Laboratory 1: Basic Signals and Systems

Project 1.1: Basic Discrete-Time Signals

For this project, you will write MATLAB functions to create some basic sequences, and use these functions to obtain plots of various discrete-time signals. As an example, suppose you are asked to write a MATLAB function that creates the *unit-sample sequence* (also referred to as the *discrete-time impulse*):

$$\delta[n] = \begin{cases} 0 & n \neq 0 \\ 1 & n = 0 \end{cases} .$$

A MATLAB function to create this sequence is:

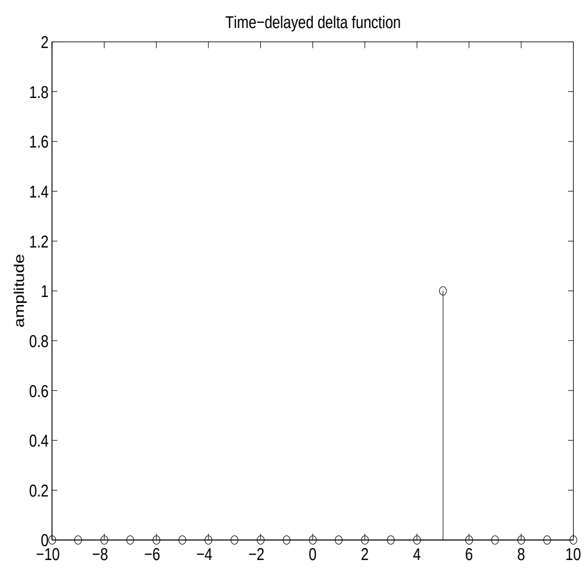
begin MATLAB m-file

```
function d = Usamp(n)
% Usamp(n): Function to create samples of the unit delta function
% evaluated at the elements of the vector n.
d = (n==0); % d(n)=1 if n=0, d(n)=0 otherwise
```

end MATLAB m-file

The following MATLAB commands would evaluate and plot the discrete-time sequence $\delta[n-5]$ over the interval $-10 \leq n \leq 10$:

```
>> n = -10:1:10; % create the sequence indices
>> stem(n, Usamp(n-5)); % plot the function
>> axis([-10 10 0 2.0]) % adjust the axes
>> xlabel('n'); % label the x-axis
>> ylabel('amplitude'); % label the y-axis
>> title('Time-delayed delta function'); % provide a title
```



Exercise 1.1.1: The Unit Sample Sequence

Obtain plots (using your `Usamp` and the `stem` function) of the following sequences over the intervals indicated.

1. $x[n] = 1.5\delta[n+3], \quad -5 \le n \le 5$
2. $x[n] = 2.5\delta[n+20] + 0.5\delta[n-10], \quad -30 \le n \le 30$

Exercise 1.1.2: The Unit Step Sequence

Write a MATLAB function to generate the unit-step sequence:

$$u[n] = \begin{cases} 1, & n \ge 0 \\ 0, & n < 0 \end{cases}.$$

A call to this function should be of the form:

```
>>u=Ustep(n);
```

where `n` is a vector of indices over which the function is to be evaluated. Use this function to obtain plots (using the `stem` function) of the following sequences over the intervals indicated.

1. $x[n] = 3.5u[n-3], \quad -10 \le n \le 10$
2. $x[n] = u[n+4] - u[n-4], \quad -20 \le n \le 20$

Exercise 1.1.3: The Discrete-Time Rect Function

Write a MATLAB function that will generate the following discrete-time rectangular pulse:

$$\mathrm{rect}_N[n] = \begin{cases} 1, & -N \le n \le N \\ 0, & |n| > N \end{cases}.$$

A call to this function should be of the form:

```
>>r=Rect(n,N);
```

where `n` is a vector of indices over which the function is to be evaluated and N defines its width. Use this function to obtain plots (using the `stem` function) of the following sequences over the intervals indicated.

1. $x[n] = 5\,\mathrm{rect}_4[n-3], \quad -10 \le n \le 10$
2. $x[n] = \mathrm{rect}_{10}[n] - \mathrm{rect}_5[n], \quad -20 \le n \le 20$

Exercise 1.1.4: The Discrete-Time Sinusoid

Write a MATLAB function that will generate the following discrete-time sequence:

$$x[n] = \sin(\omega n + \phi).$$

A MATLAB call to this function should be of the form:

```
>>x=Dtsin(n,omega,phi);
```

where `n` is a vector of indices over which the function should be evaluated. `omega` and `phi` specify the radian frequency and phase, respectively, of the sinusoid. Use this function to obtain plots (using the `stem` function) of the following sequences over the intervals indicated:

1. $x[n] = \sin(\frac{\pi}{20} n), \quad 0 \leq n \leq 60$
2. $x[n] = 5\sin(\frac{\pi}{10} n + \frac{\pi}{4}), \quad -10 \leq n \leq 30$
3. $x[n] = \cos(2\pi \frac{1}{5\sqrt{2}} n), \quad 0 \leq n \leq 30$

Determine whether or not each sequence is periodic and, if so, determine its period. Do your plots agree with this?

Exercise 1.1.5: The Discrete-Time Complex Exponential

Write a MATLAB function that will generate the following discrete-time sequence.

$$w[n] = e^{j\omega n}.$$

A MATLAB call to this function should be of the form:

```
>>w=Cexp(n,omega);
```

where `n` is a vector of indices over which the sequence should be evaluated and `omega` is the radian frequency. Use this function to create the complex-valued sequence

$$w[n] = 3.2 e^{j\left(\frac{\pi}{9} n - \frac{\pi}{4}\right)}, \quad -10 \leq n \leq 20.$$

1. Using the MATLAB commands `real` and `imag`, obtain plots of the real and imaginary part of this sequence. Use `subplot` to obtain both plots in the same figure.

2. Using the MATLAB commands `abs` and `angle`, obtain plots of the magnitude and phase of this sequence. Use `subplot` to obtain both plots in the same figure.

3. For both cases, derive analytic expressions for the sequences you have plotted and compare these expressions with your plots.

Project1.2:Discrete-TimeSystems

Forthisprojectyouwillwrite MATLAB functionstoemulatesomebasicdiscrete-timesystems, andyouwillusethesefunctionstotransforminputsequencestooutputsequences.Asanexample, supposeyouareaskedtowritea MATLAB functionthatemulatesthe *idealdelaysystem* :

$$y[n] = x[n - n_0].$$

A MATLAB functionforthissystemmightbe

begin MATLAB m-file

```
function[y,ny]=Delay(x,nx,n0)
%Delay(x,nx,n0):functiontoemulatetheidealdelaysystem
%foradelayofn0
%
N=max(size(nx));%determinethesizeofnx
ifn0>=0%checkforpositivedelay
  ny=[nx(1:N-1)nx(N):nx(N)+n0];%augmenttheindecesvectorforpos.delay
elseifn0<0
  ny=[nx(1)+n0:nx(1)nx(2:N)];%augmenttheindecesvectorforneg.delay
end;
M=max(size(ny));%determinethesizeofny
y=zeros(size(ny));%undefinedvaluesofxwillbesetto0
ifn0>0
  y(n0+1:M)=x;
else
  y(1:M+n0)=x;
end
```

end MATLAB m-file

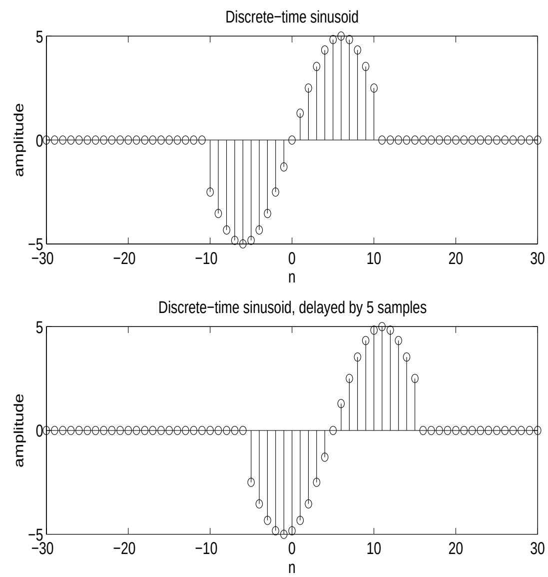
Theparameterspassedtothefunctionaretheinputsequence `x`,theindicesoverwhichitisdefined `nx`,andthenumberofsamplesbywhichitistobedelayed `n0`.Thefunctionreturnsthedelayed sequence `y` withanyvaluesforwhichtheinputsequenceisundefinedsettozero,alongwiththe indicesoverwhichitisdefined `ny`. W e c a n t h e n u s e t h i s f u n c t i o n (a n d f u n c t i o n s w r
1.1)toobtainplotsofaninputsequence

$$x[n]=5\mathrm{rect}_{10}[n]\sin\left(\frac{\pi}{12}n\right), \quad -30 \le n \le 30,$$

delayedby5samples:

```
>>nx=-30:30;
>>x=5*Rect(nx,10).*Dtsin(nx,pi/12,0);
>>[y,ny]=Delay(x,nx,5);
>>subplot(2,1,1);
>>stem(nx,x);
```

```
>>xlabel('n');
>>ylabel('amplitude');
>>title('Discrete-time sinusoid');
>>subplot(2,1,2);
>>stem(ny,y);
>>axis([-30 30 -5 5]); % set the axis the same for both plots
>>xlabel('n');
>>ylabel('amplitude');
>>title('Discrete-time sinusoid, delayed by 5 samples');
```



