ENGR 451 - Chapter 2 Laboratory

Matlab tutorial

```
clear
x = sequence([1 2 3 4 5], 1);
y = sequence([5 3 1 -1 3 -2 2 3], -1);

% test plus
test_lab1('plus(x, y)')
test_lab1('plus(y, x)')
test_lab1('plus(1, x)')
test_lab1('plus(x, 1)')

y = sequence([5 3 1 0 3 -2 2 3], -4);
test_lab1('plus(x, y)')
test_lab1('plus(y, x)')

% test minustract
test_lab1('minus(x, y)')
test_lab1('minus(y, x)')
test_lab1('minus(1, x)')
test_lab1('minus(x, 1)')

% test timesiplication
test_lab1('times(x, y)')
test_lab1('times(3, x)')
test_lab1('times(x, 3)')

% test flip
test_lab1('flip(x)')

% test shift
test_lab1('shift(y, 2)')

%combinations
test_lab1('flip(minus(shift(plus(x, 2), 4), y))')
test_lab1('plus(flip(plus(x, y)), shift(y, -5))')
test_lab1('minus(plus(times(shift(flip(x), 4), shift(y, 3)), flip(y)),
 x)')

% test stem
set(clf, 'Position', [200 200 400 200])
stem(flip(2+(x-shift(y, -4).*y-3)))
title('y[n]');

% Program Listings
fprintf('\n\n')
disp('--- sequence.m -------------------------------')
type sequence
```

```
plus(x, y): sequence O.K.
```

```
plus(y, x): sequence O.K.
plus(1, x): sequence O.K.
plus(x, 1): sequence O.K.
plus(x, y): sequence O.K.
plus(y, x): sequence O.K.
minus(x, y): sequence O.K.
minus(y, x): sequence O.K.
minus(1, x): sequence O.K.
minus(x, 1): sequence O.K.
times(x, y): sequence O.K.
times(3, x): sequence O.K.
times(x, 3): sequence O.K.
flip(x): sequence O.K.
shift(y, 2): sequence O.K.
flip(minus(shift(plus(x, 2), 4), y)): sequence O.K.
plus(flip(plus(x, y)), shift(y, -5)): sequence O.K.
minus(plus(times(shift(flip(x), 4), shift(y, 3)), flip(y)), x):
 sequence O.K.


--- sequence.m -------------------------------

classdef sequence
 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
```

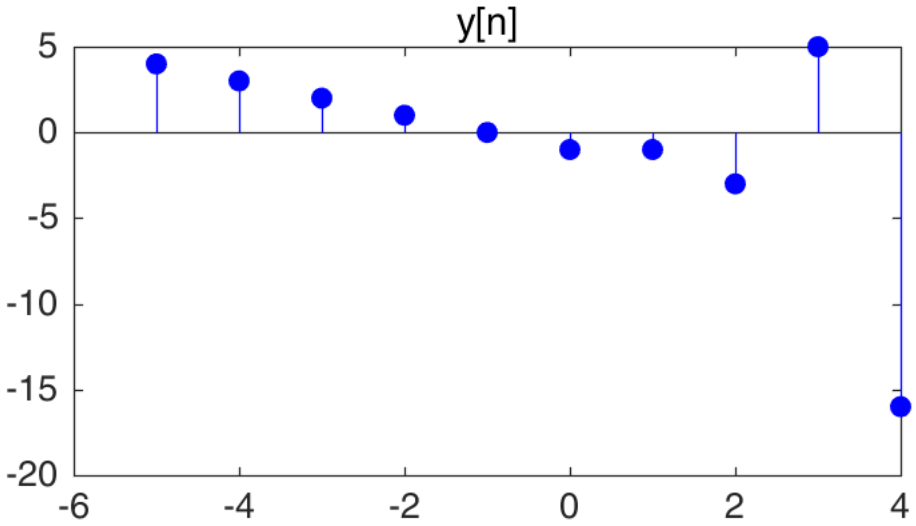
```
    if (size1 > size2 + offset1) % array1 is still goning even though
array2 has run out of values
     for i = 1:size1 - a
      s3(a) = array1(a);
     end
    end
    offsetout = offset1; % output offset
   elseif(offset2 > offset1) % offset of s2 > offset of s1
    a = 1;
    diff = offset2 - 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
  end
end
```



Published with MATLAB® R2016a