More on Arrays

Contents

1. Recap: Creating Arrays Examples

2. Data Manipulation Functions

3. Timing in MATLAB

4. Exercises

Recap: Creating Arrays Examples

Arithmetic Progressions

Question

Create the following *periodic* arithmetic progressions using ONE MATLAB statement.

$$(1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0).$$

```
m = 5;
n = 15;
mod([1:n], m)
```

Exercise: Arithmetic Progressions

Question

Create each of the following *row* vectors using ONE MATLAB statement.

- $\mathbf{v} = (1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0)$
- $\mathbf{w} = (1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4)$

Geometric and Other Progressions

Question

Create each of the following *column* vectors using ONE MATLAB statement.

- $\mathbf{v} = (1, 2, 4, 8, \ldots, 1024)^{\mathrm{T}}$
- $\mathbf{w} = (1, 4, 9, 16, \ldots, 100)^{\mathrm{T}}$

Using the colon operator:

```
v = ( 2.^[0:10] )’
w = ( [1:10].^2 )’
```

Using the `linspace` function:

```
v = ( 2.^linspace(0, 10, 11) )’
w = ( linspace(1, 10, 10).^2 )’
```

Function Evaluation

Recall that mathematical functions such as `sin, sind, log, exp` accept array inputs and return arrays of function evaluation.

Question

Create each of the the following *row* vectors using ONE MATLAB statement.

- $\mathbf{u} = (1!, 2!, 3!, \ldots, n!)$
- $\mathbf{v} = (\sin 0^\circ, \sin 30^\circ, \sin 60^\circ, \ldots, \sin 180^\circ)$
- $\mathbf{w} = (e^1, e^4, e^9, \ldots, e^{64})$

```
v = sind(0:30:180)
w = exp([1:8].^2)
```

Matrices with Patterns

Question

Generate each of the following matrices using ONE MATLAB statement.

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 1^2 & 1^3 & \cdots & 1^{10} \\ 2 & 2^2 & 2^3 & \cdots & 2^{10} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 10 & 10^2 & 10^3 & \cdots & 10^{10} \end{bmatrix}.$$

```
A = reshape(1:16, 4, 4)'
B = ((1:10)').^(1:10)
```

Matrices with Patterns

Question

Suppose `n` is already stored in MATLAB. Generate each of the following matrices using ONE MATLAB statement. All the elements not shown are 0's.

$$C = \begin{bmatrix} 2 & & & & \\ & 4 & & & \\ & & 6 & & \\ & & & \ddots & \\ & & & & 2n \end{bmatrix}, D = \begin{bmatrix} \cos 1 & -3 & & & & \\ & \cos 2 & -3 & & & \\ & & \cos 3 & -3 & & \\ & & & \ddots & \ddots & \\ & & & & \cos(n-1) & -3 \\ & & & & & \cos n \end{bmatrix}.$$

```
C = diag(2:2:2*n)
D = diag(cos(1:n)) - 3*diag(ones(n-1,1), 1)
```

Data Manipulation Functions

Data Manipulation Functions

There are a number of MATLAB functions with *spreadsheet functionalities* that are suitable for data manipulation.

- `max` and `min`
- `sum` and `prod`
- `cumsum` and `cumprod` (cumulative sum and product)
- `diff`
- `mean`, `std`, and `var` (simple statistics)
- `sort`

Example 1: Finding the Maximum Value of a Vector

Question

Write a program to find the maximum value of a vector.

- **With loops:**

```
% input: x
% output: m    % DON'T USE max FOR THE VARIABLE NAME
m = x(1);      % CODE ABORTS IF THE VECTOR IS EMPTY
for r = 2:length(x)
    if m < x(r)
        m = x(r);
    end
end
```

- **Vectorized code:**

```
m = max(x)
```

Example 1: Finding the Maximum Value of a Vector (cont')

Question

Now modify the previous program to find both the maximum value of a vector and the corresponding index.

- **With loops:**

```
% input: x
% output: m, index_m
m = x(1);
index_m = 1;
for r = 2:length(x)
    if m < x(r)
        m = x(r);
        index_m = r;
    end
end
```

- **Vectorized code:**

```
[m, index_m] = max(x)
```

Example 2: Summing Elements in a Vector

Question

Sum all elements in a vector.

- **With loops:**

```
% input: x
% output: s     % DON'T USE sum FOR THE VARIABLE NAME
s = 0;          % s begins before the first iteration
for el = 1:length(x)
    s = s + x(el);
end
```

- **Vectorized code:**

```
s = sum(x)
```

FIND Function

Basic Usage of FIND

Let `v` be an array of numbers (can be a vector or a matrix). Then

```
find(<condition>)
```

returns the (linear) indices of `v` satisfying `<condition>`.

