MATLAB Assignment 2

Alex Benasutti

ELC 321 / Signals and Systems

May 9th, 2019

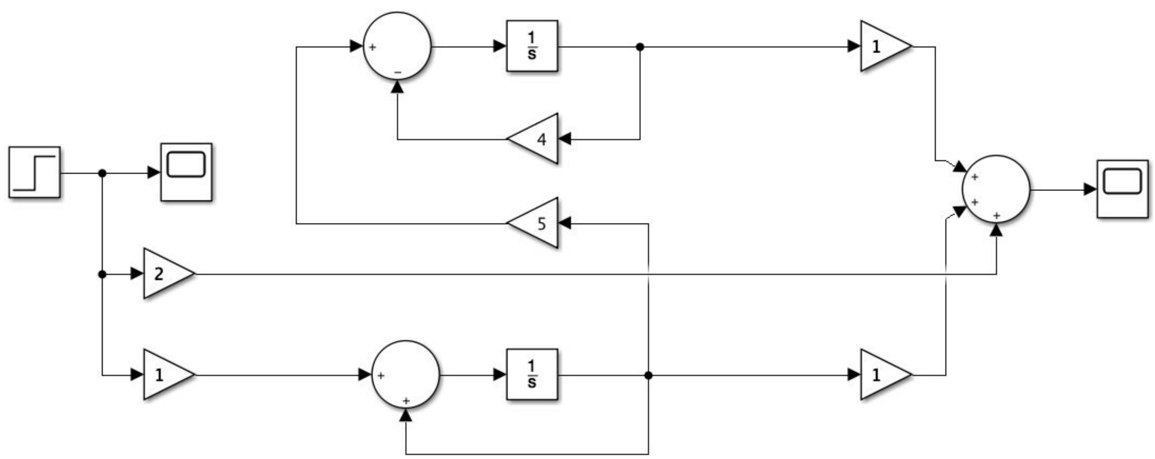
THE COLLEGE OF NEW JERSEY

SCHOOL OF ENGINEERING

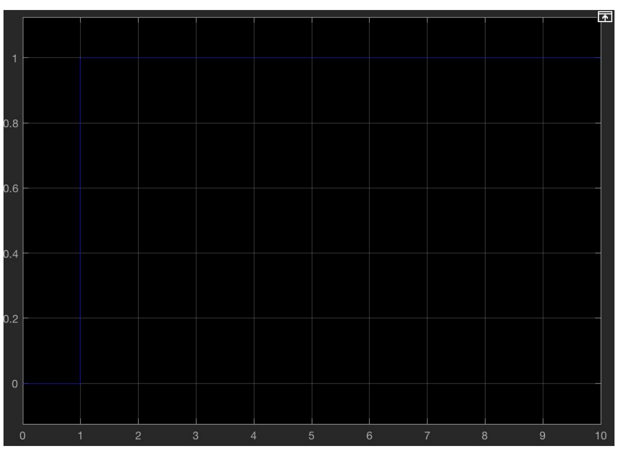
Results

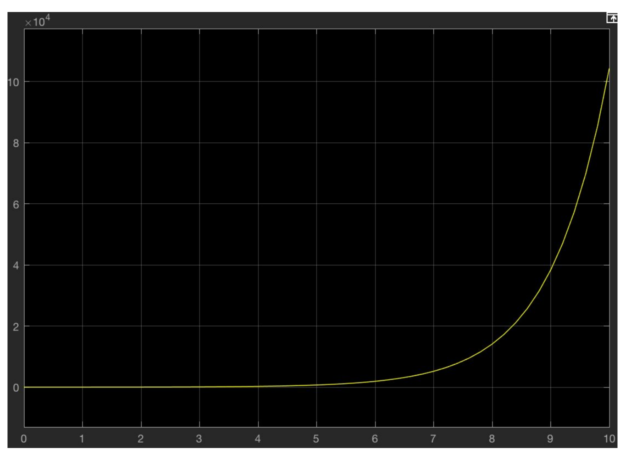
Problem 1.

