HW5 Problem 1

Table of Contents

Initialize

```
fprintf('\n');
clearvars -except function_list pub_opt
close all
% Bring in answers to compare
hw5_p1_answers
```

a) "Truth" solution

Matlab's ode45 integrator was used, RelTol = 1e-12 and AbsTol = 1e-20. Time interval of 0.01 Time Units used for integration.

```
Xt0 = [1;0;0;1];
num_time_units = 100;
dt = 0.01; %TU
times = 0:dt:num_time_units;

ode_opts = odeset('RelTol', 1e-13, 'AbsTol', 1e-20);
[T,Xout] = ode45(@hw5_deriv, times, Xt0, ode_opts);

% Record only at integer time units from t0
num_record_step=10;
Xti = zeros(num_record_step+1, length(Xt0));
Xti(1,:) = Xt0';

for ii = 1:num_record_step
    Xti(ii+1,:) = Xout(num_record_step*ii/dt+1,:); %+1?
end

%Comparison
% fprintf('Nominal diffs:\n');
% Xti(2,:)'-X_10
% Xti(11,:)'-X_100

fprintf('Truth trajectory, t=10 TU:\n')
for ii = 1:4
```

```
    fprintf('%.9f\n', Xti(2,ii))
end
fprintf('\nTruth trajectory, t=100 TU:\n')
for ii = 1:4
    fprintf('%.9f\n', Xti(11,ii))
end
fprintf('\n')
```

```
Truth trajectory, t=10 TU:
-0.839071529
-0.544021111
0.544021111
-0.839071529

Truth trajectory, t=100 TU:
0.862318872
-0.506365641
0.506365641
0.862318872
```

b) Reference Trajectory

```
Xreft0 = Xt0 - [1e-6; -1e-6; 1e-6;1e-6];
STM = reshape(eye(4),16,1);

[T,Xout] = ode45(@hw5_deriv, times, [Xreft0; STM], ode_opts);

% Record only at integer time units from t0
Xrefti = zeros(num_record_step+1, length(Xt0));
STM_i=zeros(4,4, num_record_step+1); % 4x4xt
Xrefti(1,:) = Xreft0';
STM_i(:,:,1) = eye(4);

for ii = 1:num_record_step
    Xrefti(ii+1,1:4) = Xout(num_record_step*ii/dt+1,1:4); %+1?
    STM_i(:,:,ii+1) = reshape(Xout(num_record_step*ii/dt+1,5:20),4,4);
end

%Comparison
% fprintf('Ref trajectory, STM diffs:\n');
% Xrefti(2,:)'-Xref_10
% STM_i(:,:,2)-STM_10
% Xrefti(11,:)'-Xref_100
% STM_i(:,:,end)-STM_100
% Xti(2,:)'-Xrefti(2,:)'-dX_10
% Xti(11,:)'-Xrefti(11,:)'-dX_100
% STM_i(:,:,2)*(Xti(1,:)'-Xrefti(1,:)')-STM_dX_10
% STM_i(:,:,end)*(Xti(1,:)'-Xrefti(1,:)')-STM_dX_100
fprintf('Reference trajectory, t=10 TU:\n')
for ii = 1:4
    fprintf('%.9f\n', Xrefti(2,ii))
end
```

```
fprintf('\nReference trajectory, t=100 TU:\n')
for ii = 1:4
    fprintf('%.9f\n', Xrefti(11,ii))
end
fprintf('\n')
```

```
Reference trajectory, t=10 TU:
-0.839031098
-0.544071486
0.544076120
-0.839041244

Reference trajectory, t=100 TU:
0.862623360
-0.505843963
0.505845689
0.862623303
```

c) Show STM is symplectic

```
dim=2;
J = [zeros(dim) eye(dim); -eye(dim) zeros(dim)];

inv_STM = -(J*STM_i(:,:,end)*J)';
fprintf('STM inverse, t=100 TU:\n')
disp(inv_STM)

prod = STM_i(:,:,end)*inv_STM;
fprintf('\nSTM*inv(STM), t=100 TU:\n')
for ii = 1:4
fprintf('%.9f %.9f %.9f %.9f\n', prod(ii,1), prod(ii,2), prod(ii,3), ...
    prod(ii,4));
end
```

```
STM inverse, t=100 TU:
   1.0e+02 *

  Columns 1 through 3

   0.012367484371046  -0.001388295702833   0.005751839913344
   2.600263802477707  -1.516392131659342   1.525394552910912
  -2.591544475361987   1.521279107676095  -1.512840323289880
  -0.003746434527943  -0.003667128573764  -0.000696433460503

  Column 4

  -0.000191322894662
   2.606700884422467
  -2.602345144293468
   0.008812356066156


STM*inv(STM), t=100 TU:
```

```
1.000000000 0.000000000 0.000000000 -0.000000000
0.000000000 1.000000000 0.000000000 0.000000000
0.000000000 -0.000000000 1.000000000 0.000000000
0.000000000 0.000000000 0.000000000 1.000000000
```

d) Calculate perturbation vector

The different methods of calculating dX are pretty small (<0.1%), different due to the numerical propagation of the truth, reference, and STM.

```
dX_method1 = Xti(11,:)'-Xrefti(11,:)'
dX_method2 = STM_i(:,:,end)*(Xti(1,:)'-Xrefti(1,:)')
dX_diff = dX_method1 - dX_method2
```

```
dX_method1 =

   1.0e-03 *

  -0.304487398147146
  -0.521677944295695
   0.519951933400931
  -0.304430783532483


dX_method2 =

   1.0e-03 *

  -0.304329028274100
  -0.521766706203722
   0.520042932783209
  -0.304272666369938


dX_diff =

   1.0e-06 *

  -0.158369873046464
   0.088761908026555
  -0.090999382278438
  -0.158117162544905
```

