**amplitudeScaling.m**

clc;

clear;

close all;

x=input(' Enter the sequence x =') % x must be vector

n= input(' The index of the sequence n =') % n indicates the starting and ending position of a sequence x

a=input(' The scaling factor a =') % a is a scalar

y=a.\*x;

subplot(2,1,1); % or subplot(211);

stem(n,x);

subplot(2,1,2); % or subplot(212);

stem(n,y);

Enter the sequence x =[2 4 6 8 10]

x =

2 4 6 8 10

The index of the sequence n =[0 1 2 3 4]

n =

0 1 2 3 4

The scaling factor a =2

a =

2



**timeshifting.m**

clc;

clear;

close all;

x=input('Enter the sequence x =') % x must be vector

n=input(' The index of the sequence n =')

k=input(' The shifting factor k =') % k is a scalar

n1=n+k

% k is positive -- delayed version

% k is negative -- advanced version

subplot(2,1,1);

stem(n,x);

subplot(2,1,2);

stem(n1,x);

Enter the sequence x =[1 3 5 7 9]

x =

1 3 5 7 9

The index of the sequence n =[0 1 2 3 4]

n =

0 1 2 3 4

The shifting factor k =3

k =

3

n1 =

3 4 5 6 7



**folding.m**

clc;

clear;

close all;

x=input(' Enter the sequence x =') % x must be vector (magnitudes)

n=input('The index of the sequence n=') %length of x and n must be same

m=-fliplr(n);

y=fliplr(x);

subplot(2,1,1);

stem(n,x);

subplot(2,1,2);

stem(m,y);

Enter the sequence x =[2 3 4 5 6]

x =

2 3 4 5 6

The index of the sequence n=[2 3 4 5 6]

n =

2 3 4 5 6

>>



**fold1.m**

clc;

clear;

close all;

x=[1 2 3 6 2 -3 1];

n=[-2 -1 0 1 2 3 4];

subplot(211);

stem(n,x);

y=fliplr(x);

m=-fliplr(n);

subplot(2,1,2);

stem(m,y);

% folding of signal

clc;

clear;

close all;

x=[1 2 3 6 2 -3 1];

n=[-2 -1 0 1 2 3 4];

subplot(211);

stem(n,x);

y=fliplr(x);

m=-fliplr(n);

subplot(2,1,2);

stem(m,y);

% folding of signal



**Down and Up sampling**

clc;

clear;

close all;

x=input(' Enter the sequence x =') % x must be vector

y1=downsample(x,2)

y2=downsample(x,2,1)

z1=upsample(x,2)

z2=upsample(x,2,1)

subplot(2,2,1);

stem(y1);

subplot(2,2,2);

stem(y2);

subplot(2,2,3);

stem(z1);

subplot(2,2,4);

stem(z2);



Enter the sequence x =[1 2 3 4 5 6 7 8 9]

x =

1 2 3 4 5 6 7 8 9

y1 =

1 3 5 7 9

y2 =

2 4 6 8

z1 =

Columns 1 through 17

1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9

Column 18

0

z2 =

Columns 1 through 17

0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0

Column 18

9

**5. Addition of two signals**

****

****

****

function[y,n]=sigadd()

x1=input(' Enter the sequence x1 = ') % x1 and x2 are vectors

n1=input(' The index of the sequence x1 is n1 = ')

subplot(311);

stem(n1,x1);

x2=input(' Enter the sequence x2 = ') % x1 and x2 are vectors

n2=input(' The index of the sequence x2 is n2 = ')

subplot(312);

stem(n2,x2);

n=min(min(n1),min(n2)):max(max(n1),max(n2));

y1=zeros(1,length(n));

y2=y1;

y1(find((n>=min(n1))&(n<=max(n1))))=x1;

y2(find((n>=min(n2))&(n<=max(n2))))=x2;

y=y1+y2;

% subplot(number of rows,number of columns,figure number);

subplot(313);

stem(n,y);

>> sigadd()

