M.E. 530.647 Lab 5

Revision 01

Louis L. Whitcomb*
Department of Mechanical Engineering
G.W.C. Whiting School of Engineering
Johns Hopkins University
`https://dscl.lcsr.jhu.edu/home/courses/530-647-adaptive-systems-fall-2017`

In this lab you will simulate the performance of non-adaptive and adaptive controllers for the 2-DOF revolute-revolute robot you examined in Problem Set 5, Figure 1), for the case where

$$\begin{array}{ll} l_1 & = 1\ m \\ l_2 & = 1\ m \\ m_1 & = 2\ kg \\ m_2 & = 2\ kg. \end{array} \tag{1}$$

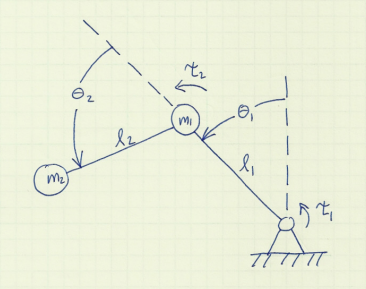

Figure 1: RR Arm: A planar arm with two point-masses and two revolute joints.

Supporting Files

The following matlab m-files are available via a link on the class dropbox web site: https://jh.box.com/v/530-647-Dropbox in a file entitled `rr_arm_mfiles.zip`

rrm.m: The function $rrm(q)$ in file `RRM.M` returns the 2×2 positive definite symmetric inertia matrix $M(q)$.

rrc.m: The function $rrc(q, \dot{q})$ in file `RRC.M` returns the 2×2 coriolis matrix $C(q, \dot{q})$. Note that this $C(q, \dot{q})$ factored such that $(\frac{d}{dt}M(q)) - 2C(q, \dot{q})$ is skew symmetric.

rrg.m: The function $rrg(q)$ in file `RRG.M` returns the 2×1 vector $q(q)$ of joint torques due to gravity when normal $9.8 m/s^2$ gravity is present.

rrdyn.m: The function $rrdyn(q, \dot{q}, \tau)$ in file `RRDYN.M` returns the 2×1 vector $\ddot{q}$ for the robot shown in Figure 1 with joint position vector q, joint velocity vector $\dot{q}$, and joint torque vector τ in the case when gravity is zero.

rrdyng.m: The function $rrdyng(q, \dot{q}, \tau)$ in file `RRDYNG.M` returns the 2×1 vector $\ddot{q}$ for the robot shown in Figure 1 with joint position vector q, joint velocity vector $\dot{q}$, and joint torque vector τ in the case when normal $9.8 m/s^2$ gravity is present.

rrw.m: The function $rrw(\ddot{q}, \dot{q}, \dot{q}, q)$ in file `RRW.M` returns the 2×2 matrix $W(\ddot{q}, \dot{q}, \dot{q}, q)$ where

$$\begin{array}{rl} \tau & = W(\ddot{q}, \dot{q}_1, \dot{q}_2, q)\ \theta \\ & = M(q)\ddot{q} + C(q, \dot{q}_1)\dot{q}_2 + g(q) \end{array} \tag{2}$$

under normal $9.8 m/s^2$ gravity where θ is the plant parameter vector

$$\theta = \begin{bmatrix} m_1 \\ m_2 \end{bmatrix}. \tag{3}$$

rrmhat.m: The function $rrmhat(q, \hat{\theta})$ in file `RRMHAT.M` returns the 2×2 *estimated* inertia matrix $\hat{M}(q, \hat{\theta})$ as computed from the plant parameter estimate $\hat{\theta} = \begin{bmatrix} \hat{m}_1 \\ \hat{m}_2 \end{bmatrix}$.

