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 21.6 usecs
     which is 1.89 times Holton's elapsed time (11.4 usecs)
     and 3.79 times Matlab's elapsed time (5.7 usecs)

Problem #2
   Your data are correct
   Your offset is correct
     Your elapsed time is 13.6 usecs
     which is 1.81 times Holton's elapsed time (7.5 usecs)
     and 6.48 times Matlab's elapsed time (2.1 usecs)

Problem #3
   Your data are correct
   Your offset is correct
     Your elapsed time is 12.1 usecs
     which is 0.203 times Holton's elapsed time (59.5 usecs)
     and 0.134 times Matlab's elapsed time (90.5 usecs)

Problem #4
   Your data are correct
   Your offset is correct
     Your elapsed time is 20.5 usecs
     which is 0.869 times Holton's elapsed time (23.6 usecs)
     and 4.46 times Matlab's elapsed time (4.6 usecs)

Problem #5
   Your data are correct
   Your offset is correct
     Your elapsed time is 25021 usecs
     which is 15.5 times Holton's elapsed time (1612.8 usecs)
     and 113 times Matlab's elapsed time (220.6 usecs)

Problem #6
   Your data are correct
   Your offset is correct
     Your elapsed time is 21099 usecs
     which is 15 times Holton's elapsed time (1408.9 usecs)
     and 117 times Matlab's elapsed time (180.9 usecs)

Problem #7
   Your data are correct
   Your offset is correct
     Your elapsed time is 9.4 usecs
     which is 1.92 times Holton's elapsed time (4.9 usecs)
     and 2.41 times Matlab's elapsed time (3.9 usecs)

