Matlab Miscellaneous and Basic Numeric and Array Operations

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

Divide an Array into Sub-segments

There is a loop, divide N elements into O segments of M elements or less in each segment.

```
it_count_m = 100;
for it_ctr=1:5
    rng(it_ctr);
    it_elements_n = round(rand()*1000);
    ar_seg_ends = unique([1:it_count_m:it_elements_n it_elements_n]);
    disp(ar_seg_ends);
end
```

```
     1   101   201   301   401   417

     1   101   201   301   401   436

     1   101   201   301   401   501   551

     1   101   201   301   401   501   601   701   801   901   967

     1   101   201   222
```

Divisor, Quotient and Remainder

Given an array of integer values, and some divisor, find the quotient and remainder.

```
it_divisor = 10;
for bl_int=[0,1]
    if (bl_int == 1)
        ar_integers = int16([1,2,3, 11,12,13, 21,22,23]);
    else
        ar_integers = [1,2,3, 11,12,13, 21,22,23];
    end
    for it_integer=ar_integers
        % Modulus and quotient
        if(isinteger(it_integer))
            it_quotient = idivide(it_integer, it_divisor);
        else
            it_quotient = fix(it_integer/it_divisor);
        end
        it_remainder = rem(it_integer, it_divisor);

        if (it_remainder == 1)
            fl_power = 1.0;
        elseif (it_remainder == 2)
            fl_power = 1.5;
        elseif (it_remainder == 3)
            fl_power = 2.0;
        end
```

```
        if (it_quotient == 0)
            fl_base = 2;
        elseif (it_quotient == 1)
            fl_base = exp(1);
        elseif (it_quotient == 2)
            fl_base = 10;
        end

        fl_value = fl_base^fl_power;

        % Print
        st_print = strjoin(...
            ["Divide test:", ...
            ['bl_int=' num2str(bl_int)], ...
            ['it_integer=' num2str(it_integer)], ...
            ['it_remainder=' num2str(it_remainder)], ...
            ['it_quotient=' num2str(it_quotient)], ...
            ['fl_value=' num2str(fl_value)], ...
            ], ";");
        disp(st_print);

    end
end
```

```
Divide test:;bl_int=0;it_integer=1;it_remainder=1;it_quotient=0;fl_value=2
Divide test:;bl_int=0;it_integer=2;it_remainder=2;it_quotient=0;fl_value=2.8284
Divide test:;bl_int=0;it_integer=3;it_remainder=3;it_quotient=0;fl_value=4
Divide test:;bl_int=0;it_integer=11;it_remainder=1;it_quotient=1;fl_value=2.7183
Divide test:;bl_int=0;it_integer=12;it_remainder=2;it_quotient=1;fl_value=4.4817
Divide test:;bl_int=0;it_integer=13;it_remainder=3;it_quotient=1;fl_value=7.3891
Divide test:;bl_int=0;it_integer=21;it_remainder=1;it_quotient=2;fl_value=10
Divide test:;bl_int=0;it_integer=22;it_remainder=2;it_quotient=2;fl_value=31.6228
Divide test:;bl_int=0;it_integer=23;it_remainder=3;it_quotient=2;fl_value=100
Divide test:;bl_int=1;it_integer=1;it_remainder=1;it_quotient=0;fl_value=2
Divide test:;bl_int=1;it_integer=2;it_remainder=2;it_quotient=0;fl_value=2.8284
Divide test:;bl_int=1;it_integer=3;it_remainder=3;it_quotient=0;fl_value=4
Divide test:;bl_int=1;it_integer=11;it_remainder=1;it_quotient=1;fl_value=2.7183
Divide test:;bl_int=1;it_integer=12;it_remainder=2;it_quotient=1;fl_value=4.4817
Divide test:;bl_int=1;it_integer=13;it_remainder=3;it_quotient=1;fl_value=7.3891
Divide test:;bl_int=1;it_integer=21;it_remainder=1;it_quotient=2;fl_value=10
Divide test:;bl_int=1;it_integer=22;it_remainder=2;it_quotient=2;fl_value=31.6228
Divide test:;bl_int=1;it_integer=23;it_remainder=3;it_quotient=2;fl_value=100
```