Exercise 1.2.1: First-order Moving Average System

Write a MATLAB function to emulate the *first-order moving average system*:

$$y[n] = a_0 x[n] + a_1 x[n-1].$$

A call to this function should be of the form

```
>>[y,ny]=Mave1(x,nx,a0,a1);
```

where `nx` is the vector of indices for which the input sequence `x` is defined, and `ny` is the vector of indices for which the output sequence `y` is defined. Assume that any undefined values of `x` are zero. Use this function (and functions written in Project 1.1) to obtain plots of an input sequence

$$x[n] = 5\,\mathrm{rect}_{20}[n] \sin\left(\frac{\pi}{12} n\right), \quad -30 \le n \le 30,$$

and its corresponding output sequence for:

1. $a_0 = 1$ and $a_1 = -1$
2. $a_0 = a_1 = 1/2$

Laboratory2:Discrete-TimeConvolution

Inthislaboratoryassignment,youwillstudytheconceptsofdiscrete-timeconvolution.Recall thatthediscrete-timeconvolutionofthesequences $x[n]$ and $h[n]$isdefinedas

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k], \quad -\infty < n < \infty.$$

Ifthesequencesarenonzeroonlyoverfiniteintervals,thatis

$$x[k] = 0, k < K_1 \text{ or } k > K_2,$$

and

$$h[n] = 0, n < N_1 \text{ or } n > N_2,$$

thentheconvolutionsumcanbewrittenas

$$y[n] = \sum_{k=K_1}^{K_2} x[k]h[n-k], N_1 + K_1 \leq n \leq N_2 + K_2,$$

andthesequence $y[n]$willbenonzeroonlyoveranintervalof $N_2 - N_1 + K_2 - K_1 + 1$samples. The MATLAB function `conv` canbeusedtoconvolvetwosequences;however,youmustdoall ofthebookkeepingfortheindicesoverwhich $x[n]$, $h[n]$,and $y[n]$aredefined.Tolearnmore aboutthe `conv` function,explorethe `conv` sub-categorywithinthe `datafun` categoryoftheon-line documentation(usingthe `doc` command).

Writea MATLAB functiontoconvolvetwosequences(usingthe `conv` function)andkeeptrack oftheindicesoverwhichthefunctionsaredefined.Acalltoyourfunctionshouldbeoftheform

```
>>[y,ny]=Convolve(x,nx,h,nh);
```

where `x` and `h` arethesequencestobeconvolved, `nx` and `nh` aretheindicesoverwhichtheyare defined, `y` istheconvolvedsequence,and `ny` isavectorofindicesoverwhichitisdefined.

Example1 Supposeyouareaskedtoconvolvethesequences

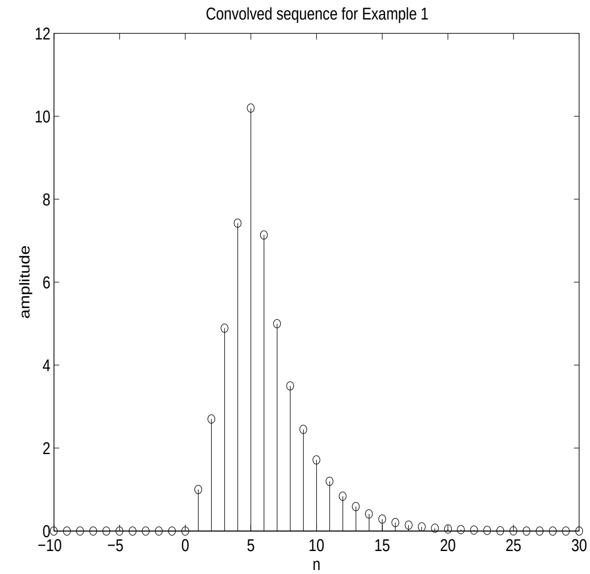
$$x[n] = \begin{cases} n & 0 \leq n \leq 5 \\ 0 \text{otherwise} & \end{cases}, \quad -10 \leq n \leq 10,$$

and

$$h[n] = (\ 07)^n u[n], \quad 0 \leq n \leq 20.$$

Your `Convolve` couldbeusedwithfunctionsfrompreviouslabsasfollows:

```
>>nx=-10:10;%maketheindicesforx
>>x=nx.*(Ustep(nx)-Ustep(nx-6));%makex
>>nh=0:20;%maketheindicesforh
>>h=(0.7).^nh.*Ustep(nh);%makeh
>>[y,ny]=Convolve(x,nx,h,nh);%convolve
>>stem(ny,y);%plot
>>xlabel('n');
>>ylabel('amplitude');
>>title('ConvolvedsequenceforExample1');
```



Exercise 2.1.1:

Use your `Convolve` function to convolve the following sequences:

$$x[n] = u[n] - u[n-6], \quad -10 \le n \le 10,$$

and

$$h[n] = (0.4)^n u[n], \quad 0 \le n \le 10.$$

Use `stem` to plot $x[n]$, $h[n]$, and the result. Derive an analytic expression for the result and compare this with your numerical result.

Exercise 2.1.2:

Use your `Convolve` function to convolve the following sequences:

$$x[n] = \begin{cases} \frac{1}{4} & n = 0 \\ \frac{1}{4}\frac{\sin(\pi n/4)}{\pi n/4} & n \ne 0 \end{cases} \tag{1}$$

$$= \frac{1}{4}\mathrm{sinc}(n/4), \quad -100 \le n \le 100, \tag{2}$$

and

$$h[n] = x[n], \quad -100 \le n \le 100.$$

(You will probably want to use the MATLAB function `sinc` to create $x[n]$.) Plot $x[n]$ and the convolved sequence, call it $y[n]$, over the interval $-100 \le n \le 100$. Use the `axis` command to ensure that the limits on the x-axis are the same for the plots of both $x[n]$ and $y[n]$. Do you find the result surprising? Comment on this.

Exercise 2.1.3: First-order Moving Average System

In Laboratory 1, you considered a system with the following input-output relationship:

$$y[n] = a_0 x[n] + a_1 x[n-1].$$

Derive the impulse response for this system. Consider the input

$$x[n] = 5\,\mathrm{rect}_{20}[n]\,\sin\left(\frac{\pi}{12}n\right), \quad -30 \le n \le 30,$$

and use your `Convolve` function to compute the output sequence for:

1. $a_0 = 1$ and $a_1 = -1$
2. $a_0 = a_1 = 1/2$

Compare these results with those obtained using the `Mave1` function you wrote for Laboratory 1.

Exercise 2.1.4: Cascade Connection of LTI Systems

Consider two LTI systems with the impulse responses:

$$h_1[n] = (\,08)^n u[n],$$

and

$$h_2[n] = \delta[n] - 0.8\delta[n-1].$$

1. Use your `Convolve` function to compute the output of system 1 when its input is

   $$x[n] = \mathrm{rect}_5(n).$$

   When creating the input and impulse-response sequences, use your judgment as to the appropriate indices over which the sequences should be defined. (That is, you need to define `nx` and `nh`).

2. Use your `Convolve` function to compute the output of system 2 when its input is the output of system 1 with the input described above.

3. Use your `Convolve` function to compute the overall impulse response for the cascade connection of systems 1 and 2. Is this result consistent with your previous results? Comment on this result, and on the relationship between systems 1 and 2.

Laboratory 3: The Continuous-Time Fourier Transform

Introduction The purpose of this exercise is to illustrate numerically the concept of the Fourier transform of continuous-time aperiodic signals. In addition, this exercise severs to illustrate the computational questions arising in the numerical calculation of Fourier transforms.

Let $f(t)$ be a real-valued function defined for $-\infty < t < \infty$, satisfying conditions of existence of Fourier transform (such as absolute integrability, a finite number of maxima, minima and discountinuities in any finite interval). We recall that

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$$

is the definition of direct Fourier transform of $f(t)$ where $-\infty < \omega < \infty$.

We will now try to compute numerically $F(\omega)$ for a few examples of $f(t)$.

