Array Index Slicing and Subsetting to Replace and Expand

back to Fan's Intro Math for Econon, Matlab Examples, or MEconTools Repositories

Find the First Negative Element of Array

There is an array with positive, negative and NaN values, find the first negative number's index, and the last positive number index. Use the find function.

```
for it_array=1:1:2
    % Construct Array
    if (it_array == 1)
        ar_alpha_sum_devi_one = [0.1, 0.7, -0.001, NaN, NaN];
    elseif (it_array ==2)
        ar_alpha_sum_devi_one = [0.1, 0.7, 0.001, NaN, NaN];
    end
    % Find last positive
    it_last_pos_idx = find(ar_alpha_sum_devi_one > 0, 1, 'last');
    % Find first negative
    it_first_neg_idx = find(ar_alpha_sum_devi_one < 0, 1, 'first');
    if (isempty(it_first_neg_idx))
        it_first_neg_idx = NaN;
    end
    disp(['it_last_pos_idx=' num2str(it_last_pos_idx) ...
        ', it_first_neg_idx=' num2str(it_first_neg_idx)])
end
```

```
it_last_pos_idx=2, it_first_neg_idx=3
it_last_pos_idx=3, it_first_neg_idx=NaN
```

First the First Negative Element of Last Segment of Array

There is an array with some negative numbers, then some positive numbers, then some negative numbers again. Find the first element of the final segment of negative numbers.

```
for it_array=1:5

    % Construct Array
    if (it_array == 1)
        ar_alpha_sum_devi_one = [0.1, 0.7, -0.001, NaN, NaN];
    elseif (it_array ==2)
        ar_alpha_sum_devi_one = [0.1, 0.7, 0.001, NaN, NaN];
    elseif (it_array ==3)
        ar_alpha_sum_devi_one = [-0.001, 0.1, 0.7, 0.001, NaN, NaN];
    elseif (it_array ==4)
        ar_alpha_sum_devi_one = [-0.001, 0.1, 0.7, 0.001, -0.0002, -0.05, NaN, NaN];
    elseif (it_array ==5)
        ar_alpha_sum_devi_one = [NaN, NaN, -0.001, 0.1, 0.7, 0.001, -0.0002, NaN];
    end

    % Find last positive
    it_last_pos_idx = find(ar_alpha_sum_devi_one > 0, 1, 'last');
    % Find first negative
    it_first_neg_idx = find(ar_alpha_sum_devi_one < 0, 1, 'first');
    % Find first negative of last negative segment
    ar_alpha_sum_after_last_pos = ...
        ar_alpha_sum_devi_one(it_last_pos_idx:length(ar_alpha_sum_devi_one));
    it_first_neg_last_neg_seg_idx = it_last_pos_idx - 1 + ...
        find(ar_alpha_sum_after_last_pos < 0, 1, 'first');

    if (isempty(it_first_neg_idx))
        it_first_neg_idx = NaN;
    end
    if (isempty(it_first_neg_last_neg_seg_idx))
        it_first_neg_last_neg_seg_idx = NaN;
    end

    disp(['it_last_pos_idx=' num2str(it_last_pos_idx) ...
        ', it_first_neg_idx=' num2str(it_first_neg_idx)...
        ', it_first_neg_last_neg_seg_idx=' num2str(it_first_neg_last_neg_seg_idx)])
end
```

```
it_last_pos_idx=2, it_first_neg_idx=3, it_first_neg_last_neg_seg_idx=3
it_last_pos_idx=3, it_first_neg_idx=NaN, it_first_neg_last_neg_seg_idx=NaN
it_last_pos_idx=4, it_first_neg_idx=1, it_first_neg_last_neg_seg_idx=NaN
it_last_pos_idx=4, it_first_neg_idx=1, it_first_neg_last_neg_seg_idx=5
it_last_pos_idx=6, it_first_neg_idx=3, it_first_neg_last_neg_seg_idx=7
```

Index Select Rows and Columns of a 2D matrix

In the example below, select by entire rows and columns:

```
% There is a 2D Matrix
rng(123);
randMatZ = rand(3,6);
disp(randMatZ);
```

```
    0.6965    0.5513    0.9808    0.3921    0.4386    0.7380
    0.2861    0.7195    0.6848    0.3432    0.0597    0.1825
    0.2269    0.4231    0.4809    0.7290    0.3980    0.1755
```

```
% Duplicate Select Row sand Columns of Elements
disp(randMatZ([1,2,3,3,3,2], [1,1,2,2,2,1]))
```

```
    0.6965    0.6965    0.5513    0.5513    0.5513    0.6965
    0.2861    0.2861    0.7195    0.7195    0.7195    0.2861
    0.2269    0.2269    0.4231    0.4231    0.4231    0.2269
    0.2269    0.2269    0.4231    0.4231    0.4231    0.2269
    0.2269    0.2269    0.4231    0.4231    0.4231    0.2269
    0.2861    0.2861    0.7195    0.7195    0.7195    0.2861
```

