CS368 MATLAB Programming

Lecture 5

Young Wu

Based on lecture slides by Michael O'Neill and Beck Hasti

February 23, 2022

Guess Two-Thirds of the Average Game

Quiz

- Enter an integer between 0 and 100 (including 0 and 100) that is the closest to $\frac{2}{3}$ of the average of everyone's integer.

Comment on Vectorization

Admin

- Please try to avoid using *for* loops and *if* conditionals in the first half of the course.
- The main difference between MATLAB and other programming languages is its very efficient matrix operation implementation.

Boolean Variables

Math

- A Boolean variable, also called *logical* variable type in MATLAB, is a variable with two possible values *true* and *false* .
- A Boolean variable is stored as either 1 for *true* or 0 for *false* .

Indicator Functions

Math

- Indicator functions, also called dummy variables, are functions that return 1 if a condition is satisfied and 0 if the condition is not satisfied.

1. $x == y$ is the indicator of $x = y$, meaning $\begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}.$
2. $x \;\tilde{}{=}\; y$ is the indicator of $x \neq y$, meaning $\begin{cases} 1 & \text{if } x \neq y \\ 0 & \text{if } x = y \end{cases},$

   $x \,!{=}\, y$ does not work in MATLAB.
3. $x > y$, $x >= y$ are indicators of $x > y$ and $x \geqslant y$.
4. $x < y$, $x <= y$ are indicators of $x < y$ and $x \leqslant y$.

Other Logical Functions

Code

- `~` is not: `~0` is 1 and `~1` is 0.
- `&` is and: `0 & 0` is 0, `0 & 1` is 0, `1 & 0` is 0, `1 & 1` is 1.
- `|` is or: `0 | 0` is 0, `0 | 1` is 1, `1 | 0` is 1, `1 | 1` is 1.
- `==`, `~=`, `<`, `<=`, `>`, `>=`, `~`, `&`, `|` can be applied element-wise to a vector directly.

Short Circuit Evaluation

Code

- $\&\&$ is and, but only works on scalars.
- $||$ is or, but only works on scalars.
- $\&\&$ and $||$ use short-circuit evaluation, for example, when evaluating $a \ \&\& \ b$, if a is false, then b will not be evaluated, and when evaluating $a \ || \ b$, if a is true, then b will not be evaluated.

Vector Reduction Logical Functions

Code

- *any(x)* returns whether any of the elements in the matrix or vector *x* is non-zero.
- *all (x)* returns whether all of the elements in the matrix or vector *x* is non-zero.
- *find (x)* finds the index of all the non-zero elements in the vector *x*.
- *find (x, 1)* finds the index of the first non-zero element in the vector *x*.

Other Reduction Functions

Code

- *sum(x)* and *prod(x)* compute the sum and product of the elements in a matrix or vector x.
- *sum(x, 1)* and *prod(x, 1)* compute the column sums and products of the elements in a matrix x, for example, *sum([1 2; 3 4], 1)* returns the column sums $\begin{bmatrix} 4 & 6 \end{bmatrix}$.
- *sum(x, 2)* and *prod(x, 2)* compute the row sums and products of the elements in a matrix x, for example, *sum([1 2; 3 4], 2)* returns the row sums $\begin{bmatrix} 3 \\ 7 \end{bmatrix}$.
- *mean(x)* computes the average of the numbers in a matrix or vector x.
- *max(x)* and *min(x)* compute the maximum and minimum of the elements in a matrix or vector x.

Indicator, Quiz Grade

Quiz

- (Compute the number of questions a student gets incorrect if the student's answers are B, C, D and the correct answers are B, D, D?)
- **2**

1. $a = ['B',\ 'C',\ 'D'];\ s = ['B',\ 'D',\ 'D'];$

- C : $sum(a == s)$
- D : $sum(a\ \tilde{}= s)$
- E : $sum(a\ != s)$ (this is not MATLAB)

Indicator, Grade Point Average

Quiz

- (Compute the GPA if C is worth 1 point and N is worth 0 point for a student whose grades are C, C, N.)
- **0.5**

1. $g = ['C', \ 'C', \ 'N', \ 'N'];$

- C : $(1 * (g == 'C') + 0 * (g == 'N')) \ / \ length(g)$
- D : $(1 * sum(g == 'C') + 0 * sum(g == 'N')) \ / \ length(g)$

