tag{8}$$

The $\dot{\mathbf{q}}^{\mathrm{T}}(\partial \mathbf{J}(\mathbf{q})/\partial \mathbf{q})\dot{\mathbf{q}}$ term is the source of Coriolis and centripetal forces, which are quadratic in velocity.

Transforming Stiffness and Damping

What is the relationship between stiffness in joint coordinates ($\Delta\tau = \mathbf{K_q}\Delta\mathbf{q}$) and stiffness in end effector coordinates ($\Delta\mathbf{f} = \mathbf{K_x}\Delta\mathbf{x}$)?

$$\begin{aligned}
\Delta\mathbf{f} &= \mathbf{K_x}\Delta\mathbf{x} \\
\Delta\mathbf{f} &= \mathbf{K_x}\mathbf{J}(\mathbf{q})\Delta\mathbf{q} \\
\mathbf{J}^{\mathrm{T}}(\mathbf{q})\Delta\mathbf{f} &= \mathbf{J}^{\mathrm{T}}(\mathbf{q})\mathbf{K_x}\mathbf{J}(\mathbf{q})\Delta\mathbf{q} \\
\Delta\tau &= \mathbf{J}^{\mathrm{T}}(\mathbf{q})\mathbf{K_x}\mathbf{J}(\mathbf{q})\Delta\mathbf{q} \\
\mathbf{K_q} &= \mathbf{J}^{\mathrm{T}}(\mathbf{q})\mathbf{K_x}\mathbf{J}(\mathbf{q})
\end{aligned} \tag{9}$$

The same transformation applies to damping:

$$\begin{aligned}
\mathbf{f} &= \mathbf{B_x}\dot{\mathbf{x}} \\
\mathbf{f} &= \mathbf{B_x}\mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \\
\mathbf{J}^{\mathrm{T}}(\mathbf{q})\mathbf{f} &= \mathbf{J}^{\mathrm{T}}(\mathbf{q})\mathbf{B_x}\mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \\
\tau &= \mathbf{J}^{\mathrm{T}}(\mathbf{q})\mathbf{B_x}\mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \\
\mathbf{B_q} &= \mathbf{J}^{\mathrm{T}}(\mathbf{q})\mathbf{B_x}\mathbf{J}(\mathbf{q})
\end{aligned} \tag{10}$$

Note that one can transform a controller in end effector coordinates to joint coordinates in a straightforward way. However, specifying a controller in joint coordinates and asking what end effector controller that corresponds to requires inverting the Jacobian matrix.

What happens at singularities where $\mathbf{J}(\mathbf{q})$ is not full rank?

What happens when the robot is redundant?

Operational Space Control

Suppose we wanted to control in end effector coordinates $\mathbf{x}$ with $\mathbf{K_x}$ and $\mathbf{B_x}$. We could calculate the corresponding $\mathbf{K_q}(\mathbf{q})$ and $\mathbf{B_q}(\mathbf{q})$ which would vary as $\mathbf{q}$ changed. This approach is well behaved both when the posture is singular and when the robot has more than six joints (it is redundant).

Another approach is to do inverse kinematics (actually inverting the Jacobian) as part of the control loop. The goal is to implement arbitrary dynamics in end effector (task) space. This means entirely cancelling the robot dynamics. For free motion, we would specify accelerations as the output of the end effector dynamics, as there are no contact forces: $\ddot{\mathbf{x}}_d = \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}})$. The robot end effector could kinematically behave like another system after a perturbation or after starting from some initial state. The acceleration could be transformed into joint coordinates, and then a computed torque controller could be applied to exactly achieve it:

$$\ddot{\mathbf{q}}_d = \mathbf{J}^{-1}(\mathbf{q})\ddot{\mathbf{x}}_d - \mathbf{J}^{-1}(\mathbf{q})(\dot{\mathbf{q}}^{\mathrm{T}}\frac{\partial \mathbf{J}(\mathbf{q})}{\partial \mathbf{q}}\dot{\mathbf{q}}) \tag{11}$$

This is a form of "virtual model" control, where the goal is to make the robot behave as some other physical system. This version makes the most sense if the robot is not in contact with anything.

