A Design Study Approach to Classical Control

Randal W. Beard Timothy W. McLain
Brigham Young University

Updated: April 27, 2016

Homework A.a

Create a simulink animation of the single link robot arm. The input should be a slider for θ.

Solution

Consider the image of the single link robot arm shown in Figure **??**, where the configuration is completely specified by the angle θ of the arm from vertical. The physical parameters of the system that impact the animation are the arm length ℓ, and the arm width w. The first step in developing the

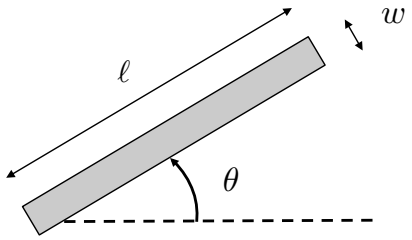

Figure 1: Drawing for single link robot arm. The first step in developing an animation is to draw a figure of the object to be animated and identify all of the physical parameters.

animation is to determine the position of points that define the animation.

For example, for the single link robot arm in Figure **??**, the four corners of the arm, when $\theta = 0$ are given by

$$p_0 = \begin{pmatrix} 0 & \ell & \ell & 0 \\ -\frac{w}{2} & -\frac{w}{2} & \frac{w}{2} & \frac{w}{2} \end{pmatrix},$$

where the first column of p_0 is the lower left corner, the second column is the lower right corner, the third column is the upper right corner, and the last column is the upper left corner. When $\theta \neq 0$, the points are rotated by θ to obtain

$$p(\theta) = R(\theta)p_0,$$

where

$$R = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

is a rotation matrix that rotates points in $\mathbb{R}^2$ by θ.

The Matlab code that draws the arm link and returns a handle to the image of the link is given by

```
1  function handle = drawLink(theta, L, w, handle)
2      pts = [ 0, L, L, 0; -w/2, -w/2, w/2, w/2];
3      R = [cos(theta), -sin(theta); sin(theta), cos(theta)];
4      pts = R*pts;
5      X = pts(1,:);
6      Y = pts(2,:);
7
8      if isempty(handle),
9          handle = fill(X,Y,'b');
10     else
11         set(handle,'XData',X,'YData',Y);
12     end
13 end
```

Line 2 defines the points when $\theta = 0$, and line 3 defines the rotation matrix R. The rotated points are given by line 4, and the associated X and Y coordinates are extracted in lines 5 and 6. Note that in Line 1, `handle` is both an input and an output. If an empty array is passed into the function, then the `fill` command is used to plot the arm in Line 9. On the other hand, if a valid handle is passed into the function, then the arm is redrawn using the `set` command in Line 11.

The main routine for the robot arm animation is listed below.

```
1  function drawArm(u)
2      % process inputs to function
3      theta    = u(1);
4      t        = u(2);
5      % drawing parameters
6      L = 1;
7      w = 0.3;
8      % define persistent variables
9      persistent link_handle
10     % at time 0, initialize plot and persistent vars
11     if t==0,
12         figure(1), clf
13         plot([0,L],[0,0],'k--'); % plot track
14         hold on
15         link_handle = drawLink(theta, L, w, []);
16         axis([-2*L, 2*L, -2*L, 2*L]);
17     % at every other time step, redraw link
18     else
19         drawLink(theta, L, w, link_handle);
20     end
21 end
```

The routine `drawArm` is called from the Simulink file shown in Figure **??**, where there are two inputs: the angle θ, and the time t. Lines 3-4 rename the inputs to θ, and t. Lines 6-7 define the drawing parameters ℓ and w. We require that the graphics handle persist between function calls to `drawArm`. We define a persistent variable `handle` in Line 9. The `if` statement in Lines 11-20 is used to produce the animation. Lines 12-16 are called once at the beginning of the simulation, and draw the initial animation. Line 12 brings the figure 1 window to the front and clears it. Lines 13 and 14 draw the zero angle line as a visual reference. Line 15 calls the `drawLink` routine with an empty handle as input, and returns the handle to the link. Line 16 sets the axes of the figure. After the initial time step, all that needs to be changed are the locations of link. Therefore, in Line 19, the `drawLink` routines are called with the figure handles as inputs.

Complete Matlab and Simulink files are on the wiki associated with this book.

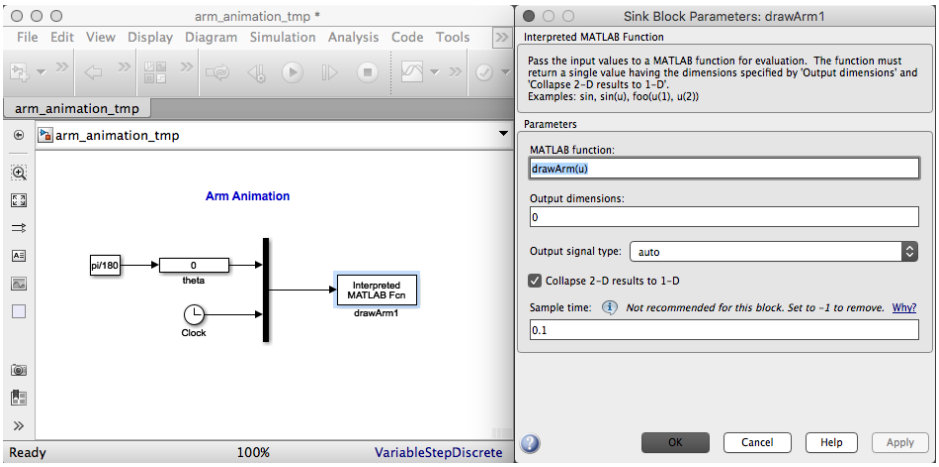

Figure 2: Simulink file for debugging the arm animation. There are two inputs to the Matlab m-file `drawArm`: the angle θ, and the time t. A slider gain for θ is used to verify the animation.