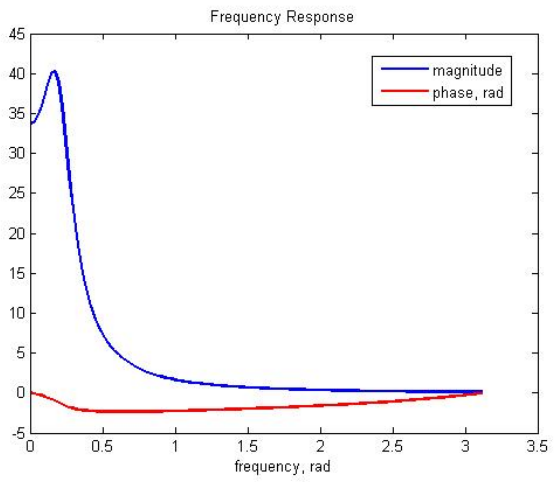
**Matlab Exercise 2**

**1.a.**

```
b = [1 0.5]'; a = [1 -1.8*cos(pi()/16) 0.81]';
[h,w] = freqz(b,a,128);
plot(w, mag(h)); hold on;
plot(w, abs(h));
```



1.b

$$H(z) = \frac{1 + 0.5z^{-1}}{1 - 1.8\cos\left(\frac{\pi}{16}\right)z^{-1} + 0.81z^{-2}}$$

c

$$H(z) = \frac{z\left(z+\frac{1}{2}\right)}{z^2 - Az + 0.81} \quad \text{where} \quad A = 1.8\cos\left(\frac{\pi}{16}\right)$$

$$\boxed{\text{zeros} = \left\{0, -\frac{1}{2}\right\}}$$

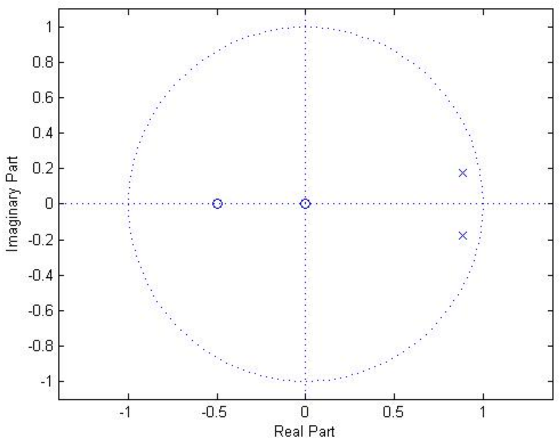
poles : let $b = -A$, $c = 0.81$, $a = 1$

$$z = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \Rightarrow z = \underbrace{\boxed{0.883 \pm j\,0.1756}}_{\text{poles}}$$

**1.d.**

```
>> b = [1 0.5 0] ; zros = roots(b)

zros =

         0
   -0.5000

>> a = [1 -1.8*cos(pi()/16) 0.81]; pols = roots(a)

pols =

   0.8827 + 0.1756i
   0.8827 - 0.1756i
```

**1.e.** `zplane(b,a);`



$$\boxed{2.a}\quad H(z) = \frac{3 - 7z^{-1} + 5z^{-2}}{1 - \frac{5}{2}z^{-1} + z^{-2}}$$

```
b = [3 -7 5]; a = [1 -5/2 1];
zros = roots(b)

zros =

   1.1667 + 0.5528i
   1.1667 - 0.5528i

pols = roots(a)

pols =

    2.0000
    0.5000
```

**2.b.**

$$\begin{array}{r|l} & \quad 5 \\ \hline 1 - \frac{5}{2}z^{-1} + z^{-2} & \;3 - 7z^{-1} + 5z^{-2} \\ & -5 + \frac{25}{2}z^{-1} - 5z^{-2} \\ \hline & -2 + \frac{11}{2}z^{-1} \end{array}$$

$$H(z) = 5 + \frac{-2 + \frac{11}{2}z^{-1}}{\left(1 - \frac{1}{2}z^{-1}\right)\left(1 - 2z^{-1}\right)}$$

$$H(z) = 5 + \frac{A}{\left(1-\frac{1}{2}z^{-1}\right)} + \frac{B}{\left(1-2z^{-1}\right)}$$

<u>where</u> $A\left(1-2z^{-1}\right) + B\left(1-\frac{1}{2}z^{-1}\right) = -2 + \frac{11}{2}z^{-1}$

$\Rightarrow A + B = -2$ <u>and</u> $-2A - \frac{1}{2}B = \frac{11}{2}$

$\hookrightarrow \quad 2A + 2B = -\frac{8}{2}$

$\cdot\frac{3}{2}B = \frac{3}{2} \Rightarrow \boxed{\begin{array}{l} B = 1 \\ A = -3 \end{array}}$

