Table of Contents

```
% AUTHORS: JUAN ANGELES & MOSES MARTINEZ
% 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
```

Questions: Q: Why do we have to pad these sequences with zeros? A: We pad the sequences to be multiplied correct part of the sequence as h travels through x.

Q: In heneral, how does the number of sequeces we have to sum and the

```
% length of these sequences on the sizes of x[n] and h[n]?
% A: If made efficient, the number of sequences we have to sum is the
 sum
% of the lengths minus 1. The length of the sequences would depend on
% whichever is shorter, x or l.
%
% Q: Why the difference?
% A: If x[n] and h[n] vary in length, the algorithm can be made more
%   efficient if the shorter sequence is chosen as x[n]*h[n], due to
 the
%   less number of sequence additions in the end.
%
% Q: Why the difference?
% A: Depending base on whether we convolve the input with the impulse
 or
%   the impulse with the input we will obtain different rows in the
 Matrix
%   but same columns.
%
% Q: In general, how do the number of rows and columns of the matrix
 depend
%   on the lengths of the sequences x[n] and h[n]?
% A: The number of columns is always the length of x[n] plus the
 length of
%   h[n] minus one. The number of rows depends on which sequence we
 decided
%   to convolve.
%
% Q: When is it more advantageous to compute x[n]*h[n] and when is it
%   better to compute h[n]*x[n]?
% A: The advantage all depends on which sequence has the shorter
 length.
%   This will give us a smaller Matrix to multiply.

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 643.776 usecs
     which is 3.24 times Holton's elapsed time (198.985 usecs)
     and 5.41 times Matlab's elapsed time (118.938 usecs)

Problem #2
   Your data are correct
   Your offset is correct
     Your elapsed time is 231.835 usecs
     which is 2.37 times Holton's elapsed time (97.7936 usecs)
     and 3.01 times Matlab's elapsed time (77.0266 usecs)
```

```
Problem #3
   Your data are correct
   Your offset is correct
     Your elapsed time is 134.419 usecs
     which is 3.52 times Holton's elapsed time (38.1357 usecs)
     and 3.83 times Matlab's elapsed time (35.1151 usecs)

Problem #4
   Your data are correct
   Your offset is correct
     Your elapsed time is 184.637 usecs
     which is 6.27 times Holton's elapsed time (29.4514 usecs)
     and 7.41 times Matlab's elapsed time (24.9204 usecs)

Problem #5
   Your data are correct
   Your offset is correct
     Your elapsed time is 9503.73 usecs
     which is 0.974 times Holton's elapsed time (9758.22 usecs)
     and 23.3 times Matlab's elapsed time (407.788 usecs)

Problem #6
   Your data are correct
   Your offset is correct
     Your elapsed time is 8992.86 usecs
     which is 0.909 times Holton's elapsed time (9891.88 usecs)
     and 33.6 times Matlab's elapsed time (267.705 usecs)

Problem #7
   Your data are correct
   Your offset is correct
     Your elapsed time is 107.233 usecs
     which is 2.76 times Holton's elapsed time (38.8909 usecs)
     and 4.06 times Matlab's elapsed time (26.4307 usecs)

Problem #8
   Your data are correct
   Your offset is correct
     Your elapsed time is 110.254 usecs
     which is 3.48 times Holton's elapsed time (31.7168 usecs)
     and 5.51 times Matlab's elapsed time (20.0118 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
---------------------------------------------------

% Juan Angeles Acuna and Moses Martinez
```

```
classdef sequence
 properties
  data
  offset
 end

 methods
  function s = sequence(data, offset)
   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)
            Lin = length(x.data) + x.offset -1;
            data = fliplr(x.data);
            offset = -Lin;
            y = sequence(data, offset);
        end

  function y = shift(x, n0)
            offset = x.offset + n0;
            y = sequence(x.data, offset);
        end

        function [xdata, ydata] = seqData(x, y)
            Lx = length(x.data);
