

# Robot Control Exercise 4: State Feedback Control

`inv_pend_test.m`

`segway_test.m`

Nicholas Shindler, [shindler25@gmail.com](mailto:shindler25@gmail.com)

December 3, 2019

## 1 PID Control Stabilization

1) [Matlab]

2) Stabilizing in the initial state of  $r = 0$  we determined a PID calibration of  $k_p = 9.86$ ;  $k_i = 0.048$ ;  $k_d = 3.87$

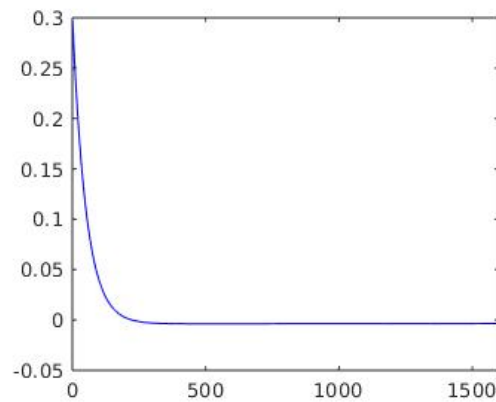


Figure 1: PID calibrated Theta(t) curve

## 2 State Feedback Stabilization

### 2.1 Calculate gain matrix

```
1 syms k1 k2 lambda Theta dTheta;
2
3
4 l = 1; % length of pendulum
5 g = 9.81; % gravity
6 c = 1; % control constant
7 K = [k1 k2]; % gain matrix <— what we want to know;
8 x = [Theta;dTheta];
9 u = -K*x;
10
11 % Using the function: x_dot = A*x - B*K*x
12 % solve for the vector k by obtaining the eigenvalues.
13
14 % ddTheta = (g/l)*sin(Theta) + c*u;
15 % by small angle theorem sin(Th) —> Th
16 % ddTheta = (g/l)*Theta + c*u;
17
18 % calculate A and B values plugging in the above equation
19 A = [0,1;(g/l),0];
20 B = [0;c];
21
22 % calculate THETA dot
23 THETA = A*x + B*u;
24 % THETA:
25 % [ dTheta, dTheta]
26 % [ (981*Theta)/100 - k1*x, (981*Theta*x)/100 - k2]
27
28 I = eye(2); % the identity matrix
29
30 eigValMat = (A - B*K)-lambda*I; % calculate the matrix to be ...
    solved to obtain eigenvalues
31 % eigValMat:
32 % [ -lambda, 1]
33 % [ 981/100 - k1, - k2 - lambda]
34 det_eigMat = det(eigValMat); % take determinite
35 % det_eigMat:
36 % lambda^2 + k2*lambda + k1 - 981/100
37 %
38 % given f(lam) = (lam - lam1)(lam - lam2)
39 % lam1 = -1, lam2 = -2
40 % equivocate to determinite:
41 % lambda^2 + 3*lambda + 2 ==> lambda^2 + k2*lambda + k1 - 981/100
42 % k1 = 2+9.81 = 11.81;
43 % k2 = 3
```

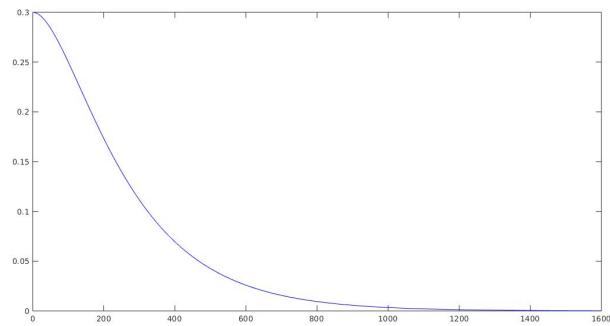


Figure 2: Plotting  $\Theta(t)$  based on  $u = -Kx$

## 2.2 Segway - linear state space

```

1 %% Problem 2.2 %%
2 syms TH dd_X ddTH u;
3
4 % constants
5 g = 9.81;
6 l = 1;
7 m = 1;
8 M = 1;
9
10 % small angle approx
11 sin.TH = TH;
12 cos.TH = 1;
13 dTH2 = 0;
14
15 ddTH = (g/l)*sin.TH+(1/l)*dd_X*cos.TH;
16 ddX = (1/(M+m))*(u + m*l*ddTH*cos.TH - m*l*dTH2*sin.TH);
17 % (9.81*TH)/2 + dd_X/2 + u/2
18 ddX = ddX*2 - dd_X;
19 % 9.81 * TH + u

