Table of Contents

..... 1
Convolution, Part I 1
Real-time Convolution 3
Deconvolution 3
Code 3


```
% lab2.m
% Please place lab2.m in your working directory
% Provide the print-out from running this function
% using 'publish lab2'
%
% T. Holton 10 Sept 08

test_lab2;
```

Convolution, Part I

Convolution #1

```
x = sequence([1 2 6 -3 5], 1);
h = sequence([4 -1 5 3 2], -3);
test_lab2(x, h);

% Convolution #2
test_lab2(h, x);

% Convolution #3
h = sequence(1, 0);
test_lab2(x, h);

% Convolution #4
test_lab2(h, x);

% Convolution #5
x = sequence(cos(2 * pi * (1:50000) / 16), -5); % nice, big sequence
h = sequence(ones(1, 10), 10);
test_lab2(x, h);

% Convolution #6
test_lab2(h, x);

% Convolution #7
x = sequence(1, 2);
h = sequence(1, -1);
test_lab2(x, h);

% Convolution #8
test_lab2(h, x);
```

Problem #1

```
   Your data are correct
   Your offset is correct
     Your elapsed time is 36.127 usecs
     which is 4.31 times Holton's elapsed time (8.385 usecs)
     and 11.2 times Matlab's elapsed time (3.212 usecs)

Problem #2
   Your data are correct
   Your offset is correct
     Your elapsed time is 21.189 usecs
     which is 2.71 times Holton's elapsed time (7.806 usecs)
     and 8.53 times Matlab's elapsed time (2.483 usecs)

Problem #3
   Your data are correct
   Your offset is correct
     Your elapsed time is 24.453 usecs
     which is 5.01 times Holton's elapsed time (4.885 usecs)
     and 8.17 times Matlab's elapsed time (2.992 usecs)

Problem #4
   Your data are correct
   Your offset is correct
     Your elapsed time is 15.888 usecs
     which is 1.12 times Holton's elapsed time (14.162 usecs)
     and 6.43 times Matlab's elapsed time (2.469 usecs)

Problem #5
   Your data are correct
   Your offset is correct
     Your elapsed time is 1514 usecs
     which is 0.999 times Holton's elapsed time (1515.74 usecs)
     and 19.7 times Matlab's elapsed time (76.845 usecs)

Problem #6
   Your data are correct
   Your offset is correct
     Your elapsed time is 1573.91 usecs
     which is 1.2 times Holton's elapsed time (1311.3 usecs)
     and 17.5 times Matlab's elapsed time (89.682 usecs)

Problem #7
   Your data are correct
   Your offset is correct
     Your elapsed time is 19.929 usecs
     which is 3.99 times Holton's elapsed time (4.996 usecs)
     and 6.82 times Matlab's elapsed time (2.922 usecs)

Problem #8
   Your data are correct
   Your offset is correct
     Your elapsed time is 12.191 usecs
     which is 3.29 times Holton's elapsed time (3.7 usecs)
     and 6.35 times Matlab's elapsed time (1.92 usecs)
```

Real-time Convolution

Real-time convolution #1

```
x = [1 4 2 6 5];
h = [4 -1 3 -5 2];
test_lab2a;
test_lab2a(x, h);

% Real-time convolution convolution #2
test_lab2a(h, x);

% Real-time convolution #3
x = cos(2 * pi * (1:50000) / 16); % nice, big sequence
h = ones(1, 10);
test_lab2a(x, h);
```

```
Real-time convolution #1
   Your data are correct

Real-time convolution #2
   Your data are correct

Real-time convolution #3
   Your data are correct
```

Deconvolution

Deconvolution #1

```
h = sequence([1 3 2], 2);
y = sequence([1 6 15 20 15 7 2], -1);
test_lab2b;
test_lab2b(y, h);

% Deconvolution #1
y = sequence([-1 -2 0 0 0 0 1 2], 2);
test_lab2b(y, h);
```

```
Deconvolution problem #1
   Your data are correct
   Your offset is correct

Deconvolution problem #2
   Your data are correct
   Your offset is correct
```