Index Select Set of Elements from 2D matrix

Rather than selecting entire rows and columns, suppose we want to select only one element at row 1 col 2, the element at row 2 col 4, element at row 5 col 1, etc.

```
% Select Subset of Elements
it_row_idx = [1,2,3,1,3,2];
it_col_idx = [1,1,5,4,2,3];
```

```
% Select sub2idx
ar_lin_idx = sub2ind(size(randMatZ), it_row_idx, it_col_idx);
ar_sel_val = randMatZ(ar_lin_idx);
disp(ar_sel_val');
```

```
    0.6965
    0.2861
    0.3980
    0.3921
    0.4231
    0.6848
```

Find Closest Element of Array to Each Element of Another Array

Given scalar value, find the cloest value in array:

```
fl_a = 3.4;
ar_bb = [1,2,3,4];
[fl_min, it_min_idx] = min(abs(ar_bb-fl_a));
disp(it_min_idx);
```

```
     3
```

Given a scalar value and an array, find the closest smaller value in the array to the scalar value:

```
fl_a = 2.1;
ar_bb = [1,2,3,4];
disp(sum(ar_bb<fl_a));
```

```
     2
```

Array A is between 0 and 1, on some grid. Array B is also between 0 and 1, but scattered. Find for each element of B the index of the cloest value on A that is smaller than the element in B.

```
rng(1234);
ar_a = linspace(0,10,5);
ar_b = rand([5,1])*10;
mt_a_less_b = ar_a<ar_b;
mt_a_less_b_idx = sum(ar_a<ar_b, 2);
disp(ar_a);
```

```
         0    2.5000    5.0000    7.5000   10.0000
```

```
disp(ar_b);
```

```
    1.9152
    6.2211
    4.3773
    7.8536
    7.7998
```

```
disp(mt_a_less_b);
```

```
   1   0   0   0   0
   1   1   1   0   0
   1   1   0   0   0
   1   1   1   1   0
   1   1   1   1   0
```

```
disp(mt_a_less_b_idx);
```

```
     1
     3
     2
     4
     4
```

Matlab Index based Replacement of Subset of Matrix Values

```
rng(123);
randMatZ = rand(3,6)+1;
randMat = rand(3,6)-0.5;

output = max(-randMat,0);
randMatZ(output==0) = 999;
min(randMatZ,[],2);
 randMatZ((max(-randMat,0))==0) = 999;
disp(randMatZ);
```

```
  999.0000  999.0000  999.0000    1.3921    1.4386    1.7380
  999.0000  999.0000    1.6848    1.3432    1.0597    1.1825
  999.0000  999.0000    1.4809  999.0000    1.3980    1.1755
```

```
disp(min(randMatZ,[],2));
```

```
    1.3921
    1.0597
    1.1755
```

Matlab Matrix Index Based Matrix Expansion (Manual)

In the example below, we start with a 4 by 2 matrix, than we expand specific rows and columns of the matrix. Specifically, we expand the matrix such that the result matrix repeats the 1st, 2nd, 1st, 2nd, then 3rd, than 1st, 1st, and 1st rows. And repeats column 1, then 2nd, then 2nd, then 2nd, and finally the first column.

```
% Original Matrix
Z = 2;
N = 2;
Q = 2;
base_mat = reshape(1:(Z*N*Q),Z*N,Q);
disp(base_mat);
```

```
     1     5
     2     6
     3     7
     4     8
```

```
% Expanded Matrix
base_expand = base_mat([1,2,1,2,3,1,1,1],[1,2,2,2,1]);
disp(base_expand);
```

```
     1     5     5     5     1
     2     6     6     6     2
     1     5     5     5     1
     2     6     6     6     2
     3     7     7     7     3
     1     5     5     5     1
     1     5     5     5     1
     1     5     5     5     1
```