Check if Array is All Above or Below Zero

There is an array that contains possible NaN values, check if all elements of array are positive, or all elements are negative, ignoring the NaN values.

```
for it_arrays=[1,2,3,4,5,6]
    if (it_arrays == 1)
        ar_values = [0.0001, 0.0002, 0.0005, 0.0012, 0.0013, NaN, NaN, NaN, NaN];
    elseif (it_arrays == 2)
        ar_values = [NaN, -0.0002, -0.0005, -0.0012, -0.0013, NaN, NaN, NaN, NaN];
    elseif (it_arrays == 3)
        ar_values = [0.0001, 0.0002, 0.0005, 0.0012, 0.0013];
    elseif (it_arrays == 4)
```

```
        ar_values = [-0.0002, -0.0005, -0.0012, -0.0013];
    elseif (it_arrays == 5)
        ar_values = [-0.0002, 0.0005, -0.0012, -0.0013];
    elseif (it_arrays == 6)
        ar_values = [-0.0002, 0.0005, -0.0012, NaN, -0.0013];
    end
    bl_all_pos = min(ar_values(~isnan(ar_values))>=0);
    bl_all_neg = min(ar_values(~isnan(ar_values))<=0);
    st_print = ['str=' num2str(it_arrays) ...
        ' has bl_all_pos=' num2str(bl_all_pos) ' and bl_all_neg=' num2str(bl_all_neg)];
    disp(st_print);
end
```

```
str=1 has bl_all_pos=1 and bl_all_neg=0
str=2 has bl_all_pos=0 and bl_all_neg=1
str=3 has bl_all_pos=1 and bl_all_neg=0
str=4 has bl_all_pos=0 and bl_all_neg=1
str=5 has bl_all_pos=0 and bl_all_neg=0
str=6 has bl_all_pos=0 and bl_all_neg=0
```

Check Parameter Types

There parameter input can either be a cell array or an integer, conditional processing based on parameter input type. To distinguish between an array or container map, for example, can use isnumeric or isfloat.

```
% Float and Cell
curEstiParamA = 1;
curEstiParamB = {146, 'R3'};
curEstiParamC = rand([1,5]);
curEstiParamD = [1,2,3,4.5];
curEstiParamE = containers.Map('KeyType','char', 'ValueType','any');
param_map('share_unbanked_j') = 12;
param_map('equi_r_j') = 2;
% test if is float
st_test = strjoin(...
    ["", ...
    ['isfloat(curEstiParamA)=' num2str(isfloat(curEstiParamA))], ...
    ['isfloat(curEstiParamB)=' num2str(isfloat(curEstiParamB))], ...
    ['isfloat(curEstiParamC)=' num2str(isfloat(curEstiParamC))], ...
    ['isfloat(curEstiParamD)=' num2str(isfloat(curEstiParamD))], ...
    ['isfloat(curEstiParamE)=' num2str(isfloat(curEstiParamE))], ...
    ], ";");
disp(st_test);
```

```
;isfloat(curEstiParamA)=1;isfloat(curEstiParamB)=0;isfloat(curEstiParamC)=1;isfloat(curEstiParamD)=1;isfloat(curEsti
```

```
% test if is cell
st_test = strjoin(...
    ["", ...
    ['iscell(curEstiParamA)=' num2str(iscell(curEstiParamA))], ...
    ['iscell(curEstiParamB)=' num2str(iscell(curEstiParamB))], ...
    ['iscell(curEstiParamC)=' num2str(iscell(curEstiParamC))], ...
    ['iscell(curEstiParamD)=' num2str(iscell(curEstiParamD))], ...
    ['iscell(curEstiParamE)=' num2str(iscell(curEstiParamE))], ...
    ], ";");
```

```
disp(st_test);
```

```
;iscell(curEstiParamA)=0;iscell(curEstiParamB)=1;iscell(curEstiParamC)=0;iscell(curEstiParamD)=0;iscell(curEstiParam
```

```
% test if is numeric
st_test = strjoin(...
    ["", ...
    ['isnumeric(curEstiParamA)=' num2str(isfloat(curEstiParamA))], ...
    ['isnumeric(curEstiParamB)=' num2str(isfloat(curEstiParamB))], ...
    ['isnumeric(curEstiParamC)=' num2str(isfloat(curEstiParamC))], ...
    ['isnumeric(curEstiParamD)=' num2str(isfloat(curEstiParamD))], ...
    ['isnumeric(curEstiParamE)=' num2str(isfloat(curEstiParamE))], ...
    ], ";");
disp(st_test);
```

```
;isnumeric(curEstiParamA)=1;isnumeric(curEstiParamB)=0;isnumeric(curEstiParamC)=1;isnumeric(curEstiParamD)=1;isnumer
```