```

## 2.3 Segway - controllability

```

1 %% Problem 2.3 %%
2 % r = [B AB ... A^1 B] * [u_n-1;...;u_1;u_0]
3 % rank(r) = n
4
5 syms x dx th dth g_;
6 A = [0,1,0,0;0,0,9,0;0,0,0,1;0,0,2*g_,0];

```

```

7 x_ = [x;dx;th;dth];
8 B = [0;1;0;1];
9
10 r = [B A*B A*A*B A*A*A*B];
11 n = rank(r);
12 % n == 4
13 % therefore controllable...

```

## 2.4 Segway - pole placement

```

1 %% Problem 2.4 %%
2 syms k1 k2 k3 k4 lambda;
3 I = eye(4); % the identity matrix
4 K = [k1,k2,k3,k4];
5 u = -K*x_;
6 fx = A*x_+B*u;
7 % fx is equal to:
8 %
9 %          dx
10 %      9*th - dth*k4 - dx*k2 - k3*th - k1*x
11 %          dth
12 % (981*th)/50 - dth*k4 - dx*k2 - k3*th - k1*x
13 eigValMat = (A - B*K)-lambda*I; % calculate the matrix to be ...
14 % eigValMat is equal to:
15 %[-lambda,      1,      0,      0]
16 %[-k1, -k2 - lambda,      9 - k3,      -k4]
17 %[      0,      0,      -lambda,      1]
18 %[-k1,      -k2, 981/50 - k3, -k4 - lambda]
19 det_eigMat = det(eigValMat); % take determinite
20 % 9*k1 - 2*g_*k1 + 9*k2*lambda - 2*g_*lambda^2 + k1*lambda^2 + ...
21 %      k2*lambda^3 + k3*lambda^2 + k4*lambda^3 + lambda^4 - ...
22 %      2*g_*k2*lambda
23 %
24 some = det(lambda*I-A);
25 %lambda = sqrt(2*g_-);
26 eig = -1;
27 l1=eig;l2=eig;l3=eig;l4=eig;
28 f_lam=expand((lambda-l1)*(lambda-l2)*(lambda-l3)*(lambda-l4));
29 %disp(f_lam);
30 % lambda^4 - 4*lambda^3 + 6*lambda^2 - 4*lambda + 1
31 %
32 % given f(lambda) = lambda^4 - 4*lambda^3 + 6*lambda^2 - ...
33 %      4*lambda + 1
34 % lambda^4 + (k2+k4)lambda^3 + (k3+k1-2*g)lambda^2 + (9*k2 - ...
35 %      2*g*k2)lambda + (9*k1 - 2*g*k1)
36 %
37 % -4 = k2+k4
38 % 6 = (k3+k1-2*g)
39 % -4 = (9*k2 - 2*g*k2)
40 % 1 = (9*k1 - 2*g*k1)
41 %
42 K_ = [0,1,0,1,-4;1,0,1,0,6+2*g_-;0,9-2*g_-,0,0,-4;9-2*g_-,0,0,0,1];