For real-valued even function $f(t)$, the formula for $F(\omega)$ takes the form

$$F(\omega) = \int_{-\infty}^{\infty} f(t)cos(\omega t)dt \quad (1)$$

In order for $F(\omega)$ to exist, $f(t)$ must decay to 0 as $t \to \infty$ or $t \to -\infty$. Therefore, the computation of $F(\omega)$ can be approximated by

$$F(\omega) = \int_{-a}^{a} f(t)cos(\omega t)dt \quad (2)$$

where a is large enough so that the contribution of the neglected parts of the integral, i.e.

$$\int_{a}^{\infty} f(t)cos(\omega t)dt \quad and \quad \int_{-\infty}^{-a} f(t)cos(\omega t)dt$$

is small compared to the principal part given by formula (2).

This will, for instance, be the case of functions which are zero for $|t| > a$, such as a pulse, or a finite sequence of pulses.

Example Consider a standard unit pulse of width $2a$, centered at 0. The Fourier transform of the pulse function is

$$\begin{aligned} F(\omega) &= \int_{-\infty}^{\infty} P_a(t)e^{-j\omega t}dt \\ &= \int_{-a}^{a} e^{-j\omega t}dt \\ &= 2\frac{sin(\omega a)}{\omega} \end{aligned}$$

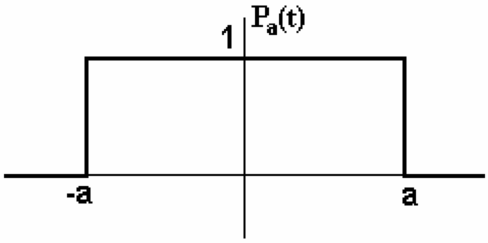


Figure 1: Pulse function

Now, suppose we do not know the analytical form of $F(\omega)$ and we want to compute a numerical approximation to $F(\omega)$

$$\begin{aligned} F(\omega) &= \int_{-\infty}^{\infty} P_a(t)e^{-j\omega t}dt \\ &= \int_{-\infty}^{\infty} P_a(t)cos(\omega t)dt \\ &= \int_{-b}^{b} P_a(t)cos(\omega t)dt \end{aligned} \tag{3}$$

where we choose b such that $P_a(t) = 0,\ t \geq b$. Hence, $b \geq a$. Numerical computation of eq. (3) can be done, for each fixed ω, by one of the numerical integration routines. The simplest one, but not very accurate, is the Euler formula.

We divide the interval $[-b, b]$ into N subintervals of length $h = 2b/N$. Then

$$\int_{-b}^{b} P_a(t)cos(\omega t)dt \cong h \sum_{n=0}^{N-1} P_a(-b+nh)cos[\omega(-b+nh)]$$

Let $a = 1,\ b = 5,\ N = 500$. Then $h = 0.02$.

We now compute this result using Matlab. Let us take a discrete sequence of values of ω, for example, $-10 \leq \omega \leq 10$ with a mesh 0.2 rad/sec.

Matlab script

```
%computation of Fourier transform of a pulse
a=input('pulse width a = ');
A=input('pulse amplitude A = ');
h=input('stepsize h = ');
aT=1.2*a;
T=-aT:h:aT;
om=-20:0.2:20;
%defining the pulse function
pa=zeros(1,length(T));
```

```
for k=1:length(T)
t=(k-1)*h+T(1);
if abs(t) <= a
pa(k)=A;
end
end
%defining an auxiliary string of ones
uv=ones(length(pa),1);
%rapid computation of the sum
for j=1:length(om)
omt=om(j);
Ft(j)=(pa.*cos(omt*T))*uv*h;
end
plot(om,Ft)
title('Fourier transform of a pulse')
xlabel('Frequency in rad per sec')
```

Problems:

1. Retype the Matlab script above and test run it with various values of pulse width and amplitude. Compare the results with the exact values of the Fourier transform given by the analytic formula, and plot the error between the exact values and the numerical approximation. For the lab report, include only two such plots, accompanied by your summary observations on how well the numerical approximation reproduces the true Fourier Transform.

2. Modify the Matlab script to enable you to compute a Fourier Transform of any time function defined by a separate Matlab statement. For example, you can define the pulse function outside of the program, and then call the program computing the Fourier Transform. Since the program provided above works only for even functions of time, you will have to add the imaginary part component (an integral involving $i\ sin(\omega t)$), or replace *cos* by *exp*. You then need to add a computation of the modulus and phase (argument) of the complex Fourier Transform.

3. Compute the Fourier Transform of a unit pulse modulated by a function $cos(\omega_0 t)$ and, in a separate calculation, by $sin(\omega_0 t)$, with $\omega_0 = 2,\ 5,\ 10$. Compare the result with an appropriate analytical result.

4. Compute the Fourier Transform of a sum of three different pulses of width 1, amplitudes 2, 1 and -2, and centered at $-c$, 0, c respectively, with the following values of c: 2, 4, 6. Compare the results with analytical results obtained by superposition.

In your report, put the above plots in a sub-plot format (use "help subplot" to figure out what to do), and print no more than three pages of the lab report. Add clear handwritten explanations of your observations.

Laboratory 4: The Discrete-Time Fourier Transform

In this laboratory assignment, you will investigate some of the basic properties of the discrete-time Fourier transform (DTFT). Recall that the DTFT *analysis* and *synthesis* equations are

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$$

$$x[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n}d\omega$$

respectively, in terms of *radian* frequency ω, or

$$X(F) = \sum_{n=-\infty}^{\infty} x[n]e^{-j2\pi Fn}$$

$$x[n] = \int_{-\frac{1}{2}}^{\frac{1}{2}} X(F)e^{j2\pi Fn}dF$$

respectively, in terms of *digital* frequency F. The DTFT analysis equation is a periodic function of ω with period 2π, or of F with period 1. Typically, the fundamental period is chosen to be $[-\pi, \pi)$ for radian frequencies and $[-1/2, 1/2)$ for digital frequencies.

When using MATLAB to compute the DTFT, we must deal with two issues:

1. Because signals are represented in MATLAB by finite-length vectors, the analysis equations can only be computed for signals that are of finite duration. (An exception will occur when we can derive an analytic expression for a signal's DTFT and simply evaluate it directly.)

2. Whereas the DTFT is a function of a continuous variable, ω or F, it can only be evaluated with MATLAB on a finite grid of points. Therefore, care must be taken to select enough frequencies so that our plots give a smooth approximation to the actual DTFT.

Project 4.1: Computing the DTFT for Finite-Length Signals

Suppose a signal is known to be zero everywhere outside of the interval $N_1 \leq n \leq N_2$. In this case, the DTFT is evaluated as

$$X(e^{j\omega}) = \sum_{n=N_1}^{N_2} x[n]e^{-j\omega n}$$

in radian frequency, or

$$X(F) = \sum_{n=N_1}^{N_2} x[n]e^{-j2\pi Fn}$$

in digital frequency. If we wish to evaluate this summation for M evenly spaced frequencies over the interval $[-\pi, \pi)$ or $[-1/2, 1/2)$, we must evaluate the following set of equations:

$$X\left(e^{j(-\pi+m\Delta_\omega)}\right) = \sum_{n=N_1}^{N_2} x[n]e^{-j(-\pi+m\Delta_\omega)n}, \quad m = 0, 1, \cdots, M-1$$

or

$$X\left(-\tfrac{1}{2} + m\Delta_F\right) = \sum_{n=N_1}^{N_2} x[n] e^{-j2\pi(-\frac{1}{2}+m\Delta_F)n}, \quad m = 0, 1, \cdots, M-1$$

where $\Delta_\omega = 2\pi/M$ and $\Delta_F = 1/M$. A good rule of thumb for obtaining a smooth plot of the DTFT is to select M to be 5 to 10 time larger than the signal duration $N = N_2 - N_1 + 1$.

The Discrete Fourier Transform (DFT) Suppose we wish to compute M evenly-spaced frequency samples over the interval $[0, 2\pi)$ or $[0, 1)$ of a sequence that is known to be zero outside of the interval $0 \le n \le M-1$. The equations for computing these samples in digital frequency are

$$\begin{aligned} X[m] &= X(m\Delta_F) \\ &= \sum_{n=0}^{M-1} x[m] e^{-j2\pi\Delta_F mn} \\ &= \sum_{n=0}^{M-1} x[m] e^{-j\frac{2\pi}{M}mn}, \quad m = 0, 1, \cdots, M-1, \end{aligned}$$

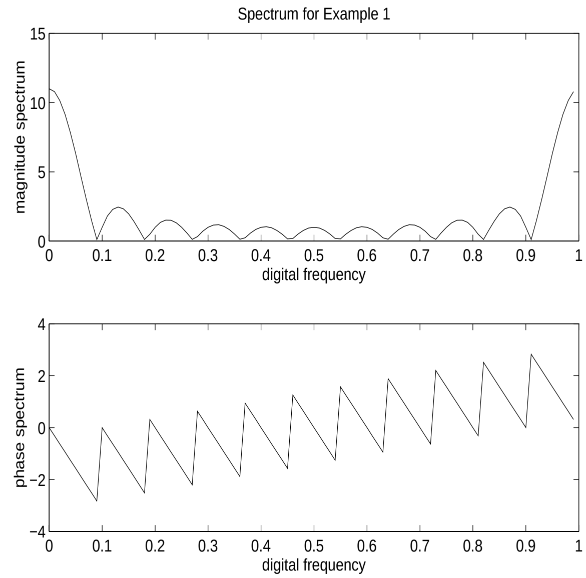
where $\Delta_F = 1/M$. Whereas direct evaluation of these equations requires on the order of M^2 floating-point operations (FLOPS), a computationally-efficient algorithm, known as the *Fast Fourier Transform* (FFT), exists for computing these equations with only the order of $M \log_2 M$ FLOPS. In MATLAB, the FFT is evaluated by the function `fft`.