Some examples of `<condition>`:

```
v > k, v >= k, v < k, v <= k, v == k, v ~= k
```

To combine more than two conditions: Use `&` (*and*) or `|` (*or*)

Example 3: Comparing Elements in Vectors

Question

Compare two real vectors of the same length, say x and y, elementwise and determine how many elements of the former are larger than the latter.

- **With loops:**

```
% input: x, y
% output: nr_gt
nr_gt = 0;
for k = 1:length(x)
    if x(k) > y(k)
        nr_gt = nr_gt + 1;
    end
end
```

- **Vectorized code:**

```
nr_gt = length(find(x > y))
```

Timing in MATLAB

CPU Time

`cputime` reads total CPU time used by MATLAB from the time it was started.

- Single measurement:

```
ct = cputime;    % total cputime as of now
    <statements>
t = cputime - ct;
```

- Average CPU time:

```
ct = cputime;    % total cputime as of now
for i = 1:nr_reps
    <statements>
end
t_avg = (cputime - ct)/nr_reps;
```

Elapsed Time

At the execution of `tic`, MATLAB records the internal time (in seconds); at `toc` command, MATLAB displays the elapsed time.

- Single measurement:

```
tic     % starts a stopwatch timer
    <statements>
toc     % reads the elapsed time from tic
```

- Average elapse time:

```
tic    % starts a stopwatch timer
for i = 1:nr_reps
    <statements>
end
t_avg = toc/nr_reps;
```

What Do You Think It Does?

Below is a modified version of an example code from MATLAB's Help documentation for `tic`. What do you think it's doing?

```
REPS = 1000; minTime = Inf; nSum = 10;
tic;
for i  = 1:REPS
    tStart = tic;
    s = 0;
    for j = 1:nsum
        s = s + besselj(j,REPS);
    end
    tElpased = toc(tStart);
    minTime = min(tElapsed, minTime);
end
t_avg = toc/REPS;
```

Example 4: Timing Elementwise Operations

Question

Generate a $10^7 \times 1$ random vector and measure the internal time and CPU time when computing elementwise squares.

```
n = 1e7;
x = rand(n, 1);
t = cputime;
x1 = x.^2;
time1 = cputime - t;

tic
x2 = x.^2;
time2 = toc();
disp([time1, time2])
```

Exercises

Pythagorean Triples

Question

Given $n \in \mathbb{N}$, find all triples $(a, b, c)\mathbb{N}^3$, with $a, b \leq n$, satisfying

$$a^2 + b^2 = c^2.$$

Notation.

- $\mathbb{N}$: the set of all natural numbers, $1, 2, 3, \ldots$.
- $\mathbb{N}[1, n] = \{1, 2, \ldots, n\}$.
- $\mathbb{N}^3 = \{(a, b, c) \mid a, b, c \in \mathbb{N}\}$.

Pythagorean Triples – Solution Using Loops

```
% input: n
% output: M
iM = 0;
M = [];
for a = 1:n
    for b = 1:n
        c = sqrt(a^2 + b^2);
        if mod(c, 1) == 0
            iM = iM + 1;
            M(iM, :) = [a, b, c];
        end
    end
end
```

Pythagorean Triples – Solution Without Loops

```
% input: x
% output: M
A = repmat([1:n], n, 1);
B = repmat([1:n]’, 1, n);
C = sqrt(A.^2 + B.^2);
M = [A(:), B(:), C(:)];
lM = ( mod(M(:, 3), 1) ~= 0 );
M(lM, :) = [];
```

Birthday Problem

Question

In a group of n randomly chosen people, what is the probability that everyone has a different birthday?

1. Find this probability by hand.
2. Let $n = 30$. Write a script that generates a group of n people randomly and determines if there are any matches.
3. Modify the script above to run a number of simulations and numerically calculate the sought-after probability. Try 1000, 10000, and 100000 simulations. Compare the result with the analytical calculation done in 1.

Birthday Problem (Hints)

- For simplicity, ignore leap years.
- Create a random (column) vector whose elements represent birthdays of individuals (denoted by integers between 1 and 365).
- Line up the birthdays in order and take the difference of successive pairs. What does the resulting vector tell you?
- For 3, to run simulation multiple times, consider creating a random matrix whose rows represent birthdays of individuals and the columns correspond to different simulations.