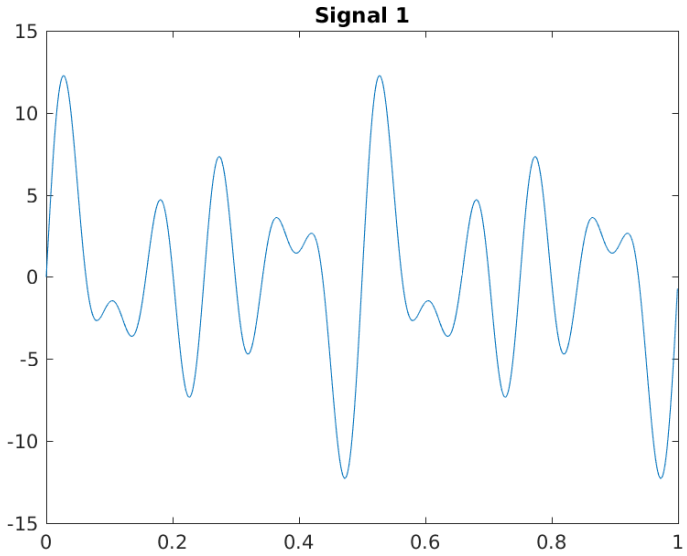
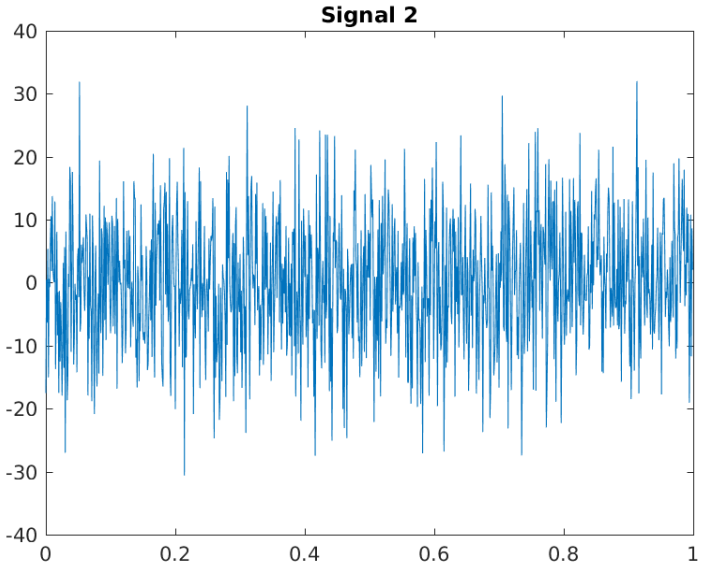
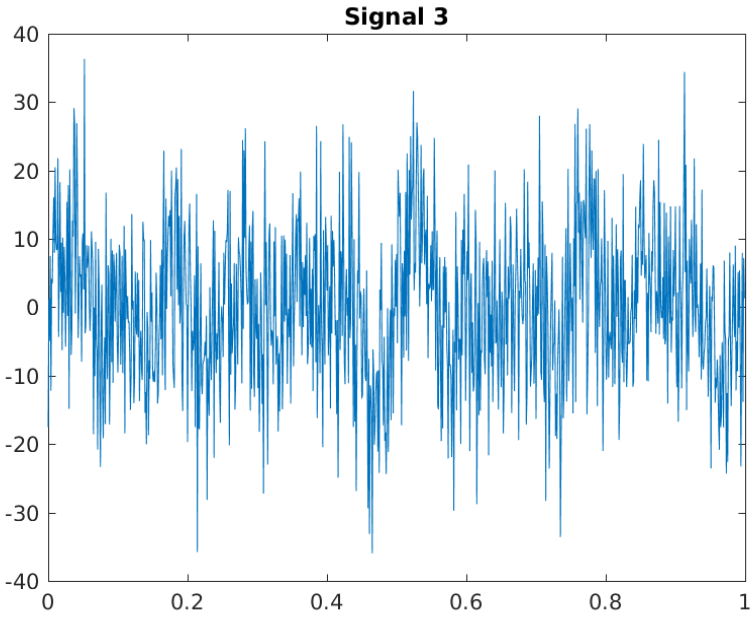
All code and outputs used or referenced can also be found on GitHub at:
https://github.com/sgoodm/apsc608/tree/master/final

Q1

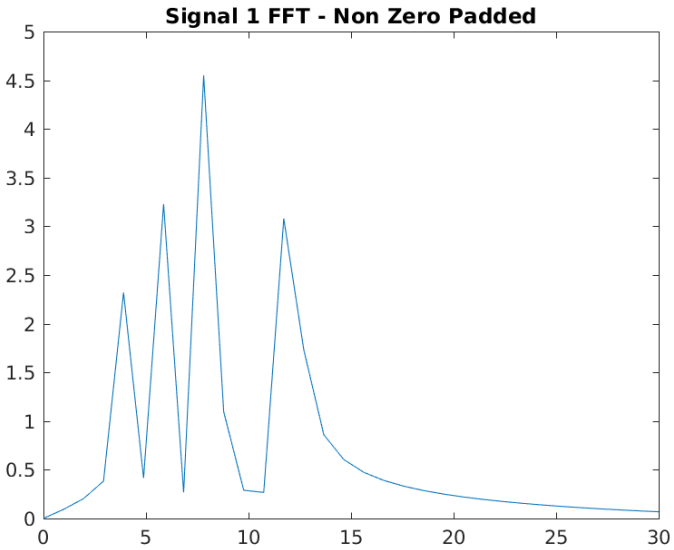
This question will explore the concepts of zero padding and frequency as they relate to three signals, shown in the plots below.. Signal 1, is composed of 4 sinusoids at frequencies of 4, 6, 8, and 12 Hz, with respective amplitudes of 2, 3, 5, and 4. Signal 2 is a normally distributed random signal that will be used as noise to perturb Signal 1. Signal 3 is Signal 1 corrupted by Signal 2 (Signal 1 + Signal 2).



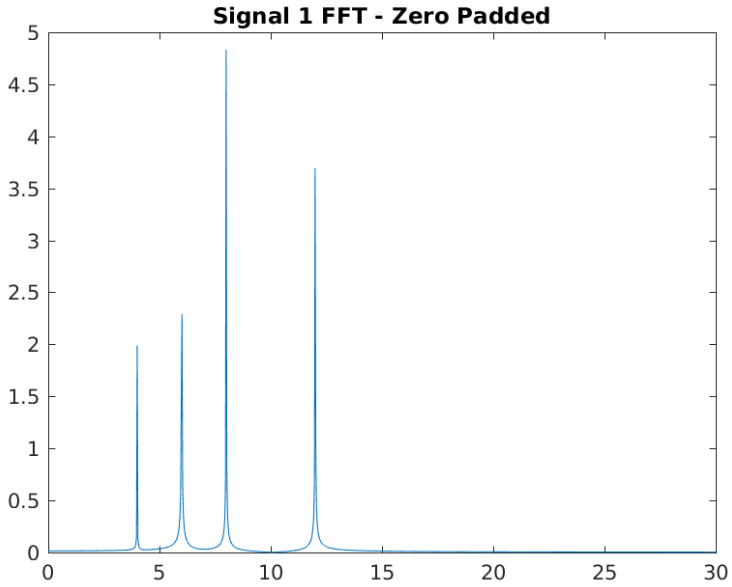




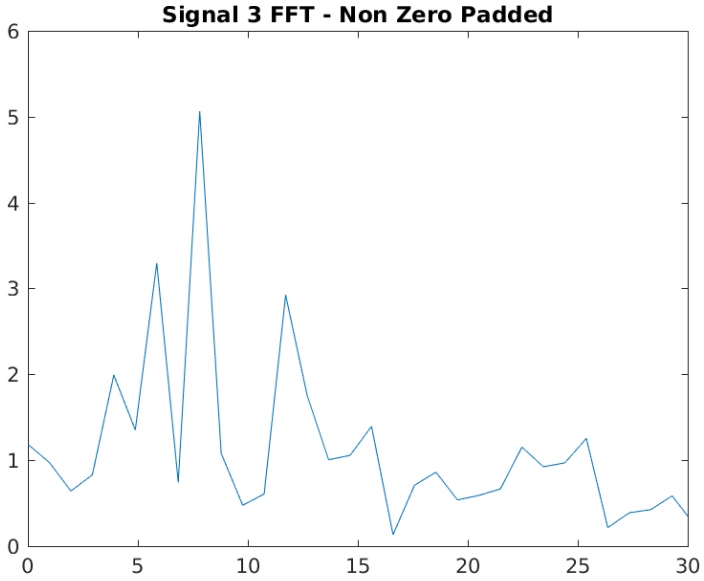
Taking the FFT of Signal 1, it can be seen that while there are clear peaks, they are not very well defined.



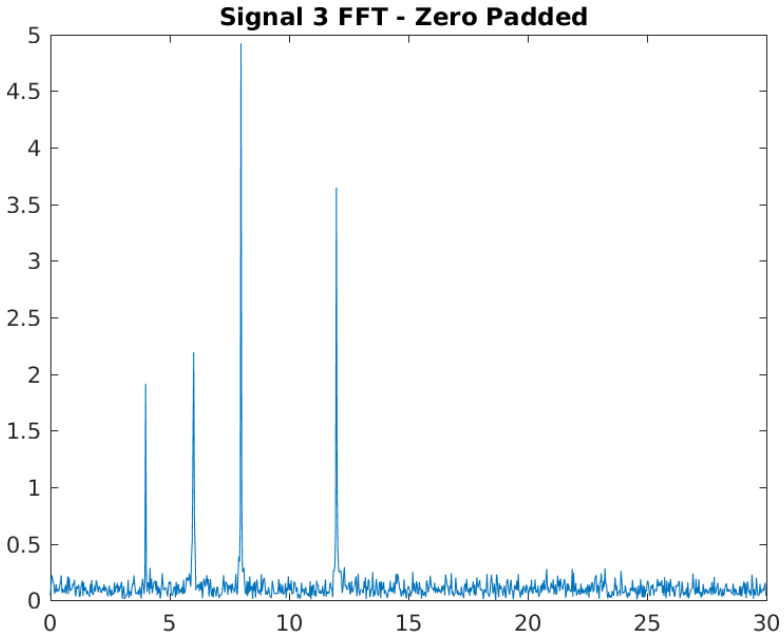
By zero padding this signal before taking the FFT, you can effectively produce an interpolation of the frequency bins utilized in the FFT to generate a more precise visual interpretation of the frequency components of the signal. Since Signal 1 consists of only clean sinusoids, the zero padded FFT clearly shows the frequency components of the signal at 4, 6, 8, and 12 Hz.



