Multi-Joint Control

Spring 2020

Hasan Poonawala

Previously on . . .

- Plan trajectory $q_d(t)$ for robot configuration q

Previously on . . .

- Plan trajectory $q_d(t)$ for robot configuration q
- Goal: Trajectory Tracking
  Choose torques τ (or motor voltages u) so that

$$q(t) \rightarrow q_d(t)$$

Previously on . . .

- Plan trajectory $q_d(t)$ for robot configuration q
- Goal: Trajectory Tracking
  Choose torques τ (or motor voltages u) so that

$$q(t) \to q_d(t)$$

- When

$$q_d(t) \equiv q_d,$$

  a constant, we get set-point regulation or goal-reaching task

Approaches to Trajectory Tracking

- Independent Joint Control.

Approaches to Trajectory Tracking

- Independent Joint Control.
  - Some robots allow us to get away with controlling each joint individually

Approaches to Trajectory Tracking

- Independent Joint Control.
  - Some robots allow us to get away with controlling each joint individually
  - Use PID controllers and frequency-domain analysis

Approaches to Trajectory Tracking

- Independent Joint Control.
  - Some robots allow us to get away with controlling each joint individually
  - Use PID controllers and frequency-domain analysis
  - Works for set-point regulation / slow trajectories

Approaches to Trajectory Tracking

- Independent Joint Control.
  - Some robots allow us to get away with controlling each joint individually
  - Use PID controllers and frequency-domain analysis
  - Works for set-point regulation / slow trajectories
- Model-based Control

Approaches to Trajectory Tracking

- Independent Joint Control.
  - Some robots allow us to get away with controlling each joint individually
  - Use PID controllers and frequency-domain analysis
  - Works for set-point regulation / slow trajectories
- Model-based Control
  - Use model

$$M(q(t))\ddot{q}(t) + C(q(t), \dot{q}(t))\dot{q}(t) + G(q(t)) = u(t)$$

Approaches to Trajectory Tracking

- Independent Joint Control.
  - Some robots allow us to get away with controlling each joint individually
  - Use PID controllers and frequency-domain analysis
  - Works for set-point regulation / slow trajectories
- Model-based Control
  - Use model

$$M(q(t))\ddot{q}(t) + C(q(t), \dot{q}(t))\dot{q}(t) + G(q(t)) = u(t)$$

  - Lyapunov-based analysis and design

Gravity-free PD Control

When: Want to regulate the robot config to a set-point q_d

Gravity-free PD Control

When: Want to regulate the robot config to a set-point q_d

Assuming no gravity (or that we canceled it out using u):

$$M(q(t))\ddot{q}(t) + C(q(t), \dot{q}(t))\dot{q}(t) = u(t)$$

Gravity-free PD Control

When: Want to regulate the robot config to a set-point q_d

Assuming no gravity (or that we canceled it out using u):

$$M(q(t))\ddot{q}(t) + C(q(t), \dot{q}(t))\dot{q}(t) = u(t)$$

Use PD control:

$$u(t) = K_P(q_d - q(t)) - K_D(\dot{q}(t))$$

Gravity-free PD Control

When: Want to regulate the robot config to a set-point q_d

Assuming no gravity (or that we canceled it out using u):

$$M(q(t))\ddot{q}(t) + C(q(t), \dot{q}(t))\dot{q}(t) = u(t)$$

Use PD control:

$$u(t) = K_P(q_d - q(t)) - K_D(\dot{q}(t))$$

Closed-loop:

$$M(q(t))\ddot{q}(t) + C(q(t), \dot{q}(t))\dot{q}(t) = K_P(q_d - q(t)) - K_D\dot{q}(t)$$

$$\implies \ddot{q}(t) = M^{-1}(q(t))\left(-C(q(t), \dot{q}(t))\dot{q}(t) + K_P(q_d - q(t)) - K_D\dot{q}(t)\right)$$

dropping t , $\ddot{q} = M^{-1}(q)\left(-C(q, \dot{q})\dot{q} + K_P(q_d - q) - K_D\dot{q}\right)$

Analysis

$$\ddot{q} = M^{-1}(q)\left(-C(q,\dot{q})\dot{q} + K_P(q_d - q) - K_D(\dot{q})\right)$$

Equilibrium occurs when $\dot{q} = \ddot{q} = 0 \implies q_{eq} = q_d$.

Analysis

$$\ddot{q} = M^{-1}(q)\left(-C(q,\dot{q})\dot{q} + K_P(q_d - q) - K_D(\dot{q})\right)$$

Equilibrium occurs when $\dot{q} = \ddot{q} = 0 \implies q_{eq} = q_d$.

We want $q \to q_d$, or asymptotic stability of equilibrium q_d

Analysis

$$\ddot{q} = M^{-1}(q)\left(-C(q,\dot{q})\dot{q} + K_P(q_d - q) - K_D(\dot{q})\right)$$

Equilibrium occurs when $\dot{q} = \ddot{q} = 0 \implies q_{eq} = q_d$.

We want $q \to q_d$, or asymptotic stability of equilibrium q_d

Can't use methods for linear systems, simulation of all cases is infeasible.

Analysis

$$\ddot{q} = M^{-1}(q)\left(-C(q,\dot{q})\dot{q} + K_P(q_d - q) - K_D(\dot{q})\right)$$

Equilibrium occurs when $\dot{q} = \ddot{q} = 0 \implies q_{eq} = q_d$.