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

```
Your elapsed time is 6.1 usecs
which is 1.56 times Holton's elapsed time (3.9 usecs)
and 2.65 times Matlab's elapsed time (2.3 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
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Code from Lab1%%%%%%%%%%%%%%%%%%%%%%%%%
%%
 properties
  data
  offset
 end
 methods
  function s = sequence(data, offset)
   s.data = data;
   s.offset = offset;
  end
  %next function
  function y = flip(dataflip) % flips data
   array = dataflip.data; % storing the input data
   off_set = dataflip.offset; % storing offset data
   size = length(array); % getting size of array
   i1 = find(array, 1,'first'); % getting the offset with the leading
 zeros removed
   off_set = off_set - i1 + 1; % calculating the offset
   a = 1;
   b = size;
   for i=1:size % for loop to flip data around
    array(b) = dataflip.data(i); % a temporary array to store the
 flipped numbers
    b = size - a;
    a = a + 1;
   end
   out = array(find(array, 1,'first'):find(array, 1,'last')); % gets
 rid of leading and trailing zero
   y = sequence(out, off_set); % output
  end
  %next function
  function y = shift(datashift, offsetshift) % shifts data
   array = datashift.data; % storing input data
   off_set = datashift.offset; % storing offset of data
   value = off_set + offsetshift; % calculating the new offset
   y = sequence(array, value); % output
  end
  %next function
  function y = plus(s1, s2) % add overload
```

```
  if(isnumeric(s1)) % if s1 is numeric
   array2 = s2.data; % data of s2
   offsetout = s2.offset; % offset of s2
   s3 = array2 + s1;
   i1 = find(s3,1,'first'); % getting the offset with the leading
zeros removed
   offsetout = offsetout - i1 + 1; % calculating the offset
   s4 = s3(find(s3,1,'first'):find(s3,1,'last')); % gets rid of
leading and trailing zero
   y = sequence(s4, offsetout);% output
  elseif(isnumeric(s2))% if s2 is numeric
   array1 = s1.data; % data of s1
   offsetout = s1.offset; % offset of s1
   s3 = array1 + s2;
   i1 = find(s3,1,'first'); % getting the offset with the leading
zeros removed
   offsetout = offsetout - i1 + 1; % calculating the offset
   s4 = s3(find(s3,1,'first'):find(s3,1,'last')); % gets rid of
leading and trailing zero
   y = sequence(s4, offsetout);% output
  else % if s1 and s2 are sequences
   array1 = s1.data; % data of first array
   offset1 = s1.offset; % offset of first array
   size1 = length(array1); % size of first array
   array2 = s2.data; % data of second array
   offset2 = s2.offset; % offset of second array
   size2 = length(array2); % size of second array
   if(offset1 == offset2) % same offset
    for i=1:size1
     s3(i) = array1(i) + array2(i);
    end
    offsetout = offset1;
   elseif(offset1 > offset2) % offset of s1 > offset of s2
    a = 1; % index of the output array
    diff = offset1 - offset2;
    for i=1:diff % first numbers of array1
     s3(i) = array1(i);
     a = a + 1;
    end
    for i=1:size2 % arrray1 plus array2
     if(a <= size1) % if they are still overlapping
      s3(a) = array1(a) + array2(i);
     else % if they are no longer overlapping
      s3(a) = array2(i);
     end
     a = a + 1;
    end
    if (size1 > size2 + offset1) % array1 is still goning even though
array2 has run out of values
     for i = 1:size1 - a
      s3(a) = array1(a);
     end
    end
    offsetout = offset1; % output offset
```

```
   elseif(offset2 > offset1) % offset of s2 > offset of s1
    a = 1;
    diff = offset2 - offset1;
    for i=1:diff % first numbers of array2
     s3(i) = array2(i);
     a = a + 1;
    end
    for i=1:size1 % arrray1 plus array2
     if(a <= size2) % if they are still overlapping
      s3(a) = array2(a) + array1(i);
     else % if they are no longer overlapping
      s3(a) = array1(i);
     end
     a = a + 1;
    end
    if (size2 > size1 + offset2)% array2 is still goning even though
array1 has run out of values
     for i = 1:size2 - a
      s3(a) = array2(a);
     end
    end
    offsetout = offset2; % output offset
   end
   i1 = find(s3,1,'first'); % getting the offset with the leading
zeros removed
   offsetout = offsetout - i1 + 1; % calculating the offset
   s4 = s3(find(s3,1,'first'):find(s3,1,'last')); % gets rid of
leading and trailing zero
   y = sequence(s4, offsetout);% output
  end
 end
 %next function
 function y = minus(s1, s2) % minus overload (not done, remove
leading and trailing zeros)
  if(isnumeric(s1)) % if s1 is numeric
   array2 = s2.data; % data of s2
   offsetout = s2.offset; % offset of s2
   s3 = array2 - s1;