Example 1 *Consider the sequence*

$$x[n] = \begin{cases} 1 & 0 \le n \le 10 \\ 0 & \text{otherwise} \end{cases} .$$

The following MATLAB *commands would compute* 100 *evenly spaced samples of the DTFT (from* 0 *to* 1 *in digital frequency) of this sequence:*

```
>> x = ones(1, 11);
>> X = fft(x, 100);  % x is augmented with enough zeros to make its length 100
>> m = 0:99;
>> F = m/100;
>> subplot(2,1,1);
>> title('Spectrum for Example 1');
>> plot(F, abs(X));
>> xlabel('digital frequency');
>> ylabel('magnitude spectrum');
>> subplot(2,1,2);
>> plot(F, angle(X));
>> xlabel('digital frequency');
>> ylabel('phase spectrum');
```



Do you understand why the phase jumps by π everywhere the magnitude spectrum is zero?

To learn more about the `fft` function, explore the `fft` sub-category within the `datafun` category of the online help (accessed with the `doc` command).

Suppose we wish to compute M samples of the DTFT over the digital frequency interval $[-1/2, 1/2)$ for a sequence that is known to be zero outside of the interval $N_1 \le n \le N_2$. In this case, we can still use the FFT by observing that

$$\begin{aligned}
X\left(-\tfrac{1}{2} + m\Delta_F\right) &= \sum_{n=N_1}^{N_2} x[n] e^{-j2\pi(-\frac{1}{2}+m\Delta_F)n} \\
&= \sum_{n=N_1}^{N_2} x[n] e^{j\pi n} e^{-j\frac{2\pi}{M}mn} \\
&= \sum_{n=0}^{N_2-N_1} x[n+N_1] e^{j\pi(n+N_1)} e^{-j\frac{2\pi}{M}m(n+N_1)} \\
&= (-1)^{N_1} e^{-j\frac{2\pi}{M}N_1 m} \sum_{n=0}^{N_2-N_1} x[n+N_1](-1)^n e^{-j\frac{2\pi}{M}mn} \\
&= (-1)^{N_1} e^{-j\frac{2\pi}{M}N_1 m} \sum_{n=0}^{M-1} \tilde{x}[n] e^{-j\frac{2\pi}{M}mn} \\
&= (-1)^{N_1} e^{-j\frac{2\pi}{M}N_1 m} \tilde{X}[m],
\end{aligned}$$

where $e^{j\pi} = -1$,

$$\tilde{x}[n] = \begin{cases} x[n+N_1](-1)^n & 0 \le n \le N_2 - N_1 \\ 0 & N_2 - N_1 + 1 \le n \le M-1 \end{cases},$$

and $\tilde{X}[m]$ is the FFT of $\tilde{x}[n]$. Based on this analysis, the following MATLAB function will evaluate M equally spaced samples over the digital frequency interval $[-1/2, 1/2)$ of the DTFT of a finite-length sequence:

begin MATLAB m-file

```
function [X, F] = DTFT(x, N1, M)
%DTFT:   Compute the DTFT of a finite-length sequence at M equally
%        spaced digital frequencies
% inputs
% ------
%   x:    the N-length input sequence
%   N1:   the starting location for the sequence x
%   M:    the number of frequencies for evaluation over the interval
%         of digital frequencies [-1/2, 1/2)
%         (M must be greater than or equal to N)
% outputs
% -------
%   X:    the DTFT values
%   F:    the frequencies for which the DTFT values are evaluated
%
M = fix(M);
N = length(x);
x = x(:);                  % make x a column vector
if (M < N)
  error('DTFT: # frequency samples must be greater than # data samples');
end
n = 0:N-1; n = n(:);        % make n a column vector
m = 0:M-1; m = m(:);        % make m a column vector
F = -0.5 + m/M;
tilde_x = zeros(M,1);
tilde_x(1:N) = x .* (-1).^n;
tilde_X = fft(tilde_x, M);
X = (-1)^N1 * exp(-j*2*pi*N1*m/M) .* tilde_X;
```

end MATLAB m-file

Digital frequencies can, of course, be converted to radian frequencies according to

$$\omega = 2\pi F.$$

Exercise 4.1.1: DTFT of a Rectangular Pulse

Consider the rectangular pulse of duration L samples:

$$x[n] = \begin{cases} 1 & 0 \le n \le L-1 \\ 0 & \text{otherwise} \end{cases}.$$

1. Show that the DTFT of $x[n]$ is

$$X(e^{j\omega}) = \begin{cases} e^{-j\omega(L-1)/2}\,\dfrac{\sin(\omega L/2)}{\sin(\omega/2)} & \omega \ne k2\pi \\ L & \omega = k2\pi \end{cases},$$

for any integer k, or, in terms of digital frequency,

$$X(F) = \begin{cases} e^{-j\pi F(L-1)} \dfrac{\sin(\pi F L)}{\sin(\pi F)} & F \neq k \\ L & F = k \end{cases} .$$

The term $\sin(\pi F L)/\sin(\pi F)$ occurs often in discrete-time signal processing and is referred to as the *aliased sinc* function or the *Dirichlet* function:

$$\text{asinc}(F, L) = \begin{cases} \dfrac{\sin(\pi F L)}{\sin(\pi F)} & F \neq k \\ L & F = k \end{cases} ,$$

for any integer k. Can you show that the asinc function can be equivalently defined as

$$\text{asinc}(F, L) = L\frac{\text{sinc}(FL)}{\text{sinc}(F)}?$$

Write a MATLAB function `Asinc(F,L)` to evaluate the aliased sinc function. A call to this function should be of the form

```
>> X = Asinc(F,L);
```

where `F` is a vector of digital frequencies over which the function should be evaluated and L is the duration parameter. The length of the returned sequence `X` should be the same as that of `F`. Use this function to obtain a plot of the magnitude and phase of the DTFT of $x[n]$ for $L = 10$. Experiment with different numbers of frequency samples. Plot your final results both as a function digital and of radian frequency.

2. Use the `DTFT` function to evaluate the DTFT of $x[n]$ for $L = 10$. Obtain plots of the magnitude and phase spectrum for this signal. Experiment with different numbers of frequency samples, and compare your results with those obtained by directly evaluating the analytic expression with your `Asinc` function.

3. Using your `DTFT` function, obtain plots of the magnitude and phase spectra for $L = 4, 8, 9,$ and 15. By inspecting these plots, can you determine a general rule for the regular spacing of zeros in the magnitude spectrum? How about the location and value of the peak of the magnitude spectrum? How about the location and value of the first side-lobe in the magnitude spectrum?

Exercise 4.1.2: The Shifting Property

Consider the discrete-time sequence

$$x[n] = \begin{cases} 2 - |n| & |n| \leq 2 \\ 0 & \text{otherwise} \end{cases} .$$

1. Use your `DTFT` function to obtain plots of the magnitude and phase spectrum for $x[n]$.

2. Use your `DTFT` function to obtain plots of the magnitude and phase spectrum for $x[n-1]$.

3. Use your `DTFT` function to obtain plots of the magnitude and phase spectrum for $x[n-2]$.
4. Comment on the similarities and differences between the spectra for these three signals. Using the theory of discrete-time signal processing, explain your observations.

Exercise 4.1.3: The Convolution Theorem

Consider the following discrete-time sequences:

$$x[n] = \mathrm{rect}_5(n-2),$$

and

$$h[n] = \begin{cases} 4-|n-4| & 0 \le n \le 8 \\ 0 & \text{otherwise} \end{cases}.$$

1. Use your `DTFT` function to compute and plot the magnitude and phase for $X(e^{j\omega})$ and $H(e^{j\omega})$.
2. Use your `Convolve` routine to compute and plot

$$y[n] = x[n] * h[n].$$

3. Use your `DTFT` function to compute and plot the magnitude and phase for $Y(e^{j\omega})$.
4. Compute and plot the magnitude and phase for the product $X(e^{j\omega})H(e^{j\omega})$. How do these plots compare with your plots of the magnitude and phase for $Y(e^{j\omega})$? Explain this.