We want $q \to q_d$, or asymptotic stability of equilibrium q_d

Can't use methods for linear systems, simulation of all cases is infeasible.

Solution: Lyapunov methods

Lyapunov Function

For this mechanical system, we choose

$V(x) =$ actual Kinetic Energy$+$ Virtual Potential Energy due to error

Lyapunov Function

For this mechanical system, we choose

$V(x) =$ actual Kinetic Energy$+$ Virtual Potential Energy due to error

$$V(x) = V(q, \dot{q}) = \frac{1}{2}\dot{q}^T M(q)\dot{q} + \frac{1}{2}(q - q_d)^T K_P(q - q_d)$$

Potential is spring-like with spring constant K_P.

Lyapunov Function

For this mechanical system, we choose

$V(x) =$ actual Kinetic Energy$+$ Virtual Potential Energy due to error

$$V(x) = V(q, \dot{q}) = \frac{1}{2}\dot{q}^T M(q)\dot{q} + \frac{1}{2}(q - q_d)^T K_P(q - q_d)$$

Potential is spring-like with spring constant K_P.

Is this a proper candidate Lyapunov function?

- Need $K_P > 0$, $M(q) > 0$ (positive definite)

$M(q) > 0$ is true for any valid Euler-Lagrangian mechanical system!

Directional Derivative of Lyapunov Function

$$V(x) = V(q, \dot{q}) = \frac{1}{2}\dot{q}^T M(q)\dot{q} + \frac{1}{2}(q - q_d)^T K_P (q - q_d)$$

How does $V(x)$ change along solutions $\bar{x}(t)$?

$$\dot{V}(t) = \frac{\partial V}{\partial x}\dot{x}$$

$$= \dot{q}^T M(q)\ddot{q} + \frac{1}{2}\dot{q}^T \dot{M}(q)\dot{q} + (q - q_d)^T K_P \dot{q}$$

Next: substitute for $\ddot{q}$

$$\ddot{q} = M^{-1}(q)\left(-C(q,\dot{q})\dot{q} + K_P(q_d - q) - K_D\dot{q}\right)$$

$$\dot{V}(t) = \dot{q}^T M(q)\ddot{q} + \frac{1}{2}\dot{q}^T \dot{M}(q)\dot{q} + (q - q_d)^T K_P \dot{q} \quad (1)$$

$$= \dot{q}^T M(q)\left(M^{-1}(q)\left(-C(q,\dot{q})\dot{q} + K_P(q_d - q) - K_D\dot{q}\right)\right) \quad (2)$$
$$+ \frac{1}{2}\dot{q}^T \dot{M}(q)\dot{q} + (q - q_d)^T K_P \dot{q}$$

The mass-matrix terms cancel, so does the term involving K_P.
Exercise: confirm that you get from the equation above to:

$$\dot{V}(t) = \frac{1}{2}\dot{q}^T\left(\dot{M}(q) - 2C(q,\dot{q})\right)\dot{q} - \dot{q}^T K_D \dot{q}$$

Skew Symmetry Property

$$\dot{V}(t) = -\dot{q}^T K_D \dot{q},$$

because for any EL-system, $\dot{M}(q) - 2C(q, \dot{q})$ is a skew-symmetric matrix!
(See Section 5.2.1 in 07_Manipulator_Kinematics_Dynamics.pdf)

So, if $\dot{q} \neq 0$, then $\dot{V} < 0$.
To apply Lyapunov's conclusions, we actually want $q \to q_d$ is that when $q \neq q_d, \dot{q} \neq 0$, THEN $\dot{V} < 0$.

Skew Symmetry Property

$$\dot{V}(t) = -\dot{q}^T K_D \dot{q},$$

because for any EL-system, $\dot{M}(q) - 2C(q, \dot{q})$ is a skew-symmetric matrix!
(See Section 5.2.1 in 07_Manipulator_Kinematics_Dynamics.pdf)

So, if $\dot{q} \neq 0$, then $\dot{V} < 0$.
To apply Lyapunov's conclusions, we actually want $q \to q_d$ is that when $q \neq q_d, \dot{q} \neq 0$, THEN $\dot{V} < 0$.

A solution comes through La Salle's invariance principle (Hello again, ME 672).

Intuition: When its impossible for $\dot{V}(t) = 0$ forever at any state where $V(q) \neq 0$, then $q \to q_d$.

Summary

- For set-point regulation,
- Assuming gravity isn't affecting dynamics, no external forces,
- PD control is enough to get $q \rightarrow q_d$.
  no coupled model issues !!!

Summary

- For set-point regulation,
- Assuming gravity isn't affecting dynamics, no external forces,
- PD control is enough to get $q \to q_d$.
  no coupled model issues !!!

Furthermore:

- if $G(q) \neq 0$, then q_{eq} satisfies

$$G(q_{eq}) = K_P(q_d - q_{eq}),$$

  and this equilibrium ($\neq q_d$) is locally asymptotically stable.
- To reduce error, increase K_P!

Summary

- For set-point regulation,
- Assuming gravity isn't affecting dynamics, no external forces,
- PD control is enough to get $q \to q_d$.
  no coupled model issues !!!

Furthermore:

- if $G(q) \neq 0$, then q_{eq} satisfies

$$G(q_{eq}) = K_P(q_d - q_{eq}),$$

  and this equilibrium ($\neq q_d$) is locally asymptotically stable.
- To reduce error, increase K_P!

Question: Will an integrator work to handle gravity, like in the case of independent joint control?