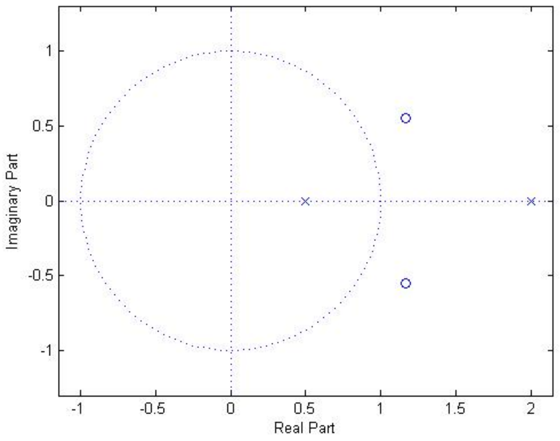
**2.b. (cont'd)**

<u>So</u> $H(z) = 5 + \frac{-3}{1-\frac{1}{2}z^{-1}} + \frac{1}{1-2z^{-1}}$

$$h[n] = 5\cdot\delta[n] - 3\left(\frac{1}{2}\right)^n u[n] + 2^n u[n]$$

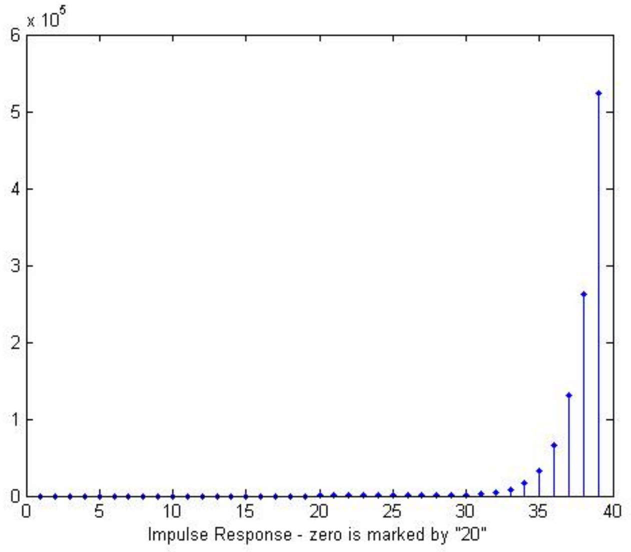
$$\boxed{h[n] = 5\delta[n] + \left(2^n - 3\left(\tfrac{1}{2}\right)^n\right)u[n]}$$

**2.c.**
```
>> zplane(b, a);
```



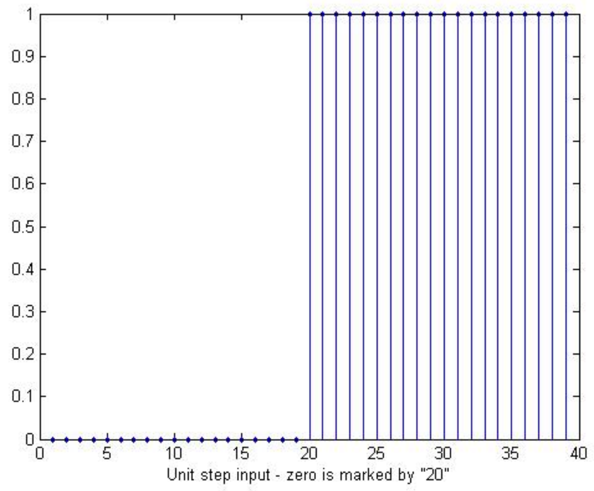
**2.d.**

```
u = [zeros(1, 19) ones(1, 20)];
x = u;
d = [zeros(1, 19) 1 zeros(1, 19)];
n = [0:1:19];
h1 = 5 * d;
h2 = [zeros(1, 19) 2.^n];
h3 = [zeros(1, 19) -3*(1/2).^n];
h = h1 + h2 + h3;
stem(h, '.');
```