Duplicate Matrix Downwards N times Using Index

The example here has the same idea, but we do the operations above in a more automated way. This could be done using alternative methods.

```
% Original Matrix
Z = 2;
N = 2;
Q = 2;
base_mat = reshape(1:(Z*N*Q),Z*N,Q);
disp(base_mat);
```

```
     1     5
     2     6
     3     7
     4     8
```

```
% Generate row Index many times automatically depending on how many times
% to replicate
vmat_repeat_count = 3;
vmat_reindex_rows_repeat = [1:(Z*N)]'* ones(1,vmat_repeat_count);
vmat_reindex_rows_repeat = vmat_reindex_rows_repeat(:);
disp(vmat_reindex_rows_repeat');
```

```
     1     2     3     4     1     2     3     4     1     2     3     4
```

```
% Duplicate Matrix by the Rows specified above, and using the same number
% of columns.
mat_repdown = base_mat(vmat_reindex_rows_repeat(:), 1:Q);
disp(mat_repdown');
```

```
     1     2     3     4     1     2     3     4     1     2     3     4
     5     6     7     8     5     6     7     8     5     6     7     8
```

Given ND Array, Get Row and Column (and other dimension) Index With Value Conditioning

There is a matrix where some values are equal to 1 (based on some prior selection), get the row and column index of the matrix.

```
% Some matrix with 1s
rng(123);
mt_some_ones = rand(3,3);
disp(mt_some_ones);
```

```
    0.6965    0.5513    0.9808
    0.2861    0.7195    0.6848
    0.2269    0.4231    0.4809
```

```
% find the location of the ones
```

```
[r_idx, c_idx] = find(mt_some_ones<0.5);
% the set of locations
disp([r_idx,c_idx]);
```

```
     2     1
     3     1
     3     2
     3     3
```

Now do the same three with a three dimensional array:

```
% Some matrix with 1s
rng(123);
mn3_some_ones = rand(3,3,3);
disp(mn3_some_ones);
```

```
(:,:,1) =

    0.6965    0.5513    0.9808
    0.2861    0.7195    0.6848
    0.2269    0.4231    0.4809


(:,:,2) =

    0.3921    0.4386    0.7380
    0.3432    0.0597    0.1825
    0.7290    0.3980    0.1755


(:,:,3) =

    0.5316    0.8494    0.7224
    0.5318    0.7245    0.3230
    0.6344    0.6110    0.3618
```

```
% find the location of the ones
[d1_idx, d2_idx, d3_idx] = ind2sub(size(mn3_some_ones), find(mn3_some_ones<0.5));
% the set of locations
disp([d1_idx, d2_idx, d3_idx]);
```

```
     2     1     1
     3     1     1
     3     2     1
     3     3     1
     1     1     2
     2     1     2
     1     2     2
     2     2     2
     3     2     2
     2     3     2
     3     3     2
     2     3     3
     3     3     3
```

Max of Matrix column by Column Linear to 2d Index

Finding max of matrix column by column, then obtain the linear index associated with the max values.

```
randMat = rand(5,3);
```

```
disp(randMat);
```

```
    0.2283    0.4309    0.8934
    0.2937    0.4937    0.9442
    0.6310    0.4258    0.5018
    0.0921    0.3123    0.6240
    0.4337    0.4264    0.1156
```

```
[maxVal maxIndex] = max(randMat);
linearIndex = sub2ind(size(randMat),maxIndex,(1:1:size(randMat,2)))
```

```
linearIndex = 1×3
     3     7    12
```

```
randMat(linearIndex)
```

```
ans = 1×3
    0.6310    0.4937    0.9442
```

```
t_pV = [1,2;3,4;5,6];
t_pV_Ind = [1,1;0,0;1,1];
[maxVal maxIndex] = max(t_pV(t_pV_Ind==1))
```

```
maxVal = 6
maxIndex = 4
```

Given Array of size M, Select N somewhat equi-distance elements

```
% Subset count
it_n = 5;

% Example 1, long array
ar_fl_a = 1:1.1:100;
ar_it_subset_idx = unique(round(((0:1:(it_n-1))/(it_n-1))*(length(ar_fl_a)-1)+1));
ar_fl_a_subset = ar_fl_a(ar_it_subset_idx);
disp(ar_fl_a_subset);
```

```
    1.0000   26.3000   50.5000   75.8000  100.0000
```

```

% Example 2, Short Array
ar_fl_a = 1:1.1:3;
ar_it_subset_idx = unique(round(((0:1:(it_n-1))/(it_n-1))*(length(ar_fl_a)-1)+1));
ar_fl_a_subset = ar_fl_a(ar_it_subset_idx);
disp(ar_fl_a_subset);
```

```
    1.0000    2.1000
```

```

% Write As function
f_subset = @(it_subset_n, it_ar_n) unique(round(((0:1:(it_subset_n-1))/(it_subset_n-1))*(it_ar_

% Select 5 out of 10
disp(f_subset(5, 10));
```

```
     1     3     6     8    10
```

```
% Select 10 out of 5
disp(f_subset(10, 5));
```

```
     1     2     3     4     5
```

```


% Select 5 out of 5
disp(f_subset(5, 5));
```

```
     1     2     3     4     5
```