Published with MATLAB® R2013b

HW5 Problem 2

Table of Contents

Initialize

```
fprintf('\n');
clearvars -except function_list pub_opt
close all
% Bring in answers to compare
hw5_p1_answers
```

Find x_est and observation error with least squares

```
y = [1 2 1]';
W = [2 0 0; 0 1 0; 0 0 1];
H = [1 1 1]';
x_bar = 2;
W_bar = 2;

% BLS algorithm, just one pass
lam = W_bar;
N=lam*x_bar;

lam = lam + H'*W*H;
N = N + H'*W*y;

x_est = lam\N

error = y - H*x_est
```

```
x_est =

   1.500000000000000


error =

  -0.500000000000000
   0.500000000000000
  -0.500000000000000
```

Published with MATLAB® R2013b

HW5 Problem 3

Table of Contents

Initialize

```
fprintf('\n');
clearvars -except function_list pub_opt
close all
% Bring in answers to compare
hw5_p1_answers
```

Estimate the initial state

```
rho = [6.37687486186586
    5.50318198665912
    5.94513302809067
    6.30210798411686
    5.19084347133671
    6.31368240334678
    5.80399842220377
    5.45115048359871
    5.91089305965839
    5.67697312013520
    5.25263404969825];
rho_dot = [-0.00317546143535849
     1.17587430814596
    -1.47058865193489
     0.489030779000695
     0.993054430595876
    -1.40470245576321
     0.939807575607138
     0.425908088320457
    -1.47604467619908
     1.42173765213734
    -0.12082311844776];

Y = [rho'; rho_dot'];
W = inv([0.0625 0; 0 0.01]);
x=[0; 0];
W_bar = inv([1000 0; 0 100]);
x_bar = x;

[state_est, BLS_info] = BLS_spring( x, Y, W, x_bar, W_bar );
```

```
fprintf('x0 is %.4f m\n', state_est(1))
fprintf('v0 is %.4f m/s\n', state_est(2))
fprintf('Range RMS is %.3f m\n', BLS_info.RMS(1))
fprintf('Range rate RMS is %.4f m/s\n', BLS_info.RMS(2))
sig_x = sqrt(BLS_info.P0(1,1));
sig_v = sqrt(BLS_info.P0(2,2));
rho_xv=BLS_info.P0(1,2)/sig_x/sig_v;
fprintf('x standard deviation is %.4f m/s\n', sig_x)
fprintf('v standard deviation is %.4f m/s\n', sig_v)
fprintf('correlation of x and v is %.4f m/s\n', rho_xv)
```

```
x0 is 2.9571 m
v0 is -0.1260 m/s
Range RMS is 0.247 m
Range rate RMS is 0.0875 m/s
x standard deviation is 0.0450 m/s
v standard deviation is 0.0794 m/s
correlation of x and v is 0.0427 m/s
```

Published with MATLAB® R2013b

```
function [true_state, BLS_info] = BLS_spring( x, Y, W, x_bar, W_bar )
%BLS_spring Batch least squares solution for spring problem in StatOD book.
%   Use Batch Least Squares algorithm to determine initial state when given
%   some measurements.  There are TODOs where I'd like to genericize this
%   function.
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

num_dim = length(x);
% x0 = x_bar;
x0 = [4.0; 0.2];
for ii = 1:4
    % Setup for this iteration
    STM = eye(num_dim);
    lam = W_bar;
    N=W_bar*x_bar;
    state = [x0; reshape(STM,num_dim*num_dim,1)];
    RMS_accum = [0;0];
    y_hist = zeros(2,1,11);
    H_hist = zeros(2,2,11);

    for measurement = 1:length(Y)
```

Integrate

```
        if measurement ~= 1
            % default to a tenth of a timestep...
            dt = 0.1; % s
            times = 0:dt:1; %Assumes 1s measurements...
            ode_opts = odeset('RelTol', 1e-13, 'AbsTol', 1e-20);
            [T,Xout] = ode45(@hw5_spring_deriv, times, state, ode_opts);
            STM = reshape(Xout(end,num_dim+1:end),num_dim, num_dim);
        end
```