   i1 = find(s3,1,'first'); % getting the offset with the leading
zeros removed
   offsetout = offsetout - i1 + 1; % calculating the offset
   s4 = s3(find(s3,1,'first'):find(s3,1,'last')); % gets rid of
leading and trailing zero
   y = sequence(s4, offsetout);% output
  elseif(isnumeric(s2))% if s2 is numeric
   array1 = s1.data; % data of s1
   offsetout = s1.offset; % offset of s1
   s3 = array1 - s2;
   i1 = find(s3,1,'first'); % getting the offset with the leading
zeros removed
   offsetout = offsetout - i1 + 1; % calculating the offset
   s4 = s3(find(s3,1,'first'):find(s3,1,'last')); % gets rid of
leading and trailing zero
   y = sequence(s4, offsetout);% output
```

```
  else % if s1 and s2 are sequences
   array1 = s1.data; % data of first array
   offset1 = s1.offset; % offset of first array
   size1 = length(array1); % size of first array
   array2 = s2.data; % data of second array
   offset2 = s2.offset; % offset of second array
   size2 = length(array2); % size of second array
   if(offset1 == offset2) % same offset
    for i=1:size1
     s3(i) = array1(i) - array2(i);
    end
    offsetout = offset1;
   elseif(offset1 > offset2) % offset of s1 > offset of s2
    a = 1; % index of the output array
    diff = offset1 - offset2;
    for i=1:diff % first numbers of array1
     s3(i) = array1(i);
     a = a + 1;
    end
    for i=1:size2 % arrray1 minus array2
     if(a <= size1) % if they are still overlapping
      s3(a) = array1(a) - array2(i);
     else % if they are no longer overlapping
      s3(a) = array2(i);
     end
     a = a + 1;
    end
    if (size1 > size2 + offset1) % array1 is still goning even though
array2 has run out of values
     for i = 1:size1 - a
      s3(a) = array1(a);
     end
    end
    offsetout = offset1; % output offset
   elseif(offset2 > offset1) % offset of s2 > offset of s1
    a = 1;
    diff = offset2 - offset1;
    for i=1:diff % first numbers of array2
     s3(i) = array2(i);
     a = a + 1;
    end
    for i=1:size1 % arrray1 minus array2
     if(a <= size2) % if they are still overlapping
      s3(a) = array2(a) - array1(i);
     else % if they are no longer overlapping
      s3(a) = array1(i);
     end
     a = a + 1;
    end
    if (size2 > size1 + offset2)% array2 is still goning even though
array1 has run out of values
     for i = 1:size2 - a
      s3(a) = array2(a);
     end
```

```
    end
    offsetout = offset2; % output offset
   end
   i1 = find(s3,1,'first'); % getting the offset with the leading
zeros removed
   offsetout = offsetout - i1 + 1; % calculating the offset
   s4 = s3(find(s3,1,'first'):find(s3,1,'last')); % gets rid of
leading and trailing zero
   y = sequence(s4, offsetout);% output
  end
 end
 %next function
 function y = mtimes(s1, s2) % mtimes overload
  if(isnumeric(s1)) % if s1 is numeric
   array2 = s2.data; % data of s2
   offsetout = s2.offset; % offset of s2
   s3 = array2 * s1;
   i1 = find(s3,1,'first'); % getting the offset with the leading
zeros removed
   offsetout = offsetout - i1 + 1; % calculating the offset
   s4 = s3(find(s3,1,'first'):find(s3,1,'last')); % gets rid of
leading and trailing zero
   y = sequence(s4, offsetout);% output
  elseif(isnumeric(s2))% if s2 is numeric
   array1 = s1.data; % data of s1
   offsetout = s1.offset; % offset of s1
   s3 = array1 * s2;
   i1 = find(s3,1,'first'); % getting the offset with the leading
zeros removed
   offsetout = offsetout - i1 + 1; % calculating the offset
   s4 = s3(find(s3,1,'first'):find(s3,1,'last')); % gets rid of
leading and trailing zero
   y = sequence(s4, offsetout);% output
  else % if s1 and s2 are sequences
   array1 = s1.data; % data of first array
   offset1 = s1.offset; % offset of first array
   size1 = length(array1); % size of first array
   array2 = s2.data; % data of second array
   offset2 = s2.offset; % offset of second array
   size2 = length(array2); % size of second array
   if(offset1 == offset2) % same offset
    for i=1:size1
     s3(i) = array1(i) * array2(i);
    end
    offsetout = offset1;
   elseif(offset1 > offset2) % offset of s1 > offset of s2
    a = 1; % index of the output array
    diff = offset1 - offset2;
    for i=1:diff % first numbers of array1
     s3(i) = array1(i);
     a = a + 1;
    end
    for i=1:size2 % arrray1 times array2
     if(a <= size1) % if they are still overlapping
```

```
      s3(a) = array1(a) * array2(i);
     else % if they are no longer overlapping
      s3(a) = array2(i);
     end
     a = a + 1;
    end
    if (size1 > size2 + offset1) % array1 is still goning even though
array2 has run out of values