Trajectory Tracking

The power of PD-feedback breaks down when the trajectory has significant accelerations, and so is not really static or quasi-static.

Trajectory Tracking

The power of PD-feedback breaks down when the trajectory has significant accelerations, and so is not really static or quasi-static.

PD-control also does not account for obstacles or contacts, since the analyzed model excludes $J(q)^T f_{tip}$.

Trajectory Tracking

The power of PD-feedback breaks down when the trajectory has significant accelerations, and so is not really static or quasi-static.

PD-control also does not account for obstacles or contacts, since the analyzed model excludes $J(q)^T f_{tip}$.

Now, we use the model to implement the Computed Torque Control, or Inverse Dynamics Control, or basic Feedback Linearization.

Trajectory Tracking

The power of PD-feedback breaks down when the trajectory has significant accelerations, and so is not really static or quasi-static.

PD-control also does not account for obstacles or contacts, since the analyzed model excludes $J(q)^T f_{tip}$.

Now, we use the model to implement the Computed Torque Control, or Inverse Dynamics Control, or basic Feedback Linearization.

Real Model: $M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = u(t)$

Trajectory Tracking

The power of PD-feedback breaks down when the trajectory has significant accelerations, and so is not really static or quasi-static.

PD-control also does not account for obstacles or contacts, since the analyzed model excludes $J(q)^T f_{tip}$.

Now, we use the model to implement the Computed Torque Control, or Inverse Dynamics Control, or basic Feedback Linearization.

Real Model: $M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = u(t)$
What you think is model: $\hat{M}(q)$, $\hat{C}(q, \dot{q})$, $\hat{G}(q)$

Trajectory Tracking

The power of PD-feedback breaks down when the trajectory has significant accelerations, and so is not really static or quasi-static.

PD-control also does not account for obstacles or contacts, since the analyzed model excludes $J(q)^T f_{tip}$.

Now, we use the model to implement the Computed Torque Control, or Inverse Dynamics Control, or basic Feedback Linearization.

Real Model: $M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = u(t)$
What you think is model: $\hat{M}(q)$, $\hat{C}(q,\dot{q})$, $\hat{G}(q)$
Choose control to get rid of nonlinearity:

$$u(t) = \hat{M}(q)\ddot{a}_q(t) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q).$$

Inverse Dynamics Control

Real Model: $M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = u(t)$
What you think is model: $\hat{M}(q)$, $\hat{C}(q,\dot{q})$, $\hat{G}(q)$
Choose control to get rid of nonlinearity:

$$u(t) = \hat{M}(q)a_q(t) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q).$$

Closed-loop:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \hat{M}(q)a_q(t) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) \quad (3)$$

Inverse Dynamics Control

Real Model: $M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = u(t)$
What you think is model: $\hat{M}(q)$, $\hat{C}(q,\dot{q})$, $\hat{G}(q)$
Choose control to get rid of nonlinearity:

$$u(t) = \hat{M}(q)a_q(t) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q).$$

Closed-loop:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \hat{M}(q)a_q(t) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) \quad (3)$$

IF $\hat{M}(q) = M(q)$, $\hat{C}(q,\dot{q}) = \hat{C}(q,\dot{q})$, $\hat{G}(q) = G(q)$, then

$$M(q)\ddot{q} = M(q)a_q(t)$$

Inverse Dynamics Control

Real Model: $M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = u(t)$
What you think is model: $\hat{M}(q)$, $\hat{C}(q,\dot{q})$, $\hat{G}(q)$
Choose control to get rid of nonlinearity:

$$u(t) = \hat{M}(q)a_q(t) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q).$$

Closed-loop:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \hat{M}(q)a_q(t) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) \quad (3)$$

IF $\hat{M}(q) = M(q)$, $\hat{C}(q,\dot{q}) = \hat{C}(q,\dot{q})$, $\hat{G}(q) = G(q)$, then

$$M(q)\ddot{q} = M(q)a_q(t)$$

Since $M(q) > 0$ for all q,

$$\ddot{q} = a_q(t)$$

Inverse Dynamics Control

Real Model: $M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = u(t)$
What you think is model: $\hat{M}(q)$, $\hat{C}(q,\dot{q})$, $\hat{G}(q)$
Choose control to get rid of nonlinearity:

$$u(t) = \hat{M}(q)a_q(t) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q).$$

Closed-loop:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \hat{M}(q)a_q(t) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) \quad (3)$$

IF $\hat{M}(q) = M(q)$, $\hat{C}(q,\dot{q}) = \hat{C}(q,\dot{q})$, $\hat{G}(q) = G(q)$, then

$$M(q)\ddot{q} = M(q)a_q(t)$$

Since $M(q) > 0$ for all q,

$$\ddot{q} = a_q(t)$$

Computed torque control gives us a linear system!
Just need to design $a_q(t)$ so that $q(t) \to q_d(t)$

Inverse Dynamics Control

$$\ddot{q} = a_q(t) \tag{4}$$

Given $q_d(t)$, one choice for $a_q(t)$ is

$$a_q(t) = \ddot{q}_d(t) + K_P\left(q_d(t) - q(t)\right) + K_D\left(\dot{q}_d(t) - \dot{q}(t)\right)$$

Inverse Dynamics Control

$$\ddot{q} = a_q(t) \tag{4}$$

Given $q_d(t)$, one choice for $a_q(t)$ is

$$a_q(t) = \ddot{q}_d(t) + K_P\left(q_d(t) - q(t)\right) + K_D\left(\dot{q}_d(t) - \dot{q}(t)\right)$$

Note that $\ddot{q}_d(t)$ is like a feed forward term, and the remainder is the feedback term for this second-order system.

