Lecture 11: Multivariable Control of Robotic Manipulators

Reading: SHV Ch.7

Mechanical Engineering
Hanz Richter, PhD

Overview

Armed with a mathematical model of the manipulator, the doors are open to the analysis of many controls-related problems.
For instance, the end-effector tracking problem will become easy to understand and solve using the tools introduced in this chapter. We examine the following approaches to model-based closed loop control:

1. Independent-joint (decoupled) PD controllers (it works for setpoint regulation)
2. Feedback linearization
3. Robust adaptive control
4. Passivity-based control

Why the need for advanced methods?
Much more powerful, allow to perform trajectory tracking very efficiently

Electromechanical Model

The model of the previous chapter is purely mechanical. The inputs are torques/forces, and the outputs are positions/velocities.
We need to account for the servomotors and the gearing used in each joint.
Remembering that the model for the servomotor used in joint k is

$$J_{m_k}\ddot{\theta}_{m_k} + B_k\dot{\theta}_{m_k} = \frac{K_{m_k}}{R_k}V_k - \tau_k r_k$$

where $B_k = B_{mk} + K_{b_k}K_{m_k}/R_k$. Since $\theta_{m_k} = r_k q_k$, we can solve for τ_k from the servomotor equation and substitute for τ_k in the manipulator equation to obtain

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

where $M(q) = D(q) + J$, with $J = \text{diag}\{r_k^2 J_{m_k}\}$ and $B = \text{diag}\{r_k^2 B_k\}$. Also:

$$u_k = r_k\frac{K_{m_k}}{R_k}V_k$$

It's important to note that the basic properties (skew-symmetry, passivity, linearity in parameters and inertia matrix bounds are still valid.

Independent-Joint PD Control

Define the setpoint error as $\tilde{q} = q - q^d$, with q^d being the vector of desired constant joint angles (setpoint). A set of n independent PD loops is equivalent to the control law

$$u = -K_P \tilde{q} - K_D \dot{q}$$

where K_P and K_D are diagonal (for the PD loops to be really decoupled). If we neglect the gravity term and the friction ($B = 0$ and $g(q) = 0$), the Lyapunov function

$$V(q) = \frac{1}{2}\dot{q}^T M(q) \dot{q} + \frac{1}{2}\tilde{q}^T K_P \tilde{q}$$

can be used to show that the errors $\tilde{q}$ converge to zero asymptotically. Follow the details of the proof in SHV, observing the following:

1. The proof relies on q^d being constant
2. The term $\frac{1}{2}\dot{q}^T(\dot{M}(q) - 2C(q, \dot{q}))\dot{q}$ is identically zero. Why?
3. LaSalle's theorem is used to be able to conclude asymptotic stability *even with negative semi-definite* $\dot{V}$.

Decoupled PD: Gravity effects

If $g(q) \neq 0$, the robot stabilizes at a nonzero $\tilde{q}$ (steady-state error). The offset satisfies

$$K_P \tilde{q} = g(q)$$

This means that the controller works until the gravity forces have been balanced, so that manipulator velocities and accelerations are zero. But the controller then calls it a day and does not want to keep working to eliminate the offset. We can either introduce integration in each loop or use a modified PD law:

$$u = -K_P \tilde{q} - K_D \dot{q} + g(q)$$

Note that this law effectively eliminates the problem, but at the cost of having to evaluate $g(q)$ as part of the real-time control algorithm. This reduces to finding the world position of the center of mass of link k and evaluating partial derivatives. Since this position depends on q_k and components of q other than q_k, we can no longer call this approach “decoupled”.

Feedback Linearization: Intuitive Idea

Suppose we have a nonlinear system

$$\begin{aligned}\dot{x} &= f(x,u)\\ y &= h(x,u)\end{aligned}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ and $y \in \mathbb{R}^p$. Suppose we are able to find a feedback function $u = g(x, \dot{x}, v)$ so that its substitution into the system results in linear closed-loop dynamics of the form

$$\frac{d^n y}{dt^n} = v$$

That is, we convert the nonlinear system in a simple linear system in the form of a multiple integrator with input v, called the *synthetic input* or *virtual control*.

Then we “simply” stabilize the linear system using v, for instance using $v = -a_0 y - a_1 \dot{y} ... - a_{n-1} \frac{d^{n-1}y}{dt^{n-1}}$ so that the closed-loop output dynamics becomes

$$\frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1}y}{dt^{n-1}} + a_{n-2} \frac{d^{n-2}y}{dt^{n-2}} + ... + a_0 y = 0$$

which is easily made asymptotically stable by choosing the $\{a_i\}$ so that the characteristic polynomial has left-half plane roots.
Finally, we substitute v into $g(x, \dot{x}, v)$ to obtain the actual control law.
Too good to be true? -There are in fact several serious issues.

Feedback Linearization: Working Example

Take the following nonlinear system

$$\begin{aligned}\dot{x}_1 &= x_2 + u \\ \dot{x}_2 &= \sin(x_1) - x_3 \\ \dot{x}_3 &= x_1 x_2 \\ y &= x_2\end{aligned}$$

Suppose that the objective is to drive y to zero asymptotically. Differentiate y repeatedly until u appears:

$$\ddot{y} = \cos(x_1)(x_2 + u) - x_1 x_2$$

Note that two differentiations of the output were needed for the input to appear with a coefficient that does not vanish in a neighborhood of the regulation point. We call the number of required differentiations *relative degree*.

Working Example...