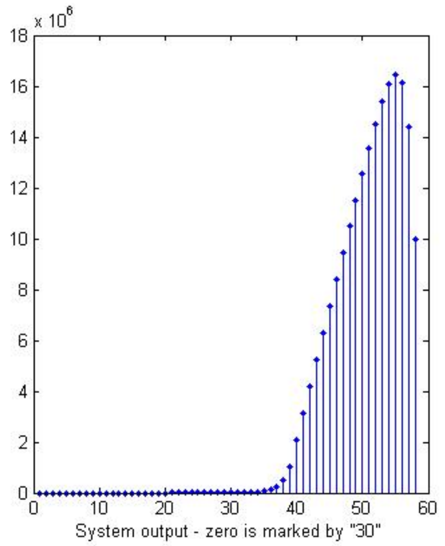


2.d. (cont’d)

```
x = u;
stem(x, '.');
```

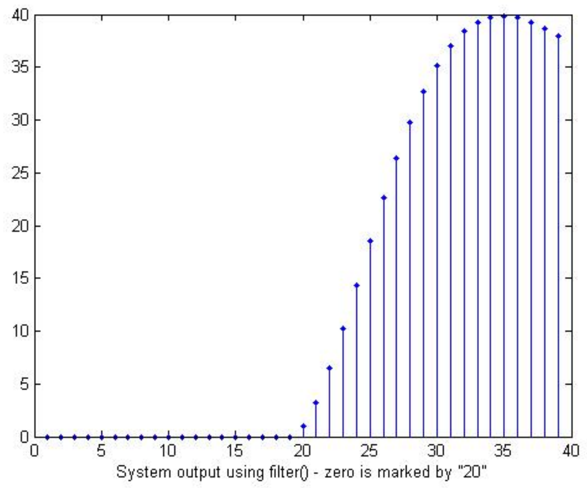


```
y = conv(h, n);
stem(y, '.');
```



2.e.

```
y1 = filter(b,a,x);
stem(y1, '.');
```



(2.f) $Y(z) = X(z)\,H(z) = \left(\frac{1}{1-z^{-1}}\right)\left(\frac{3-7z^{-1}+5z^{-2}}{1-\frac{5}{2}z^{-1}+z^{-2}}\right)$

$$Y(z) = \frac{3-7z^{-1}+5z^{-2}}{(1-z^{-1})(1-\frac{1}{2}z^{-1})(1-2z^{-1})}$$

zeros $= \{1.1667 \pm 0.5528i\}$ poles $= \{\frac{1}{2}, 1, 2\}$

(2.g) $Y(z) = \frac{A}{1-z^{-1}} + \frac{B}{1-\frac{1}{2}z^{-1}} + \frac{C}{1-2z^{-1}}$

where

$$A(1-\tfrac{1}{2}z^{-1})(1-2z^{-1}) + B(1-z^{-1})(1-2z^{-1}) + C(1-z^{-1})(1-\tfrac{1}{2}z^{-1})$$
$$= 3-7z^{-1}+5z^{-2}$$

$$A(1-\tfrac{5}{2}z^{-1}+z^{-2}) + B(1-3z^{-1}+2z^{-2}) + C(1-\tfrac{3}{2}z^{-1}+\tfrac{1}{2}z^{-2})$$
$$= 3-7z^{-1}+5z^{-2}$$

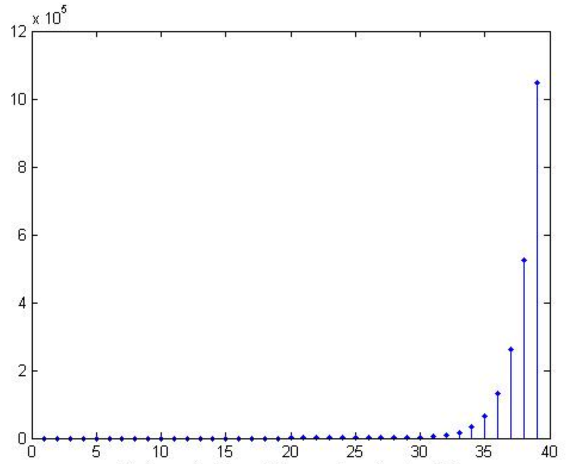
$$\Rightarrow \begin{cases} A+B+C = 3 \\ -\frac{5}{2}A - 3B - \frac{3}{2}C = -7 \\ A + 2B + \frac{1}{2}C = 5 \end{cases} \Rightarrow$$

```
coeff =

  1.0000e+000  1.0000e+000  1.0000e+000
 -2.5000e+000 -3.0000e+000 -1.5000e+000
  1.0000e+000  2.0000e+000  5.0000e-001

>> D = [3 -7 5]';
>> ABC = linsolve(coeff, D)

ABC =

    -2
     3
     2
```

$$y[n] = -2u[n] + 3\left(\tfrac{1}{2}\right)^n u[n] + 2(2)^n u[n]$$

$$\boxed{y[n] = \left[-2 + 3\left(\tfrac{1}{2}\right)^n + 2(2)^n\right] u[n]}$$

```
y2 = [zeros(1,19) -2*n] + [zeros(1,19) 3*(1/2).^n] + [zeros(1,19) 2*(2).^n];
stem(y2, '.');
```