Inverse Dynamics Control

$$\ddot{q} = a_q(t) \tag{4}$$

Given $q_d(t)$, one choice for $a_q(t)$ is

$$a_q(t) = \ddot{q}_d(t) + K_P\left(q_d(t) - q(t)\right) + K_D\left(\dot{q}_d(t) - \dot{q}(t)\right)$$

Note that $\ddot{q}_d(t)$ is like a feed forward term, and the remainder is the feedback term for this second-order system.

Defining the error as $e(t) = q(t) - q_d(t)$, we can rewrite the equation (4) as

$$\ddot{e}(t) + K_D\dot{e}(t) + K_P e(t) = 0.$$

Choosing $K_D > 0$ and $K_P > 0$ will ensure $e(t) \to 0$!

Inverse Dynamics Control

Note that the control we wrote down is

$$
\begin{aligned}
u(t) &= \hat{M}(q)a_q(t) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) \\
&= \hat{M}(q)\left(\ddot{q}_d(t) + K_P\left(q_d(t) - q(t)\right) + K_D\left(\dot{q}_d(t) - \dot{q}(t)\right)\right) \\
&\quad + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q)
\end{aligned}
$$

We don't need to construct the matrices $\hat{M}(q)$ and $\hat{C}(q,\dot{q})$, and vector $\hat{G}(q)$ explicitly in order to implement this control law.

Inverse Dynamics Control

Note that the control we wrote down is

$$
\begin{aligned}
u(t) &= \hat{M}(q)a_q(t) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) \\
&= \hat{M}(q)\left(\ddot{q}_d(t) + K_P\left(q_d(t) - q(t)\right) + K_D\left(\dot{q}_d(t) - \dot{q}(t)\right)\right) \\
&\quad + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q)
\end{aligned}
$$

We don't need to construct the matrices $\hat{M}(q)$ and $\hat{C}(q,\dot{q})$, and vector $\hat{G}(q)$ explicitly in order to implement this control law.

We can use the recursive Newton-Euler Algorithm to calculate $u(t)$ given $a_q(t)$ and the relevant frames, link geometry, and inertia parameters.

Inverse Dynamics Control

Note that the control we wrote down is

$$
\begin{aligned}
u(t) &= \hat{M}(q)a_q(t) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) \\
&= \hat{M}(q)\left(\ddot{q}_d(t) + K_P\left(q_d(t) - q(t)\right) + K_D\left(\dot{q}_d(t) - \dot{q}(t)\right)\right) \\
&\quad + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q)
\end{aligned}
$$

We don't need to construct the matrices $\hat{M}(q)$ and $\hat{C}(q,\dot{q})$, and vector $\hat{G}(q)$ explicitly in order to implement this control law.

We can use the recursive Newton-Euler Algorithm to calculate $u(t)$ given $a_q(t)$ and the relevant frames, link geometry, and inertia parameters.

Physics simulators for robots use this method.

Task Space Inverse Dynamics

Let X be the end-effector pose with orientation given by a minimal representation of $SO(3)$. Then,

$$\dot{x} = J_a(q)\dot{q} \implies \ddot{X} = J_a(q)\ddot{q} + \dot{J}_a(q)\dot{q} \tag{5}$$

Task Space Inverse Dynamics

Let X be the end-effector pose with orientation given by a minimal representation of $SO(3)$. Then,

$$\dot{x} = J_a(q)\dot{q} \implies \ddot{X} = J_a(q)\ddot{q} + \dot{J}_a(q)\dot{q} \tag{5}$$

If we choose

$$a_q = J_a(q)^{-1}\left(a_X - \dot{J}_a(q)\dot{q}\right) \tag{6}$$

then the joint space inverse dynamics control implies a task space dynamics of

$$\ddot{X} = a_X \tag{7}$$

and we can now track task space trajectories $X_d(t)$.
BUT $J_a(q)$ must be non-singular.
In some cases, Jacobian pseudoinverses may be used.

Robust Inverse Dynamics Control

What happens when $\hat{M}(q) \neq M(q)$, $\hat{C}(q, \dot{q}) \neq \hat{C}(q, \dot{q})$, $\hat{G}(q) \neq G(q)$?

Our closed-loop under inverse dynamics control is

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \hat{M}(q)a_q(t) + \hat{C}(q, \dot{q})\dot{q} + \hat{G}(q)$$

Rewrite above as

$$M(q)\ddot{q} = \hat{M}(q)a_q(t) + \hat{C}(q, \dot{q})\dot{q} - C(q, \dot{q})\dot{q} + \hat{G}(q) - G(q)$$

$$\begin{aligned} M(q)\ddot{q} &= M(q)a_q(t) + \left(\hat{M}(q) - M(q)\right) a_q(t) \\ &\quad + \left(\hat{C}(q, \dot{q}) - C(q, \dot{q})\right) \dot{q} + \left(\hat{G}(q) - G(q)\right) \\ &= M(q)a_q(t) + \tilde{M}a_q(t) + \tilde{C}\dot{q} + \tilde{G} \end{aligned}$$

$$\implies \ddot{q} = a_q(t) + M^{-1}(q)\left(\tilde{M}a_q(t) + \tilde{C}\dot{q} + \tilde{G}\right)$$

Robust Inverse Dynamics Control

$$\ddot{q} = a_q(t) + M^{-1}(q)\left(\tilde{M}a_q(t) + \tilde{C}\dot{q} + \tilde{G}\right) \quad (8)$$

$$= a_q + \eta(q, \dot{q}, \ddot{q}, a_q) \quad (9)$$

If we had perfect knowledge of the model parameters, $\eta(q, \dot{q}, \ddot{q}, a_q) = 0$, because $\tilde{M} = \hat{M}(q) - M(q) = 0$ and so on.