Code

```
disp('--------------------------------------------------')
```

```
disp('                    Code')
disp('-------------------------------------------------')
type sequence
type conv_rt

-------------------------------------------------
                     Code
-------------------------------------------------

classdef sequence
    properties
        data
        offset
    end

    methods
        function s = sequence(data, offset)
            % SEQUENCE   Sequence object
            %            S = SEQUENCE(DATA, OFFSET) creates sequence S
            %            using DATA and OFFSET
            %
            %            Kevin Baltazar Reyes  13 Feb 2019
            s.data = data;
            s.offset = offset;
        end

        function display(s)
            var = inputname(1);
            if (isempty(var))
                disp('ans =');
            else
                disp([var '=']);
            end
            switch length(s.data)
                case 0
                    disp('    data: []')
                case 1
                    disp(['    data: ', num2str(s.data)])
                otherwise
                    disp(['    data: [' num2str(s.data) ']'])
            end
            disp(['  offset: ' num2str(s.offset)])
        end

        function y = flip(x)
            % FLIP Flip a Matlab sequence structure, x, so y = x[-n]
            tempData = x.data(end:-1:1);    %start with the end of the
 sequence then count down 1 each time

            tempOffest = -(x.offset+length(x.data) - 1);
            y = sequence(tempData,tempOffest);
        end

        function y = shift(x, n0)
```

```
            % SHIFT Shift a Matlab sequence structure, x, by integer
amount n0 so that y[n] = x[n - n0]

            sameDataX=x.data;  %data sequence remains the same, we are
only shifting the offset
            newOffset=(x.offset+n0);   %new offset = previous offset +
value you are shifting
            y=sequence(sameDataX,newOffset);
        end

        function x = trim(x)
            %takes zeros off from each side of sequence

            while x.data(end)==0
                x.data(end)=[];
            end

            while x.data(1)==0
                x.data(1)=[];
                x.offset=x.offset+1;
            end

        end


        function z = plus(x, y)
            % PLUS  Add x and y. Either x and y will both be sequence
structures, or one of them may be a number.

            if isa(x,'sequence') == 0     %checks if x is a constant
                z=sequence(y.data+x,y.offset);  %if x is a constant,
add constant x to every data pt in y sequence, leave offest untouched
                z=trim(z);
                return;
            end

            if isa(y,'sequence') == 0    %same as above but instead
                z=sequence(x.data+y,x.offset);
                z=trim(z);
                return;
            end

            lx=length(x.data);  %length of data in sequence x
            ly=length(y.data);  %length of data in sequence y

            ody=y.offset-x.offset;  %difference between sequence
offsets IF Y HAS GREATER OFFSET THAN X
            odx=x.offset-y.offset;  %%difference between sequence
offsets IF X HAS GREATER OFFSET THAN Y

            x.data=[zeros(1,odx) x.data zeros(1,ody-(lx-ly))];  %add
zeros to the beginning & end of sequence x as a "filler". You cannot
perform operations between x & y if there is no data at a given
index. The zeros are put in to fill these empty spots.
```

```
            y.data=[zeros(1,ody) y.data zeros(1,odx-(ly-lx))];

            off=min(x.offset,y.offset);     %minimum offset between x
& y
            z=sequence(x.data+y.data,off);  %create sequence z as a
result of adding sequence x & y together

            z=trim(z);

        end

        function z = minus(x, y)
            % MINUS Subtract x and y. Either x and y will both be
sequence structures, or one of them may be a number.

            if isa(x,'sequence')==0
                z=sequence(x-y.data,y.offset);
                z=trim(z);
                return;
            end
            if isa(y,'sequence')==0
                z=sequence(x.data-y,x.offset);
                z=trim(z);
                return;
            end

            Lx=length(x.data);
            Ly=length(y.data);

            ody=y.offset-x.offset;
            odx=x.offset-y.offset;

            x.data=[zeros(1,odx) x.data zeros(1,ody-(Lx-Ly))];
            y.data=[zeros(1,ody) y.data zeros(1,odx-(Ly-Lx))];

            off=min(x.offset,y.offset);
            z=sequence(x.data-y.data,off);