System output from Y(z) - zero is marked by "20"

```
3.a.
Rs = 8000; f = 500; psi = 0; start_t = 0; A = 1; n = f * 2; fig_idx = 0; fig_chr = '.';
[x, t] = sin_samples(Rs, f, psi, start_t, A, n, fig_idx, fig_chr);
sound(x, Rs);

3.b.
f = 1000;
[x, t] = sin_samples(Rs, f, psi, start_t, A, n, fig_idx, fig_chr);
sound(x, Rs);

function [X, t] = sin_samples(Rs, f, psi, start_t, A, n, fig_idx, fig_chr)
% Returns the vector of sample magnitudes of a discrete sinusoidal time series.
% Output vector X contains n periods of the amplitude.
% Output vector t contains n periods of the time variable.
% Rs is the sampling rate in samples per second.
% f is the signal frequency.
% psi is the phase of the signal with respect to the first sample.
% start_t is the first index of time.
% A is the signal amplitude.
% fig_idx is an integer; (0 < fig_idx) => render Figure(fig_idx)
% fig_chr is a single character used as valid argument to stem();
%   .. (fig_ndx < 1) => fig_chr is ignored.
%
% D.Bozarth 02-21-08 SSU Engineering Science
%
Ts = 1 / Rs;                             % Time duration of one sample.
T  = 1 / f;                              % Signal period.
t = [start_t:Ts:(start_t + n*T - Ts)];  % Time vector.

% Obtain the return value.
X = A * sin(2*pi()*f*t + psi);

if 0 < fig_idx
    figure(fig_idx);
    stem(X, fig_chr);
end
% end of program
```

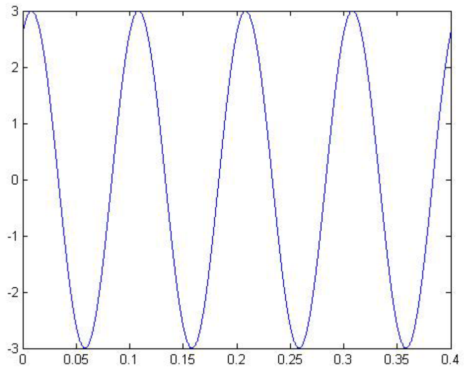
---

4. Sampling and reconstruction.

a. Consider  x(t) = 3*sin(2pi*10*t + pi/3)

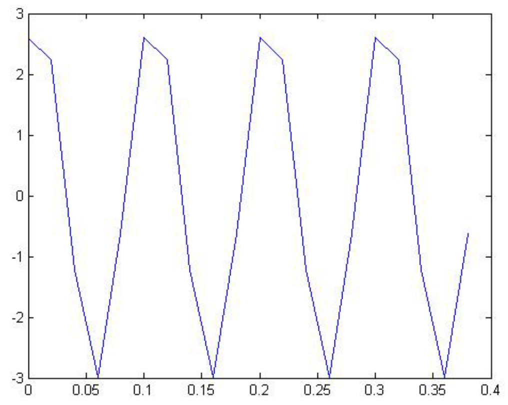
Use a discrete-time vector with a very small time resolution, to approximate a continuous-time function.

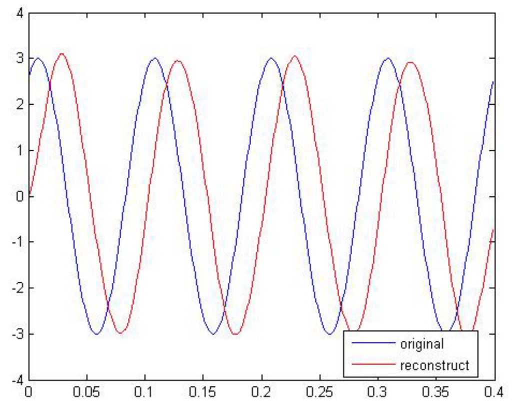
```
Rs = 1000000; f = 10; psi = pi()/3; t_base = -0.9; A = 3; N = 18; fig_idx = 0;
[x, t] = sin_samples(Rs, f, psi, t_base, A, N, fig_idx);
plot(t, x);
```



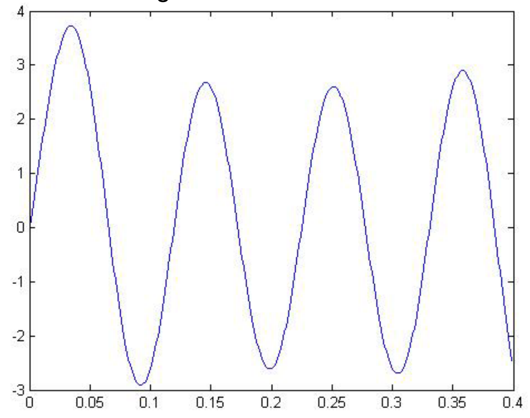
b. Create a vector x1 with samples of a sinusoid signal with sampling frequency 50 Hz.

```
Rs = 50; f = 10; psi = pi()/3; t_base = -0.9; A = 3; N = 18; fig_idx = 2;
[x1, t1] = sin_samples(Rs, f, psi, t_base, A, N, fig_idx);
plot(t1, x1);
```