Observations

Function handles would be good here, making it useful for any dynamics for any number of dimensions... G = [compute_range(state); compute_range_rate(state)]; H_tilda = [dRange(state); dRangeRate(state)]; TODO: Handle this a little better I think...

```
        if measurement ~= 1
            x = Xout(end,1);
            v = Xout(end,2);
        else
            x = x0(1);
            v = x0(2);
        end
        % TODO: Not in ideal algorithm...
        h = 5.4; % m
        rho = sqrt(x*x+h*h);
        rho_dot = x*v/rho;
```

```
        G = [rho; rho_dot];
        H_tilda = [x/rho 0; (v/rho - x*x*v/(rho*rho*rho)) x/rho];
        y = Y(:,measurement) - G;
        H = H_tilda*STM;

        % Accumulate
        lam = lam + H'*W*H;
        N = N + H'*W*y;
        if measurement ~= 1
            state = [Xout(end,1:2)'; reshape(STM,num_dim*num_dim,1)];
        end

        % Accumulate residuals
        y_hist(:,:,measurement) = y;
        H_hist(:,:,measurement) = H;

    end
    x_hat = lam\N;

    % Determine RMS
    for measurement = 1:length(Y)
        epsilon = y_hist(:,:,measurement) - H_hist(:,:,measurement)*x_hat;
        %This is per measurement type, not overall RMS error...
        RMS_accum = RMS_accum + epsilon.*epsilon;
    end
    RMS = sqrt(RMS_accum/11);

    % Test for convergence
    if ii ~= 4 %Later a convergence thing can go in here.
        x_bar = x_bar-x_hat;
        x0 = x0 + x_hat;
    end
end

true_state = x0 + x_hat;
BLS_info.RMS = RMS;
BLS_info.P0 = inv(lam);
BLS_info.xhat = x_hat;
end
```

Published with MATLAB® R2013b

```
function fcnPrintQueue( filename )
global function_list;
if exist('function_list', 'var')
    file_in_list = 0;
    for idx = 1:length(function_list)
        if strcmp(function_list(idx), filename);
            file_in_list = 1;
            break
        end
    end
    if ~file_in_list
%         fprintf('%s\n', filename);
        function_list = [function_list; filename];
    end
end
end
```

Published with MATLAB® R2013b

```
function state_dot = hw5_deriv(times, state)
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app
num_states = 4;
x = state(1);
y = state(2);
r = sqrt(x*x + y*y);
r3 = r*r*r;
state_dot = [state(3); state(4); -x/(r3); -y/(r3)];

if length(state) > num_states %Need to integrate the STM
    STM = reshape(state(num_states+1:end),num_states,num_states);

    r5=r3*r*r;
    A = [
        0, 0, 1, 0;
        0, 0, 0, 1;
        (-1/r3+3*x*x/r5), (3*x*y/r5), 0, 0;
        (3*x*y/r5), (-1/r3+3*y*y/r5), 0, 0];

    STM_dot = A*STM;
    state_dot = [state_dot; reshape(STM_dot, num_states*num_states, 1)];
end
```

Published with MATLAB® R2013b

```
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app
X_10 = [-0.839071529 -0.544021111 0.544021111 -0.839071529]';
Xref_10 = [-0.839031098 -0.544071486 0.544076120 -0.839041244]';
STM_10 = [
-19.2963174705 -1.0005919528 -1.5446240948 -20.5922746780
24.5395368984 2.5430400375 3.3820224390 24.9959638293
-26.6284485803 -1.2470410802 -2.0860289935 -27.5413748340
-15.0754226454 -1.4570972848 -2.0011442064 -14.6674122500];

dX_10 = [-0.000040431037 0.000050375590 -0.000055009526 -0.000030284890]';
STM_dX_10 = [-0.000040432624 0.000050374483 -0.000055008811 -0.000030286882]';
dX_min_STM_dX_10 = [0.000000001587 0.000000001107 -0.000000000715 0.000000001992]'

X_100 = [0.862318872 -0.506365641 0.506365641 0.862318872]';
Xref_100 = [0.862623360 -0.505843963 0.505845689 0.862623303]';
STM_100 =[
-151.2840323254 -0.0696433460 -0.5751839913 -152.5394552874
-260.2345144322 0.8812356066 0.0191322895 -260.6700884451
259.1544475393 0.3746434528 1.2367484371 260.0263802508
-152.1279107642 0.3667128574 -0.1388295703 -151.6392131624];

dX_100 = [-0.000304487370 -0.000521677895 0.000519951885 -0.000304430755]';
STM_dX_100 = [-0.000304329028 -0.000521766706 0.000520042933 -0.000304272666]';
dX_min_STM_dX_100 = [ -0.000000158342 0.000000088810 -0.000000091047 -0.0000001580
```

Published with MATLAB® R2013b

```
function state_dot = hw5_spring_deriv( times, state )
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

w2 = (2.5+3.7)/1.5;
state_dot = zeros(6,1);
state_dot(1) = state(2);
state_dot(2) = -w2*state(1);

STM = reshape(state(3:end),2,2); %2x2 for one dimension
A = [0 1; -w2 0];
STM_dot = A*STM;

state_dot(3:6) = reshape(STM_dot,4,1);

end
```

Published with MATLAB® R2013b