To account for non-zero $\eta(q, \dot{q}, \ddot{q}, a_q)$, we choose a_q as

$$a_q(t) = \ddot{q}_d(t) + K_P\left(q_d(t) - q(t)\right) + K_D\left(\dot{q}_d(t) - \dot{q}(t)\right) + \delta a$$

Robust Inverse Dynamics Control

$$\ddot{q} = a_q(t) + M^{-1}(q)\left(\tilde{M}a_q(t) + \tilde{C}\dot{q} + \tilde{G}\right) \tag{8}$$

$$= a_q + \eta(q, \dot{q}, \ddot{q}, a_q) \tag{9}$$

If we had perfect knowledge of the model parameters, $\eta(q, \dot{q}, \ddot{q}, a_q) = 0$, because $\tilde{M} = \hat{M}(q) - M(q) = 0$ and so on.

To account for non-zero $\eta(q, \dot{q}, \ddot{q}, a_q)$, we choose a_q as

$$a_q(t) = \ddot{q}_d(t) + K_P\left(q_d(t) - q(t)\right) + K_D\left(\dot{q}_d(t) - \dot{q}(t)\right) + \delta a$$

Let $e(t) = \begin{bmatrix} q(t) - q_d(t) \\ \dot{q}(t) - \dot{q}_d(t) \end{bmatrix}$. Our closed-loop is now

$$\dot{e} = \begin{bmatrix} 0 & I \\ -K_P & -K_D \end{bmatrix} e + \begin{bmatrix} 0 \\ I \end{bmatrix} (\delta a + \eta) \tag{10}$$

Robust Inverse Dynamics Control

$$\dot{e} = \begin{bmatrix} 0 & I \\ -K_P & -K_D \end{bmatrix} e + \begin{bmatrix} 0 \\ I \end{bmatrix} (\delta a + \eta(q, \dot{q}, \ddot{q}, a_q))$$

We can now easily see why the new term δa is what we use to account for non-zero $\eta(q, \dot{q}, \ddot{q}, a_q)$.

Robust Inverse Dynamics Control

$$\dot{e} = \begin{bmatrix} 0 & I \\ -K_P & -K_D \end{bmatrix} e + \begin{bmatrix} 0 \\ I \end{bmatrix} (\delta a + \eta(q, \dot{q}, \ddot{q}, a_q))$$

We can now easily see why the new term δa is what we use to account for non-zero $\eta(q, \dot{q}, \ddot{q}, a_q)$.

Remember that we can't compute $\eta(q, \dot{q}, \ddot{q}, a_q)$, because it depends on the true model, which we assume we don't know.

Robust Inverse Dynamics Control

$$\dot{e} = \begin{bmatrix} 0 & I \\ -K_P & -K_D \end{bmatrix} e + \begin{bmatrix} 0 \\ I \end{bmatrix} (\delta a + \eta(q, \dot{q}, \ddot{q}, a_q))$$

We can now easily see why the new term δa is what we use to account for non-zero $\eta(q, \dot{q}, \ddot{q}, a_q)$.

Remember that we can't compute $\eta(q, \dot{q}, \ddot{q}, a_q)$, because it depends on the true model, which we assume we don't know.

How do we choose δa ?

Robust Inverse Dynamics Control

$$\dot{e} = \begin{bmatrix} 0 & I \\ -K_P & -K_D \end{bmatrix} e + \begin{bmatrix} 0 \\ I \end{bmatrix} (\delta a + \eta(q, \dot{q}, \ddot{q}, a_q))$$

We can now easily see why the new term δa is what we use to account for non-zero $\eta(q, \dot{q}, \ddot{q}, a_q)$.

Remember that we can't compute $\eta(q, \dot{q}, \ddot{q}, a_q)$, because it depends on the true model, which we assume we don't know.

How do we choose δa ? Lyapunov methods

Robust Inverse Dynamics Control

$$\dot{e} = \begin{bmatrix} 0 & I \\ -K_P & -K_D \end{bmatrix} e + \begin{bmatrix} 0 \\ I \end{bmatrix} (\delta a + \eta(q, \dot{q}, \ddot{q}, a_q))$$

Suppose we can bound η as

$$\|\eta\| \leq \rho(e, t),$$

we can then design δa to guarantee ultimate $e(t) \to 0$.

Robust Inverse Dynamics Control

$$\dot{e} = \begin{bmatrix} 0 & I \\ -K_P & -K_D \end{bmatrix} e + \begin{bmatrix} 0 \\ I \end{bmatrix} (\delta a + \eta(q, \dot{q}, \ddot{q}, a_q))$$

Suppose we can bound η as

$$\|\eta\| \leq \rho(e, t),$$

we can then design δa to guarantee ultimate $e(t) \to 0$.

Let $A = \begin{bmatrix} 0 & I \\ -K_P & -K_D \end{bmatrix}$, and $B = \begin{bmatrix} 0 \\ I \end{bmatrix}$. $\dot{e} = Ae + B(\delta a + \eta)$.