Simulation Diagram

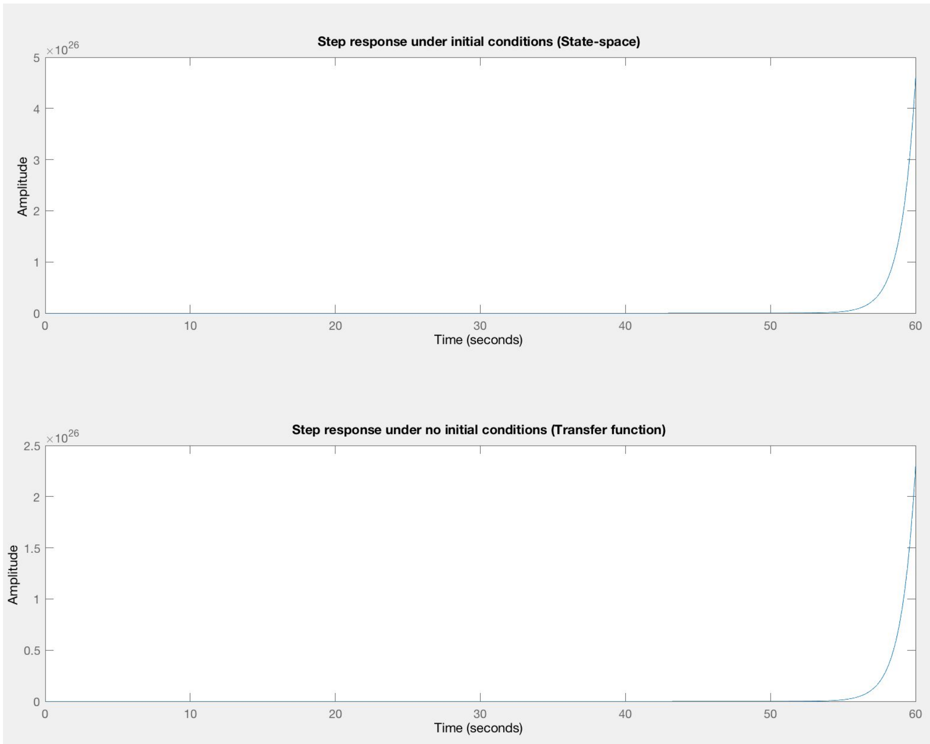


Waveforms (blue = x(t), yellow = y(t))