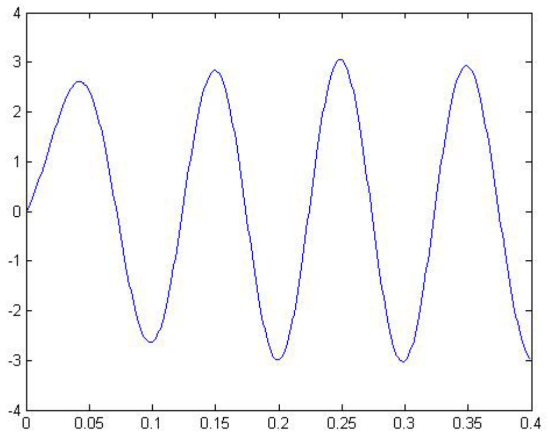


c. Reconstruct the sinusoid from the samples, by using an ideal low-pass filter.
```
xr1 = reconstruct_filter(t1, x1, 1/f);
plot(t1, xr1);
```


d. The Nyquist frequency (highest frequency of interest) is 10 Hz, so the minimum sampling frequency (Nyquist rate) is 20 Hz. Sample the signal at this frequency, and repeat step c.
```
Rs = 20; f = 10; psi = pi()/3; t_base = 0; A = 3; N = 4;
[x1, t1] = sin_samples(Rs, f, psi, t_base, A, N);
xr1 = reconstruct_filter(t, x1, Rs);
plot(t, xr1);
```
Some aliasing is evident.



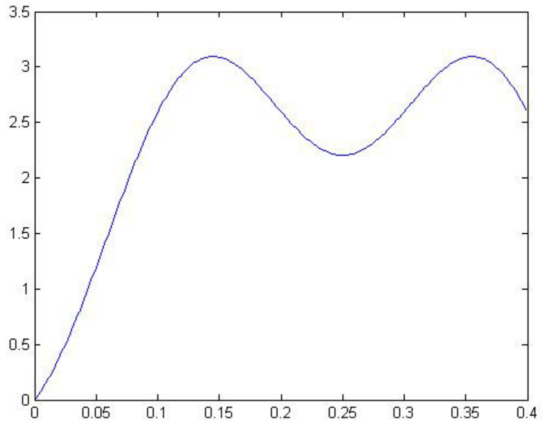
e. Set the sampling frequency to 25 Hz and repeat.
```
Rs = 25; f = 10; psi = pi()/3; t_base = 0; A = 3; N = 4;
[x1, t1] = sin_samples(Rs, f, psi, t_base, A, N);
xr1 = reconstruct_filter(t, x1, Rs);
plot(t, xr1);
```
Aliasing is less pronounced.



f. Set the sampling frequency to 10 Hz and repeat.

```
Rs = 10; f = 10; psi = pi()/3; t_base = 0; A = 3; N = 4;
[x1, t1] = sin_samples(Rs, f, psi, t_base, A, N);
xr1 = reconstruct_filter(t, x1, Rs);
plot(t, xr1);
```

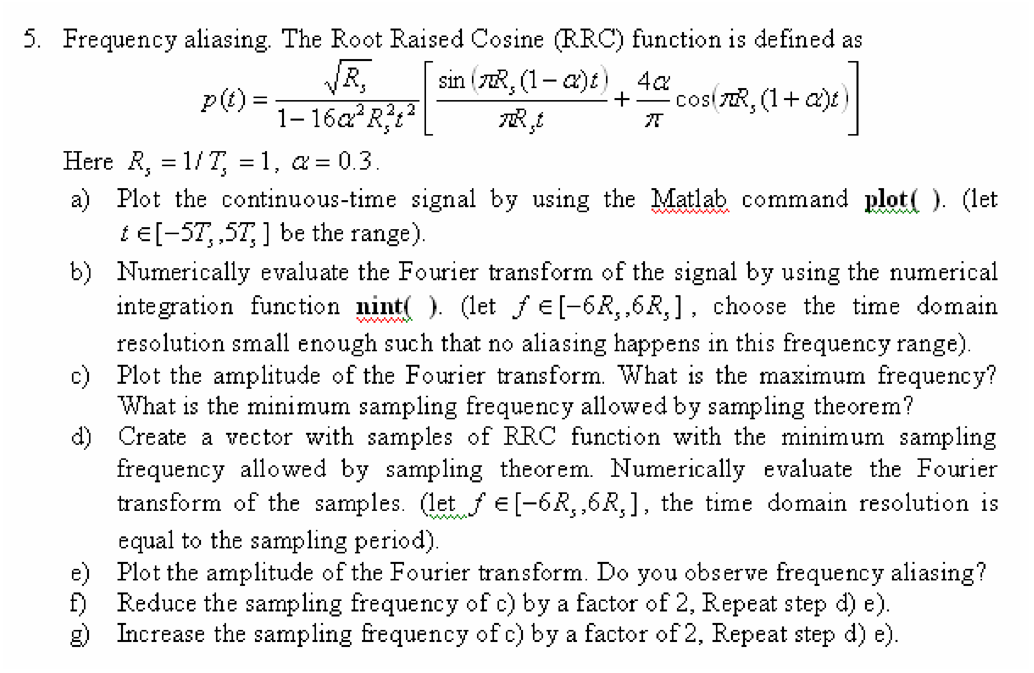
Aliasing is severe.