            Ly = length(y.data);
            ody= y.offset - x.offset;
            odx= x.offset - y.offset;
            xdata = [zeros(1, odx) x.data zeros(1, ody - (Lx - Ly))];
            ydata = [zeros(1, ody) y.data zeros(1, odx - (Ly - Lx))];
        end

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

            while(x.data(end) == 0)
                x.data(end) = [];
            end;
            z = sequence(x.data, x.offset);
        end

  function z = plus(x, y)
            if((isa(x,'sequence')) && (isa(y,'sequence')))
                [x.data, y.data] = seqData(x,y);
                z_data = x.data + y.data;
                z_offset = min(x.offset, y.offset);
            else
                if(isa(x,'sequence'))
                    z_data = x.data + y;
                    z_offset= x.offset;
                elseif(isa(y,'sequence'))
                    z_data = y.data + x;
                    z_offset=y.offset;
                end
            end
            z = sequenceTrimmer(sequence(z_data, z_offset));
        end

  function z = minus(x, y)
            if((isa(x,'sequence')) && (isa(y,'sequence')))
                [x.data, y.data] = seqData(x,y);
                z_data = x.data - y.data;
                z_offset = min(x.offset, y.offset);
            else
                if(isa(x,'sequence'))
                    z_data = x.data - y;
                    z_offset= x.offset;
                elseif(isa(y,'sequence'))
                    z_data = x - y.data;
                    z_offset=y.offset;
                end
            end
            z = sequenceTrimmer(sequence(z_data, z_offset));
        end

  function z = times(x, y)
            if((isa(x,'sequence')) && (isa(y,'sequence')))
                [x.data, y.data] = seqData(x,y);
                z_data = x.data .* y.data;
                z_offset = min(x.offset, y.offset);
            else
                if(isa(x,'sequence'))
                    z_data = x.data * y;
                    z_offset= x.offset;
                elseif(isa(y,'sequence'))
                    z_data = x * y.data;
                    z_offset=y.offset;
                end
            end
            z = sequenceTrimmer(sequence(z_data, z_offset));
        end

  function stem(x)
            Lin = length(x.data) + x.offset -1;
            stem((x.offset:Lin), x.data);
        end

        function y = conv(x, h)
            Lx = length(x.data);
            Lh = length(h.data);

            if(Lx == min(Lx, Lh))
                s = x.data;
                b = h.data;
            else
                s = h.data;
                b = x.data;
            end

            Ls = length(s);
            Lb = length(b);
            numRow = Ls;
            numCol = Lb+Ls-1;
            h_matrix = zeros(numRow, numCol);
            for i = 1:1:numRow
                h_matrix(i,:) = [zeros(1,i-1) b zeros(1, numCol-Lb-i
+1)];
            end

            y.data = s*h_matrix;
            y.offset = x.offset + h.offset;
            y = sequence(y.data, y.offset);
        end



      function x = deconv(y, h)
          Ly = length(y.data);
          Lh = length(h.data);
          Lx = Ly - Lh + 1;

          x_matrix = zeros(Lx,Lx);
          if(Lx < Lh)
              x_matrix = toeplitz([h.data(1:Lx)]);
          else
              x_matrix = toeplitz([h.data zeros(1, Lx-Lh)]);
          end
          y_matrix = 1*Lx;
          y_matrix = y.data(1:Lx);
          for i = 1:1:Lx
              x_matrix(i,1:i-1) = 0 ;
          end
          x_matrix = inv(x_matrix);
          x.data = y_matrix * x_matrix;
          x.offset = y.offset - h.offset;
          x = sequence(x.data,x.offset);
        end
 end
end

function y = conv_rt(x,h)
    Lx = length(x);
    Lh = length(h);
    y = [];
    h_hat = h(end:-1:1);
    x_hat = [zeros(1,Lh-1) x zeros(1, Lh-1)];
    for i = 1:1:Lx+Lh-1
        y = [y sum(h_hat .* x_hat(i:i+Lh-1))];
    end
end
```

Published with MATLAB® R2015a