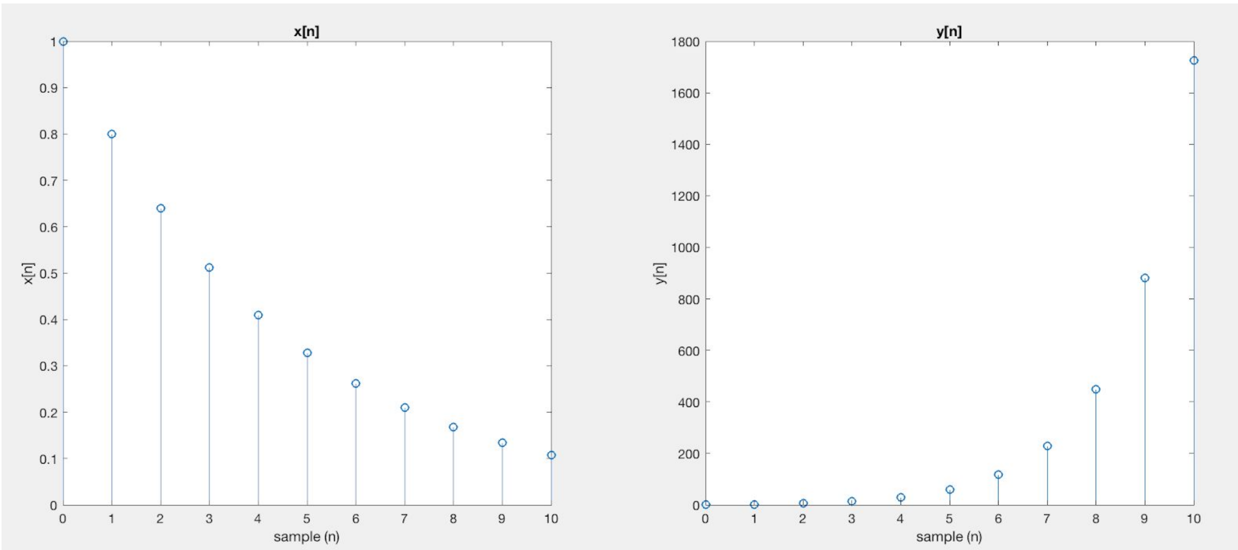




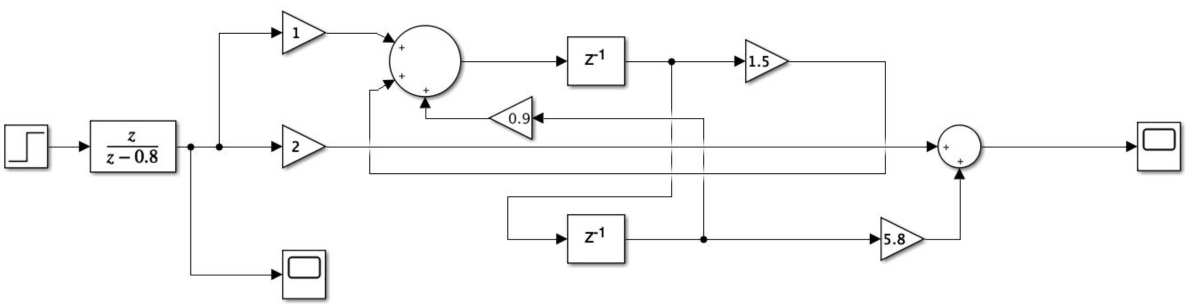
Step Responses



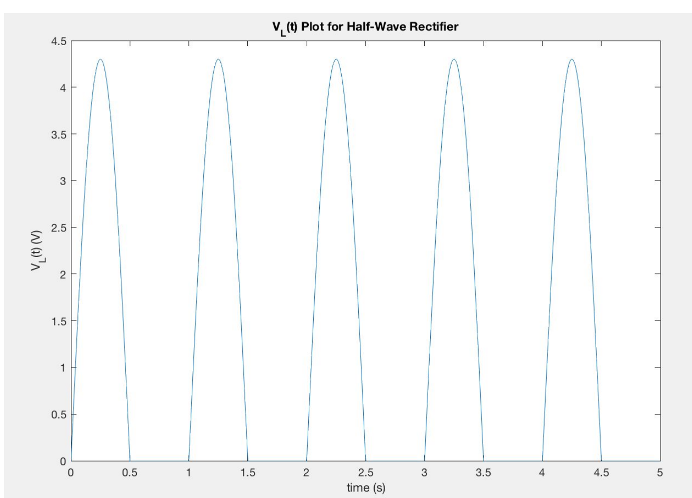
Problem 2.