     for i = 1:size1 - a
      s3(a) = array1(a);
     end
    end
    offsetout = offset1; % output offset
   elseif(offset2 > offset1) % offset of s2 > offset of s1
    a = 1;
    diff = offset2 - offset1;
    for i=1:diff % first numbers of array2
     s3(i) = array2(i);
     a = a + 1;
    end
    for i=1:size1 % arrray1 times array2
     if(a <= size2) % if they are still overlapping
      s3(a) = array2(a) * array1(i);
     else % if they are no longer overlapping
      s3(a) = array1(i);
     end
     a = a + 1;
    end
    if (size2 > size1 + offset2)% array2 is still goning even though
array1 has run out of values
     for i = 1:size2 - a
      s3(a) = array2(a);
     end
    end
    offsetout = offset2; % output offset
   end
   i1 = find(s3,1,'first'); % getting the offset with the leading
zeros removed
   offsetout = offsetout - i1 + 1; % calculating the offset
   s4 = s3(find(s3,1,'first'):find(s3,1,'last')); % gets rid of
leading and trailing zero
   y = sequence(s4, offsetout);% output
  end
 end
 %next function
 function stem(s1)% overloads stem function
  array = s1.data;
  off_set = s1.offset;
  size = length(array);
  endpoint = size - off_set-1; % calcuating the endpoint of the array
with the offset and the -1 is needed to get the same size array as
the data
  startpoint = 0 - off_set; % calcuating the startpoint with offset
  n = startpoint:endpoint; % making the x-axis array
```

```
  stem(n, array);
 end
 %%%%%%%%%%%%%%%%%%%%%%%%%Convovle Code%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 function y = conv(x,h)
  size_x = length(x.data);
  size_h = length(h.data);
  offset_x = x.offset;
  offset_h = h.offset;
  size_output = size_x + size_h - 1; %size of the output array
  if(size_x < size_h)
   first = x.data; %smaller array
   second = h.data; %larger array
  else
   first = h.data;
   second = x.data;
  end
  output_offset = offset_x + offset_h;
  nextrow = 0;
  for i=1:length(first) % makes the matrix
   for j=1:size_output
    if(j > length(second)) % when the second array is all copied,
then add zeros until you reac the end
     martix(i,j) = 0;
    else
     martix(i,j) = second(j); % copies the values from the second
array
    end
    if(nextrow > 0) % for all rows after the first one
     if(j == 1) % first point of the next row has to have a zero
added to it
      martix(i,j) = 0;
     else
      martix(i,j) = martix(i - 1,j - 1); %copies the previous row
without the last value to make it seem like a shift
     end
    end
   end
   nextrow = nextrow + 1; % makes sure to let the program know that
the first row of the matrix has been copied
  end
  output = first* martix; % multiple martix together
  y = sequence(output, output_offset);
 end
 %next function
 function y = deconv(y,h)
  y_data = y.data;
  h_data = h.data;
  output_offset = y.offset;
  h_offset = h.offset;
  input_offset = output_offset - h_offset;
  input_size = length(y_data) - length(h_data) + 1;
  nextrow = 0;
  for i=1:input_size % makes the matrix
   for j=1:length(y_data)
```

```
    if(j > length(h_data)) % when the second array is all copied,
 then add zeros until you reac the end
     martix(i,j) = 0;
    else
     martix(i,j) = h_data(j); % copies the values from the second
 array
    end
    if(nextrow > 0) % for all rows after the first one
     if(j == 1) % first point of the next row has to have a zero
 added to it
      martix(i,j) = 0;
     else
      martix(i,j) = martix(i - 1,j - 1); %copies the previous row
 without the last value to make it seem like a shift
     end
    end
   end
   nextrow = nextrow + 1; % makes sure to let the program know that
 the first row of the matrix has been copied
  end
  input = y_data/martix;
  y = sequence(input,input_offset);
 end
end
end

function y = conv_rt(x,h)
 size_x = length(x);
 size_h = length(h);
 if(size_x < size_h)
  first = x;
  second = h;
 else
  first = h;
  second = x;
 end
 outputlength = size_x + size_h - 1;
 temp = zeros(1,length(first));
 index = 1;
 shift = length(temp);
 output = length(outputlength);
 while(index < outputlength + 1)
  for i=1:length(temp) %shifts the elements of temp
   if(i == length(temp))
    if(index > length(second)) %if second is out of elements add 0
     shift(1) = 0;
    else
     shift(1) = second(index);
    end
   else
    shift(i+1) = temp(i);
   end
  end
  temp = shift; %putting the shifted array back to the temp array
```

```
  y1 = 0;
  for i=1:length(first)
   value = first(i)*temp(i);
   y1 = y1 + value;
  end
  output(index) = y1;
  index = index + 1;
 end
 y = output;
end
```

Published with MATLAB® R2020a