            z=trim(z);

        end

        function z = times(x, y)
            % TIMES Multiply x and y (i.e. .*) Either x and y will
both be sequence structures, or one of them may be a number.
            if isa(x,'sequence')==0
                z=sequence(y.data*x,y.offset);
                return;
            end
            if isa(y,'sequence')==0
                z=sequence(x.data*y,x.offset);
                return;
            end
```

```
            Lx=length(x.data);
            Ly=length(y.data);

            ody=y.offset-x.offset;
            odx=x.offset-y.offset;

            x.data=[zeros(1,odx) x.data zeros(1,ody-(Lx-Ly))];
            y.data=[zeros(1,ody) y.data zeros(1,odx-(Ly-Lx))];

            off=min(x.offset,y.offset);
            z=sequence(x.data.*y.data,off);

            z=trim(z);

        end

        function stem(x)
            % STEM Display a Matlab sequence, x, using a stem plot.
            stem( x.offset : length(x.data )+x.offset-1,x.data);
        end

        function y = conv(x,h)
            %CONV two finite-length Matlab sequence objects, x and h
            %   returning sequence object y

            if (length(x.data) > length(h.data))    %if sequence h is
shorter, convolve sequence h with matrix X (y=h*X)
                X = zeros(length(h.data),
((length(h.data)+length(x.data))-1));  %define X's dimensions and
stuff with zeros:
                i=1; %X row length = sequence h data length
                j=1; %Y column length =
length(h.data)+length(x.data)-1
                P = [x.data zeros(1,length(h.data)-1)]; %sequence x
duplicated in P. length(x.data) + length of zeros stuffed after MUST
EQUAL length of sequence y for flip&shift method
                for n = 1:(length(h.data))  %outer loop for row
increment
                    for m = 1:((length(h.data)+length(x.data))-1)
%inner loop for column increment
                        X(i,j)=P(j);    %value stored in P's current
index goes to corresponding X index
                        j=j+1;          %index points to next column
(next value of data in P)
                    end
                    P = circshift(P,[0,1]); %circular shift-right on
vector P. Value of 0 inputted one time to open data slot (leftmost
slot)
                    i=i+1;  %index points to next row (P[k-n])
                    j=1;    %start from P's first data point (leftmost
column)
                end
                y = sequence((h.data*X),(x.offset+h.offset)); %now
that the matrix dimensions are correct, we can multiply them together
```

```
            else    %convolve x with matrix H (y=x*H)
                H = zeros(length(x.data),
 ((length(h.data)+length(x.data))-1));
                i=1;
                j=1;
                P = [h.data zeros(1,length(x.data)-1)];

                for n=1:(length(x.data))
                    for m = 1:((length(x.data)+length(h.data))-1)
                        H(i,j)=P(j);
                        j=j+1;
                    end
                    P = circshift(P,[0,1]);
                    i=i+1;
                    j=1;
                end
                y=sequence((x.data*H),(x.offset+h.offset));
            end
        end

        function x = deconv(y,h)    % x = y*H^(-1)
            % DECONV Convolve finite-length Matlab sequence object, y,
            %        given impulse response object, h
            %        returning sequence object, x.

            Lx = length(y.data)-length(h.data)+1;   %length of
 sequence x
            h_hat = zeros(Lx,Lx);    %define Lx by Lx matrix
            F = [h.data zeros(1, Lx-length(h.data))];   %sequence h
 duplicated in F. Add zeros after h.data so the length of F == Lx


            %traverse vector F through Lx by Lx matrix h_hat. Shift
 right, cut
            %off data shifted out at the end of the vector
            for i=1:Lx
                h_hat(i,:) = [zeros(1,i-1) F(1:Lx-i+1)];     %shift
 right by 1 each row by adding one zero per row in front, and cutting
 1 data per row at end
            end
            x = sequence(y.data(1:Lx)*inv(h_hat), y.offset-h.offset);
  %only need to sample enough y.data to equal length(x.data).
 (length(y.data) MUST equal row & column length for h_hat
        end

    end
end

function y = conv_rt(x,h)
% Convolve two finite-length arrays, x and h returning array, y
h_hat = h(end:-1:1);
x_hat = [zeros(1,length(h)-1) x zeros(1,length(h)-1)];
y = zeros(1,length(x)+length(h)-1);
```

```
for i=1:length(x)+length(h)-1
    y(i) = sum(h_hat.*x_hat(i:i+length(h)-1));
end
end
```

Published with MATLAB® R2018b