Enter the sequence x1 = [1 1 2 2 3 4]

x1 =

1 1 2 2 3 4

The index of the sequence x1 is n1 = [-4 -3 -2 -1 0 1]

n1 =

-4 -3 -2 -1 0 1

Enter the sequence x2 = [2 3 4 5]

x2 =

2 3 4 5

The index of the sequence x2 is n2 = [-1 0 1 2]

n2 =

-1 0 1 2

ans =

1 1 2 4 6 8 5

>>

****

**----------------------------------------------------------------**

>> x=[1 2 3 1 -1 2 3 4]

x =

1 2 3 1 -1 2 3 4

>> downsample(x,2)

ans =

1 3 -1 3

>> downsample(x,3)

ans =

1 1 3

>> upsample(x,2)

ans =

1 0 2 0 3 0 1 0 -1 0 2 0 3 0 4 0

>> upsample(x,3)

ans =

Columns 1 through 20

1 0 0 2 0 0 3 0 0 1 0 0 -1 0 0 2 0 0 3 0

Columns 21 through 24

0 4 0 0

>> downsample(x,2,1)

ans =

2 1 2 4

>> upsample(x,2,1)

ans =

0 1 0 2 0 3 0 1 0 -1 0 2 0 3 0 4

>> downsample(x,2,3)

Error using updownsample>parseUpDnSample (line 55)

Offset must be from 0 to N-1.

Error in updownsample (line 17)

phase = parseUpDnSample(str,N,varargin{:});

Error in downsample (line 33)

y = updownsample(x,N,'Down',varargin{:});

>> downsample(x,4,2)

ans =

3 3

>> downsample(x,2,2)

Error using updownsample>parseUpDnSample (line 55)

Offset must be from 0 to N-1.

Error in updownsample (line 17)

phase = parseUpDnSample(str,N,varargin{:});

Error in downsample (line 33)

y = updownsample(x,N,'Down',varargin{:});

>> downsample(x,2,1)

ans =

2 1 2 4

>>

**Assignments:**

1. **Signal multiplication**



clc;

clear;

close all;

x1=input(' Enter the sequence x1 = ') % x1 and x2 are vectors

n1=input(' The index of the sequence x1 is n1 = ')

x2=input(' Enter the sequence x2 = ') % x1 and x2 are vectors

n2=input(' The index of the sequence x2 is n2 = ')

n=min(min(n1),min(n2)):max(max(n1),max(n2));

y1=zeros(1,length(n));

y2=y1;

y1(find((n>=min(n1))&(n<=max(n1))))=x1;

y2(find((n>=min(n2))&(n<=max(n2))))=x2;

y=y1.\*y2; % signal multiplication

% subplot(number of rows,number of columns,figure number);

subplot(311);

stem(n1,x1);

subplot(312);

stem(n2,x2);

subplot(313);

stem(n,y);

Enter the sequence x1 = [1 1 2 2 3 4]

x1 =

1 1 2 2 3 4

The index of the sequence x1 is n1 = [-4 -3 -2 -1 0 1]

n1 =

-4 -3 -2 -1 0 1

Enter the sequence x2 = [2 3 4 5]

x2 =

2 3 4 5

The index of the sequence x2 is n2 = [-1 0 1 2]

n2 =

-1 0 1 2

>>

1. Folding operation
2. Even and Odd Component



clc;

clear;

close all;

x=input(' Enter the sequence x = ')

n=input(' The index of the sequence n = ')

n1=-fliplr(n);

x1=fliplr(x);

n2=min(min(n),min(n1)):max(max(n1),max(n));

y1=zeros(1,length(n2));

y2=y1;

y1(find((n2>=min(n))&(n2<=max(n))))=x;

y2(find((n2>=min(n1))&(n2<=max(n1))))=x1;

even=.5\*(y1+y2)

odd=.5\*(y1-y2)

subplot(311);

stem(n,x);

subplot(312);

stem(even);

subplot(313);

stem(odd);

Enter the sequence x = [2 4 6 8 10]

x =

2 4 6 8 10

The index of the sequence n = [1 2 3 4 5]

n =

1 2 3 4 5

even =

5 4 3 2 1 0 1 2 3 4 5

odd =

-5 -4 -3 -2 -1 0 1 2 3 4 5

>>

Assignment 4:

Precedence Rule



clc;

clear;

close all;

x=input(' Enter the sequence x = ')

n=input(' The index of the sequence n = ')

% time shifting

k=input(' The shifting factor k =') % k is a scalar

n1=n+k

% amplitude scaling

a=input(' The scaling factor a =') % a is a scalar

y=a.\*x;

