Lab 2: Convolution

ENGR 451 Digital Signal Processing

Kevin Baltazar Reyes

916353599

Questions:

1. **Why do we have to pad these sequences with zeros?**
    a. We pad the sequence with zeros to be able to easily add all the sequences in matlab. Matlab only lets you effectively add sequences of the same length
2. **In general, how does the number of sequences we have to sum the length of these sequences depend on the sizes of x[n] and h[n]?**
    a. When convolving X and H, the length always turns out to be the length of X and length of H minus 1
3. **Why the difference?**
    a. There is a difference because the length is going to be the same no matter how you convolve it but X and H could have different lengths so the one with the shorter length will require more sequences to complete the convolution.
4. **Why the difference?**
    a. There is a difference in matrix size because the lengths of X and H are different. Due to the difference property of matrices, the shorter length will have a small matrix.
5. **In, general how do the number of rows and columns of matrix depend on the lengths of the sequences x[n] and h[n]?**
    a. When multiplying the two vectors with size i and k to make a matrix of the size i x k, we know that since we pad on the matrices that i won't change, what we can change is k. The smaller the vector, the smaller the size which means we will not to calculate as much when creating the new matrix.
6. **When is it more advantageous to compute x[n]*h[n] and when is it better to compute h[n]*x[n]?**
    a. The factor that really dictates what will be the most efficient way to calculate the convolution is the length of the X and H vectors. Whichever is the shorter length is the one you want to convolve with to make all the calculations shorter even though in the end, both will give the same result. Essentially, it is better to convolve the signal with the shortest length

Matlab Published Code

Contents


```
% 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 30.437 usecs
    which is 3.05 times Holton's elapsed time (9.995 usecs)
    and 3.89 times Matlab's elapsed time (7.831 usecs)

Problem #2
  Your data are correct
  Your offset is correct
    Your elapsed time is 21.691 usecs
    which is 3.1 times Holton's elapsed time (7.007 usecs)
    and 8.39 times Matlab's elapsed time (2.584 usecs)

Problem #3
  Your data are correct
  Your offset is correct
    Your elapsed time is 45.585 usecs
    which is 0.527 times Holton's elapsed time (86.475 usecs)
    and 0.459 times Matlab's elapsed time (99.265 usecs)

Problem #4
  Your data are correct
  Your offset is correct
    Your elapsed time is 24.63 usecs
    which is 0.875 times Holton's elapsed time (28.143 usecs)
    and 3.66 times Matlab's elapsed time (6.722 usecs)

Problem #5
  Your data are correct
  Your offset is correct
    Your elapsed time is 1854.75 usecs
    which is 1.4 times Holton's elapsed time (1327.47 usecs)
    and 18.6 times Matlab's elapsed time (99.553 usecs)

Problem #6
  Your data are correct
  Your offset is correct
    Your elapsed time is 2039.65 usecs
    which is 1.46 times Holton's elapsed time (1395.56 usecs)
    and 22.8 times Matlab's elapsed time (89.622 usecs)
```

```
Problem #7
  Your data are correct
  Your offset is correct
   Your elapsed time is 21.07 usecs
   which is 3.28 times Holton's elapsed time (6.425 usecs)
   and 3.79 times Matlab's elapsed time (5.562 usecs)

Problem #8
  Your data are correct
  Your offset is correct
   Your elapsed time is 13.614 usecs
   which is 3.84 times Holton's elapsed time (3.547 usecs)
   and 7.16 times Matlab's elapsed time (1.901 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