Robust Inverse Dynamics Control

$$\dot{e} = \begin{bmatrix} 0 & I \\ -K_P & -K_D \end{bmatrix} e + \begin{bmatrix} 0 \\ I \end{bmatrix} (\delta a + \eta(q, \dot{q}, \ddot{q}, a_q))$$

Suppose we can bound η as

$$\|\eta\| \leq \rho(e, t),$$

we can then design δa to guarantee ultimate $e(t) \to 0$.

Let $A = \begin{bmatrix} 0 & I \\ -K_P & -K_D \end{bmatrix}$, and $B = \begin{bmatrix} 0 \\ I \end{bmatrix}$. $\dot{e} = Ae + B(\delta a + \eta)$.

Let $V = e^T P e$ where $A^T P + PA = -Q$. Since A can be made Hurwitz by choosing K_P and K_D, we know that for each $Q > 0$ there exists $P > 0$ that satisfies the Lyapunov equation $A^T P + PA = -Q$.

Robust Inverse Dynamics Control

We have that

$$\begin{aligned} \dot{V} &= e^T PAe + e^T A^T Pe + 2e^T PB(\delta a + \eta) \\ &= -e^T Qe + 2e^T PB(\delta a + \eta) \end{aligned} \quad (11)$$

We choose

$$\delta a = \begin{cases} -\rho(e,t)\frac{B^T Pe}{\|B^T Pe\|} & , \quad \text{if } \|B^T Pe\| \neq 0 \\ 0 & , \quad \text{if } \|B^T Pe\| = 0 \end{cases} \quad (12)$$

Let $w = B^T Pe$. Then the second term in (11) is then

$$\begin{aligned} w^T\left(-\rho\frac{w}{\|w\|} + \eta\right) &\leq -\rho\|w\| + \|w\|\|\eta\| \quad (w^T\eta \leq \|w\|\|\eta\|) \\ &\leq \|w\|(-\rho + \|\eta\|) \\ &\leq 0, \qquad (\|\eta\| \leq \rho(e,t)) \end{aligned}$$

when $e \neq 0$.

Robust Inverse Dynamics Control

So,

$$\begin{aligned} \dot{V} &= -e^T Qe + 2w^T(\delta a + \eta) \\ &\leq -e^T Qe + 0 && \text{(from previous slide)} \\ &< 0 && (Q > 0) \end{aligned}$$

In summary, if we can bound η (see notes Section 5.5),

Robust Inverse Dynamics Control

So,

$$\begin{aligned}
\dot{V} &= -e^T Qe + 2w^T(\delta a + \eta) \\
&\leq -e^T Qe + 0 && \text{(from previous slide)} \\
&< 0 && (Q > 0)
\end{aligned}$$

In summary, if we can bound η (see notes Section 5.5), which depends on our model errors,

Robust Inverse Dynamics Control

So,

$$\begin{aligned}
\dot{V} &= -e^T Q e + 2w^T(\delta a + \eta) \\
&\leq -e^T Q e + 0 && \text{(from previous slide)} \\
&< 0 && (Q > 0)
\end{aligned}$$

In summary, if we can bound η (see notes Section 5.5), which depends on our model errors, we can achieve $e(t) \to 0$ using the robust version of the inverse dynamics control.

Robust Inverse Dynamics Control

So,

$$\begin{aligned}\dot{V} &= -e^T Qe + 2w^T(\delta a + \eta) \\ &\leq -e^T Qe + 0 && \text{(from previous slide)} \\ &< 0 && (Q > 0)\end{aligned}$$

In summary, if we can bound η (see notes Section 5.5), which depends on our model errors, we can achieve $e(t) \to 0$ using the robust version of the inverse dynamics control.

Issues:

- If the bound ρ is large (due to large errors η), then the demanded control u becomes larger than motor capacity
- The control is discontinuous at $w = 0$, which is tricky to implement; overheats electric motors.

Adaptive Inverse Dynamics Control

- The error in model estimate affects $\rho(\epsilon, t)$ which ruins the lowest achievable error in continuous robust inverse dynamics control.

Adaptive Inverse Dynamics Control

- The error in model estimate affects $\rho(\epsilon, t)$ which ruins the lowest achievable error in continuous robust inverse dynamics control.
- Ideally, we want smaller model errors to achieve lower error.

Adaptive Inverse Dynamics Control

- The error in model estimate affects $\rho(\epsilon, t)$ which ruins the lowest achievable error in continuous robust inverse dynamics control.
- Ideally, we want smaller model errors to achieve lower error.
- Luckily, we can learn models on-the-fly using adaptive control theory.

Adaptive Inverse Dynamics Control

Key idea: EL model is linear in parameters!

Adaptive Inverse Dynamics Control

Key idea: EL model is linear in parameters!

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) \to Y(q,\dot{q},\ddot{q})\Theta$$

- Θ: a vector function that depend on link masses, lengths, g
- $Y(q,\dot{q},\ddot{q})$ a vector function that DOES NOT DEPEND on robot parameters

Adaptive Inverse Dynamics Control

Key idea: EL model is linear in parameters!

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) \rightarrow Y(q,\dot{q},\ddot{q})\Theta$$

- Θ: a vector function that depend on link masses, lengths, g
- $Y(q,\dot{q},\ddot{q})$ a vector function that DOES NOT DEPEND on robot parameters

We don't know true parameters Θ.

Adaptive Inverse Dynamics Control

Key idea: EL model is linear in parameters!