% folding

y1=-fliplr(n1);

m=fliplr(y);

subplot(2,2,1);

stem(n,x);

subplot(2,2,2);

stem(n1,x);

subplot(2,2,3); % or subplot(212);

stem(n1,y);

subplot(2,2,4);

stem(y1,m);

Enter the sequence x = [1 2 3 4 5 6 7 8 9 10]

x =

1 2 3 4 5 6 7 8 9 10

The index of the sequence n = [-5 -4 -3 -2 -1 0 1 2 3 4]

n =

-5 -4 -3 -2 -1 0 1 2 3 4

The shifting factor k =2

k =

2

n1 =

-3 -2 -1 0 1 2 3 4 5 6

The scaling factor a =2

a =

2

>>

------------------------------------------------------------------------------------------------------------------------------------------

>> x1=[4 3 2 -5 3 2 7 8]

x1 =

4 3 2 -5 3 2 7 8

>> n1=-3:4

n1 =

-3 -2 -1 0 1 2 3 4

>> n2=-5:5

n2 =

-5 -4 -3 -2 -1 0 1 2 3 4 5

>> x2=[3 -1 2 4 3 2 1 1 -2 -3 4]

x2 =

3 -1 2 4 3 2 1 1 -2 -3 4

>> x1[3]

x1[3]

↑

Error: Invalid expression. When calling a function or indexing a variable, use parentheses. Otherwise, check for

mismatched delimiters.

>> x1(3)

ans =

2

>> x1(3)=-3 % replacing data

x1 =

4 3 -3 -5 3 2 7 8

>> x1

x1 =

4 3 -3 -5 3 2 7 8

>> x2

x2 =

3 -1 2 4 3 2 1 1 -2 -3 4

>> x=x1+x2

Matrix dimensions must agree.

>> help find

find Find indices of nonzero elements.

I = find(X) returns the linear indices corresponding to

the nonzero entries of the array X. X may be a logical expression.

Use IND2SUB(SIZE(X),I) to calculate multiple subscripts from

the linear indices I.

I = find(X,K) returns at most the first K indices corresponding to

the nonzero entries of the array X. K must be a positive integer,

but can be of any numeric type.

I = find(X,K,'first') is the same as I = find(X,K).

I = find(X,K,'last') returns at most the last K indices corresponding

to the nonzero entries of the array X.

[I,J] = find(X,...) returns the row and column indices instead of

linear indices into X. This syntax is especially useful when working

with sparse matrices. If X is an N-dimensional array where N > 2, then

J is a linear index over the N-1 trailing dimensions of X.

[I,J,V] = find(X,...) also returns a vector V containing the values

that correspond to the row and column indices I and J.

Example:

A = magic(3)

find(A > 5)

finds the linear indices of the 4 entries of the matrix A that are

greater than 5.

[rows,cols,vals] = find(speye(5))

finds the row and column indices and nonzero values of the 5-by-5

sparse identity matrix.

See also sparse, ind2sub, relop, nonzeros.

Reference page for find

Other functions named find

>> find(x1>2)

ans =

1 2 5 7 8

>> % find -- returns index values

>> % index starts from 1

>> min(n1,n2):max(n1,n2) % to get time index

Matrix dimensions must agree.

>> min(m1),min(m2):max(m1),max(m2)

Undefined function or variable 'm1'.

>> min(n1),min(n2):max(n1),max(n2)

ans =

-3

ans =

-5 -4 -3 -2 -1 0 1 2 3 4

ans =

5

>> n=min(n1),min(n2):max(n1),max(n2)

n =

-3

ans =

-5 -4 -3 -2 -1 0 1 2 3 4

ans =

5

>> length=n

length =

-3

>> y1=zeros(1,length)

y1 =

1×0 empty double row vector

>> y1=zeros(1,length(n))

Array indices must be positive integers or logical values.

>> y1=zeros(1,length(n))

Array indices must be positive integers or logical values.

>> y1=zeros(1,len(n))

Undefined function or variable 'len'.

>> y1=zeros(1,length(n))

Array indices must be positive integers or logical values.

>> y2=y1

y2 =

1×0 empty double row vector

>> y1=zeros(1,length(n))

y1 =

0

>> y2=y1

y2 =

0

>> find(n>=(min(n1))

find(n>=(min(n1))

↑

Error: Invalid expression. When calling a function or indexing a variable, use parentheses. Otherwise, check for

mismatched delimiters.

Did you mean:

>> find(n>=(min(n1)))

ans =

1

>>