```

```

38 k_rd = rref(K_);
39 %disp(k_rd);
40 % [ 1, 0, 0, 0, -1/(2*g_ - 9)]
41 % [ 0, 1, 0, 0, 4/(2*g_ - 9)]
42 % [ 0, 0, 1, 0, -(4*g_^2 + 6*g_ + 53)/(2*g_ - 9)]
43 % [ 0, 0, 0, 1, -(8*(g_ - 4))/(2*g_ - 9)]
44
45 k1 = -0.0942;
46 k2 = 0.3766;
47 k3 = 25.7142;
48 k4 = -4.3766;
49
50 % alt:
51 eig = -1;
52 l1=eig;l2=eig;l3=eig*2;l4=eig*2;
53 f_lam=expand((lambda-l1)*(lambda-l2)*(lambda-l3)*(lambda-l4));
54 %disp(f_lam);
55 % lambda^4 + 6*lambda^3 + 13*lambda^2 + 12*lambda + 4
56 l_4 = 1;
57 l_3 = 6;
58 l_2 = 13;
59 l_1 = 12;
60 l_0 = 4;
61
62 K_ = [0,1,0,1,l_3;1,0,1,0,l_2+2*g;0,9-2*g,0,0,l_1;9-2*g,0,0,0,l_0];
63 k_rd = rref(K_);
64 %disp(k_rd);
65 % [ 1, 0, 0, 0, -4/(2*g_ - 9)]
66 % [ 0, 1, 0, 0, -12/(2*g_ - 9)]
67 % [ 0, 0, 1, 0, (4*g_^2 + 8*g_ - 113)/(2*g_ - 9)]
68 % [ 0, 0, 0, 1, (6*(2*g_ - 7))/(2*g_ - 9)]
69
70 k1 = -0.3766;
71 k2 = -1.1299;
72 k3 = 32.9966;
73 k4 = 7.1299;
74
75 % alt:
76 eig = -1;
77 l1=eig;l2=eig*2;l3=eig*3;l4=eig*4;
78 f_lam=expand((lambda-l1)*(lambda-l2)*(lambda-l3)*(lambda-l4));
79 %disp(f_lam);
80 % lambda^4 + 10*lambda^3 + 35*lambda^2 + 50*lambda + 24
81 l_4 = 1;
82 l_3 = 10;
83 l_2 = 35;
84 l_1 = 50;
85 l_0 = 24;
86
87 K_ = [0,1,0,1,l_3;1,0,1,0,l_2+2*g;0,9-2*g,0,0,l_1;9-2*g,0,0,0,l_0];
88 k_rd = rref(K_);
89 %disp(k_rd);
90 % [ 1, 0, 0, 0, -24/(2*g_ - 9)]
91 % [ 0, 1, 0, 0, -50/(2*g_ - 9)]
92 % [ 0, 0, 1, 0, (4*g_^2 + 52*g_ - 291)/(2*g_ - 9)]
93 % [ 0, 0, 0, 1, (20*(g_ - 2))/(2*g_ - 9)]
94 k1 = -2.2599;

```

```

95 k2 = -4.7081;
96 k3 = 56.8799;
97 k4 = 14.7081;

```

This was tested using 3 sets of pole placement values to get a good result. The initial placement has not been included in the plot as it did not converge. the red line represents the 2<sup>nd</sup> trial ( $\lambda = [1, 1, 2, 2]$ ) and the green represents the 3<sup>rd</sup> ( $\lambda = [1, 2, 3, 4]$ ).

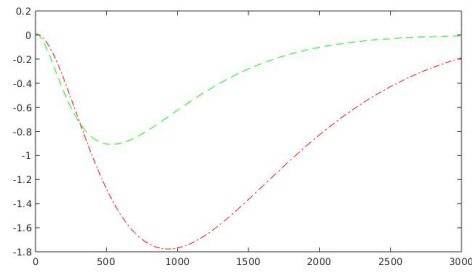


Figure 3: Plotting  $x(t)$  based on  $u = -Kx$

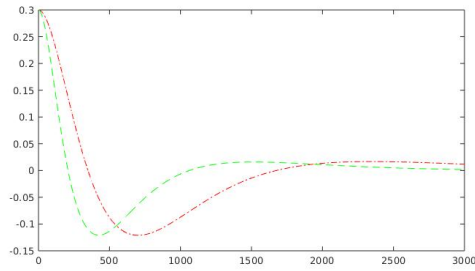


Figure 4: Plotting  $\Theta(t)$  based on  $u = -Kx$

## 2.5 Segway - pole placement

Using the best  $K$  from the previous section we determined that  $x_0$  could be scaled by 3.33 and be able to converge.