Exercise 4.1.4: The Modulation Theorem

Consider the discrete-time sequence

$$x[n] = \mathrm{rect}_{100}(n).$$

1. Use your `DTFT` function to compute and plot the real-part of $X(F)$.
2. Consider the signal

$$y[n] = x[n]\cos\left(\frac{\pi}{4}n\right).$$

   Use your `DTFT` function to compute and plot the real-part of $Y(F)$.
3. Consider the signal

$$z[n] = x[n]\cos\left(\frac{5\pi}{4}n\right).$$

   Use your `DTFT` function to compute and plot the real-part of $Z(F)$.
4. Compare these spectra and comment on their similarities and differences.
5. In all cases, what is the imaginary-part of the sequence's DTFT? Why?

Laboratory 5: Sampling of Continuous-Time Signals

In this laboratory assignment, you will investigate some of the basic principles of the sampling process.

Project 5.1: Sampling a Sinusoid

Consider the continuous-time sinusoidal signal

$$x(t) = \cos(2\pi f_0 t),$$

with frequency f_0 in Hz. If we sample this signal at the rate $f_s = 1/T_s$, we will obtain the discrete-time signal

$$\begin{aligned} x[n] &= x(nT_s) \\ &= \cos(2\pi f_0 T_s n) \\ &= \cos\left(2\pi \frac{f_0}{f_s} n\right). \end{aligned}$$

For each part of this project, use a sampling frequency of $f_s = 8192$ Hz. Also, use frequency-domain sketches in your explanations.

1. For $f_0 = 128, 256, 384,$ and 512 Hz, sample each signal over an interval of about 16 ms, and plot the resulting signal. Use the `subplot` command to put your plots in the same figure. Does the frequency of the discrete-time signal appear to be increasing? Explain.

2. For $f_0 = 7680, 7808, 7936,$ and 8064 Hz, sample the signal over an interval of about 16 ms, and plot the resulting signal. Use the `subplot` command to put your plots in the same figure. Does the frequency of the discrete-time signal appear to be increasing? Explain.

3. For the frequencies specified in part (1), sample each signal over an interval of about 0.25 s. Make a new signal by concatenating the four sampled signals together. This new signal will contain the four 0.25 s segments. Using a machine with a speaker and a D/A converter[1], use the MATLAB `sound` command to listen to this signal. Can you hear four distinct tones? Are they increasing in frequency?

4. For the frequencies specified in part (2), sample each signal over an interval of about 0.25 s. Make a new signal by concatenating the four sampled signals together. Again, use the `sound` command to listen to this signal. Can you hear four distinct tones? Are they increasing in frequency? Explain.

5. For $f_0 = 3840, 3968, 4096, 4224,$ and 4352 Hz, sample each signal over an interval of about 0.25 s. Make a new signal by concatenating the five sampled signals together. This new signal will contain the five 0.25 s segments. Again, use the `sound` command to listen to this signal. Can you hear five distinct tones? Are they increasing in frequency? Explain.

[1] All terminals in the Maxwell lab except `maxwell1 (b & c)` and `maxwell2 (b & c)` have this capability.

Project 5.2: Sampling a Chirp

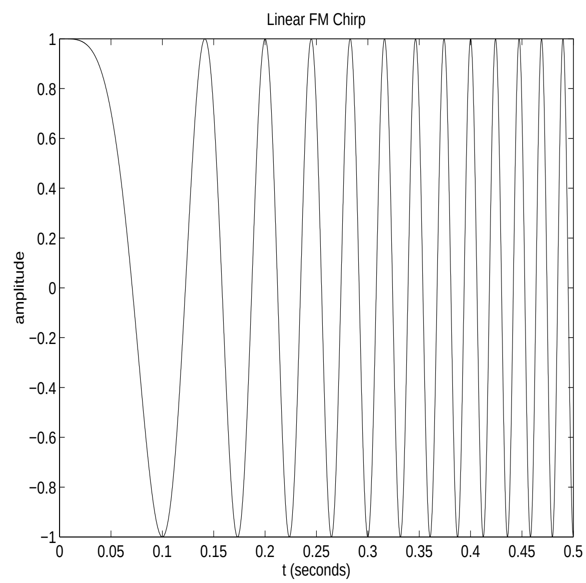
A continuous-time, linear frequency-modulated (LFM) chirp is a sinusoidal signal whose frequency changes linearly with time:

$$x(t) = \cos(\pi \alpha t^2).$$

This waveform is of particular importance in radar and sonar applications. As an example, a plot of the LFM signal

$$x(t) = \cos(\pi 100 t^2)$$

over the time interval $[0, 0.5]$ seconds is shown below:



The instantaneous frequency of this signal is found by taking the time derivative of the phase:

$$\begin{aligned} f(t) &= \frac{\partial \phi(t)}{\partial t} \\ &= \frac{\partial \pi \alpha t^2}{\partial t} \\ &= 2\pi \alpha t, \end{aligned}$$

from which we see that the instantaneous frequency of the signal is αt Hz and exhibits a linear variation in time. For each part of this project, use a sampling frequency of $f_s = 8192$ Hz.

1. Let $\alpha = 2048$. Sketch the instantaneous frequency of this signal as a function of time over the interval $[0, 2]$ seconds, clearly showing the starting and ending values. Sample the signal over an interval of 2 seconds, and use the `sound` command to listen to this signal. Does the frequency appear to be increasing linearly with time?

2. Let $\alpha = 8192$. Sketch the instantaneous frequency of this signal as a function of time over the interval $[0, 2]$ seconds, clearly showing the starting and ending values. Sample the signal over an interval of 2 seconds, and use the `sound` command to listen to this signal. Does the frequency appear to be increasing linearly with time? If not, use your sketch of the instantaneous frequency and its relationship to the sampling frequency to explain the sound that you hear.

Project 5.3: Sampling Multiple Sinusoids

Consider the continuous-time signal

$$x(t) = \cos(2\pi f_0 t) - \cos(2\pi f_1 t),$$

with f_0 and f_1 in Hz. For each part of this project, use $f_0 = 100$ Hz and $f_1 = 200$ Hz.

1. Let the sampling frequency be $f_s = 400$ Hz, and sample the signal over the time interval $[0, 0.1]$ seconds. Plot the discrete-time sequence using the `stem` command.

2. Let the sampling frequency be $f_s = 1600$ Hz, and sample the signal over the time interval $[0, 0.1]$ seconds. Plot the discrete-time sequence using the `stem` command.

3. Let the sampling frequency be $f_s = 300$ Hz, and sample the signal over the time interval $[0, 0.1]$ seconds. Use a frequency-domain sketch to predict what the sampled signal should look like. Plot the discrete-time sequence and compare the plot with your prediction. (Be sure to notice the scale of the amplitude axis.)

Use the `subplot` command to put all of the plots on the same figure.

Project 5.4: Reconstruction From Samples

The *Nyquist sampling theorem* states that if $x_c(t)$ is a bandlimited signal with

$$X_c(j\Omega) = 0 \quad \text{for} \quad |\Omega| > \Omega_N,$$

or, equivalently,

$$X_c(f) = 0 \quad \text{for} \quad |f| > f_N,$$

then $x_c(t)$ is uniquely determined by its samples $x[n] = x_c(nT)$, for $n = 0, \pm1, \pm2, \ldots$, provided that the *sampling period* satisfies

$$T < \frac{\pi}{\Omega_N} = \frac{1}{2f_N},$$

or, equivalently, the *sampling frequency* satisfies

$$f_s > 2f_N,$$

or

$$\Omega_s > 2\Omega_N.$$

In this project, you will investigate the importance of using the proper form of interpolation when attempting to reconstruct a signal from its samples.

Consider the Gaussian pulse signal:

$$x(t) = e^{-a^2 t^2}.$$

The Fourier transform of this signal is

$$X(j\Omega) = \frac{\sqrt{\pi}}{a} e^{-\left(\frac{\Omega}{2a}\right)^2},$$

or

$$X(f) = \frac{\sqrt{\pi}}{a} e^{-(\pi f/a)^2}.$$

For the following exercises, let $a = 100$.

1. Use the MATLAB `plot` command to plot the magnitude spectrum for the Fourier transform of this signal over the interval $[-150, 150]$ Hz. Verify that the spectrum is approximately zero for $|f| > 75$ Hz. Although this signal is not bandlimited (its tails extend to $\pm\infty$), we can approximate it as bandlimited. Based on this information, the minimum sample spacing required to uniquely specify this signal by its samples is approximately

$$T < \frac{1}{150} \text{seconds.}$$

2. Use the `plot` command and a sample spacing of 0.0001 s to plot $x(t)$ over the interval $[-1/30, 1/30]$ s. Because the `plot` command connects each sample with a straight line, the continuous plot represents a reconstruction of $x(t)$ from its samples through *linear interpolation*. Notice, however, that the sampling frequency used here is approximately 70 times larger than is required in the Nyquist theorem.