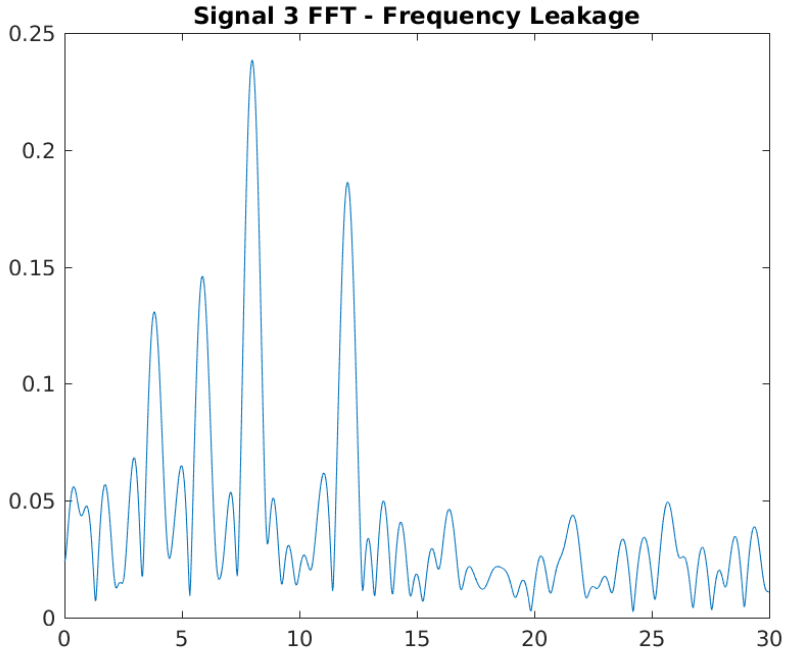
The FFT of Signal 3 has clear markers of the noise introduced due to Signal 2. It is now unclear that the sinusoid at 4Hz is truly present, and “ghost” sinusoids may be detected at 16Hz or other frequencies

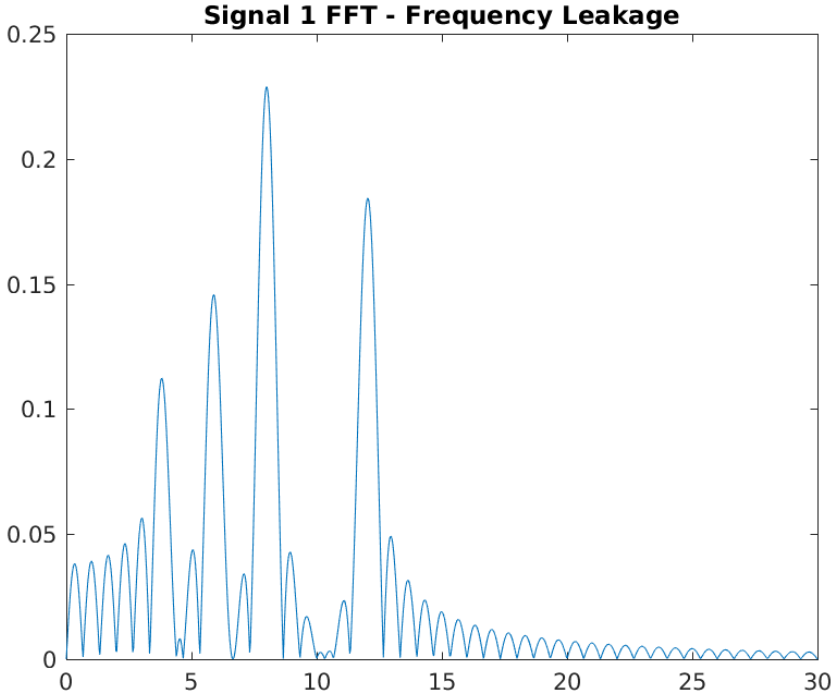


After zero padding Signal 3, the frequency bins used are substantially closer which produced a FFT with clear peaks at the expected frequencies.



Frequency leakage can be seen in both Signal 1 and Signal 3 by decreasing the number of periods provided in the FFT sample. By reducing a periodic signal to small and finite sample, especially when the number of sample periods are not an integer multiple of every component's period, the FFT may be unable to accurately account for the signals. This results in the power attributed to missed frequencies being distributed across other frequencies. In practice, such as with Signal 1 and 3 below, this is seen as a reduction and smoothing of the peaks, with smaller peaks being spread out across nearby frequencies.


Signal 3 FFT - Frequency Leakage
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30


Signal 1 FFT - Frequency Leakage
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30

```
clear
close all

Fs = 1000;
N = 100;
t = 0:1/Fs:N-1/Fs;

x1a = 2 * sin(2*pi*4*t);
x1b = 3 * sin(2*pi*6*t);
x1c = 5 * sin(2*pi*8*t);
x1d = 4 * sin(2*pi*12*t);

x1 = x1a + x1b + x1c + x1d;

% normally distributed random signal
x2 = normrnd(0, 10, [1, length(t)]);

% sine waves combined with normal random signal
x3 = x1 + x2;

figure
plot(t, x1)
title('Signal 1')
xlim([0 1])
saveas(gcf, [pwd, '/outputs/q1_s1.png'], 'png');

figure
plot(t, x2)
title('Signal 2')
xlim([0 1])
saveas(gcf, [pwd, '/outputs/q1_s2.png'], 'png');

figure
plot(t, x3)
title('Signal 3')
xlim([0 1])
saveas(gcf, [pwd, '/outputs/q1_s3.png'], 'png');




x = x1;

n = 2^10;
y = fft(x, n);
m = abs(y) / (n/2);
f = (0:length(y)-1)*Fs/length(y);

figure
plot(f,m)
title('Signal 1 FFT - Non Zero Padded')
xlim([0 30])
saveas(gcf, [pwd, '/outputs/q1_s1a.png'], 'png');


n = 2^15;
y = fft(x,n);
m = abs(y) / (n/2);
```

```
f = (0:length(y)-1)*Fs/length(y);

figure
plot(f,m)
title('Signal 1 FFT - Zero Padded')
xlim([0 30])
saveas(gcf, [pwd, '/outputs/q1_s1b.png'], 'png');


x = x2;

n = 2^10;
y = fft(x, n);
m = abs(y) / (n/2);
f = (0:length(y)-1)*Fs/length(y);

figure
plot(f,m)
title('Signal 2 FFT - Non Zero Padded')
xlim([0 30])
saveas(gcf, [pwd, '/outputs/q1_s2a.png'], 'png');


n = 2^15;
y = fft(x,n);
m = abs(y) / (n/2);
f = (0:length(y)-1)*Fs/length(y);

figure
plot(f,m)
title('Signal 2 FFT - Zero Padded')
xlim([0 30])
saveas(gcf, [pwd, '/outputs/q1_s2b.png'], 'png');


x = x3;

n = 2^10;
y = fft(x, n);
m = abs(y) / (n/2);
f = (0:length(y)-1)*Fs/length(y);

figure
plot(f,m)
title('Signal 3 FFT - Non Zero Padded')
xlim([0 30])
saveas(gcf, [pwd, '/outputs/q1_s3a.png'], 'png');


n = 2^15;
y = fft(x,n);
m = abs(y) / (n/2);
f = (0:length(y)-1)*Fs/length(y);

figure
plot(f,m)
title('Signal 3 FFT - Zero Padded ')
xlim([0 30])
saveas(gcf, [pwd, '/outputs/q1_s3b.png'], 'png');
```

```
N = 1.5;
t = 0:1/Fs:N-1/Fs;

x1a = 2 * sin(2*pi*4*t);
x1b = 3 * sin(2*pi*6*t);
x1c = 5 * sin(2*pi*8*t);
x1d = 4 * sin(2*pi*12*t);

x1 = x1a + x1b + x1c + x1d;

% normally distributed random signal
x2 = normrnd(0, 10, [1, length(t)]);

% sine waves combined with normal random signal
x3 = x1 + x2;


x = x1;

n = 2^15;
y = fft(x,n);
m = abs(y) / (n/2);
f = (0:length(y)-1)*Fs/length(y);

figure
plot(f,m)
title('Signal 1 FFT - Frequency Leakage')
xlim([0 30])
saveas(gcf, [pwd, '/outputs/q1_s1c.png'], 'png');



x = x3;

n = 2^15;
y = fft(x,n);
m = abs(y) / (n/2);
f = (0:length(y)-1)*Fs/length(y);

figure
plot(f,m)
title('Signal 3 FFT - Frequency Leakage')
xlim([0 30])
saveas(gcf, [pwd, '/outputs/q1_s3c.png'], 'png');
```

Q6

Attached tables compare the outputs of Heun's method and the modified Euler method with the basic method used in the Coleman text (Table 11.1) and the actual results across time steps for both n=10 and n=100.

Both Heun's method and the modified Euler method produce nearly identical results, which makes sense since they are both second order Runge Kutta methods. As expected, using a smaller step size (n=100) produces more accurate results.

n=10

t	**Original**	**Heun**	**Euler**	**Actual**
0	5	5	5	5
0.1	4.8	4.81	4.81	4.8096748361
0.2	4.62	4.63805	4.63805	4.6374615062
0.3	4.458	4.48243525	4.48243525	4.4816364414
0.4	4.3122	4.3416039013	4.3416039013	4.3406400921
0.5	4.18098	4.2141515306	4.2141515306	4.2130613194
0.6	4.062882	4.0988071352	4.0988071352	4.0976232722
0.7	3.9565938	3.9944204574	3.9944204574	3.9931706076
0.8	3.86093442	3.8999505139	3.8999505139	3.8986579282
0.9	3.774840978	3.8144552151	3.8144552151	3.8131393195
1	3.6973568802	3.7370819697	3.7370819697	3.7357588823

n=100

t	Original	Heun	Euler	Actual
0	5	5	5	5
0.01	4.98	4.9801	4.9801	4.9800996675
0.02	4.9602	4.960398005	4.960398005	4.9603973466
0.03	4.940598	4.9408920449	4.9408920449	4.9408910671
0.04	4.92119202	4.921580169	4.921580169	4.9215788783
0.05	4.9019800998	4.9024604463	4.9024604463	4.902458849
0.06	4.8829602988	4.8835309649	4.8835309649	4.8835290672
0.07	4.8641306958	4.8647898318	4.8647898318	4.8647876398
0.08	4.8454893889	4.846235173	4.846235173	4.8462326928
0.09	4.827034495	4.827865133	4.827865133	4.8278623705
0.1	4.80876415	4.8096778749	4.8096778749	4.8096748361
0.11	4.7906765085	4.7916715801	4.7916715801	4.7916682706
0.12	4.7727697434	4.7738444478	4.7738444478	4.7738408734
0.13	4.755042046	4.7561946956	4.7561946956	4.7561908618
0.14	4.7374916255	4.7387205584	4.7387205584	4.7387164708
0.15	4.7201167093	4.7214202888	4.7214202888	4.7214159529
0.16	4.7029155422	4.7042921569	4.7042921569	4.7042875779