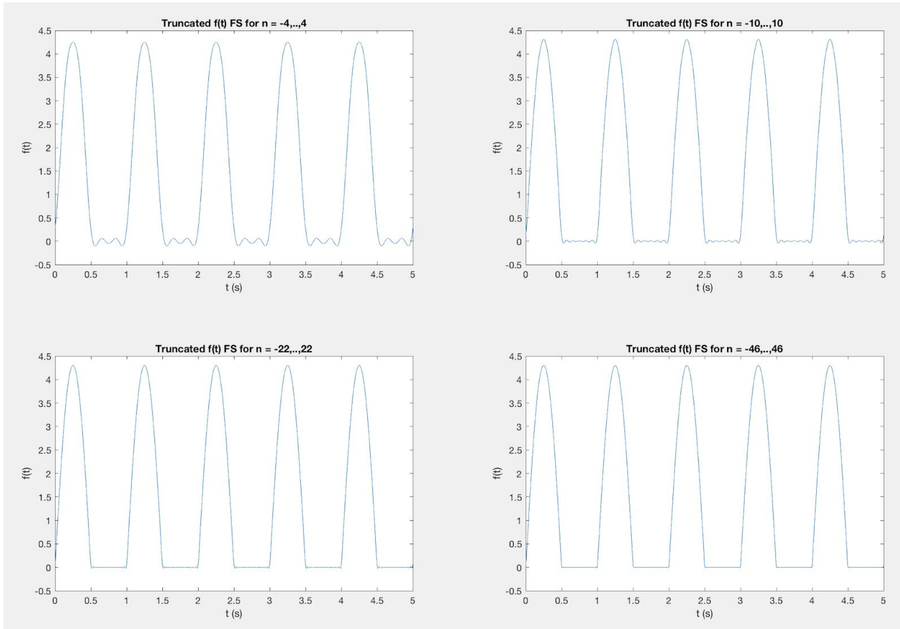


Simulation Diagram



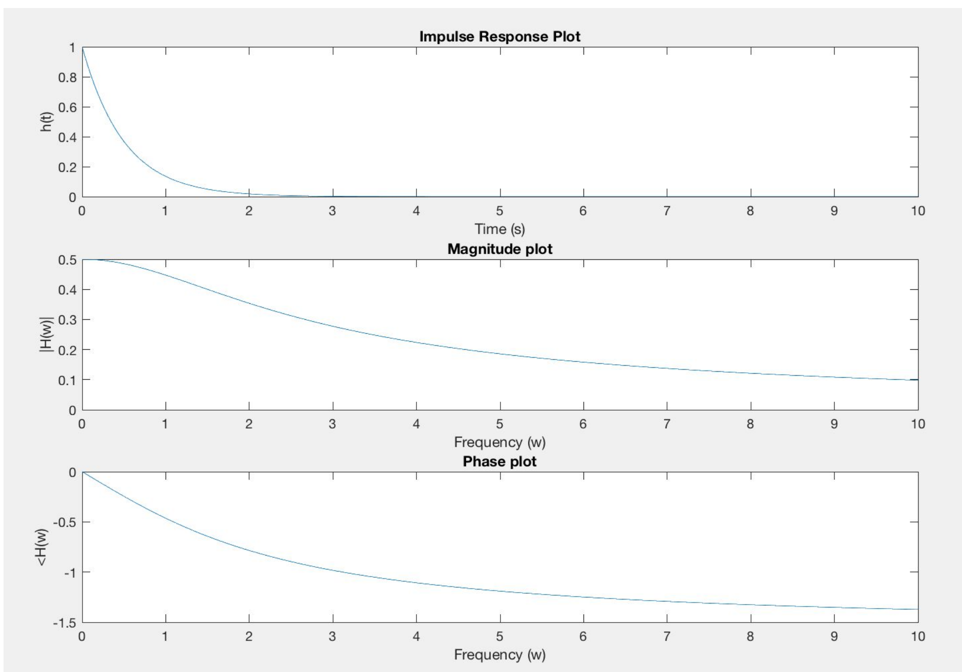
Problem 3.