3. Use the sampling frequency $f_s = 150$ Hz to sample the signal over the time interval $[-1/30, 1/30]$ seconds. Use the `plot` command to connect the samples with straight lines. This sampling frequency satisfies the Nyquist criterion; however, does the signal look like a Gaussian pulse when reconstructed by linear interpolation?

4. Recall that reconstruction of a signal from its Nyquist samples requires that the signal be reconstructed with an ideal low-pass filter. In the time domain, this corresponds to a *sinc interpolation* filter:

$$x_r(t) = \sum_{n=-\infty}^{\infty} x[n] \text{sinc}\left(\frac{t - nT}{T}\right).$$

   (a) Let $T = 1/150$ seconds, and use the `plot` command and a sample spacing of 0.0001 s to plot $\text{sinc}(t/T)$ over the interval $[-0.04, 0.04]$ s. The process of sinc interpolation simply scales and shifts this function according to $x[n]$ and the sample spacing, and then adds all of the shifted functions.

   (b) If

$$x[n] = x_c(nT), \quad n = n(1), n(2), \cdots, n(N),$$

   then the command

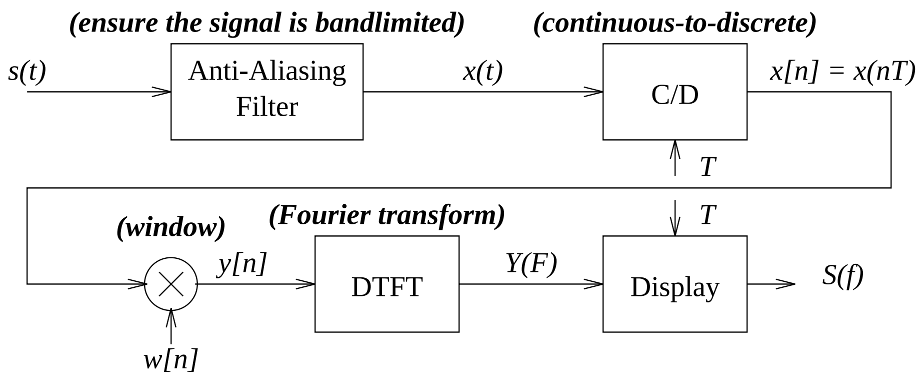
```
>> [xr, tr] = SincInterp(x, n, T);
```

   will perform sinc interpolation from the samples stored in the vector `x`, to create a new vector `xr` which contains samples of the original signal sampled at a sampling rate which is 20 times greater than $1/T$. The times at which the new samples are taken are stored in the vector `tr`. Type `help SincInterp` to learn more about this function. For the Nyquist-sampled signal obtained in Part 3, use this function to plot the reconstructed signal. Adjust the limits on the x- and y-axes with the `axis` command so that they are the same as for your linearly-interpolated plot from Part 3. Compare the two plots. Comment on the importance of using the appropriate interpolation when samples are obtained at or near the Nyquist rate.

Laboratory 6: Spectrum Analysis

Spectrum analysis often refers to the task of processing a continuous-time signal to compute the signal's frequency spectrum; either magnitude, phase, or both. In this laboratory assignment you will investigate and explore some of the basic methods used for the frequency-domain analysis of signals.

Many modern instruments for spectrum analysis use digital signal processing techniques. An example of one such instrument is the SR770 FFT Network Analyzer manufactured by Stanford Research Systems[1]. The basic components of a digital spectrum analyzer are shown below:



The anti-aliasing filter is used to ensure that the input signal is bandlimited to a frequency that is appropriate for the sampling frequency. As an example, if the sampling frequency for the continuous to discrete (C/D) converter is $f_s = 128$ kHz, then the anti-aliasing filter should suppress frequencies greater than 64 kHz. The C/D converter converts the continuous-time (CT) signal to a discrete-time (DT) signal; the sampling rate is $T = 1/f_s$. The window function is needed to truncate the DT signal to an interval of length N; this then allows for the numerical computation of the signal's spectrum using an algorithm such as the one developed in our laboratory on the discrete-time Fourier transform (DTFT). After the Fourier transform of the windowed signal has been computed, the final step is the display of the spectrum with the frequency axis appropriately labeled as specified by the sampling period T.

Because

$$y[n] = x[n]w[n],$$

the modulation or windowing property of the DTFT tells us that

$$Y(e^{j\omega}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\theta})W(e^{j(\omega-\theta)})d\theta,$$

in radian frequency, or

$$Y(F) = \int_{-1/2}^{1/2} X(F')W(F-F')dF'$$

in digital frequency, where X and W are the DTFT of $x[n]$ and $w[n]$, respectively. Because of this, the DTFT of the window function should be a function that is highly concentrated around $\omega = 0$ or $F = 0$. For instance, if the DTFT of the window function is an impulse train (implying that the window function is constant and of infinite duration)

$$W(f) = \sum_{k=-\infty}^{\infty} \delta(F-k),$$

[1]This is the spectrum analyzer used in our communications laboratories.

then $Y(F)$ will simply be $X(F)$. However, any practical window function must be of finite duration and some *blurring* of the spectrum will occur.

Upon displaying the signal's spectrum, the frequency axis should be adjusted according to the following conversion rules:

- *Digital frequency to Hz*: $f = F/T$
- *Digital frequency to Radians/Second*: $\Omega = 2\pi F/T$
- *Radian frequency to Hz*: $f = \omega/(2\pi T)$
- *Radian frequency to Radians/Second*: $\Omega = \omega/T$

Given that our DTFT algorithm (MATLAB function `Dtft`) produces samples of the spectrum in digital frequency, and that most spectrum analyzers specify frequency in Hz, we will focus on the conversion from digital frequency to Hz:

$$Y(F) = S(F/T),$$

or

$$S(f) = Y(fT).$$

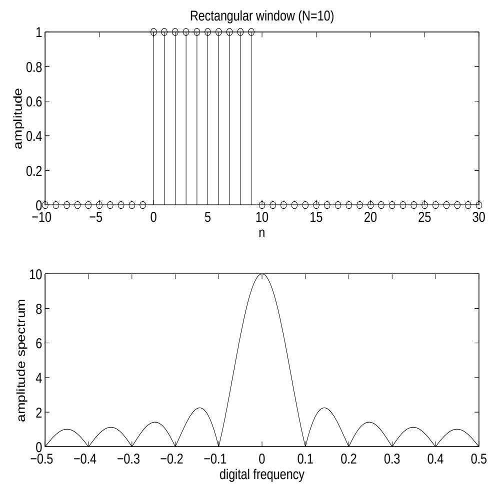
Project 6.1: Rectangular Window

The simplest and most straight-forward of the window functions is the *Rectangular* window function:

$$w[n] = \begin{cases} 1 & 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases}.$$

The following MATLAB commands will create and plot a rectangular window with $N = 10$, and will also compute and plot its magnitude spectrum:

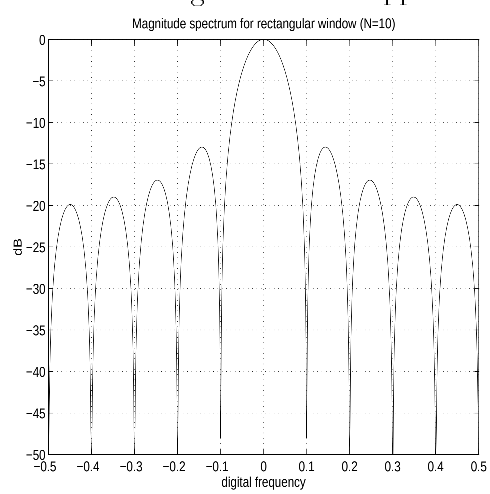
```
>> n = -10:30;
>> N = 10;
>> w = Ustep(n) - Ustep(n-N);
>> [W,F] = Dtft(w, n(1), 1024);
>> subplot(2,1,1);
>> stem(n,w);
>> xlabel('n')
>> ylabel('amplitude')
>> title('Rectangular window (N=10)')
>> subplot(2,1,2)
>> plot(F, abs(W))
>> xlabel('digital frequency')
>> ylabel('magnitude')
```



It is often convenient to make magnitude spectrum plots on a normalized dB scale, where the peak of the magnitude spectrum is normalized to 0 dB. The commands

```
>> dBplot(F, abs(W), -50);
>> xlabel('digital frequency');
>> title('Magnitude spectrum for rectangular window (N=10)')
```

will accomplish this with the normalized magnitude axis clipped at -50 dB:



Notice that the y-axis is automatically labeled as "dB". Type `help dBplot` to learn more about this command.