0.17	4.6858863868	4.68733445	4.68733445	4.6873296332
0.18	4.6690275229	4.6705454722	4.6705454722	4.6705404228
0.19	4.6523372477	4.6539235447	4.6539235447	4.6539182679
0.2	4.6358138752	4.6374670055	4.6374670055	4.6374615062
0.21	4.6194557364	4.6211742088	4.6211742088	4.6211684919
0.22	4.6032611791	4.6050435254	4.6050435254	4.6050375959
0.23	4.5872285673	4.5890733423	4.5890733423	4.589067205
0.24	4.5713562816	4.5732620626	4.5732620626	4.5732557221
0.25	4.5556427188	4.557608105	4.557608105	4.5576015661
0.26	4.5400862916	4.5421099044	4.5421099044	4.5421031716
0.27	4.5246854287	4.5267659108	4.5267659108	4.5267589887
0.28	4.5094385744	4.51157459	4.51157459	4.5115674829
0.29	4.4943441887	4.4965344229	4.4965344229	4.4965271352
0.3	4.4794007468	4.4816439053	4.4816439053	4.4816364414
0.31	4.4646067393	4.4669015485	4.4669015485	4.4668939124
0.32	4.4499606719	4.4523058781	4.4523058781	4.4522980741
0.33	4.4354610652	4.4378554346	4.4378554346	4.4378474669
0.34	4.4211064545	4.423548773	4.423548773	4.4235406455
0.35	4.40689539	4.4093844627	4.4093844627	4.4093761794
0.36	4.3928264361	4.3953610873	4.3953610873	4.3953526521
0.37	4.3788981717	4.3814772445	4.3814772445	4.3814686613
0.38	4.36510919	4.3677315459	4.3677315459	4.3677228184
0.39	4.3514580981	4.354122617	4.354122617	4.354113749
0.4	4.3379435171	4.340649097	4.340649097	4.3406400921
0.41	4.324564082	4.3273096385	4.3273096385	4.3273005003
0.42	4.3113184411	4.3141029076	4.3141029076	4.3140936396
0.43	4.2982052567	4.3010275837	4.3010275837	4.3010181894
0.44	4.2852232042	4.2880823592	4.2880823592	4.2880728422
0.45	4.2723709721	4.2752659397	4.2752659397	4.2752563032
0.46	4.2596472624	4.2625770436	4.2625770436	4.262567291
0.47	4.2470507898	4.250014402	4.250014402	4.2500045366
0.48	4.2345802819	4.2375767587	4.2375767587	4.2375667836
0.49	4.2222344791	4.22526287	4.22526287	4.2252527884
0.5	4.2100121343	4.2130715044	4.2130715044	4.2130613194
0.51	4.1979120129	4.201001443	4.201001443	4.2009911576
0.52	4.1859328928	4.1890514786	4.1890514786	4.1890410959
0.53	4.1740735639	4.1772204164	4.1772204164	4.1772099394
0.54	4.1623328282	4.1655070733	4.1655070733	4.1654965047
0.55	4.1507095	4.1539102779	4.1539102779	4.1538996208
0.56	4.139202405	4.1424288706	4.1424288706	4.1424181277
0.57	4.1278103809	4.1310617033	4.1310617033	4.1310508774
0.58	4.1165322771	4.1198076394	4.1198076394	4.1197967331
0.59	4.1053669543	4.1086655534	4.1086655534	4.1086545695

0.6	4.0943132848	4.0976343311	4.0976343311	4.0976232722
0.61	4.0833701519	4.0867128695	4.0867128695	4.0867017381
0.62	4.0725364504	4.0759000765	4.0759000765	4.0758888752
0.63	4.0618110859	4.0651948707	4.0651948707	4.065183602
0.64	4.0511929751	4.0545961818	4.0545961818	4.0545848481
0.65	4.0406810453	4.0441029497	4.0441029497	4.0440915535
0.66	4.0302742348	4.0337141254	4.0337141254	4.033702669
0.67	4.0199714925	4.0234286699	4.0234286699	4.0234171556
0.68	4.0097717776	4.0132455546	4.0132455546	4.0132339847
0.69	3.9996740598	4.0031637613	4.0031637613	4.0031521381
0.7	3.9896773192	3.9931822819	3.9931822819	3.9931706076
0.71	3.979780546	3.9833001182	3.9833001182	3.9832883949
0.72	3.9699827405	3.973516282	3.973516282	3.9735045119
0.73	3.9602829131	3.963829795	3.963829795	3.9638179802
0.74	3.950680084	3.9542396885	3.9542396885	3.954227831
0.75	3.9411732832	3.9447450036	3.9447450036	3.9447331055
0.76	3.9317615503	3.9353447909	3.9353447909	3.935332854
0.77	3.9224439348	3.9260381102	3.9260381102	3.9260261366
0.78	3.9132194955	3.916824031	3.916824031	3.9168120226
0.79	3.9040873005	3.9077016319	3.9077016319	3.9076895906
0.8	3.8950464275	3.8986700006	3.8986700006	3.8986579282
0.81	3.8860959633	3.8897282341	3.8897282341	3.8897161324
0.82	3.8772350036	3.8808754382	3.8808754382	3.880863309
0.83	3.8684626536	3.8721107276	3.8721107276	3.8720985726
0.84	3.859778027	3.8634332259	3.8634332259	3.8634210469
0.85	3.8511802468	3.8548420653	3.8548420653	3.8548298639
0.86	3.8426684443	3.8463363867	3.8463363867	3.8463241646
0.87	3.8342417599	3.8379153397	3.8379153397	3.8379030985
0.88	3.8258993423	3.829578082	3.829578082	3.8295658234
0.89	3.8176403488	3.8213237801	3.8213237801	3.8213115055
0.9	3.8094639454	3.8131516085	3.8131516085	3.8131393195
0.91	3.8013693059	3.80506075	3.80506075	3.8050484481
0.92	3.7933556128	3.7970503955	3.7970503955	3.7970380822
0.93	3.7854220567	3.7891197441	3.7891197441	3.7891074207
0.94	3.7775678361	3.7812680027	3.7812680027	3.7812556707
0.95	3.7697921578	3.773494386	3.773494386	3.7734820469
0.96	3.7620942362	3.7657981169	3.7657981169	3.765785772
0.97	3.7544732938	3.7581784256	3.7581784256	3.7581660762
0.98	3.7469285609	3.7506345503	3.7506345503	3.7506221977
0.99	3.7394592753	3.7431657365	3.7431657365	3.743153382
1	3.7320646825	3.7357712374	3.7357712374	3.7357588823

```
% q6

% colemen chapter 11.1 #13

clear;
close all;

% format long;

n = 10;

tmin = 0;
tmax = 1;

h = (tmax - tmin) / n;
tsteps = tmin+h:h:tmax;

y0 = 5;

actual = 3 + 2 .* exp(-tsteps);
actual = [y0 actual];

% Table 11.1 Results from Example 1
y1(1) = y0;
for t=1:length(tsteps)
    y1(t+1) =  h * (3 - y1(t)) + y1(t);
end


% Heun's method
y2(1) = y0;
for t=1:length(tsteps)
    y2a = y2(t) + h * (3-y2(t));
    y2(t+1) =  y2(t) + (h/2) * (3-y2(t) + 3-y2a);
end

% modified Euler's method
y3(1) = y0;
for t=1:length(tsteps)
    y3a = y3(t) + 0.5 * h * (3-y3(t));
    y3b = 3-y3a;
    y3(t+1) =  y3(t) + h * y3b;
end

out = [[0 tsteps]; y1; y2; y3; actual]';

output = array2table(out, 'VariableNames',{'t', 'Original', 'Heun', ' Euler',
'Actual'});
writetable(output, [pwd, '/outputs/q6_', num2str(n), '.csv']);
```

Q7

x	t0	t1	t2	t2exact
0	0	0	0	0
0.2	0.6	0.45	0.196875	0.375
0.4	1.2	1.125	0.7875	0.75
0.6	1.8	1.8	1.4625	1.125
0.8	2.4	2.475	2.1375	1.5
1	3	3.15	2.8125	1.875
1.2	3.6	3.825	3.4875	2.25
1.4	4.2	4.5	0	2.625
1.6	0	0	0	3

```
% q7
% colemen 11.2 #6

c = 1;
tstep = 0.1;
xstep = 0.2;

e = c * tstep / xstep;

t = 0:tstep:0.2;
x = 0:xstep:1+length(t)*xstep;

factual = @(x,t) (3.*x) ./ (1 + 3.*t);
actual = factual(x, 0.2);

u = zeros(length(x), length(t));

ic = @(x) 3*x;

for j = 1:length(t)-1
    for i = 1:length(x)-j

        if j == 1
            u(i,j) = ic(x(i));
        end

        if i == 1
            u(i,j) = 0;
        else
            u(i,j+1) = (1-e^2) * u(i,j) + (e/2) * (1+e) * u(i-1,j) + (e/2) * (e-1) *
u(i+1,j);
        end

    end
end

out = [x; u'; actual]';
output = array2table(out, 'VariableNames',{'x', 't0', 't1', ' t2', 't2exact'});
writetable(output, [pwd, '/outputs/q7.csv']);
```

Q8

This question will utilize the SystemID Toolbox to generate a model equivalent to the model generating in HW #6 (continuous system defined by ABCD below).

```
A = [-0.0558 -0.9968  0.0802  0.0415
      0.5980 -0.1150 -0.0318  0
     -3.0500  0.3880 -0.4650  0
      0       0.0805  1.0000  0];

B = [0.0073  0
    -0.4750  0.0077
     0.1530  0.1430
     0       0];

C = [0 1 0 0
     0 0 0 1];

D = [0 0
     0 0];
```

For this example we will use two sinusoidal inputs defined below. One has a frequency of 30 Hz and the other 75 Hz, and both sampled at a frequency of 1000 Hz.

```
Fs = 1000;
N = 1;
Ts = 1/Fs;
t = 0:1/Fs:N-1/Fs;

u1 = sin(2*pi*30*t);
u2 = sin(2*pi*75*t);
u = [u1; u2];
```

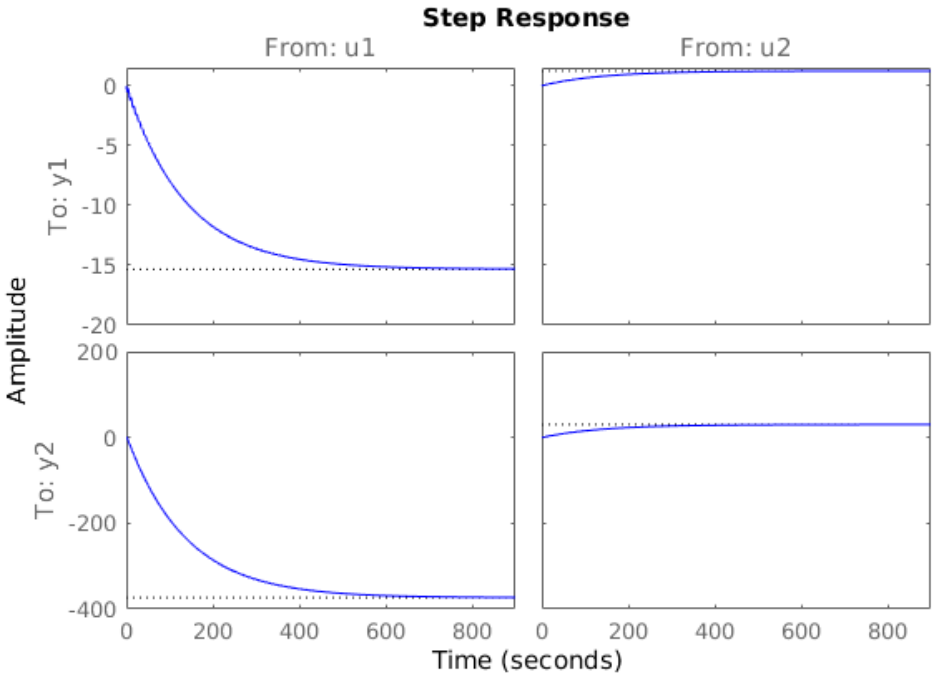
The state space representation of the system is then defined, discretized, and the output generated.

```
% build system
sysc = ss(A ,B, C, D);

% discretize
sysd = c2d(sysc, Ts);

% output
y = lsim(sysd, u);
```