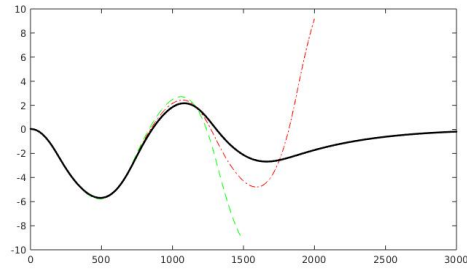


Figure 5:  $x(t)$  based on scale values 3.35(green), 3.34(red), 3.33(black)

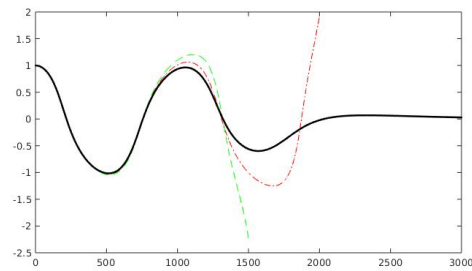


Figure 6:  $\Theta(t)$  based on scale values 3.35(green), 3.34(red), 3.33(black)

## 3 Observer Design

### 3.1 Observably

```

1 % Observability
2 %% 3.1
3 % the system is observable if the rank of the observability ...
  matrix is such that
4 syms g-;
5 A = [0,1,0,0;0,0,9,0;0,0,0,1;0,0,2*g-,0];
6 C = [1 0 0 0];
7 O = [C;C*A;C*(A*A);C*(A*A*A)];
8 % O =
9 % [ 1, 0, 0, 0]
10 % [ 0, 1, 0, 0]
11 % [ 0, 0, 9, 0]
12 % [ 0, 0, 0, 9]
13 % rank(O) = 4

```

```

14 % n = 4 == rank(O) = 4
15 % Observable? yes.

```

## 3.2 Observer Gain Matrix

```

1 %% 3.2
2 syms x dx th dth g_ x_hat L u;
3 A = [0,1,0,0;0,0,9,0;0,0,0,1;0,0,2*g_,0];
4 A_ = [0,1,0,0;0,0,9,0;0,0,0,1;0,0,2*9.81,0];
5 B = [0;1;0;1];
6 C = [1 0 0 0];
7 x_ = [x;dx;th;dth];
8 y = C*x_;
9
10
11 Dx_hat = A*x_hat + B*u + L*(y-C*x_hat);
12
13 syms k1 k2 k3 k4 lambda l1 l2 l3 l4 e_;
14 I = eye(4); % the identity matrix
15 K = [k1,k2,k3,k4];
16 u = -K*x_;
17 Dx = A*x_+B*u;
18
19 e = (x_-x_hat);
20 L = [l1;l2;l3;l4];
21 De = (A-L*C)*e_;
22 % (A-L*C) =
23 % [-l1, 1, 0, 0]
24 % [-l2, 0, 9, 0]
25 % [-l3, 0, 0, 1]
26 % [-l4, 0, 2*g_, 0]
27
28 p = det((A-L*C)-lambda*I);
29 % ACTUAL: p = 9*14 - 2*g_*12 + 9*13*lambda - 2*g_*lambda^2 + ...
% 11*lambda^3 + 12*lambda^2 + lambda^4 - 2*g_*11*lambda
30 % DESIRED: (lambda + 1)^4 = lambda^4 + 4*lambda^3 + 6*lambda^2 + ...
% 4*lambda + 1
31 L_ = [1,0,0,0,4;0,-2*g_,0,0,6;-2*g_,0,9,0,4;0,-2*g_,0,9,1];
32 l_rd = rref(L_);
33 % [ 1, 0, 0, 0, 4]
34 % [ 0, 1, 0, 0, -3/g_]
35 % [ 0, 0, 1, 0, (8*g_-)/9 + 4/9]
36 % [ 0, 0, 0, 1, -5/9]
37 l1 = 4;
38 l2 = -3/9.81;
39 l3 = (8*9.81)+4/9;
40 l4 = -5/9;