The quality of a window function is often specified by the width of the mainlobe and by the height of the largest sidelobe of its magnitude spectrum. Often these two qualities are specified by the frequency at which the normalized magnitude of the mainlobe falls to -3dB (called the 3dB point), and by the height (or attenuation) of the largest sidelobe (usually specified in normalized dB). For instance, inspection of the previous figure shows that the 10-point rectangular window has a 3dB width of approximately 0.06 cycles/sample, and the height of its largest sidelobe is approximately -13 dB.

For the following window lengths: 16, 32, 41, and 64, compute the DTFT (using `Dtft`) and plot the magnitude spectrum on a dB scale (using `dBplot`). Use the `subplot` command to put the plots in the same figure.

1. What is the height of the first sidelobe as a function of the window length? Can you determine an analytic expression for determining this height for arbitrary N?

2. What is the 3dB width of the mainlobe as a function of the window length? Can you determine an analytic expression for determining this width for arbitrary N?

Project 6.2: Other Commonly Used Windows

Some of the most commonly used windows in signal processing and spectrum analysis include the *Rectangular*, *Bartlett*, *Hanning*, *Hamming*, and *Blackman* windows[2]. Each of these windows can be generated through MATLAB function calls and are defined by the following equations:

- `boxcar(N)` (the rectangular window):

$$w[n] = \begin{cases} 1, & 0 \le n \le N-1 \\ 0, & \text{otherwise} \end{cases}$$

- `bartlett(N)`:

$$w[n] = \begin{cases} \frac{2n}{N-1}, & 0 \le n \le \frac{N-1}{2} \\ 2 - \frac{2n}{N-1}, & \frac{N-1}{2} < n \le N-1 \\ 0, & \text{otherwise} \end{cases}$$

- `hanning(N)`:

$$w[n] = \begin{cases} 0.5 - 0.5\cos\left(\frac{2\pi(n+1)}{N+1}\right), & 0 \le n \le N-1 \\ 0, & \text{otherwise} \end{cases}$$

- `hamming(N)`:

$$w[n] = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), & 0 \le n \le N-1 \\ 0, & \text{otherwise} \end{cases}$$

- `blackman(N)`:

$$w[n] = \begin{cases} 0.42 - 0.5\cos\left(\frac{2\pi(n-1)}{N-1}\right) + 0.08\cos\left(\frac{2\pi(n-1)}{N-1}\right), & 0 \le n \le N-1 \\ 0, & \text{otherwise} \end{cases}$$

1. Using the `plot` command with multiple arguments, obtain plots of the *Rectangular*, *Bartlett*, *Hanning*, *Hamming*, and *Blackman* windows for $N = 100$, all on the same axes. Based on these time-domain plots, does any window seem "best"? Is it the rectangular window?

2. Obtain dB plots of the magnitude spectrum for each of these windows. Use the `axis` command or the MATLAB colon operator to display only the frequency interval that contains the first few sidelobes. Comment on the differences between the mainlobe width and sidelobe height for each of these windows. Which window has the most narrow mainlobe? Which window has the lowest sidelobes? Which window do you think would be best for a spectrum analyzer? Why?

[2]The Bartlett, Hanning, Hamming, and Blackman windows are all named after their originators. The Hanning window is associated with Julius von Hann, an Austrian meteorologist, and is sometimes referred to as the Hann or von Hann window. The term "hanning" was used by Blackman and Tukey (*The Measurement of Power Spectra*, 1958) to describe the operation of applying this window to a signal. (From Oppenheim and Schafer, *Discrete Time Signal Processing*.)

Project 6.3: Spectrum Analysis for a Tone

Consider the continuous-time signal

$$x_c(t) = \cos(2\pi f_0 t),$$

where $f_0 = 30$ kHz. Suppose you are using a digital spectrum analyzer with a sampling frequency $f_s = 128$ kHz. Using a window size of $N = 128$ samples, compute the DTFT of the sampled signal after it has been multiplied by the window, and display the magnitude spectrum with the frequency axis labeled in Hz (you must convert the digital frequencies returned by the `Dtft` function to Hz). Using each of the windows (*Rectangular*, *Bartlett*, *Hanning*, *Hamming*, and *Blackman*), obtain plots of the magnitude spectrum for this signal. Be sure to label the frequency axis appropriately. Use the `axis` command to "zoom-in" on the frequency interval that contains the mainlobe and a few sidelobes of the spectrum of this signal. In your judgment, which window gives the best performance?

Project 6.4: Spectral Resolution for Two Tones

Consider a combination of continuous-time tones at closely spaced frequencies f_0 and $f_0 + \Delta_f$:

$$x_c(t) = \cos(2\pi f_0 t) + \cos(2\pi[f_0 + \Delta_f]t),$$

where $f_0 = 30$ kHz and Δ_f is the frequency offset. Again, suppose your are using a digital spectrum analyzer with a sampling frequency $f_s = 128$ kHz. Our goal here is to determine the minimum frequency offset Δ_f such that our spectrum analyzer can clearly distinguish the two frequency components of this signal. For this project, focus only on the *Rectangular* and *Hanning* windows. Beginning with $\Delta_f = 10$ kHz and using a window size of $N = 128$ samples, compute the DTFT of the sampled signal after it has been multiplied by the window, and display the magnitude spectrum with the frequency axis labeled in Hz. Use the `axis` command to localize the frequency axis to the region of interest. Can you clearly distinguish the two frequency components with both windows? Does one window seem to perform better than the other? Can you suggest modifications that would increase the resolution of your spectrum analyzer?

Now, decrease Δ_f until you can no longer distinguish two separate frequency components. Do this for both windows. Does one window have better "resolution" than the other? That is, can one of the windows distinguish the two tones for a smaller Δ_f than the other?

Laboratory 7: FIR Filter Design Using Window Functions

In this laboratory assignment you will investigate and explore a method of FIR filter design known as the *window method*. This method generally begins with the specification of a desired frequency response for an LTI system: $H_d(e^{j\omega})$ in radian frequency, or $H_d(F)$ in digital frequency. The impulse response for this system is then obtained through the Fourier synthesis equation:

$$\begin{aligned} h_d[n] &= \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\omega})e^{j\omega n}d\omega \\ &= \int_{-1/2}^{1/2} H_d(F)e^{j2\pi Fn}dF. \end{aligned}$$

However, because the specification of the desired system often includes a piecewise constant or piecewise functional frequency response, the desired system's impulse response is often non-causal and of infinite duration. As an example, suppose the desired system is an ideal low-pass filter with a cut-off frequency of $\pi/4$ in radian frequency or $1/8$ in digital frequency. The desired impulse response is then

$$\begin{aligned} h_d[n] &= \frac{1}{2\pi}\int_{-\pi/4}^{\pi/4} e^{j\omega n}d\omega \\ &= \frac{e^{j\pi n/4} - e^{-j\pi n/4}}{j2\pi n} \\ &= \frac{\sin(\pi n/4)}{\pi n} \\ &= \frac{1}{4}\mathrm{sinc}(n/4). \end{aligned}$$

Obviously this impulse response is neither causal nor of finite duration. To make the impulse response finite, we might truncate the sequence:

$$h_t[n] = \begin{cases} h_d[n], & -M \le n \le M \\ 0, & \text{otherwise} \end{cases}.$$

This system, however, would not be causal. To make the system causal and of finite duration, we could delay the truncated impulse response by M samples:

$$\begin{aligned} h[n] &= h_t[n-M] \\ &= \begin{cases} h_d[n-M], & 0 \le n \le 2M \\ 0, & \text{otherwise} \end{cases}. \end{aligned}$$

This two-step process can be viewed as

$$h[n] = h_d[n-M]\mathrm{rect}_M[n-M],$$

where $\mathrm{rect}_M[n]$ is the rectangular window function of width $2M+1$. In general, any window function $w[n]$ could be used to truncate the impulse response:

$$h[n] = h_d[n-M]w[n-M].$$

Much insight can be gained by examining the windowing operation in the frequency domain. Most times we specify the desired frequency response as a real-valued symmetric function so that the desired impulse response is also a real-valued symmetric function. Then, because of the time–delay property for the DTFT:

$$h_d[n-M] \longleftrightarrow H_d(e^{j\omega})e^{-j\omega M},$$

or

$$h_d[n-M] \longleftrightarrow H_d(F)e^{-j2\pi FM},$$

the frequency response for the delayed system will have *generalized linear phase*. Furthermore, if the window function is a real-valued symmetric function, then its Fourier transform ($W(e^{j\omega})$ or $W(F)$) will also be a real-valued symmetric function, and, as with the shifted impulse response,

$$w[n-M] \longleftrightarrow W(e^{j\omega})e^{-j\omega M},$$

or

$$w[n-M] \longleftrightarrow W(F)e^{-j2\pi FM}.$$

Because multiplication in the time domain results in convolution in the frequency domain, we have

$$\begin{aligned} h_d[n-M]w[n-M] &\longleftrightarrow \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\theta})e^{-j\theta M} W(e^{j(\omega-\theta)})e^{-j(\omega-\theta)M}\,d\theta \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\theta})W(e^{j(\omega-\theta)})d\theta\ e^{-j\omega M}, \end{aligned}$$

or

$$\begin{aligned} h_d[n-M]w[n-M] &\longleftrightarrow \int_{-1/2}^{1/2} H_d(F')e^{-j2\pi F'M} W(F'-F)e^{-j2\pi(F'-F)M}\,dF' \\ &= \int_{-1/2}^{1/2} H_d(F')W(F'-F)dF'\ e^{-j2\pi FM}, \end{aligned}$$

and the resulting frequency response will also have generalized linear phase. Because the resulting frequency response is the convolution of the desired frequency response with the Fourier transform of the window function (times the linear phase term), windows that are highly concentrated in the frequency domain will result in the best approximation of the desired frequency response.