(modified sin_samples)

```
function [X, t] = sin_samples(Rs, f, psi, t_base, A, N, fig_idx)
% Returns the vector of sample magnitudes of a discrete sinusoidal time series.
% Output vector X contains N periods of the amplitude.
% Output vector t contains N periods of the time variable.
% Rs is the sampling rate in samples per second.
% f is the signal frequency.
% psi is the phase of the signal with respect to the first sample.
% t_base is the starting time of the sequence.
% A is the signal amplitude.
%
% D.Bozarth 02-23-08 SSU Engineering Science
%
Ts = 1 / Rs;                            % Time duration of one sample.
T  = 1 / f;                             % Signal period.
t = [t_base:Ts:(t_base + N*T - Ts)];    % Time vector.

% Obtain the return value.
X = A * sin(2*pi()*f*t + psi);

if 0 < fig_idx
    figure(fig_idx);
end
% end of program
------------------

function xr = reconstruct_filter(t, x, T)
% Returns the approximate ideal low-pass filter output vector
%    .. corresponding to the periodic sinusoidal input vector x(t).
% t is the vector t[n].
% x is the vector x[n].
% T is the period of the input signal.
```

```
%
% D.Bozarth 02-23-08 SSU Engineering Science
% using Oppenheimer, et.al, Discrete-Time Signal Processing, 2nd Ed., p.150
%
s = size(t);
sz = s(1, 2);

for tt = 1:sz       % For each distinct time value..
    xr(1, tt) = 0;
    ttt = t(tt);    % The time value..

    for n = 1:sz    % Form a summation.
        if (ttt == (n - ceil(sz/2)) * T)
            tnt = 0.00000001;
        else
            tnt = ttt - (n - ceil(sz/2)) * T;
        end

        ptnt = pi() * tnt;
        xr(1, tt) = xr(1, tt) + x(1, n) * T * sin(ptnt / T) / ptnt;
    end
end
% end of program
-------------------
```



5. Frequency aliasing. The Root Raised Cosine (RRC) function is defined as

$$p(t) = \frac{\sqrt{R_s}}{1-16\alpha^2 R_s^2 t^2}\left[\frac{\sin(\pi R_s(1-\alpha)t)}{\pi R_s t} + \frac{4\alpha}{\pi}\cos(\pi R_s(1+\alpha)t)\right]$$

Here $R_s = 1/T_s = 1$, $\alpha = 0.3$.

a) Plot the continuous-time signal by using the Matlab command **plot( )**. (let $t \in [-5T_s, 5T_s]$ be the range).

b) Numerically evaluate the Fourier transform of the signal by using the numerical integration function **nint( )**. (let $f \in [-6R_s, 6R_s]$, choose the time domain resolution small enough such that no aliasing happens in this frequency range).

c) Plot the amplitude of the Fourier transform. What is the maximum frequency? What is the minimum sampling frequency allowed by sampling theorem?

d) Create a vector with samples of RRC function with the minimum sampling frequency allowed by sampling theorem. Numerically evaluate the Fourier transform of the samples. (let $f \in [-6R_s, 6R_s]$, the time domain resolution is equal to the sampling period).

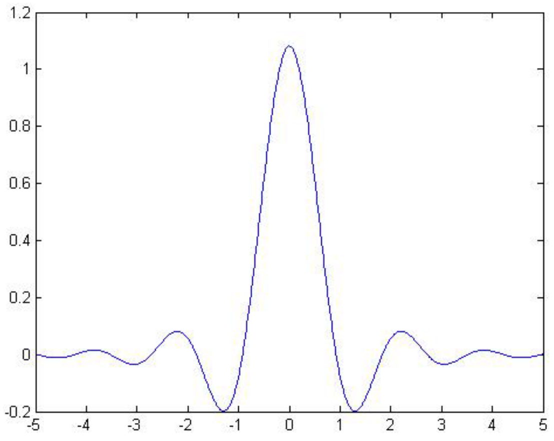
e) Plot the amplitude of the Fourier transform. Do you observe frequency aliasing?

f) Reduce the sampling frequency of c) by a factor of 2, Repeat step d) e).

g) Increase the sampling frequency of c) by a factor of 2, Repeat step d) e).