If the robot is in contact with something at the end effector, then the goal of the controller can be to generate end effector forces. This is another form of "virtual model" control Someone grabbing a handle at the end effector and moving it would be fooled into thinking they were in contact with the desired physical system and not be able to tell that a robot was attached (this is a goal of haptic interfaces). In this case $\mathbf{f}_d = \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}})$ and the controller would both cancel the robot dynamics and generate the desired force:

$$\tau = \mathbf{J}^{\mathrm{T}}(\mathbf{q})\mathbf{f}_d + \mathbf{ID}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \tag{12}$$

Unless one was trying to implement a haptic interface with a very light and weak robot that a human could easily overpower, I would not try to fully cancel the mass of the robot using acceleration feedback. Although the "mass matrix" $\mathbf{M}(\mathbf{q})$ is always square and full rank (there is always mass in all directions) and thus invertable, a clean non-delayed acceleration measurement is very hard to come by. Twice differentiating position is noisy and filtering that estimate leads to lag

which causes instabilities when trying to cancel mass. An accelerometer is just a local measurement of acceleration and given everything vibrates will be very sensitive to structural vibration. Gravity compensation is common, since it only depends on position. Coriolis and centripetal force cancellation is less common. Power assist, where the user applies a force to the end effector, and that force is multiplied ($\boldsymbol{\tau} = \mathrm{Gain} * \mathbf{J}^{\mathrm{T}}(\mathbf{q})\mathbf{f}_{\mathrm{user}}$) , is common on exoskeletons. one can add additional damping for safety. However, most current exoskeletons are exhausting to use for any length of time, so adding additional drag may not be possible.

SHOW SARCOS VIDEOS

`https://www.youtube.com/watch?v=XpiCVjpPb_8`

`https://www.youtube.com/watch?v=Ac5cowTwOPw`

Optimal Control

The above approaches to control assumed it was easy to cancel the plant dynamics. Usually, there are unmodelled dynamics that take a lot of work to model and cancel (actuator dynamics, structural vibrations, friction, ...). Also, cancelling the plant dynamics is usually energetically wasteful. Consider a glider. It glides. Now add actuation to eliminate the glider dynamics (which adds a lot of weight for actuators and fuel). Now try to make it fly like a glider (Think space shuttle).

In legged locomotion, the idea of building the behavior that you want into the mechanics is now very popular, because of a demonstration of passive dynamic walking, where robots could walk with no sensing, computing, or actuation.

`https://www.youtube.com/watch?v=WOPED7I5Lac`
`https://www.youtube.com/watch?v=FfKQSUhYjlY`
`https://www.youtube.com/watch?v=IvtFOxd4NvY`
`https://www.dailymail.co.uk/news/article-6437723/Step-ladder-`
`html`

A more general philosophy is "Instead of trying to cancel the plant, take advantage of its dynamics. Do the least actuation you can do to get what you want. Gentle nudges, not massive motors or ginormous jets."

Optimal control provides a design paradigm to simultaneously design mechanics and behaviors to maximize performance while minimizing actuation and control. In terms of the controllers above, one can specify matching desired dynamics as a goal while penalizing large actuation:

$$l(\text{state}, \text{action}) = |(\mathbf{f}_{\text{contact}} - \mathbf{f}_d)|^2 + |\boldsymbol{\tau}|^2 \tag{13}$$

where $l()$ is the one step cost, and

$$\mathbf{f}_d = \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}) \tag{14}$$

The total cost is the sum of the one step costs over all the time steps.

Optimization techniques

Optimizing functions. Examples on class web page (kin3).

Optimize policy parameters by executing behavior for a while (basically treating behavior optimization as function optimization).

`http://www.dgp.toronto.edu/~jmwang/`

Trajectory optimization: u(t) or u(x) LQR over time. Differential Dynamic Programming (DDP). Many other methods. We will talk more about this in the nonlinear control section of the course.

Model-based - all the stuff we talk about

Model-free - directly optimize on the robot - exoskeleton example

`https://www.youtube.com/watch?v=Dkp_xjbRYCo`

Simulation-based = model-based