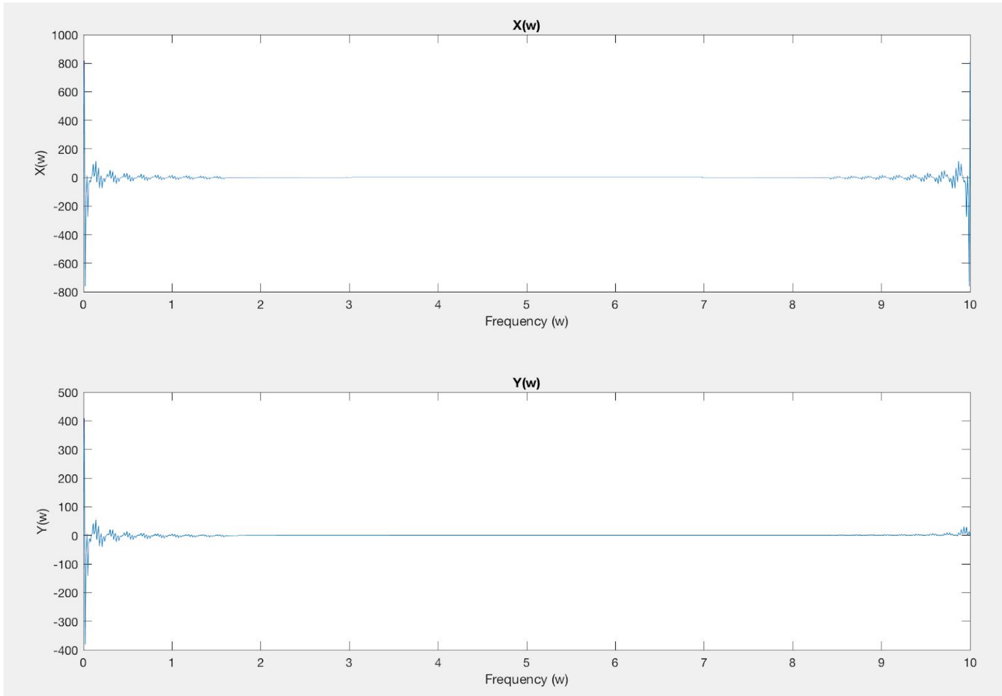


Truncated f(t) FS for n = -4,..,4
Truncated f(t) FS for n = -10,..,10
Truncated f(t) FS for n = -22,..,22
Truncated f(t) FS for n = -46,..,46
f(t)
t (s)

Problem 4.



Impulse Response Plot
h(t)
Time (s)
Magnitude plot
|H(w)|
Frequency (w)
Phase plot
<H(w)
Frequency (w)


X(w)
X(w)
Frequency (w)
Y(w)
Y(w)
Frequency (w)

Appendix

Problem 1 Code

```
A = [-4 5; 0 1];            % Define state-space matrices
B = [0; 1];
C = [1 1];
D = 2;
x0 = [1; 2];                % Define initial conditions

sys1 = ss(A,B,C,D);         % Define state-space

[num,den] = ss2tf(A,B,C,D);
sys2 = tf(num,den);         % Define transfer function

subplot(2,1,1)
initial(sys1,x0), title('Step response under initial conditions (State-space)')
subplot(2,1,2)
step(sys2), title('Step response under no initial conditions (Transfer
function)')
```

Problem 2 Code

```
t = zeros(1,11);    % Define time range
y = zeros(1,11);    % Define y[n] range
x = y;              % Define x[n] range
x(t>=0) = 1;        % Define x[n] as step function

xinit  = 0;
y1init = 0;         % Define inital conditions: y[-1]
y2init = 0;         % y[-2]

for n=1:11          % Apply transfer function based on value of n
    x(n) = x(n) * 0.8.^(n-1);
    if n==1
        % y(0) = 2*x(0) - 3*x(-1) + 4*x(-2) + 1.5*y(-1) + 0.9*y(-2);
        y(n) = 2*x(n) - 3*xinit + 4*xinit + 1.5*y2init + 0.9*y1init;
        t(n) = n-1;
    elseif n==2
        % y(1) = 2*x(1) - 3*x(0) + 4*x(-1) + 1.5*y(0) + 0.9*y(-1);
        y(n) = 2*x(n) - 3*x(n-1) + 4*xinit + 1.5*y(n-1) + 0.9*y2init;
        t(n) = n-1;
    else
        y(n) = 2*x(n) - 3*x(n-1) + 4*x(n-2) + 1.5*y(n-1) + 0.9*y(n-2);
        t(n) = n-1;
    end
```

```
end

num = [2 -3 4];
den = [1 -1.5 -0.9];
[A,B,C,D] = tf2ss(num,den)  % Define state space representation using transfer
function

% x1[n+1] = 1.5x1[n] + 0.9x2[n] + 1u[n]
% x2[n+1] = x1[n]
% y[n] = 5.8x2[n] + 2u[n]

subplot(1,2,1);     % Plot x[n]
stem(t,x)
title('x[n]'), xlabel('sample (n)'), ylabel('x[n]')
subplot(1,2,2);     % Plot y[n]
stem(t,y)
title('y[n]'), xlabel('sample (n)'), ylabel('y[n]')
```