```



### 3.3 Scaling

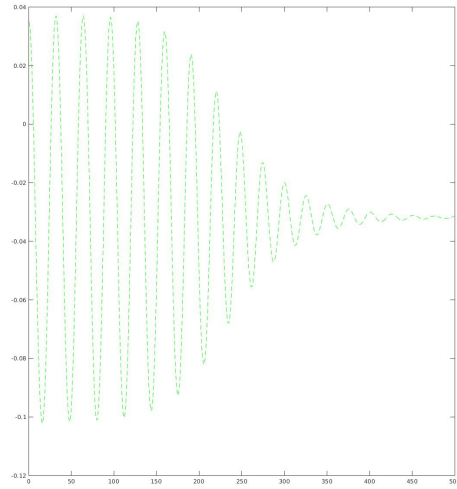


Figure 7:  $x(t)$  based on scale value 3.6

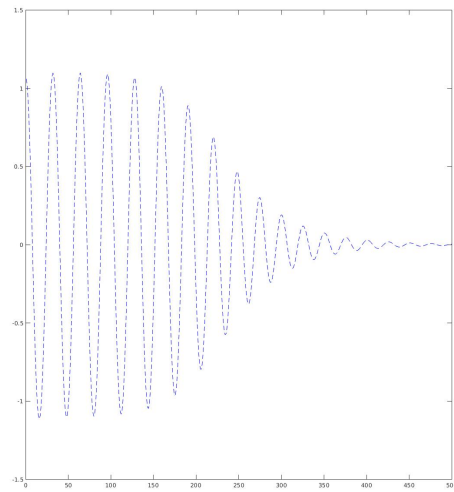


Figure 8:  $\Theta(t)$  based on scale value 3.6

Through testing we determined that the maximal scaling factor was about 3.6.

### 3.4 Initial Scaling

Setting only  $\hat{x}_0$  to be scaled resulted in a much lower potential scaling factor of only about 2.6.

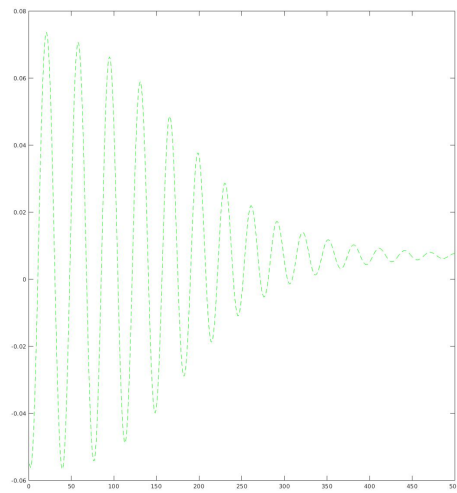


Figure 9:  $x(t)$  based on scale value 2.6 of only  $\hat{x}_0$

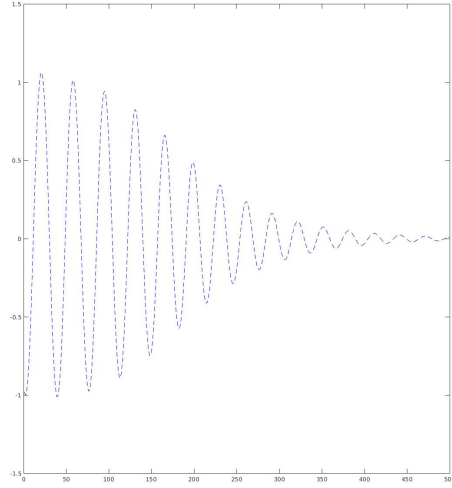


Figure 10: Theta(t) based on scale value 2.6 of only  $\hat{x}_0$

### 3.5 Tilt measurement

By changing the C matrix I needed to change the P matrix to a value that would converge (-36 -36 1 1).

