AE/BMED/CS/ECE/ME 7785 Homework

Control Module: Nonlinear Control

Due: Oct. 5, 2011 (Vela in-box in Van Leer)

For both of the problems, code stubs are available on t-square. If you have problems, please e-mail me at pvela@gatech.edu.

Problem 1. (20 pts) Recall the 3R manipulator (three revolute joints), depicted in Figure 1, that I gave as an example of feedback linearization. Well, it's time to follow through on my lectures and check it out for yourself. This problem will focus on seeing how the different types of controllers work. The code stub is available for download from the course website, and needs to be modified to get the simulations that are required below. All you have to do is fill out the control variables in the functions eom1a, eom1b, and eom1c. We want to compare the three similar control strategies.

(a) Run the simulation to convergence with the initial condition $\theta = (0,0,0)^T$ and $\dot{\theta} = (0,0,0)^T$. Let there be a control torque that tries to stabilize the system to $\theta^* = (\pi/8, \pi/5, -\pi/8)^T$ and zero joint velocity. Use the most simplest control law discussed in class,

$$\tau = -K_p e - K_v \dot{\theta}.$$

Choose $K_p = I$, and $K_v = 3I$ where I is the identity matrix. Plot the evolution of θ in one graph and $\dot{\theta}$ in another. How long did it take to converge to within about 10% error?

(b) Do the same as in part (a), but with the more sophisticated controller that tries to invert out some of the dynamics,

$$\tau = \widetilde{M}(\theta) \left(-K_p e - K_v \dot{\theta} \right),$$

where $\widetilde{M}$ is just the diagonal part of the true mass matrix. Choose $K_p = I$, and $K_v = 3I$ where I is the identity matrix. Plot the evolution of θ in one graph and $\dot{\theta}$ in another. How long did it take to converge to within about 10% error?

(c) Do the same as in part (a), but with the more sophisticated controller that tries to invert out some of the dynamics,

$$\tau = M(\theta) \left(-K_p e - K_v \dot{\theta} \right) + C(\theta, \dot{\theta})\dot{\theta} + N(\theta, \dot{\theta}).$$

Choose $K_p = I$, and $K_v = 3I$ where I is the identity matrix. Plot the evolution of θ in one graph and $\dot{\theta}$ in another. How long did it take to converge to within about 10% error?

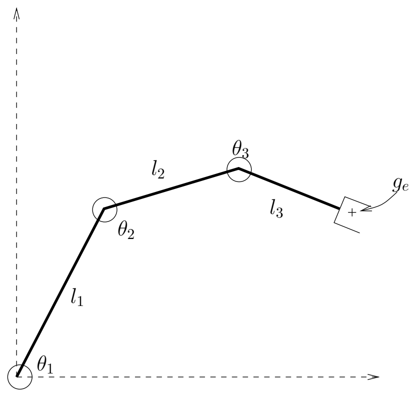

Figure 1: 3R Manipulator.

Problem 2. (50 pts) This homework problems asks for you to actually implement the position-stabilizing controller for the car-like vehicle. It's not as bad as it looks from all of the math. The equations of motion in coordinate form are:

$$\left\{ \begin{matrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{matrix} \right\} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \left\{ \begin{matrix} \nu \\ \omega \end{matrix} \right\}.$$

The position-stabilizing controller that we derived required first extending the state to include the transformation parameter λ,

$$\left\{ \begin{matrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\lambda} \end{matrix} \right\} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -c(\lambda - \epsilon) \end{bmatrix} + \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \left\{ \begin{matrix} \nu \\ \omega \end{matrix} \right\}.$$

and the actual control law was defined to be

$$u = R_\lambda^{-1} v - e_1 \dot{\lambda},$$

where

$$R_\lambda = \begin{bmatrix} \cos(\theta) & -\lambda \sin(\theta) \\ \sin(\theta) & \lambda \cos(\theta) \end{bmatrix} \quad \text{and} \quad e_1 = \left\{ \begin{matrix} 1 \\ 0 \end{matrix} \right\},$$

with

$$v \equiv -Kz$$

for some K such that $K + K^T > 0$ (positive definite) and where z is defined to be

$$z = \left\{ \begin{matrix} x \\ y \end{matrix} \right\} + \lambda R e_1, \quad \text{for} \quad R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}.$$

It may seem complicated, but it is just a succession of simple subsitutions. The best thing is to have the code follow exactly the math and it should work out OK.

If you have any questions, feel free to e-mail or drop by.

(a) Perform the numerical integration with the initial condition x0 = [-1 ; 2.5 ; pi/2 ; 0.2]; and hand in the paramtric plot of the trajectory (x vs. y plots).

(b) Play around with the initial conditions and see how it works. Turn in two additional plots that you found interesting and tell me the initial conditions.

(c) If you are ambitious, then you'll try to track a trajectory. In that case, then you need to do something like,

$$v = -K e_z \quad \text{where} \quad e_z(t) = z(t) - z_{des}(t)$$

where $z_{des}(t)$ is directly obtained from the desired trajectory $g_{des}(t) = (x_{des}(t), y_{des}(t), \theta_{des}(t))$ (no transformation of coordinates, or think of it as $\lambda_{des} = 0$ with no feedback to drive λ to λ_{des}). Notice that in spite of the lack of control on the θ variable, the system ends up doing the appropriate thing. This is what I said about how the system does what you would do using common sense, but requires a fair amount of math to actually prove. The math could have been a little bit shorter, but not much more. **You won't be graded for this part**, it's just so you get comfortable with it, because this is probably the best controller you could do for trajectory tracking, plus changing the gains K and c have physical significance in terms of how the controller stabilizes.