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) \rightarrow Y(q,\dot{q},\ddot{q})\Theta$$

- Θ: a vector function that depend on link masses, lengths, g
- $Y(q,\dot{q},\ddot{q})$ a vector function that DOES NOT DEPEND on robot parameters

We don't know true parameters Θ.

We have a guess $\hat{\Theta}$ (which we may convert into $\hat{M}(q)$, $\hat{C}(q,\dot{q})$, $\hat{G}(q)$).

Adaptive Inverse Dynamics Control

We would hope $\hat{\Theta} = \Theta$.

Adaptive Inverse Dynamics Control

We would hope $\hat{\Theta} = \Theta$.

Since guessing right is unlikely, we design a continuous update rule for $\hat{\Theta}(t)$.

Adaptive Inverse Dynamics Control

We would hope $\hat{\Theta} = \Theta$.

Since guessing right is unlikely, we design a continuous update rule for $\hat{\Theta}(t)$.

KEY ADAPTIVE CONTROL BEHAVIOR:
This update rule WILL NOT ensure that $\hat{\Theta}(t) \to \Theta$, but it ensures that

$$e(t) \to 0.$$

Adaptive Inverse Dynamics Control

We would hope $\hat{\Theta} = \Theta$.

Since guessing right is unlikely, we design a continuous update rule for $\hat{\Theta}(t)$.

KEY ADAPTIVE CONTROL BEHAVIOR:
This update rule WILL NOT ensure that $\hat{\Theta}(t) \to \Theta$, but it ensures that

$$e(t) \to 0.$$

This update rule relies on the linearity in parameters property.

Adaptive Inverse Dynamics Control

Let $\tilde{q} = q - q_d$, $\dot{\tilde{q}} = \dot{q} - \dot{q}_d$
Choosing $u = Y(q, \dot{q}, a_q)\hat{\Theta}$, where $a_q = \ddot{q}_d(t) - K_P\tilde{q} - K_D\dot{\tilde{q}}$ we get

$$\ddot{\tilde{q}} + K_1\dot{\tilde{q}} + K_0\dot{\tilde{q}} = M^{-1}Y(q, \dot{q}, \ddot{q})\tilde{\Theta} = \Phi\tilde{\Theta}, \tag{13}$$

where $\tilde{\Theta} = \hat{\Theta} - \Theta$.

Adaptive Inverse Dynamics Control

Let $\tilde{q} = q - q_d$, $\dot{\tilde{q}} = \dot{q} - \dot{q}_d$
Choosing $u = Y(q, \dot{q}, a_q)\hat{\Theta}$, where $a_q = \ddot{q}_d(t) - K_P\tilde{q} - K_D\dot{\tilde{q}}$ we get

$$\ddot{\tilde{q}} + K_1\dot{\tilde{q}} + K_0\dot{\tilde{q}} = M^{-1}Y(q, \dot{q}, \ddot{q})\tilde{\Theta} = \Phi\tilde{\Theta}, \tag{13}$$

where $\tilde{\Theta} = \hat{\Theta} - \Theta$.

Let $e = \begin{bmatrix} \tilde{q} \\ \dot{\tilde{q}} \end{bmatrix}$. We get the ODE

$$\dot{e} = Ae + B\Phi\tilde{\theta} \tag{14}$$

which is effectively the same ODE as in the robust case, but without δa.

Adaptive Inverse Dynamics Control

Consider a function of $e, \tilde{\Theta}$ given by

$$V(e, \tilde{\Theta}) = e^T P e + \tilde{\Theta}^T \Gamma \tilde{\Theta}. \tag{15}$$

For $P > 0$ and $\Gamma > 0$, $V(e, \tilde{\Theta}) = 0$ when $e = 0$ and $\Theta = \hat{\Theta}$.

Adaptive Inverse Dynamics Control

Consider a function of $e, \tilde{\Theta}$ given by

$$V(e, \tilde{\Theta}) = e^T P e + \tilde{\Theta}^T \Gamma \tilde{\Theta}. \tag{15}$$

For $P > 0$ and $\Gamma > 0$, $V(e, \tilde{\Theta}) = 0$ when $e = 0$ and $\Theta = \hat{\Theta}$.
Again, we know there exists $Q > 0$ such that $A^T P + PA = -Q$.

Adaptive Inverse Dynamics Control

Consider a function of $e, \tilde{\Theta}$ given by

$$V(e, \tilde{\Theta}) = e^T P e + \tilde{\Theta}^T \Gamma \tilde{\Theta}. \tag{15}$$

For $P > 0$ and $\Gamma > 0$, $V(e, \tilde{\Theta}) = 0$ when $e = 0$ and $\Theta = \hat{\Theta}$.
Again, we know there exists $Q > 0$ such that $A^T P + PA = -Q$.

We have

$$\dot{V}(e, \tilde{\Theta}) = -e^T Q e + 2\tilde{\Theta}^T \left(\Phi^T B^T P e + \Gamma \dot{\hat{\Theta}} \right) \tag{16}$$

If we knew Θ the second term is made zero by choosing $\hat{\Theta} = \Theta$.

Adaptive Inverse Dynamics Control

Consider a function of $e, \tilde{\Theta}$ given by

$$V(e, \tilde{\Theta}) = e^T P e + \tilde{\Theta}^T \Gamma \tilde{\Theta}. \tag{15}$$

For $P > 0$ and $\Gamma > 0$, $V(e, \tilde{\Theta}) = 0$ when $e = 0$ and $\Theta = \hat{\Theta}$.
Again, we know there exists $Q > 0$ such that $A^T P + PA = -Q$.