Lab Problems

1. Simulate the performance of the closed loop system comprising the plant (Figure 1) with gravity and the non-adaptive "computed torque" *exact linearization* inverse dynamics controller for trajectory tracking (with q_d time varying) of the form (as covered in class and on page 449 of the text):

$$\begin{array}{rl} \tau & = M(q)[\ddot{q}_d - K_d \Delta\dot{q} - K_p \Delta q] + C(q, \dot{q})\dot{q} + g(q) \\ & = W([\ddot{q}_d - K_d \Delta\dot{q} - K_p \Delta q], \dot{q}, \dot{q}, q)\ \theta \end{array} \tag{4}$$

using the "critically damped" feedback gains

$$K_p = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix} \quad K_d = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}. \tag{5}$$

Recall that the closed-loop error dynamics for this system takes the form

$$\begin{array}{rl} \dot{e} & = \begin{bmatrix} 0_{2\times2} & I_{2\times2} \\ -K_p & -K_d \end{bmatrix} e \\ \dot{e} & = Ae \end{array} \tag{6}$$

where

$$e = \begin{bmatrix} \Delta q \\ \Delta\dot{q} \end{bmatrix} \quad and \quad A = \begin{bmatrix} 0_{2\times2} & I_{2\times2} \\ -K_p & -K_d \end{bmatrix} e. \tag{7}$$

Write and hand in an m-file called `RRSYS10.M` to simulate this system. Your simulation should simulate the full nonlinear plant and nonlinear controller, do not simply simulate (6). Use a non-trivial reference trajectory.

Does the robot state converge to the desired trajectory? Numerically compute the eigenvalues of $A_{4\times4}$ in matlab. Are your *tracking error plots* consistant with the computed eigenvalues? Why or why not?

What happens to the eigenvalues and the simulation plots when you set K_d to half its original value while using the original K_p?

Hand in annotated simulation plots for one initial condition and trajectory. Be sure to plot $\Delta q(t)$ and $\Delta\dot{q}(t)$ vs time. Clearly state the Lyapunov function from your system, and plots of the value of the Lyapunov function vs time.

2. Simulate the performance of the closed loop system comprising the plant (Figure 1) with gravity and the *adaptive exact linearization* inverse dynamics controller for trajectory tracking (with q_d time varying) of the form (as covered in class and on page 450-452 of the text):

$$\begin{aligned} \tau &= \hat{M}(q)[\ddot{q}_d - K_d\Delta\dot{q} - K_p\Delta q] + \hat{C}(q,\dot{q})\dot{q} + \hat{g}(q) \\ &= W([\ddot{q}_d - K_d\Delta\dot{q} - K_p\Delta q], \dot{q}, \dot{q}, q)\ \hat{\theta} \end{aligned} \tag{8}$$

using the "critically damped" feedback gains given above. and the parameter update law

$$\dot{\hat{\theta}} = -\left[\begin{array}{c} 0_{2\times 2} \\ \hat{M}(q,\hat{\theta})^{-1}\ W(\ddot{q},\dot{q},\dot{q},q) \end{array}\right]^T P\ e \tag{9}$$

where e is as defined above and $P_{4\times 4}$ is a positive definite symmetric matrix which is the unique solution to the Lyapunov equation

$$PA + A^T P = -I_{4\times 4}. \tag{10}$$

Compute the value of P for this case. Compute and clearly show the method you employed as well as the numerical result.

Write and hand in an m-file called `RRSYS11.M` to simulate this system.

First, set the initial conditions $\hat{\theta}(0)$ to θ. Is the tracking error system stable? Asymptotically stable? Is the parameter estimation error system stable? Asymptotically stable?

Second, experiment with initial conditions where $\hat{\theta}(0) \neq \theta$. What happens? Does this agree with theoretical predictions?

What happens if $\hat{\theta}(0) = 0$? Does this agree with theoretical predictions?

What happens if $\hat{\theta}(0) = \left[\begin{array}{c} -1 \\ -1 \end{array}\right]$? Does this agree with theoretical predictions?

Hand in annotated simulation plots for one initial condition and trajectory. Be sure to plot $\Delta q(t)$, $\Delta\dot{q}(t)$, and $\Delta\theta(t)$ vs time. Clearly state the Lyapunov function from your system, and plots of the value of the Lyapunov function vs time.

See me if you have any questions.

Hand in **ANNOTATED** printouts of your plots and printouts of your m-files. Hand in your m-file functions and also email them to me as ZIP file attached to your email. Put "530.647 LAB#5 m-files from YOUR FULL NAME" in the subject line.

Check to verify that the files you hand in run. If your matlab functions call custom matlab m-files that you have written (for this course or otherwise) be sure to include *all* necessary files.