Now choose u so that nonlinearities are canceled out and we are left with a simple double-integrator system. Choose:

$$u = \frac{v + x_2(x_1 - \cos(x_2))}{\cos(x_1)}$$

where v is the virtual control input. Substitution into the differential equation for y gives

$$\ddot{y} = v$$

Now choose $v = -y - \dot{y}$ so that the output dynamics become

$$\ddot{y} + \dot{y} + y = 0$$

which is clearly asymptotically stable.

Working Example...

The above control will make $y \to 0$. We can work out the dynamics of the system under the restriction $y = 0$:

$$\begin{aligned} \dot{x}_1 &= 0 \\ \dot{x}_2 &= 0 \\ \dot{x}_3 &= 0 \end{aligned}$$

Therefore x_1 and x_3 will approach constants as x_2 approaches zero, resulting in output regulation with bounded states.

Verify with a simulation.

Feedback Linearization: Non-Working Example

Take the following linear system

$$\begin{aligned}\dot{x}_1 &= -3x_1 - x_2 - x_3 + u \\ \dot{x}_2 &= 4x_1 \\ \dot{x}_3 &= x_2 \\ y &= x_2 - x_3\end{aligned}$$

Suppose that the objective is to drive y to zero asymptotically. The relative degree is again 2. We show that the linear state feedback control

$$u = \frac{1}{4}(12x_1 + 4x_2 + 5x_3)$$

results in

$$\ddot{y} + \dot{y} + y = 0$$

However, the dynamics of the system under the restriction $y = 0$ give

$$\dot{x}_2 = x_2$$

which is unstable. This is an example of a *non-minimum phase* system (unstable zero dynamics).

Using Feedback Linearization

We saw two examples of *input-output linearization*. When control is used to linearize all state derivatives, we have *input-to-state linearization*. Linearizability and stability of the feedback-linearized system can be analyzed with the tools of *Geometric Control Theory*.

Geometric control is usually included in graduate courses in nonlinear systems.

Joint Space Inverse Dynamics

Consider the undamped manipulator dynamic equation

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

The choice to obtain linear dynamics is pretty clear:

$$u = M(q)a_q + C(q,\dot{q})\dot{q} + g(q)$$

where a_q is the virtual control (v in the previous discussion). This leaves the system in the form

$$\ddot{q} = a_q$$

The fundamental difference with the previous two examples is that here we have linearized *all* coordinates. The key to be able to do this is the invertibility of $M(q)$.

Joint Space Inverse Dynamics...

Now choose

$$a_q = \ddot{q}^d - K_0\tilde{q} - K_1\dot{\tilde{q}}$$

If we pick K_0 and K_1 to be diagonal with positive entries, we achieve decoupling and stabilization of the tracking error.
Note that q^d does not have to be constant anymore!
See Eq.(8.28) for a hint on tuning K_0 and K_1. The total control input is obtained by substituting a_q above into the expression for u (8.23).
This approach is referred to as an *inner-loop/outer-loop* architecture.
The inner loop uses u to linearize the system (invert the dynamics), while the outer loop stabilizes the linearized system.

Task Space Inverse Dynamics

In the previous approach the reference inputs are the q^d's. In practice we care about obtaining a definite trajectory for the end-effector rather than the joint angles. To achieve tracking in task space we still use the inner loop so that

$$\ddot{q} = a_q$$

Let X represent the vector of position and orientation of the end-effector relative to the world frame in terms of only six parameters (for instance the 3 rectangular coordinates and the 3 Euler angles). Then

$$\begin{aligned} \dot{X} &= J(q)\dot{q} \\ \ddot{X} &= J(q)\ddot{q} + \dot{J}(q)\dot{q} \end{aligned}$$

where $J = J_a$ is the analytical Jacobian of function $X(q)$ (matrix of partial derivatives). If we choose $a_q = \ddot{q} = J^{-1}(a_X - \dot{J}\dot{q})$ then we obtain a double integrator system in task space:

$$\ddot{X} = a_X$$

Task Space Inverse Dynamics...

Choosing

$$a_X = \ddot{X}^d - K_0(X - X^d) - K_1(\dot{X}_1 - \dot{X}^d)$$

achieves the desired result as before.

If task-space velocity is represented using our usual geometric Jacobian, then we use

$$a_q = J^{-1}(q)(a_{xw} - \dot{J}(q)\dot{q})$$

where a_{xw} is the 6-component virtual control vector. This achieves 6 double-integrators as follows:

$$\begin{aligned} \dot{x} &= a_x \\ \ddot{w} &= a_w \end{aligned}$$

a_x and a_w can be used as before the obtain stable asymptoting tracking. Note that the Jacobian cannot contain singularities, which limits the applicability to 6-joint robots. We can also use a pseudoinverse approach.

State-Space Representation of Robot Dynamics

A state-space representation can be used to facilitate simulation studies. Define states as $z_1 = q$ and $z_2 = \dot{q}$. Then we have

$$\begin{aligned} \dot{z}_1 &= z_2 \\ \dot{z}_2 &= M^{-1}(z_1)\left(u - g(z_1) - (B + C(z_1, z_2))z_2\right) \end{aligned}$$

The state $z = [z_1^T \mid z_2^T]^T$ is now $2n$-by-1. In Matlab, we would write a function evaluating the whole state derivative knowing z and u:

```
function zdot=stateder(t,z,u)
n=length(u);
z_1=z(1:n/2);
z_2=z(n/2+1:2*n);
...
%find numerical values for matrices M, C and calculate state deriva

...
dotz_1=...
dotz_2=...
zdot=[dotz_1;dotz_2];
```