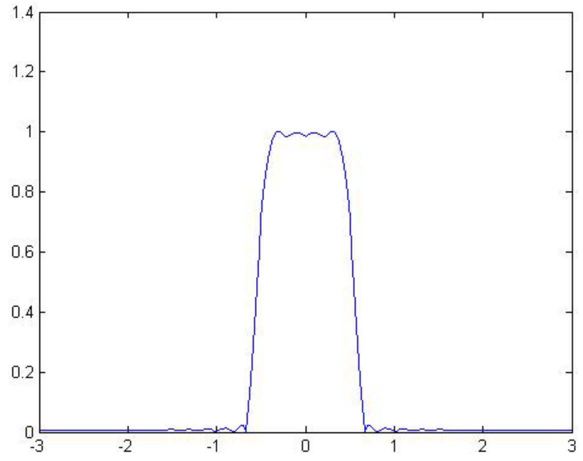
5.a.

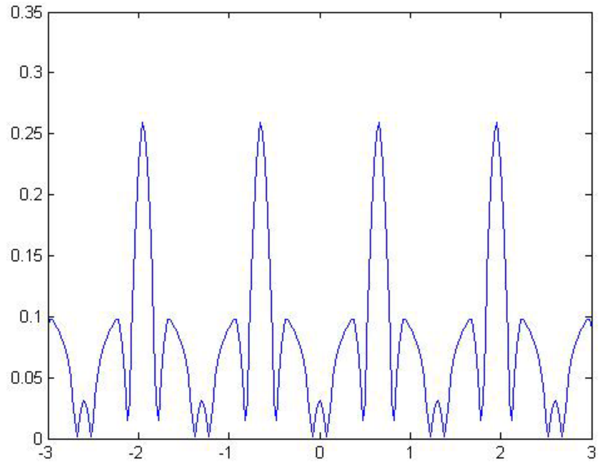
```
Ts = 1 ; alp = 0.3; t = (-5*Ts):(Ts/100):(5*Ts);
p = root_rcos(t, Ts, alp);
plot(t, p);
```



5.b.
```
Rs = 1; Ts = 1/Rs ; alp = 0.3; t = (-5*Ts):(Ts/100):(5*Ts);
p = root_rcos(t, Ts, alp);
```

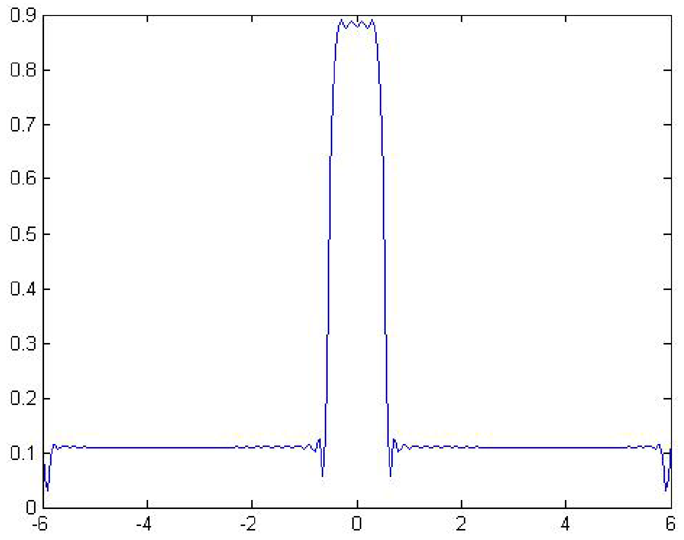
I used a Riemann sum approximation of the Fourier transform, and the transform built using the Matlab trapz(), and the transform built using the instructor's function nint(). In each case, the plot was very similar to the eye.
```
X = fourier(f, t, p);
```

5.c
The max frequency whose magnitude is greater than 0, is the 26th positive frequency, or 0.65*Rs. According to Nyquist, the minimum sample frequency is 1.3*Rs..
```
plot(f, abs(X));
```


