ME 599/699 Robot Modeling & Control
Fall 2021

Inverse Dynamics Control

Hasan A. Poonawala

Department of Mechanical Engineering
University of Kentucky

Email: hasan.poonawala@uky.edu
Web: https://www.engr.uky.edu/~hap

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:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \hat{M}(q)a_q(t) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) \quad (1)$$

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:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \hat{M}(q)a_q(t) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) \quad (1)$$

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:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \hat{M}(q)a_q(t) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) \quad (1)$$

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:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \hat{M}(q)a_q(t) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) \quad (1)$$

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

$$\ddot{q} = a_q(t) \tag{2}$$

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

$$\ddot{q} = a_q(t) \tag{2}$$

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

$$\ddot{q} = a_q(t) \tag{2}$$

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 (2) 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,

$$\dot{x} = J_a(q)\dot{q} \implies \ddot{X} = J_a(q)\ddot{q} + \dot{J}_a(q)\dot{q} \tag{3}$$

Task Space Inverse Dynamics

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

$$\dot{x} = J_a(q)\dot{q} \implies \ddot{X} = J_a(q)\ddot{q} + \dot{J}_a(q)\dot{q} \tag{3}$$

If we choose

$$a_q = J_a(q)^{-1}\left(a_X - \dot{J}_a(q)\dot{q}\right) \tag{4}$$

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

$$\ddot{X} = a_X \tag{5}$$

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.