Table of 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 116.6 usecs
     which is 3.63 times Holton's elapsed time (32.1 usecs)
     and 5.3 times Matlab's elapsed time (22 usecs)

Problem #2
   Your data are correct
   Your offset is correct
     Your elapsed time is 103 usecs
     which is 3.67 times Holton's elapsed time (28.1 usecs)
     and 11.4 times Matlab's elapsed time (9 usecs)

Problem #3
   Your data are correct
   Your offset is correct
     Your elapsed time is 101.6 usecs
     which is 0.338 times Holton's elapsed time (300.6 usecs)
     and 0.276 times Matlab's elapsed time (368.1 usecs)

Problem #4
   Your data are correct
   Your offset is correct
     Your elapsed time is 66.6 usecs
     which is 0.623 times Holton's elapsed time (106.9 usecs)
     and 3.42 times Matlab's elapsed time (19.5 usecs)

Problem #5
   Your data are correct
   Your offset is correct
     Your elapsed time is 16618.8 usecs
     which is 1.39 times Holton's elapsed time (11978.8 usecs)
     and 31.9 times Matlab's elapsed time (520.5 usecs)

Problem #6
   Your data are correct
   Your offset is correct
     Your elapsed time is 15675 usecs
     which is 1.1 times Holton's elapsed time (14234.5 usecs)
     and 9.96 times Matlab's elapsed time (1573.8 usecs)

Problem #7
   Your data are correct
   Your offset is correct
     Your elapsed time is 121.4 usecs
     which is 5.32 times Holton's elapsed time (22.8 usecs)
     and 4.58 times Matlab's elapsed time (26.5 usecs)

Problem #8
   Your data are correct
   Your offset is correct
```

```
Your elapsed time is 62.5 usecs
which is 1.41 times Holton's elapsed time (44.3 usecs)
and 8.33 times Matlab's elapsed time (7.5 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(Static)

        function [a,b] = padData(x,y)
            Lx = length(x.data) + x.offset;
            Ly = length(y.data) + y.offset;
            a = [zeros(1,x.offset-y.offset), x.data, zeros(1,Ly-Lx)];
            b = [zeros(1,y.offset-x.offset), y.data, zeros(1,Lx-Ly)];
        end

%         % My Original Implementation
%         %
%         % Pads the input sequences so that they are of the same
 length.
%         % Sequence with the lower offset will not have front
 padding. This
%         % returns the data portion of the sequences only.
%         function [a,b] = padData(x,y)
%             % Find which sequencce has the lower offset (furthest to
 the
%             % left).
%             lo = sequence([],0);
%             hi = sequence([],0);
%             if(x.offset<y.offset)
%                 lo = x;
%                 hi = y;
%             else
%                 lo = y;
%                 hi = x;
%             end
%             % Define ints for left and right padding of zeros.
%             leftPad = hi.offset-lo.offset;
%             rightPad = (length(lo.data)+lo.offset)-
(length(hi.data)+hi.offset);
%             % Padding the left side of the sequence with the higher
 offset
```

```
%             % is easiest.
%             hi.data = [zeros(1,leftPad),hi.data];
%             % Pad the right side of either the lower or higher
 offset
%             % sequence depending on whether rightPad is
%             % positive or negative.
%             if(rightPad>0)
%                 hi.data = [hi.data, zeros(1,rightPad)];
%             elseif(rightPad<0)
%                 lo.data = [lo.data, zeros(1,abs(rightPad))];
%             end
%             % Map lo and hi back to the order in which they came
 i.e. a = x
%             % and b = y.
%             if(x.offset<y.offset)
%                 a=lo.data;
%                 b=hi.data;
%             else
%                 a=hi.data;
%                 b=lo.data;
%             end
%         end
    end

 methods
  function s = sequence(data, offset)
   % SEQUENCE   Sequence object
   %            S = SEQUENCE(DATA, OFFSET) creates sequence S
   %            using DATA and OFFSET
   %
   %            Your Name  1 Jan 2014
   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)
```

```
          ofs = -(x.offset+length(x.data)-1);
 y = sequence(x.data(end:-1:1),ofs);
end

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

function z = plus(x, y)
          if(isa(x,'double'))
              z = sequence(x+y.data,y.offset);
          elseif(isa(y,'double'))
              z = sequence(x.data+y,x.offset);
          else
              [a, b] = sequence.padData(x,y);
              z = sequence(a+b,min(x.offset,y.offset));
          end
          %trim(z);
      end

function z = minus(x, y)
          if(isa(x,'double'))
              z = sequence(x-y.data,y.offset);
          elseif(isa(y,'double'))
              z = sequence(x.data-y,x.offset);
          else
              [a, b] = sequence.padData(x,y);
              z = sequence(a-b,min(x.offset,y.offset));
          end
          %trim(z);
end

function z = times(x, y)
          if(isa(x,'double'))
              z = sequence(x.*y.data,y.offset);
          elseif(isa(y,'double'))
              z = sequence(x.data.*y,x.offset);
          else
              [a, b] = sequence.padData(x,y);
              z = sequence(a.*b,min(x.offset,y.offset));
          end
          %trim(z);
      end

      function y = conv(x,h)
          lx = length(x.data);
          lh = length(h.data);
          if(lx>lh)
              y = convol(h,x,lh,lx);
          else
              y = convol(x,h,lx,lh);
          end
      end
```

```
        function H = getConvMatrix(h,lx,lh)
            widthH = lx + lh - 1;
            H=zeros( lx, widthH );
            for n = 1:lx
                zerosLeftLength     = n-1;
                dataStart           = 1;
                dataEnd             = min( lh, widthH -
 zerosLeftLength );
                zerosRightLength    = widthH - dataEnd - dataStart -
 zerosLeftLength + 1;
                left                = zeros( 1, zerosLeftLength );
                mid                 = h.data( dataStart : dataEnd );
                right               = zeros( 1, zerosRightLength );
                H(n,:)              = [left, mid, right];
            end
        end

        % Convolution
        function y = convol( x,h,lx,lh )
            H = getConvMatrix(h,lx,lh);
            y = sequence( x.data*H, x.offset+h.offset );
        end

        function x = deconv(y,h)
            ly = length(y.data);
            lh = length(h.data);
            lx = ly-lh+1;
            x_data = zeros(1,lx);
            for n=1:lx
                sub = 0;
                for k=1:(min(n,lh))
                    sub = sub + x_data(n-k+1)*h.data(k);
                end
                x_data(n)=(y.data(n)-sub)/h.data(1);
            end
            x=sequence(x_data,y.offset-h.offset);
        end

        function x = trim(x)
            while(x.data(1) == 0 && length(x.data)>1)
                x.data(1) = [];
            end
            while(x.data(end) == 0 && length(x.data)>1)
                x.data(end) = [];
            end
        end

  function stem(x)
   % STEM Display a Matlab sequence, x, using a stem plot.
            data_length = length(x.data);
            n_axis_indeces = linspace(1,data_length,data_length);
            n_axis_vals = n_axis_indeces
+linspace(x.offset,x.offset,data_length)-1;
```

```
            figure()
            stem(x.data)

            xlabel('n'); title('x');

            set(gca,'XTick', n_axis_indeces );
            set(gca,'XTickLabel', n_axis_vals );

  end
 end
end

% When finished: publish Lab1 'pdf' or 'doc'

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

% Notes
%
%                                                 [ 1 2 1 0 1 2 1 ]
 h[n]
%                                                         <-- [ 0 0 0 0
 0 0 9 8 7 1 2 ]   x[n]
% N=1
```

Published with MATLAB® R2019b