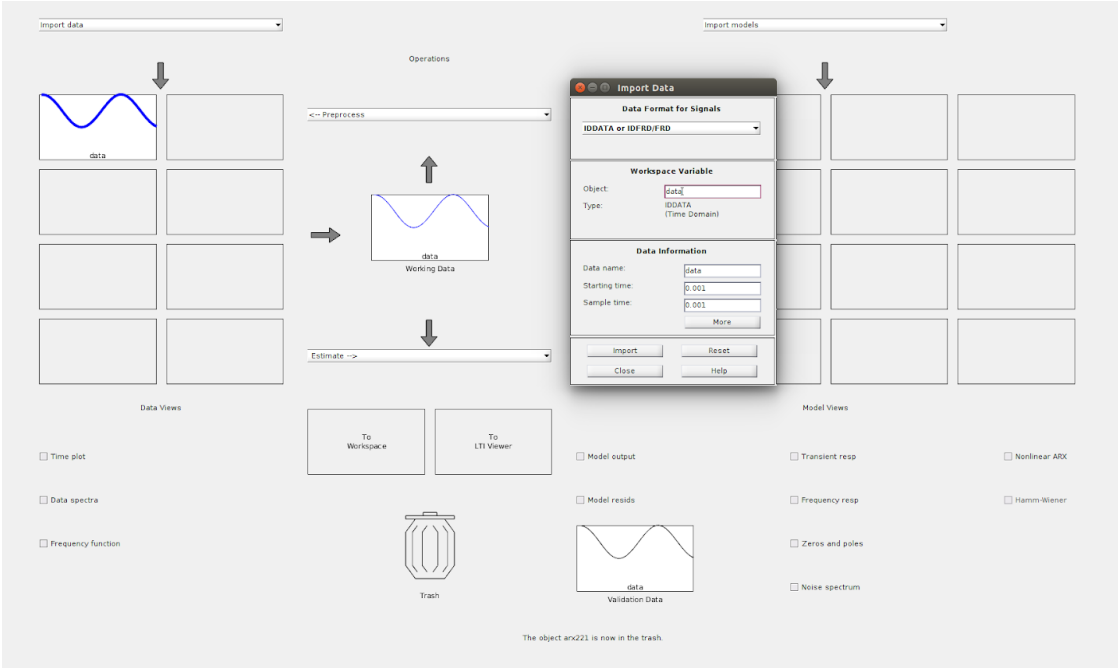
The input and output can be used to generate a data object which will be imported into the SystemID Toolbox.

```
data = iddata(y, u', Ts);
```

Before opening the SystemID Toolbox, we can fist check the step response of this system.

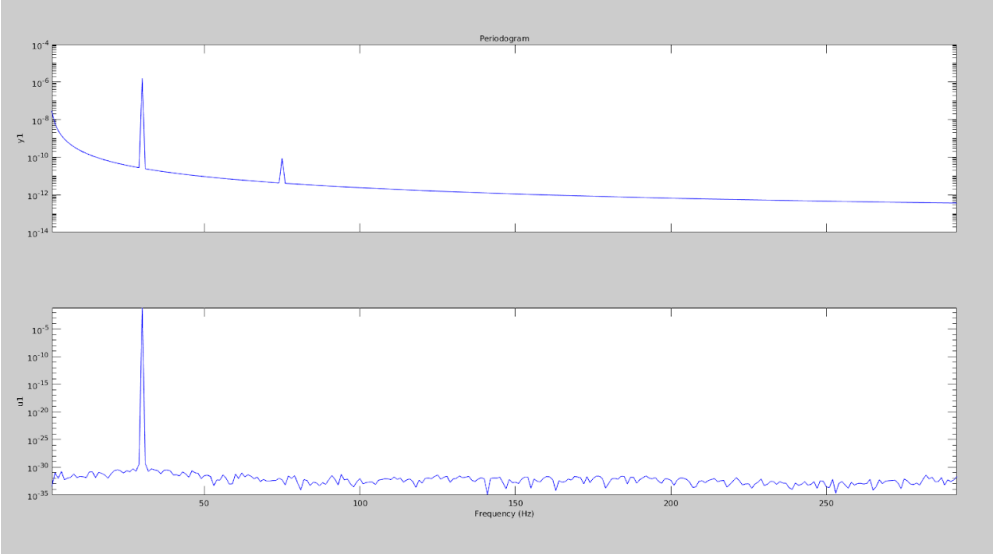


After opening the SystemID Toolbox using the `systemIdentification` command, the data object created earlier needs to be imported.

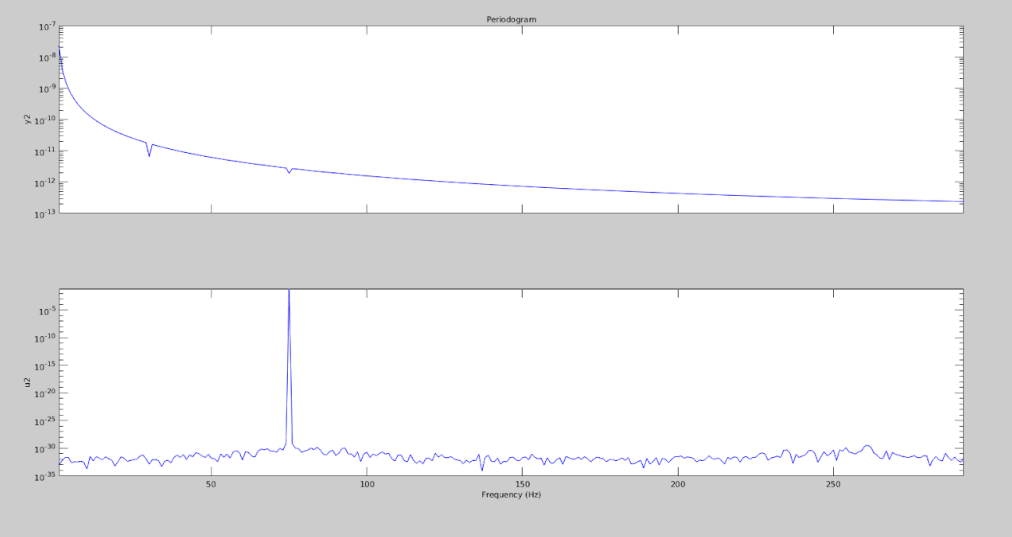


Once the data is loaded, the data spectra plots show the correct frequency for the inputs at 30 Hz and 75 Hz.

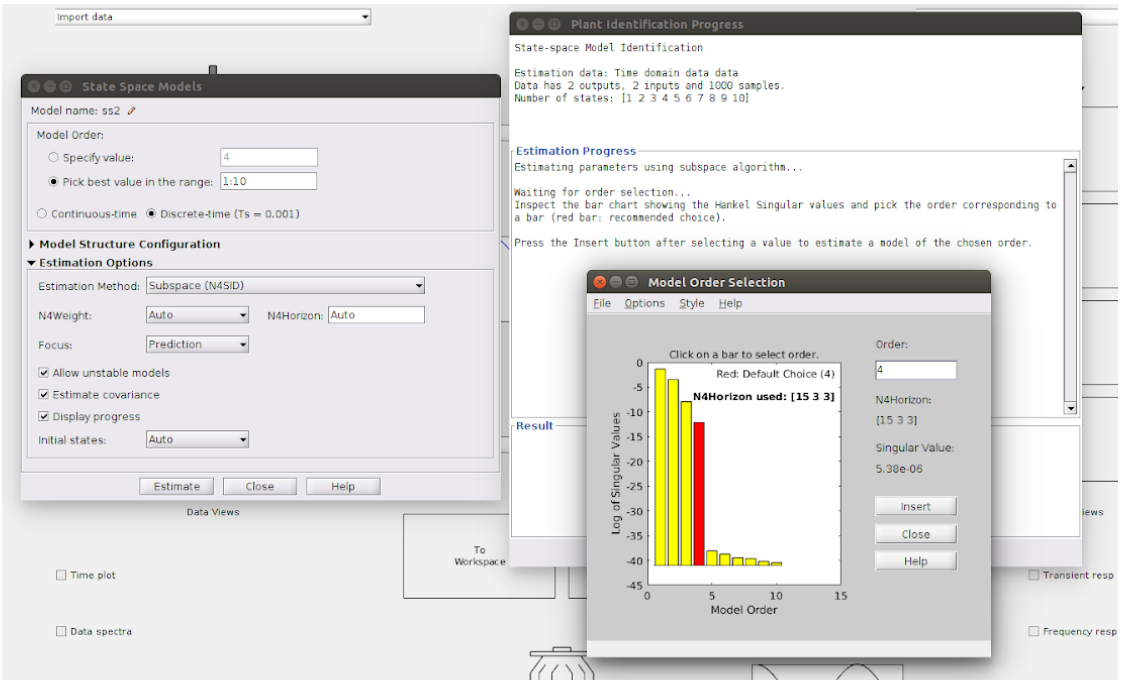
u1 (bottom), y1 (top)



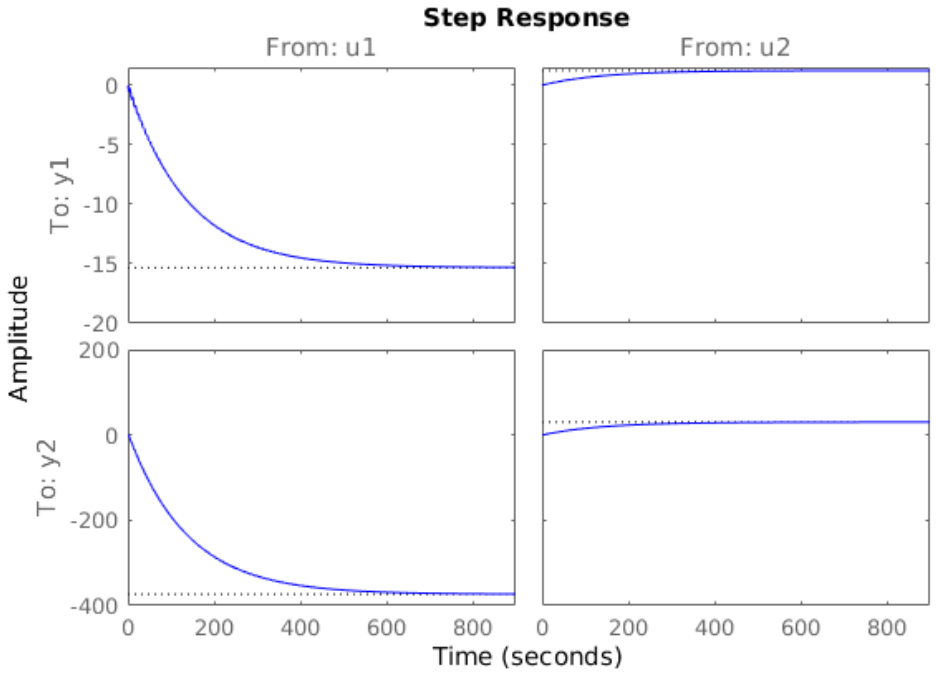
u2 (bottom), y2 (top)



Then the model can be defined by selecting the *State Space Models* option from the *Estimate Menu*. We select the Discrete Time option and the toolbox will assist in choosing the model order.