Problem 3 Code

```
% Half-Wave Rectifier Circuit
% Vs(t) = Asin(wo*t)
% A = 5V wo = 2pi RL = 1ohm

t = linspace(0,5,1000);
w0 = 2*pi;
Vs = (5-0.7)*sin(w0*t); % Define source voltage with ideal diode voltage drop
Vl = Vs;
for t1 = 1:length(t)
    if Vl(t1) < 0
        Vl(t1) = 0;
    end
end

figure(1)
plot(t,Vl);
title('V_L(t) Plot for Half-Wave Rectifier');
xlabel('time (s)');
ylabel('V_L(t) (V)');

j = sqrt(-1);

N = [4 10 22 46];       % Define +/- harmonic values at which to truncate FS
X0 = 4.3;               % Amplitude = 5 - 0.7 = 4.3 V

figure(2)
for i = 1:4             % Compute truncated FS for harmonic values
    f = zeros(size(t)); % start out with DC bias term
```

```
    for k = -N(i):N(i)                   % Loop over index k
        if mod(k,2) == 0
            Ck = -X0/(pi*(k.^2-1));      % FS coefficient (even)
        elseif k == -1 || k == 1
            Ck = -1*j*k*(X0/4);          % FS coefficient (-1 and 1)
        else
            Ck = 0;                      % FS coefficient (odd)
        end
        f = f + real(Ck*exp(j*k*w0*t)); % FS computation
    end

    subplot(2,2,i); % Plot truncated FS representation of square wave
    plot(t,f);      % and actual signal
    hold on;

    xlabel('t (s)');
    ylabel('f(t)');
    titlevec = ['Truncated f(t) FS for n = '
num2str(-N(i)),',..,',num2str(N(i))];
    title(titlevec);
end
```

Problem 4 Code

```
% h(t) = e^-2t*u(t)
% square wave input: X0 = 2, C0x = 2, w0 = 1, Ckx = 4/(-j*pi*k)

w0 = 1;                      % Define fundamental frequency
j = sqrt(-1);
t = linspace(0,10,1000);     % Define time range
h = zeros(1,length(t));      % Define impulse response range

for i=1:length(t)            % Define h(t) = u(t)*e^-2t
    if(t(i)>=0)
        h(i)=exp(-2*t(i));
    else
        h(i)=0;
    end
end

w = t;                       % Set range of frequency to time range
H = zeros(1,length(w));
for k=1:length(w)
    H(k) = 1 / (j*w(k)+2);
end
```

```
figure(1)
subplot(3,1,1)
plot(t,h), title('Impulse Response Plot'), xlabel('Time (s)'), ylabel('h(t)')
subplot(3,1,2)
plot(w, abs(H)), title('Magnitude plot'), xlabel('Frequency (w)'),
ylabel('|H(w)|')
subplot(3,1,3)
plot(w, angle(H)), title('Phase plot'), xlabel('Frequency (w)'),
ylabel('<H(w)')

N = 99;
X0 = 2;

fx = zeros(size(t)); % start out with DC bias term

for k = -N:2:N                              % Loop over index k (odd)
    if k ~= 0
        Ckx = (2*X0)/(j*pi*k*w0);           % FS coefficient for square wave
        fx = fx + real(Ckx*exp(j*k*w0*t));  % FS computation
    else
        C0x = 2;                            % Add C0x to FS
        fx = fx + C0x;
    end
end

% sq = 2*square(t);
% Fsq = fft(sq);
Fx = fft(fx);                               % DFT of input signal
Fy = Fx.*H;                                 % Compute Y(w) = X(w)*H(w)

figure(2)
subplot(2,1,1)
plot(w,Fx), title('X(w)'), xlabel('Frequency (w)'), ylabel('X(w)')
subplot(2,1,2)
plot(w,Fy), title('Y(w)'), xlabel('Frequency (w)'), ylabel('Y(w)')
```