Check if a value is an array of single scalar boolean

A function could take an array, if the array parameter input is boolean and false, then generate the array needed by the function in a different way. All that is needed is a NaN checker, works for scalar or array of NaN.

```
rng(123);
it_len = 3;
for it_case=[1,2,3]

    if (it_case == 1)
        ob_val = rand(1,it_len);
    elseif (it_case == 2)
        % Single NaN
        ob_val = NaN;
    elseif (it_case == 3)
        % Single NaN
        ob_val = NaN(1,it_len);
    end

    if (~isnan(ob_val))
        % Input is the output vector since input is not NaN
        ob_val_out = ob_val;
    else
        % Generates random output vector since input is not provided
        ob_val_out = rand(1, it_len);
    end

    st_test = strjoin(...
        ["", ...
        ['ob_val=' num2str(ob_val)], ...
        ['ob_val_out=' num2str(ob_val_out)], ...
        ], ";");
    disp(st_test);
end
```

```
;ob_val=0.69647     0.28614     0.22685;ob_val_out=0.69647     0.28614     0.22685
;ob_val=NaN;ob_val_out=0.55131     0.71947     0.42311
```

```
;ob_val=NaN  NaN  NaN;ob_val_out=0.98076     0.68483     0.48093
```

Compare Array Values That are Approximately Similar

What is the best way to compare floats for almost-equality in Python?

- rel_tol is a relative tolerance, it is multiplied by the greater of the magnitudes of the two arguments; as the values get larger, so does the allowed difference between them while still considering them equal.
- abs_tol is an absolute tolerance that is applied as-is in all cases. If the difference is less than either of those tolerances, the values are considered equal.

```
rel_tol=1e-09;
abs_tol=0.0;
if_is_close = @(a,b) (abs(a-b) <= max(rel_tol * max(abs(a), abs(b)), abs_tol));
disp(['1 and 1, if_is_close:' num2str(if_is_close(1,1))]);
```

```
1 and 1, if_is_close:1
```

```
disp(['1e-300 and 1e-301, if_is_close:' num2str(if_is_close(1e-300,1e-301))]);
```

```
1e-300 and 1e-301, if_is_close:0
```

```
disp(['1+1e-9 and 1+1e-10, if_is_close:' num2str(if_is_close(1+1e-9,1+1e-10))]);
```

```
1+1e-9 and 1+1e-10, if_is_close:1
```

Imaginary Number Examples

```
rng(123);

% Imaginary array
ar_img = rand([1,7]) + 1i*rand([1,7]);

% Regular Array
ar_real = rand([1,10]);

% Combine arrays
ar_full = [ar_real ar_img];
ar_full = ar_full(randperm(length(ar_full)));
disp(ar_full);
```

```
  Columns 1 through 8

   0.6344 + 0.0000i   0.1755 + 0.0000i   0.5316 + 0.0000i   0.2861 + 0.4809i   0.7380 + 0.0000i   0.1825 + 0.0000i

  Columns 9 through 16

   0.7245 + 0.0000i   0.8494 + 0.0000i   0.6110 + 0.0000i   0.4231 + 0.4386i   0.9808 + 0.0597i   0.5318 + 0.0000i

  Column 17

   0.7195 + 0.7290i
```

```
% real index
disp(~imag(ar_full));
```

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

```
% Get Real and not real Components
disp(ar_full(imag(ar_full) == 0));
```

```
    0.6344    0.1755    0.5316    0.7380    0.1825    0.7245    0.8494    0.6110    0.5318    0.3980
```

```
disp(ar_full(imag(ar_full) ~= 0));
```

```
   0.2861 + 0.4809i   0.6965 + 0.6848i   0.2269 + 0.3921i   0.4231 + 0.4386i   0.9808 + 0.0597i   0.5513 + 0.3432i
```