The step response of the resulting model can then be view in the LTI Viewer.

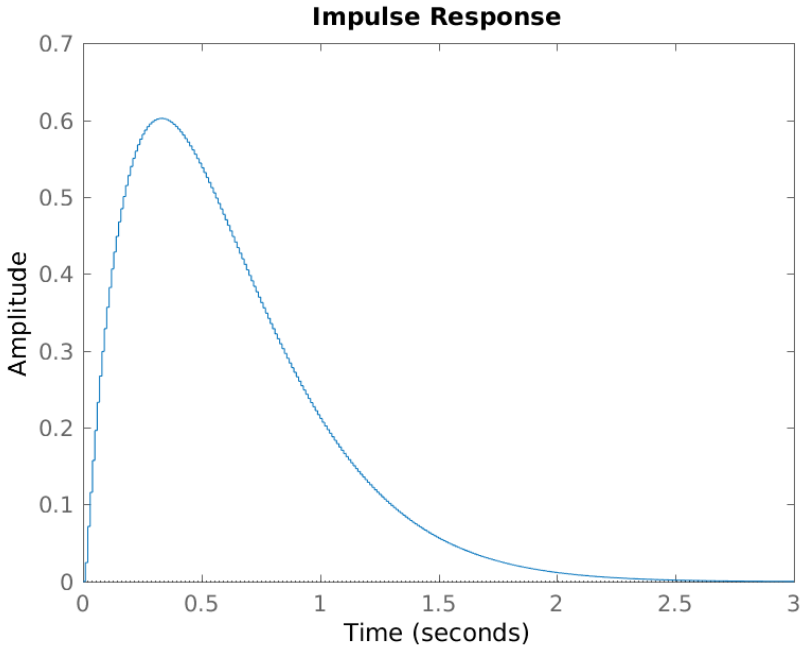


Based on the frequencies and step response of the model generated by the SystemID Toolbox, we can see that the model is equivalent to the original system from HW #6.

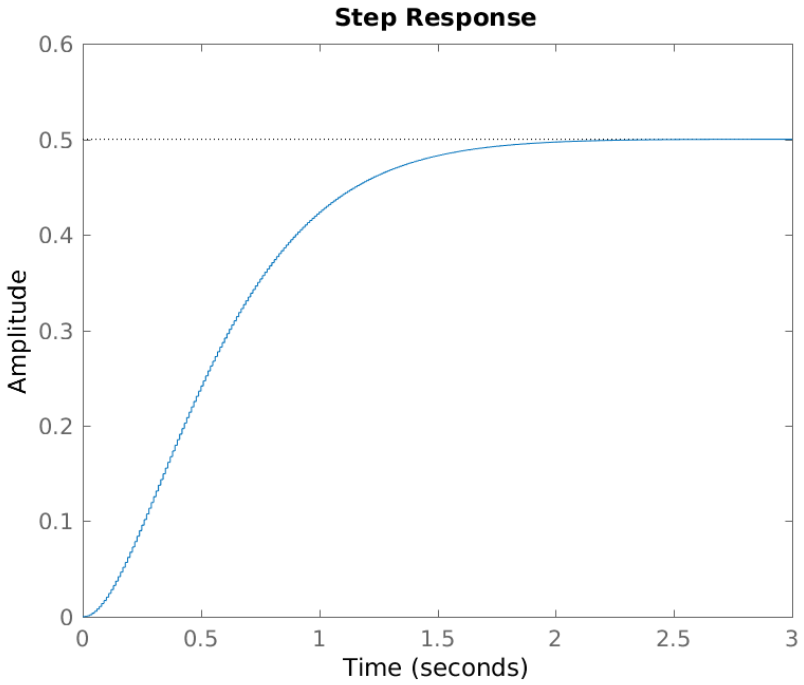
```
% q8

clear;
close all;

% continuous time system defined by ABCD

A = [-0.0558 -0.9968  0.0802  0.0415
      0.5980 -0.1150 -0.0318  0
     -3.0500  0.3880 -0.4650  0
      0       0.0805  1.0000  0];

B = [0.0073  0
    -0.4750  0.0077
     0.1530  0.1430
     0       0];

C = [0 1 0 0
     0 0 0 1];

D = [0 0
     0 0];

% input

Fs = 1000;
N = 1;
Ts = 1/Fs;
t = 0:1/Fs:N-1/Fs;

u1 = sin(2*pi*30*t);
u2 = sin(2*pi*75*t);
u = [u1; u2];

% build system
sysc = ss(A ,B, C, D);

% discretize
sysd = c2d(sysc, Ts);

% output
y = lsim(sysd, u);

% data object
data = iddata(y, u', Ts);


figure()
impulse(sysd)
saveas(gcf, [pwd, '/outputs/q8_impulse.png'], 'png');

figure()
step(sysd)
saveas(gcf, [pwd, '/outputs/q8_step.png'], 'png');

% systemIdentification
```

Q9

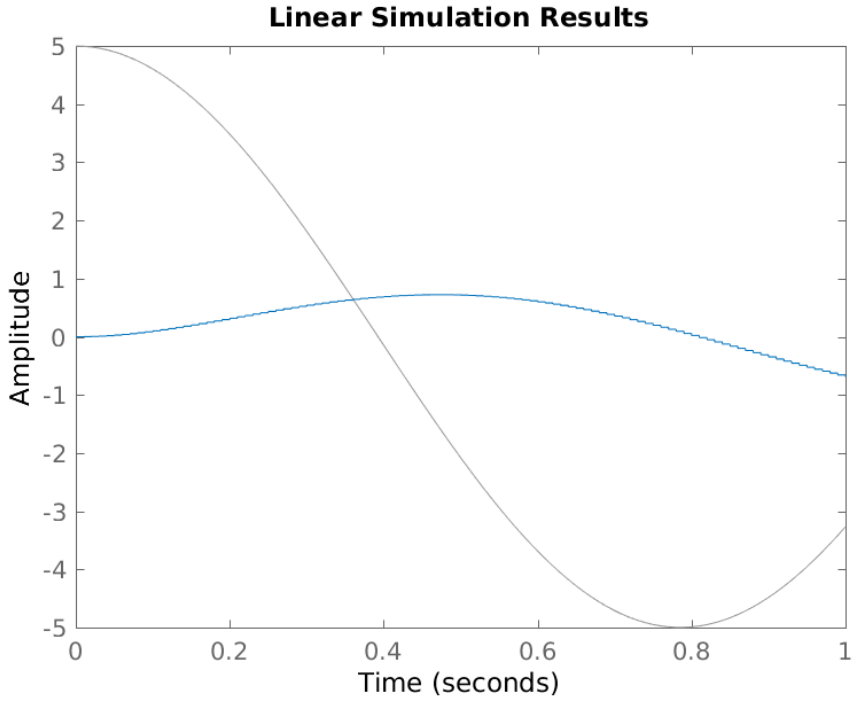
State space impulse response:



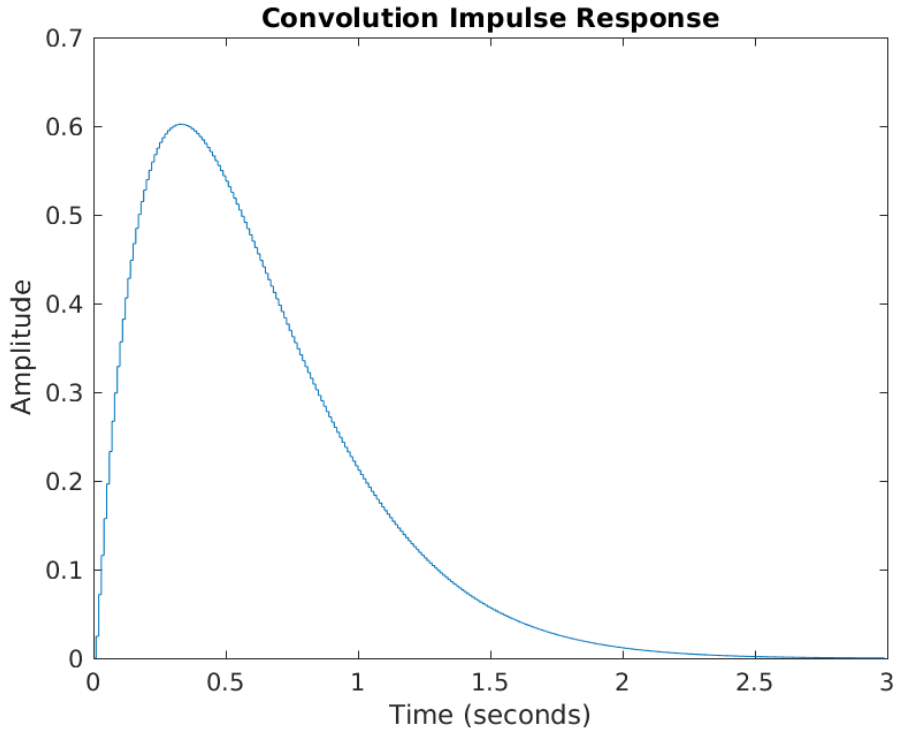
State space step response:



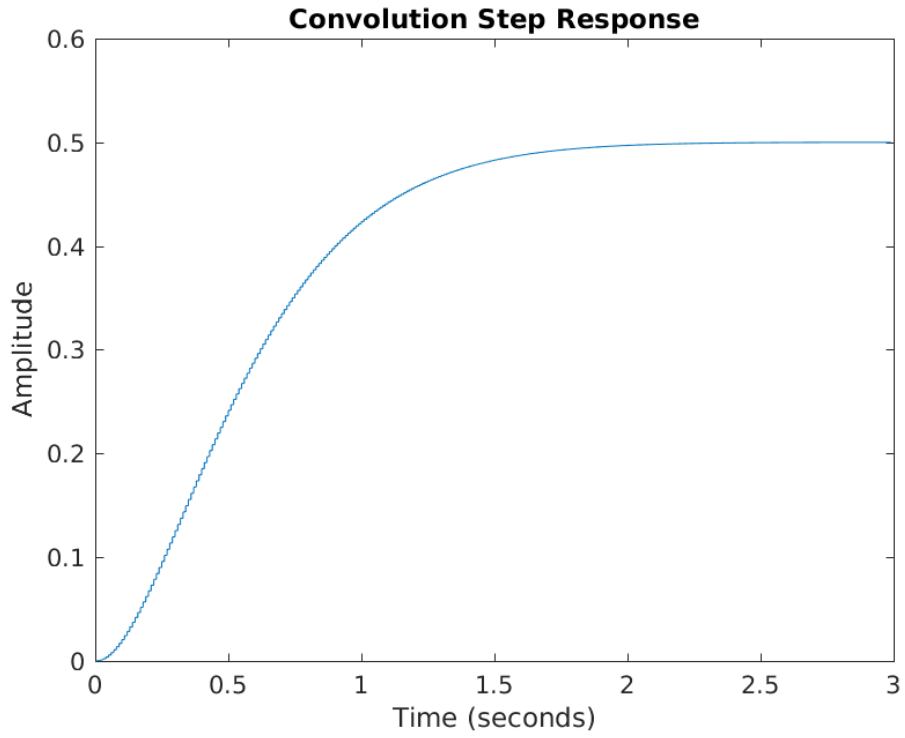
State space response due to input function:



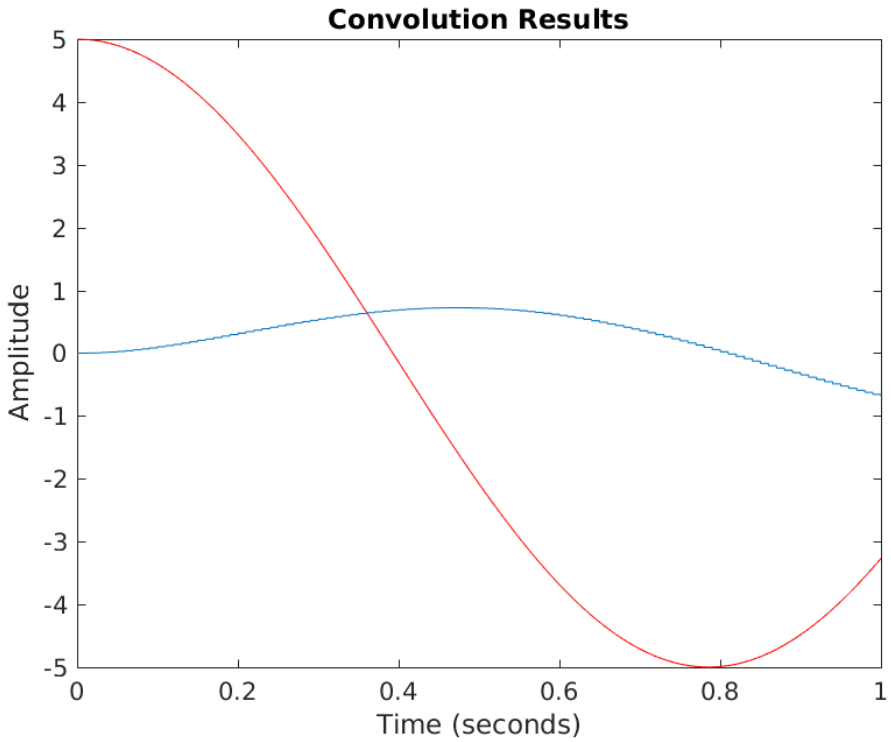
Impulse response generated by convolution:



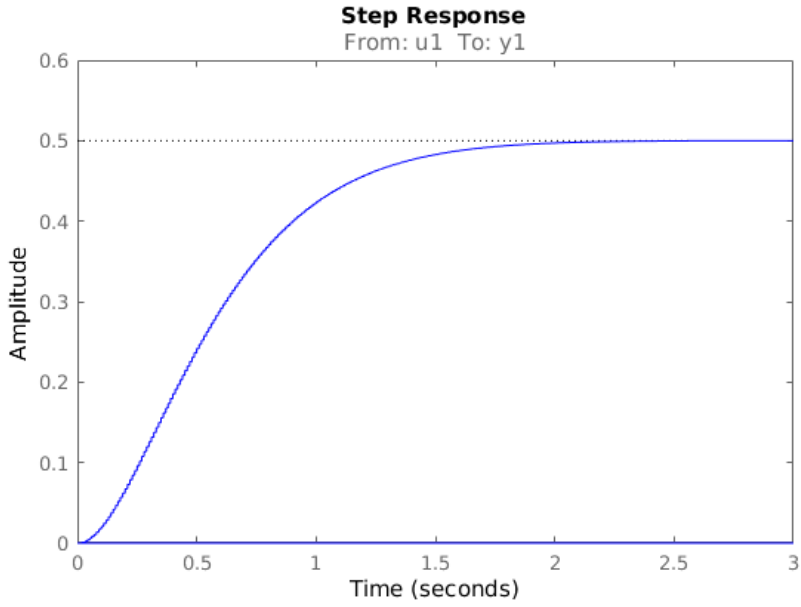
Step response generated by convolution:



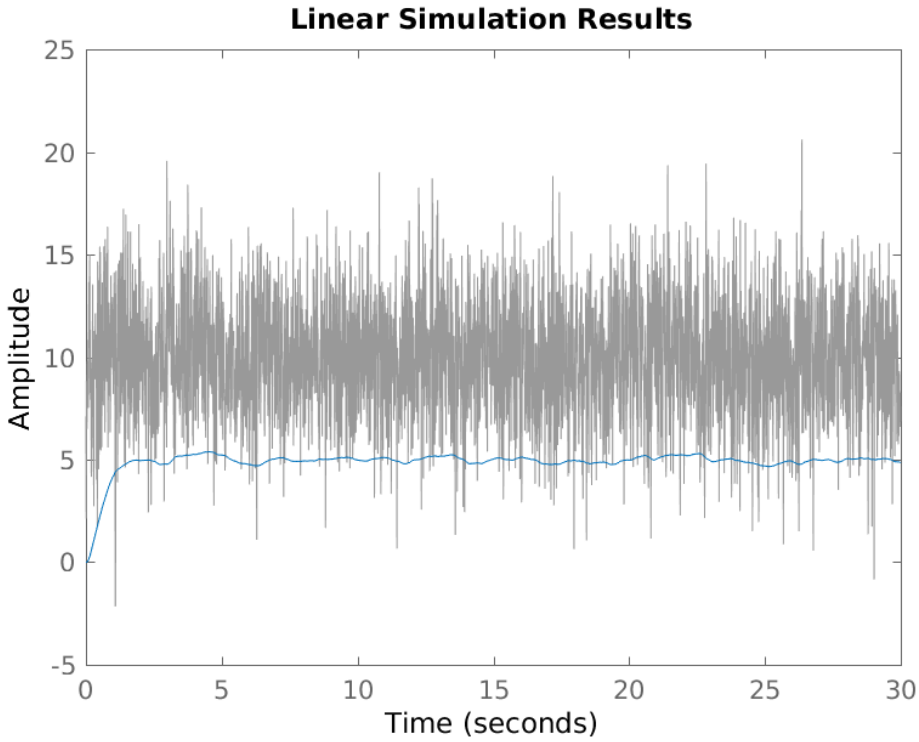
Output due to input function generated by convolution :



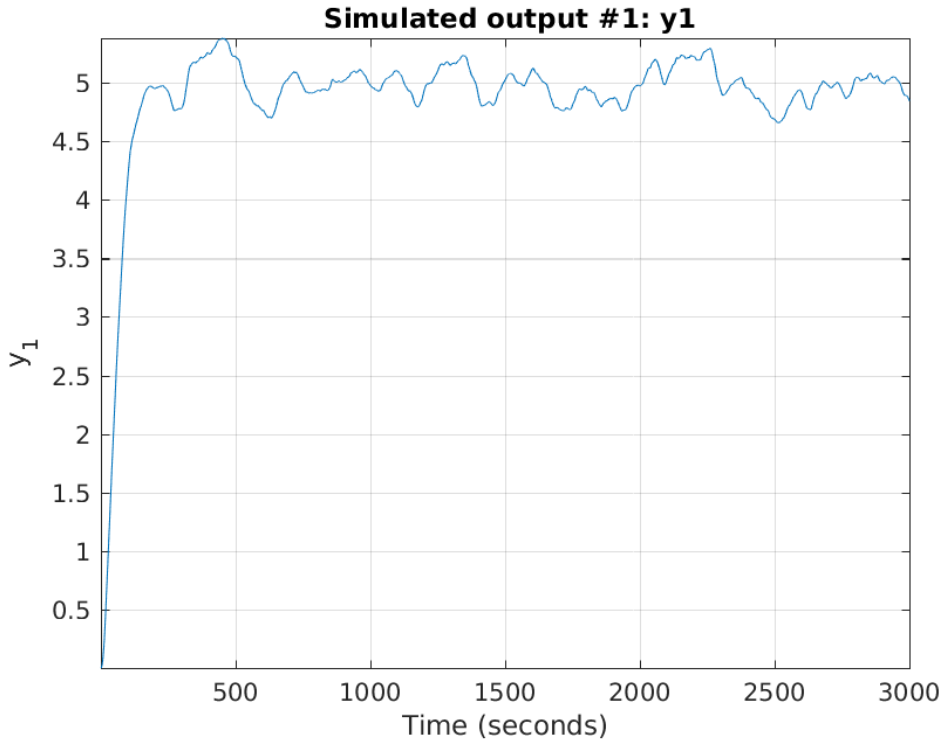
Using the input and output from the system, the SystemID ToolBox can be used to generate an equivalent state space model. The step response of this model, seen below, matches the original step response generated.



Non-zero mean Gaussian input results in an output that is linearly related to the mean of the input.



When applying a random input to a nonlinear system, we would expect an output that is not linearly related to the input.



Note: I was not really sure how to convert an existing linear system into a nonlinear system. I tried using the nlarx estimation of the SystemID Toolbox to generate a nonlinear system model but the new model behaves similarly to the linear model. The plot above is based on the nlarx model.

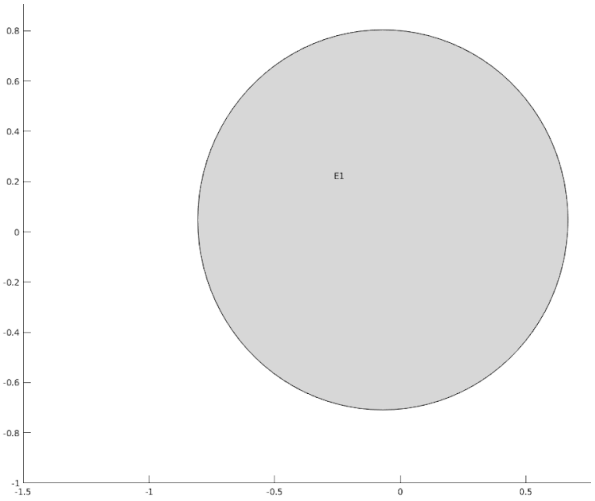
```
% q9

% https://www.mathworks.com/help/ident/gs/about-system-identification.html
%
http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=SystemModeli
ng#5

% clear;
close all;

m = 0.2;
k = 2;
c = 1.2;
f = @(t) 5 .* cos(4.*t);

A = [0 1; -k/m -c/m];
B = [0 1/m]';
C = [1 0];
D = [0];
```

```
sysc = ss(A, B, C, D);

% define time step
Ts = 0.01;
t = 0:Ts:1;

% input
u = f(t);

% discretize system
sysd = c2d(sysc, Ts);

% output
y = lsim(sysd, u);

data = iddata(y, u', Ts);

figure()
impulse(sysd)
saveas(gcf, [pwd, '/outputs/q9_impulse.png'], 'png');

figure()
step(sysd)
saveas(gcf, [pwd, '/outputs/q9_step.png'], 'png');

figure()
lsim(sysd, u)
saveas(gcf, [pwd, '/outputs/q9_func.png'], 'png');

% ------------------------------------------

[h, tc] = impulse(sysd, length(u)-1);


yci = conv(h', [1 0 0 0 0 0 0]);

figure()
stairs(tc(1:300), yci(1:300))
ylabel("Amplitude")
xlabel("Time (seconds)")
title("Convolution Impulse Response")
saveas(gcf, [pwd, '/outputs/q9_conv_impulse.png'], 'png');

ycs = conv(h', ones([length(h) 1])) * Ts;

figure()
stairs(tc(1:300), ycs(1:300))
ylabel("Amplitude")
xlabel("Time (seconds)")
title("Convolution Step Response")
saveas(gcf, [pwd, '/outputs/q9_conv_step.png'], 'png');

ycf = conv(h', u);

figure()
plot(tc(1:101), u, 'r-')
hold on
stairs(tc(1:101), ycf(1:101)*Ts)
ylabel("Amplitude")
```