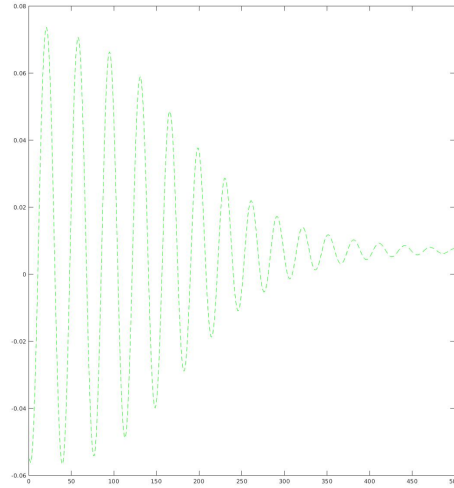


Figure 11:  $x(t)$  with gyro as part of C matrix

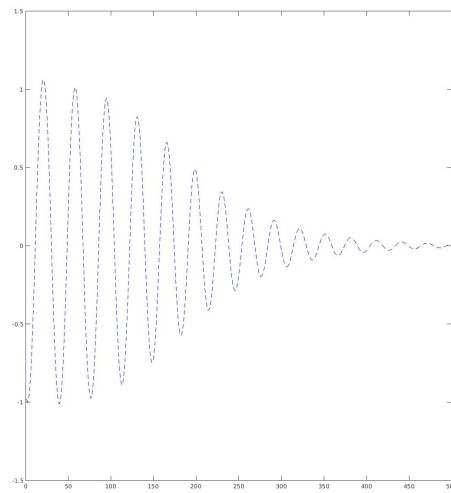


Figure 12:  $\Theta(t)$  with gyro as part of C matrix

I was able to get a higher scaling factor for a global initial scale in this situation.

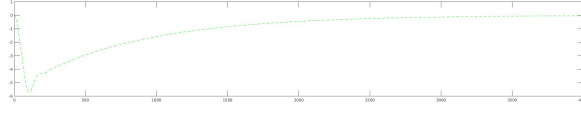


Figure 13:  $x(t)$  with gyro as part of C matrix scaled by 4.33

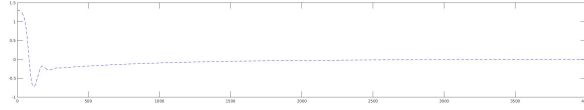


Figure 14:  $\Theta(t)$  with gyro as part of C matrix scaled by 4.33

Setting only  $\hat{x}_0$  to be scaled resulted in a much lower potential scaling factor of only about 2.6.

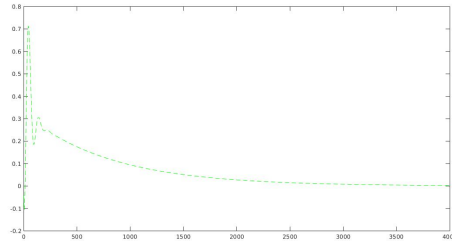


Figure 15:  $x(t)$  with gyro as part of C matrix scaled by 1.25 of only for  $\hat{x}_0$

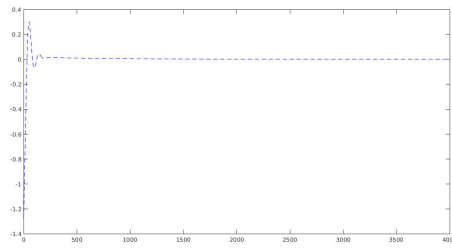


Figure 16:  $\Theta(t)$  with gyro as part of C matrix scaled by 1.25 of only for  $\hat{x}_0$

Setting only  $\hat{x}_0$  to be scaled resulted in a much much lower potential scaling factor of only about 1.25.

### 3.6 Reference Gain

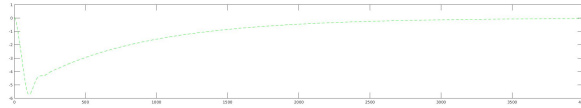


Figure 17:  $x(t)$  with gain matrix  $-r(t) = 1$

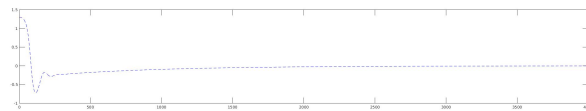


Figure 18:  $\Theta(t)$  with gain matrix  $-r(t) = \sin(t)$