We have

$$\dot{V}(e, \tilde{\Theta}) = -e^T Q e + 2\tilde{\Theta}^T \left(\Phi^T B^T P e + \Gamma \dot{\hat{\Theta}} \right) \tag{16}$$

If we knew Θ the second term is made zero by choosing $\hat{\Theta} = \Theta$.
Since we don't, we instead choose

$$\dot{\hat{\Theta}} = -\Gamma^{-1} \Phi^T B^T P e \tag{17}$$

$$(\implies \dot{V} \leq 0, \text{ and } \dot{V} < 0 \text{ when } e \neq 0) \tag{18}$$

It's like a nonlinear integral control!

Adaptive Inverse Dynamics Control

Summary:

- Parameter update: $\dot{\hat{\Theta}} = -\Gamma^{-1}\Phi^T B^T P e$

Adaptive Inverse Dynamics Control

Summary:

- Parameter update: $\dot{\hat{\Theta}} = -\Gamma^{-1}\Phi^T B^T P e$
- Ensures that $e(t) \to 0$ for any initial guess $\Theta(0)$
  Analysis uses Barbalat's Lemma to handle $\dot{V} \not< 0$.
  (ME 699 in Fall 2020)

Adaptive Inverse Dynamics Control

Summary:

- Parameter update: $\dot{\hat{\Theta}} = -\Gamma^{-1}\Phi^T B^T P e$
- Ensures that $e(t) \to 0$ for any initial guess $\Theta(0)$
  Analysis uses Barbalat's Lemma to handle $\dot{V} \not< 0$.
  (ME 699 in Fall 2020)
- DOES NOT ensure that $\hat{\Theta}(t) \to \Theta$ ...

Adaptive Inverse Dynamics Control

Summary:

- Parameter update: $\dot{\hat{\Theta}} = -\Gamma^{-1}\Phi^T B^T P e$
- Ensures that $e(t) \to 0$ for any initial guess $\Theta(0)$
  Analysis uses Barbalat's Lemma to handle $\dot{V} \nless 0$.
  (ME 699 in Fall 2020)
- DOES NOT ensure that $\hat{\Theta}(t) \to \Theta$ ...
- ...unless a condition known as Persistence of Excitation holds.

Adaptive Inverse Dynamics Control

Summary:

- Parameter update: $\dot{\hat{\Theta}} = -\Gamma^{-1}\Phi^T B^T P e$
- Ensures that $e(t) \to 0$ for any initial guess $\Theta(0)$
  Analysis uses Barbalat's Lemma to handle $\dot{V} \nless 0$.
  (ME 699 in Fall 2020)
- DOES NOT ensure that $\hat{\Theta}(t) \to \Theta$ ...
- ...unless a condition known as Persistence of Excitation holds.
  - when $e = 0$, $\dot{\Theta} = 0$. Update stops.

Adaptive Inverse Dynamics Control

Summary:

- Parameter update: $\dot{\hat{\Theta}} = -\Gamma^{-1}\Phi^T B^T P e$
- Ensures that $e(t) \to 0$ for any initial guess $\Theta(0)$
  Analysis uses Barbalat's Lemma to handle $\dot{V} \nless 0$.
  (ME 699 in Fall 2020)
- DOES NOT ensure that $\hat{\Theta}(t) \to \Theta$ …
- …unless a condition known as Persistence of Excitation holds.
  - when $e = 0$, $\dot{\Theta} = 0$. Update stops.
  - Some disturbances keep pushing $e(t)$ away from zero in a way that allows update to continue while e stays small

Adaptive Inverse Dynamics Control

Summary:

- Parameter update: $\dot{\hat{\Theta}} = -\Gamma^{-1}\Phi^T B^T P e$
- Ensures that $e(t) \to 0$ for any initial guess $\Theta(0)$
  Analysis uses Barbalat's Lemma to handle $\dot{V} \not< 0$.
  (ME 699 in Fall 2020)
- DOES NOT ensure that $\hat{\Theta}(t) \to \Theta$ ...
- ...unless a condition known as Persistence of Excitation holds.
  - when $e = 0$, $\dot{\Theta} = 0$. Update stops.
  - Some disturbances keep pushing $e(t)$ away from zero in a way that allows update to continue while e stays small
  - PoE: says when $\hat{\Theta}(t) \to \Theta$ as opposed to $\|\hat{\Theta}(t)\| \to \infty$

Adaptive Inverse Dynamics Control

Summary:

- Parameter update: $\dot{\hat{\Theta}} = -\Gamma^{-1}\Phi^T B^T P e$
- Ensures that $e(t) \to 0$ for any initial guess $\Theta(0)$
  Analysis uses Barbalat's Lemma to handle $\dot{V} \not< 0$.
  (ME 699 in Fall 2020)
- DOES NOT ensure that $\hat{\Theta}(t) \to \Theta$ ...
- ...unless a condition known as Persistence of Excitation holds.
  - when $e = 0$, $\dot{\hat{\Theta}} = 0$. Update stops.
  - Some disturbances keep pushing $e(t)$ away from zero in a way that allows update to continue while e stays small
  - PoE: says when $\hat{\Theta}(t) \to \Theta$ as opposed to $\|\hat{\Theta}(t)\| \to \infty$
  - PoE rule is important. Bad updates caused the NASA X-15 to crash in 1967.
    (Mathematical analysis is sometimes not optional).