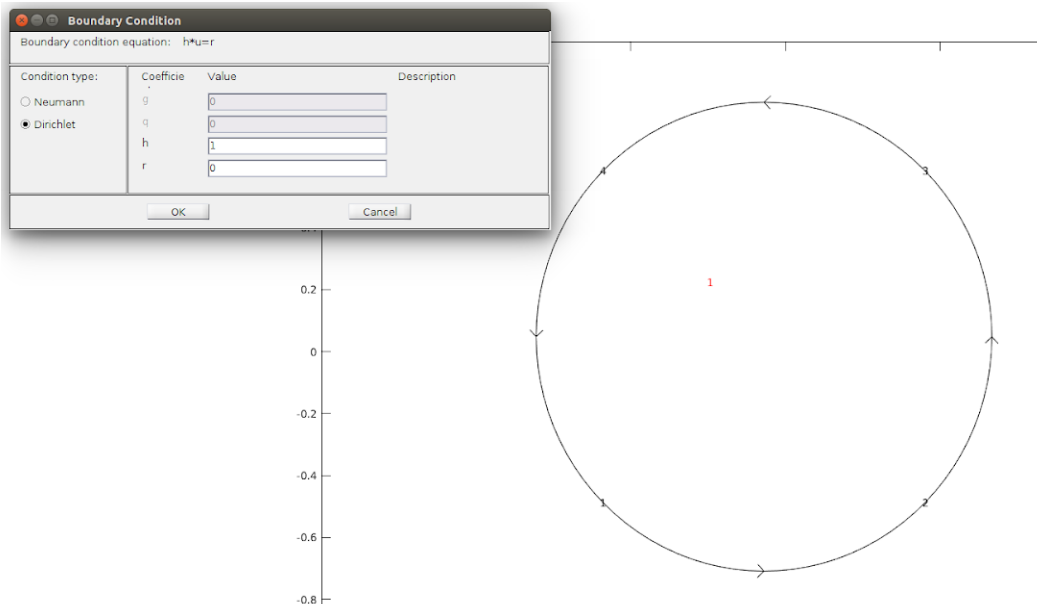
```
xlabel("Time (seconds)")
title("Convolution Results")
saveas(gcf, [pwd, '/outputs/q9_conv_func.png'], 'png');

% ------------------------------------------

ug = normrnd(10, 3, [1, 3000]);

figure()
lsim(sysd, ug)
saveas(gcf, [pwd, '/outputs/q9_gauss_func.png'], 'png');

% figure()
% sim(nlarx1, ug')
% saveas(gcf, [pwd, '/outputs/q9_nonlinear.png'], 'png');
```

Q10

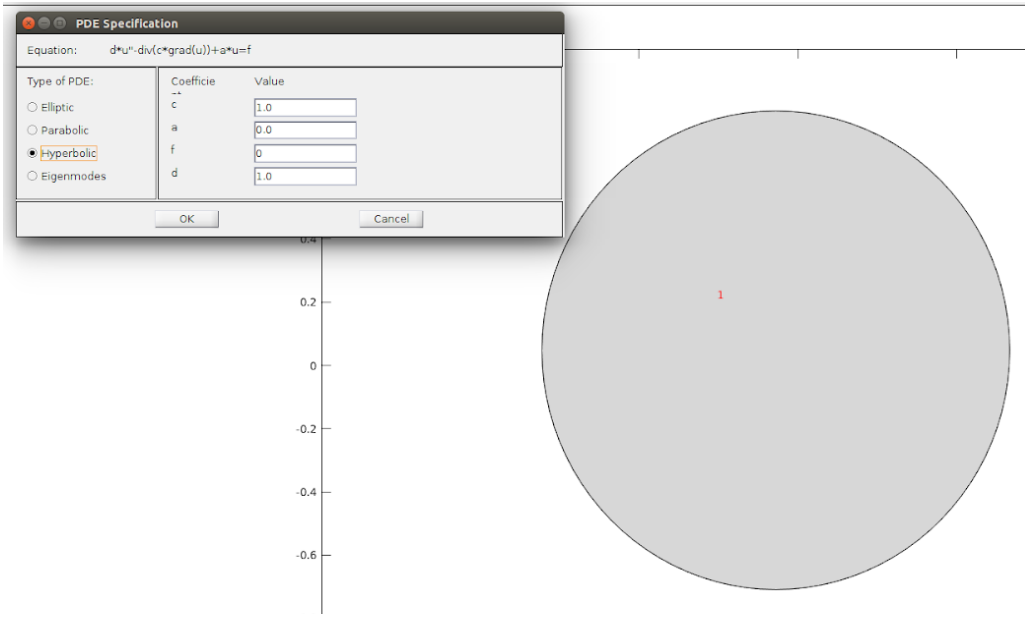
This question will utilize the pdetool toolbox in Matlab to demonstrate the effect of grid density and time step for a wave equation over a circular domain.



After drawing the circle (above), the boundary conditions are initialized (below).

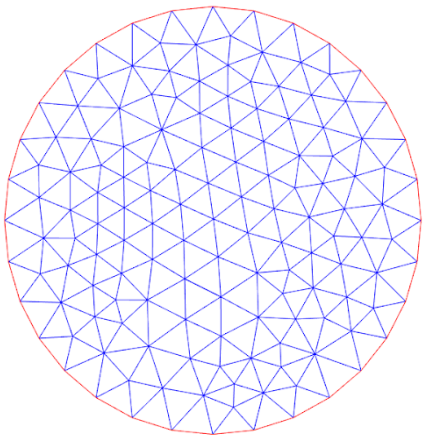


Then the PDE is defined. Since we are exploring the wave equation we will use the hyperbolic option and set appropriate coefficients (below).

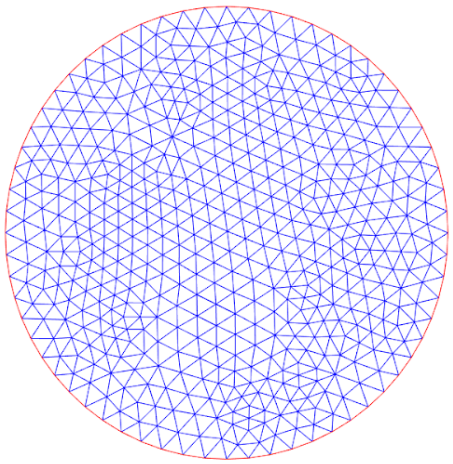
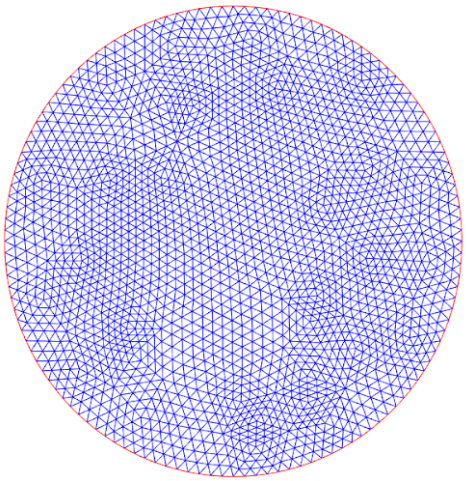


Next the mesh (grid) will be initialized.

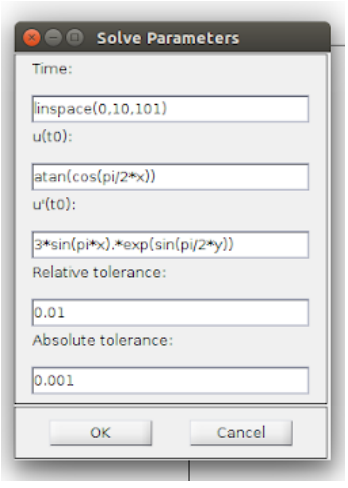
Grid A (Initial Mesh)
147 nodes and 260 triangles



Two additional layers of mesh refinement (grid density) will also be tested (both shown below).

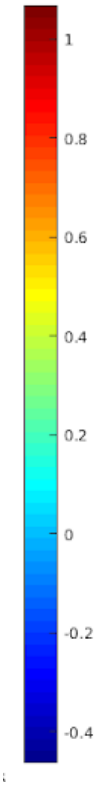
Grid B (First Refinement) 553 nodes and 1040 triangles	Grid C (Second Refinement) 2145 nodes and 4160 triangles
	

Each grid will then be tested using varying time steps. This can be specified in the solve parameters using linspace by specifying the start time, end time, and number of steps.



(Other parameters based on https://www.mathworks.com/help/pde/examples/wave-equation-on-a-square-domain.html)

Output color scale:



Both increasing grid density and decreasing time steps improve the effectiveness of the pdetool solution by increasing the precision of calculations - spatially for grid density, and temporally for time steps. This is illustrated in the following page(s) as best as possible in static format, but to best visualize these effects you should load the accompanying q10_pdetool.m file and use the animation feature to view the solution plots under the varying conditions.

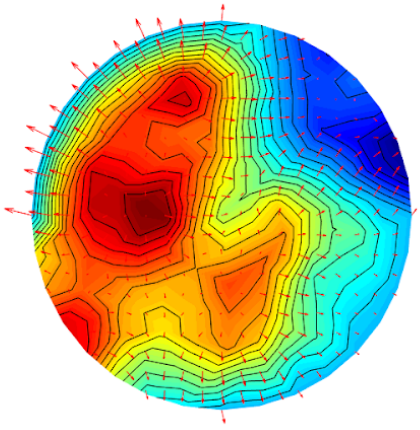
Greater grid density effectives enables finer resolution mapping over the domain. In Grid A, the lowest grid density, there are fewer and much smoother transitions , while in Grid C, with the greatest grid density, there are many more sharper transitions which reflect the true behavior across the domain in greater detail. This is shown at two arbitrary time steps (t=5 and t=10) to illustrate the differences between grid densities.

A similar effect can be observed when adjusting the time step. When comparing time steps of 0.1 and 1.0 at t=5 (stepping to t=5.1 and t=6.0 respectively) we see much more abrupt transitions between steps at the larger time step. With the smaller time step, the changes between each step illustrate a clear pattern of flow more indicative of the true process being modeled.