All of the windows we have previously considered – *Rectangular*, *Bartlett*, *Hanning*, *Hamming*, and *Blackman* – can be used for FIR filter design[1]. In addition to these, a more flexible and general window is the *Kaiser* window:

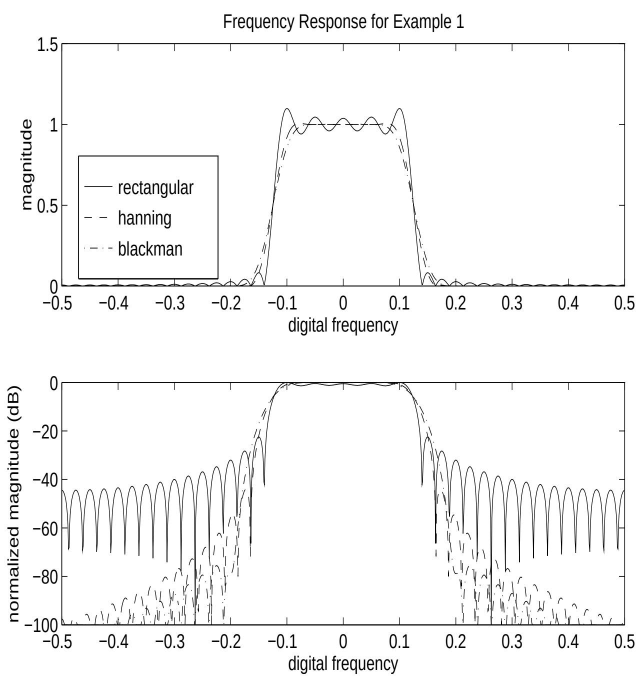
$$w[n] = \begin{cases} \dfrac{I_0\left(\beta\sqrt{1-[2(n-M)/(N-1)]^2}\right)}{I_0(\beta)}, & 0 \le n \le N-1 \\ 0, & \text{otherwise} \end{cases},$$

where $M = (N-1)/2$, $I_0(x)$ is the zeroth-order modified Bessel function of the first kind, and β is a parameter that adjusts the width and shape of the window. The MATLAB function call for an `N`-length Kaiser window with parameter `beta` is `kaiser(N,beta)`. The utility of this window lies in its ability to adjust the trade-off between mainlobe width and sidelobe height. These window characteristics are important because they control the transition bandwidth and passband ripple when the window method is used for FIR filter design.

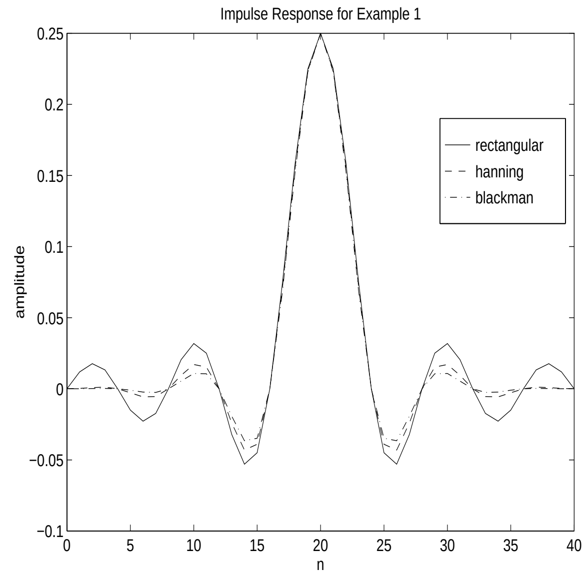
[1] Recall that the MATLAB definition for each of these windows is such that they are already shifted so that they begin at $n = 0$.

Example 1 Suppose we wish to design a length-41 low-pass filter with the cutoff frequency $\pi/4$ (in radian frequency) or 1/8 (in digital frequency). The following MATLAB code would accomplish this using the *Rectangular*, *Hanning*, and *Blackman* windows:

```
>> N = 41; n=0:N-1;
>> hd = (0.25)*sinc((n-20)/4);      % the desired impulse response (truncated)
>> w1 = boxcar(N);                  % the rectangular window
>> h1 = hd(:) .* w1(:);
>> [H1,F] = Dtft(h1,n(1),1000);
>> w2 = hanning(N);                 % the hanning window
>> h2 = hd(:) .* w2(:);
>> [H2,F] = Dtft(h2,n(1),1000);
>> w3 = blackman(N);                % the blackman window
>> h3 = hd(:) .* w3(:);
>> [H3,F] = Dtft(h3,n(1),1000);
>> subplot(2,1,1)
>> plot(F,abs(H1),'-',F,abs(H2),'--',F,abs(H3),'-.');
>> legend('-','rectangular','--','hanning','-.','blackman');
>> xlabel('digital frequency');
>> ylabel('magnitude');
>> title('Frequency Response for Example 1');
>> subplot(2,1,2)
>> plot(F,dB(abs(H1),-100),'-',F,dB(abs(H2),-100),'--',F,dB(abs(H3),-100),'-.');
>> xlabel('digital frequency');
>> ylabel('normalized magnitude (dB)');
```



To learn more about the function `dB`, type `help dB` at the MATLAB prompt. The impulse response for each of the filters is shown in the following figure:



Project 7.1: High-Pass Filter Design

Design a length-61 linear-phase FIR high-pass filter with a band edge of $7\pi/8$ in radian frequency or $7/16$ in digital frequency using the *Rectangular*, *Bartlett*, *Hanning*, *Hamming* and *Blackman* windows. For each window, plot the impulse response and the magnitude of the frequency response (on a dB scale). Which filter do you feel is "best"?

Project 7.2: Band-Pass Filter Design

Design a length-61 linear-phase FIR band-pass filter with the passband $\pi/4$ through $\pi/2$ in radian frequency or $1/8$ through $1/4$ in digital frequency using the window function that you classified as "best" in the high-pass filter design. Plot the impulse response and the magnitude response (on a dB scale). Repeat for a length-23 and a length-401 filter. Comment on your results.

Project 7.3: Kaiser Window

For a frequency selective filter, let δ define the maximum percent ripple in the passband (percent ripple $= 100 \times \delta$), and

$$\Delta\omega = |\omega_s - \omega_p|,$$

define the width of the transition band, where ω_s is the stop-band frequency and ω_p is the pass-band frequency. Furthermore, let

$$A = -20\log_{10}\delta.$$

As determined by Kaiser, empirical formulas for the β and N that are needed to achieve specified values of δ and $\Delta\omega$ are

$$\beta = \begin{cases} 0.1102(A-8.7), & A > 50 \\ 0.5842(A-21)^{0.4} + 0.07886(A-21), & 21 \le A \le 50 \\ 0, & A < 21 \end{cases},$$

and

$$N - 1 = \frac{A - 8}{2.285\Delta\omega}.$$

1. Write a MATLAB function to evaluate N and β for specified values of δ and $\Delta\omega$. A call to your function should be of the form

   ```
   >> [N, beta] = KaiserParam(delta, delta_omega);
   ```

   where `delta` is the ripple parameter and `delta_omega` is the transition region width. Use this function to obtain a plot of β versus δ to get a feel for the typical range for β. For your plot, vary δ from 0.0001 to 0.1. (Hint: write your function so that `delta`, `N` and `beta` can be vectors.)

2. Design a length-61 linear-phase FIR high-pass filter using the same specifications as in Project 7.1, but using a Kaiser window with $\beta = 2, 6$, and 9. Plot the impulse response and the magnitude response (in dB). Compare these with each other and with the filters designed with the other windows. Comment on your results.

3. Design a linear-phase FIR high-pass filter using the same specifications as in Project 7.1 and a Kaiser window, but select N and β for 1% ripple in the passband and a transition region of width $\pi/10$ in radians. Use your function `KaiserParam` to determine the window width N and β. Plot the impulse response and the magnitude response (in dB). Compare these with the filters designed with the other windows. Comment on your results.