Indicator, Letter Grade

Quiz

- (Compute letter grade if A corresponds to a grade $\geqslant 90$, B for a grade $\geqslant 80$, C for a grade $\geqslant 70$, and D otherwise.)
- **'C'**

1. $g = 75; c = [101\ 90\ 80\ 70\ 0];\ s = ['A'\ 'B'\ 'C'\ 'D'];$

- C : $s(sum(g >= c) + 1)$
- D : $s(sum(g < c))$

Indicator, Letter Grades

Quiz

- (Compute letter grades if A corresponds to a grade $\geqslant 90$, B for a grade $\geqslant 80$, C for a grade $\geqslant 70$, and D otherwise.)
- **'ACD'**

1. $$g = [95\ 75\ 65];\ c = [101\ 90\ 80\ 70\ 0];\ s = ['A'\ 'B'\ 'C'\ 'D'];$$

- C : $s(sum(repmat(g', 1, 5) < repmat(c, 3, 1)) + 1)$
- D : $s(sum(repmat(g', 1, 5) < repmat(c, 3, 1), 2))$

Functions

Math

- A function $y = f(x)$ is a mapping from a list of inputs x, also called arguments or parameters, to a list of outputs y.
- The previous lectures covered many built-in functions in MATLAB, for example, *log* has 1 input and 1 output, $+$ has 2 inputs and 1 output, and *size* has 1 input and 2 outputs.
- New functions can be defined in *.m* files and used in commands.

Function Definition

Code

- A function with name f should be put in a file named $f.m$.
- The first line of the file is $function\ \ y = f(x)$ or $function\ \ [y1,\ y2,\ \ ...]\ \ = f(x1,\ x2,\ \ ...)$, where y is the name or names of the variables to return, and x is the list of arguments of the function.
- The second line of the file is usually comments describing what the function does. Comments start with $\%$ the line after $\%$ is not executed by the program.
- The last line of the file should be end, but it can be omitted.

Helper Functions

Code

- Multiple functions can be defined in the same file $f.m$, but only f can be used outside the file in commands.
- The functions in $f.m$ that is not f are helper functions.

Function Example, Addition

Code

- The addition function $x + y$ is usually written in infix notation (argument 1, then function name, then argument 2).
- The following function is the addition function in prefix notation (function name, then argument 1, then argument 2).

```
function z = add(x, y)
  z = x + y;
end
```

- *add(1, 2)* returns **3** .

Function Example, Linear Combination

Code

- The linear combination of x and y with coefficients u and v is $ux + vy$.
- Sometimes, u, v are not specified, so the default value $u = v = 1$ is used.

```
function z = lincom(x, y, u, v)
  arguments
    x; y; u = 1; v = 1;
  end
  z = u * x + v * y;
end
```

- *arguments* block is also used for input validation. More detail in a later lecture.

Function Example, Linear Combination Too

Code

```
function z = lincom(x, y, u, v)
   arguments
      x; y; u = 1; v = 1;
   end
   z = u * x + v * y;
end
```

- *lincom(1, 2, 3, 4)* returns **11** .
- *lincom(1, 2, 3)* returns **5** .
- *lincom(1, 2)* returns **3** .

Function Example, Max and Min

Code

- Multiple values can be returned from a function, *[y1, y2, ..., yn] = f(x)* stores the value of i-th output in y_i for $i = 1, 2, ..., n$ and *f(x)* only returns first output.

```
function [mx, mn] = mxn(x)
   mx = max(x);
   mn = min(x);
end
```

- *mxn([1, 2, 3])* returns **3**
- *[a b] = mxn([1, 2, 3])* sets *a* to **3** and *b* to **1** .

Functions, Vector Output

Quiz

1. *function v = f1(x)*
2. *v = [x, x + 1];*
3. *end*

1. *sum(f1(2))*

- *B* : 2
- *C* : 3
- *D* : 5

Functions, Multiple Outputs

Quiz

```
function [u, v] = f2(x)
  u = x; v = x + 1;
end
```

```
sum(f2(2))
```

- $B : 2$
- $C : 3$
- $D : 5$

Blank Slide