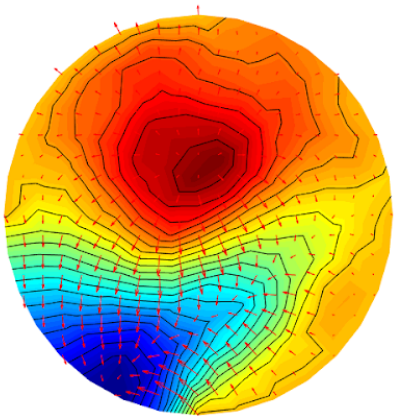
Grid Density

Grid A

T = 5

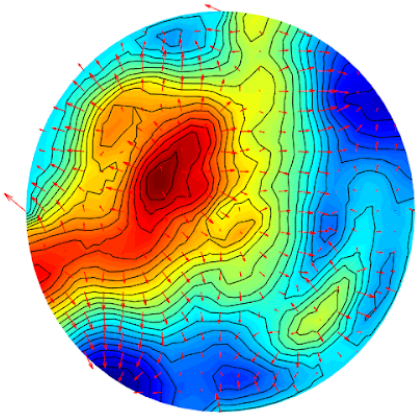


T = 10

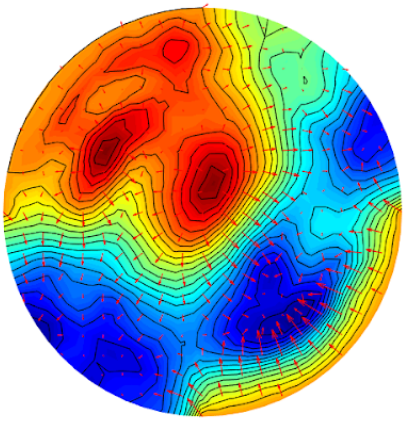


Grid B

T = 5

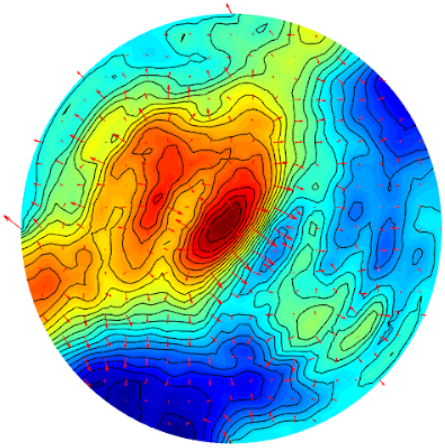


T = 10

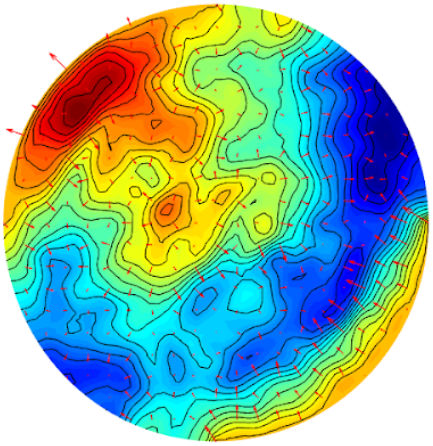


Grid C

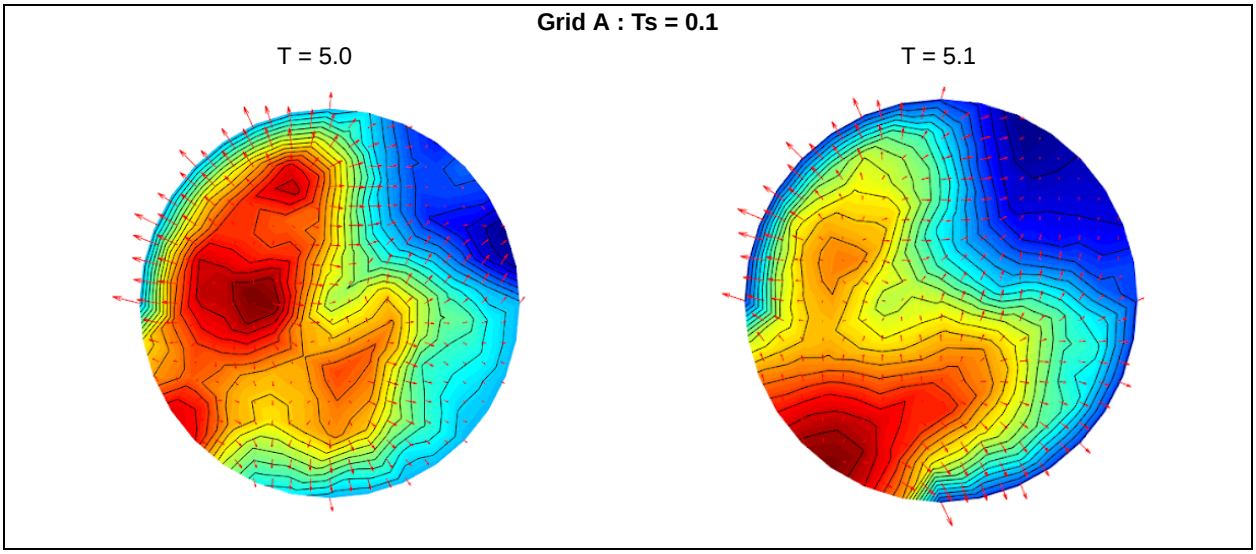
T = 5

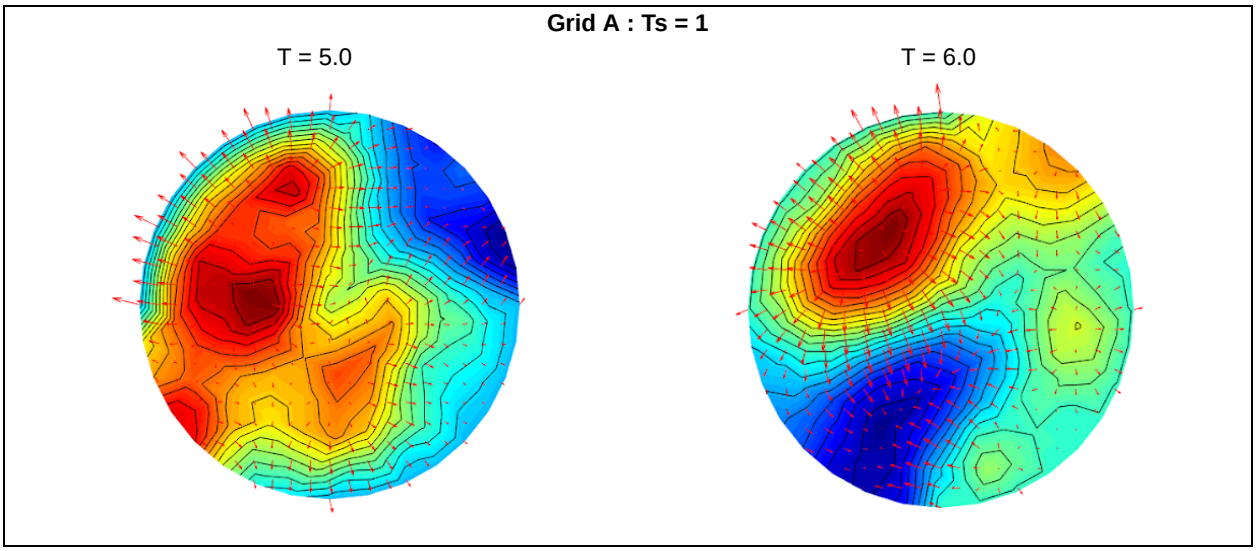


T = 10



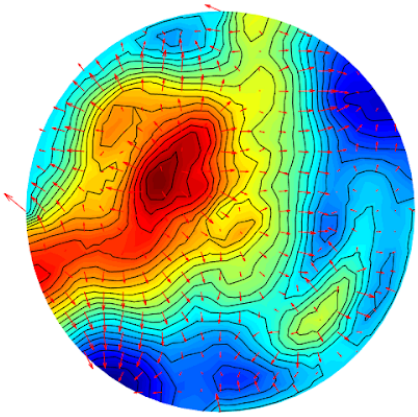
Time Step



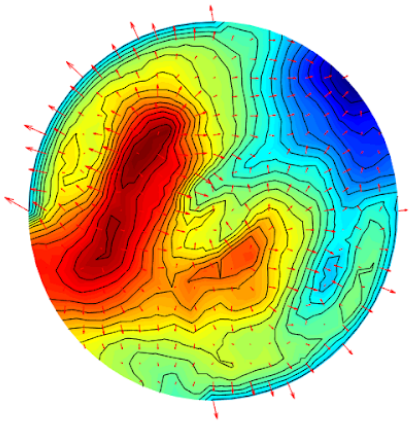


Grid B : Ts = 0.1

T = 5.0

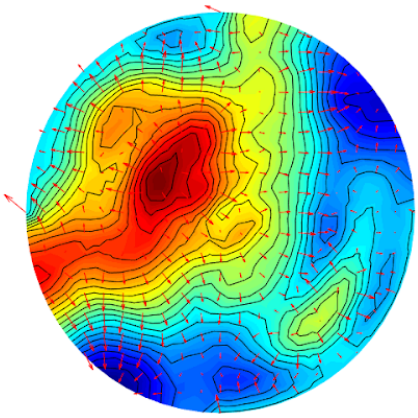


T = 5.1

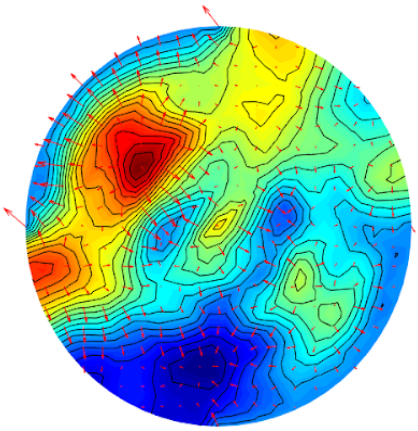


Grid B : Ts = 1

T = 5.0

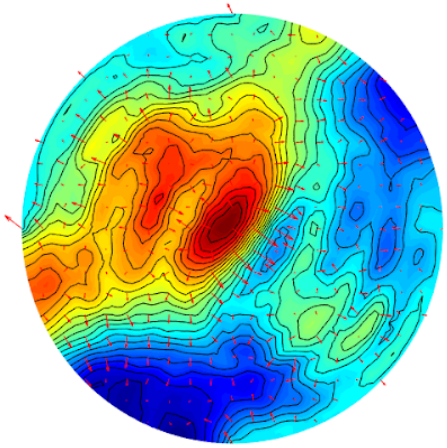


T = 6.0

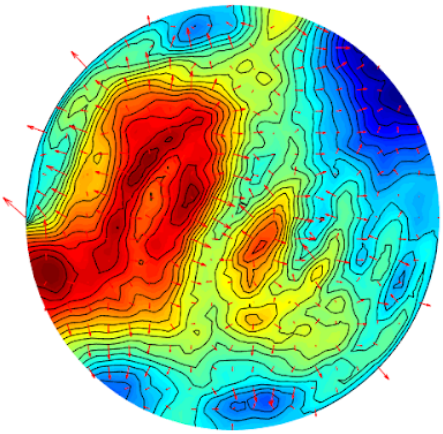


Grid C : Ts = 0.1

T = 5.0

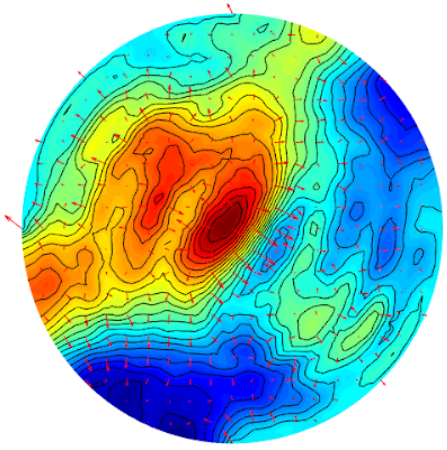


T = 5.1



Grid C : Ts = 1

T = 5.0



T = 6.0