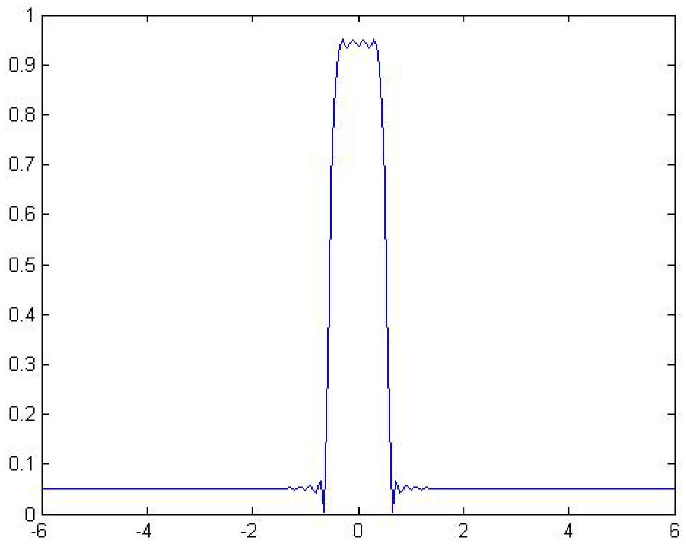
5d-g. frequency aliasing observed in parts e through g. For example:
```
Rs = 1.3; Ts = 1/Rs ; alp = 0.3; t = (-5*Ts):(Ts):(5*Ts);
p = root_rcos(t, Ts, alp);
plot(t, p);
stem(p, '.');
X = fourier(f, t, p);
plot(f, abs(X));
```


The Fourier transform always displays aliasing, but the form of it varies.

I chose to try increasing the sampling freq until no aliasing is observed. With sampling rate Ts/13, aliasing is evident at the margins of this frequency range.



With rate Ts/14, aliasing is not present:



There remains a pronounced white noise floor, though. With sampling rate Ts/100, the noise floor is much smaller.

These effects can be partly explained by the fact that we are restricting the time domain of sampling. With an infinite impulse response, the Nyquist criterion would hold.

Another potential corrupting element is limited precision of the numerical integration.

(See following pages for Matlab code used in Problem 5.)

```
function [p] = root_rcos(t, Ts, alp)
% Returns the Root Raised Cosine function.
% t is a time vector.
% Ts is the time period of sampling.
% alp is the roll-off factor.
%
% 02-25-08 Adapted from rcos.m - D.Bozarth, SSU Engineering Science
%
Rs = 1 / Ts;                    % Sampling rate.
st = size(t); szt = st(1, 2);   % Size of time vector.

for (tt = 1:szt)
    ttt = t(tt);                % One time value.
    a   = sin(pi * ttt * Rs * (1 - alp));
    b   = cos(pi * ttt * Rs * (1 + alp));
    s   = a + 4 * alp * Rs * ttt * b;

    if (ttt == 0) | (abs(ttt) == Ts / (4 * alp))
        p(1, tt)= sign(s) * Inf;
    else
        c       = pi * ttt * Rs;
        d       = 1 - (4 * alp * ttt * Rs)^2;
        p(1, tt)= sqrt(Rs) * s / (c * d);
    end
end
% end of program
```

---

```
function [X] = fourier_Riemann(f, t, x)
% Returns a Riemann approximation of the Fourier transform of x(t) as a function of f.
% f is a frequency vector.
% t is a time vector.
% x is a magnitude vector.
%
% 03-02-08 D.Bozarth, SSU Engineering Science
%
sf = size(f); szf = sf(1, 2);       % Size of frequency vector.
st = size(t); szt = st(1, 2);       % Size of time vector.

for (ff = 1:szf)
    fff = f(ff);                    % One frequency value.
    X(1, ff) = 0;
    del      = 0;
    del_prev = (t(szt) - t(1)) / szt;

    for (tt = 1:szt)
        ttt = t(tt);

        if (tt < szt)
            del = t(tt + 1) - ttt;
        else
            del = del_prev;
        end

        if isnan(x(1, tt))
            xx(1, tt) = 0 + 0i;
        else
            xx(1, tt) = x(1, tt) * exp(-i*2*pi*fff*ttt); % Fourier integrand at fff.
        end

        X(1, ff)  = X(1, ff) + xx(1, tt) * del;          % Accumulate Riemann sum.
        del_prev = del;
    end
end
% end of program
```

---

```
function [X] = fourier(f, t, x)
% Returns the Fourier transform of x(t) as a function of f.
% f is a frequency vector.
% t is a time vector of odd size.
% x is a magnitude vector keyed to t.
%
% 03-02-08 D.Bozarth, SSU Engineering Science
%
sf = size(f); szf = sf(1, 2);       % Size of frequency vector.
st = size(t); szt = st(1, 2);       % Size of time vector.
h  = (t(szt) - t(1)) / szt;         % Step size.

for (ff = 1:szf)
    fff = f(ff);                    % One frequency value.

    for (tt = 1:szt)
        ttt = t(tt);

        if ~isfinite(x(1, tt))
            xx(1, tt) = 0;
        else
            xx(1, tt) = x(1, tt) * exp(-i*2*pi*fff*ttt); % Fourier integrand at fff.
        end
    end

    %X(1, ff) = trapz(t, xx);        % Numerical integration.
    X(1, ff) = nint(xx, h);

end
% end of program
```