Topic 8: Functions I

Built-in Functions

Dr. Wei Liu

Credits: Dr. Peter Kovesi, Dr. Eun-Jung Holden, Dr. Cara MacNish

Functions

• We have already used a number of functions without really examining what's going on.
• In mathematics, the term function means is some operation which is defined over a domain of values such that for any input there is a unique result.
• In computer science, the term function has a similar meaning, though we tend to follow a rather general interpretation.
• A function is simply a piece of code that receives some data, using the data to return one or more new values to the calling program.
• Calling a function - “call up” a specified piece of code to do a particular operation.
• The data that is passed to a function is known as the function's arguments (or operands or parameters).

Built-in Functions

• Matlab functions can take zero, one, or more arguments, and can return a scalar or a matrix.
• For example,
  >> s = sin(x)
  >> value = input('Enter a number ');
  >> m = zeros(n, m);
  >> [rows, cols] = size(m);

• Standard Math functions
• Special matrix construction functions
• Data analysis functions

Examples

• For example:
  >> s = sin(a);
  – The sin function takes a single argument, an angle in radians, and returns to the calling program a single value, the sine of the angle.
• Another (non-mathematical) function we have used is the input function:
  >> value = input('Enter a number ');
  – The input function takes a single argument, a string of characters, prints it on the screen and waits for the user to type something at the keyboard.
  – Whatever the user types at the keyboard is treated as a Matlab expression. If the input is valid, the expression is evaluated and the result is assigned to the variable value.
Examples

• Some functions return an array. For example:

\[
\text{>> } [\text{rows, cols}] = \text{size(m)};
\]

  – The size function takes a single argument, a matrix, and returns two values: the number of rows and columns in the matrix.

• Functions can take more than one argument. For example:

\[
\text{>> } m = \text{zeros(n, m)};
\]

  – The zeros function takes two arguments, \(n\) and \(m\), and returns a \(nxm\) matrix of zeroes.

• Some functions do not require any arguments. For example:

\[
\text{>> } r = \text{rand};
\]

  – The rand function, can accept zero (or more) arguments and returns a pseudo-random number.

More Maths functions:

• abs(a) - Returns the magnitude of \(a\).
• mod(a, b) - Returns the remainder of dividing \(a\) by \(b\).
• round(a) - Returns the nearest integer closest to \(a\).
• fix(a) - Returns the nearest integer to \(a\) that is closest to zero.
• ceil(a) - Returns the nearest integer to \(a\) that is closest to positive infinity.
• floor(a) - Returns the nearest integer to \(a\) that is closest to negative infinity.
• sqrt(v) - Returns the square root of \(v\).
• exp(v) - Returns \(e^v\).
• log(v) - Returns the natural logarithm of \(v\).
• log2(v) - Returns the logarithm to base 2 of \(v\).
• log10(v) - Returns the logarithm to base 10 of \(v\).

  For a full list of the "elementary" functions, type 'help elfun' at the Matlab command prompt.
Special matrix construction functions

- Matlab provides a number of matrix construction functions.
- Some examples include:
  - `zeros(n)` - Returns a $n \times n$ matrix of zeroes.
  - `ones(n)` - Returns a $n \times n$ matrix of ones.
  - `eye(n)` - Returns a $n \times n$ identity matrix.

```matlab
>> eye(3)
ans =
    1     0     0
    0     1     0
    0     0     1
```

Matrix construction functions

- Each of these matrix construction functions may accept one or two arguments.
- If the function receives two arguments, the arguments specify the number of rows and columns of the matrix separately. For example,
  ```matlab
  >> ones(n, m) % Generates a n x m matrix of ones.
  ```
- You can also have a function call, or expression, as an argument to a function. For example, constructing a matrix of zeroes the same size as an existing matrix:
  ```matlab
  >> zeros(size(arr));
  ```
  - Firstly, the function call `size(arr)` is evaluated. This returns the number of rows and columns in `arr`. The calculated size is then passed to the `zeros` function.

Matrix construction functions

- A common use of Matrix construction functions is to pre-allocate memory for matrices prior to a computation. For example,
  ```matlab
  [nrows, ncols] = size(m); % Find the size of m.
  mcopy = zeros(nrows, ncols); % Create a matrix to put the answer in.
  for r = 1 : nrows
      for c = 1 : ncols
          mcopy(r, c) = 2*m(r, c);
      end
  end
  ```
  - What happens if you don’t include the first two lines? How will it affect performance for very large matrices?

Why pre-allocating?

- Each time an array is extended, Matlab has to:
  - Create a new array
  - Copy the contents of the old array to the new longer array
  - Add the new values to the array, and then
  - Delete the old array
- This process is very time-consuming for long arrays.
- A good programming practice is to pre-allocate an array to its maximum size.
Data analysis functions

- Matlab provides a number of data analysis functions.
- Some examples include:
  - `max(a)` - Returns the maximum value in `a`.
  - `min(a)` - Returns the minimum value in `a`.
  - `mean(a)` - Returns the mean value in `a`.
  - `median(a)` - Returns the median value in `a`.
  - `std(a)` - Returns the standard deviation of the elements in `a`.
  - `sort(a)` - Returns the elements in `a` sorted in ascending order.
  - `sum(a)` - Returns the sum of the elements in `a`.
  - `hist(a)` - Returns a histogram of the frequency of the elements in `a`.

General rules on Data Analysis Functions

- All of these functions follow the following rules:
  - If the argument `a` is a 1D vector, the result will be a single number.
  - If the argument `a` is a 2D matrix, the specified operation is applied separately to each column of the matrix. The results are returned as an array of values.

Examples

- For example:
  ```matlab
  >> a = [1 2 3
         4 5 6
         7 8 9];
  >> min(a)
  ans =
        1  2  3
  >> median(a)
  ans =
        4  5  6
  >> sum(a)
  ans =
        12  15  18
  ```

Examples – one step further

- To find the minimum value within a matrix we need to apply the `min` function twice:
  ```matlab
  >> min(min(a))    % The second call to min finds
  ans = 1            % the minimum value in the array
  % obtained from the first call
  % to min.
  ```

- Similarly, to find the sum of all the values in a matrix we need to apply the `sum` function twice:
  ```matlab
  >> sum(sum(a))
  ans = 45
  ```

- Familiarity with these functions will save you from having to write many `for` loops.
- For more information, type `help datafun` at the Matlab command prompt.
Historical note

• Matlab has a general convention of working down columns of matrices rather than across rows.

• This is a result of its Fortran heritage and the large Fortran numerical function libraries that Matlab was built on.

• Fortran is short for FORMula TRANslator.

• The first Fortran compiler was designed and written in 1954-57 by an IBM team led by John W. Backus.

• Fortran was the first successful high level language to be developed and was rapidly adopted by the engineering and scientific community.

• Over the years the language has been revised a number of times; its current version is Fortran 90, or more recently Fortran 95, though Fortran 77 is still widely used.

• Fortran remains the language of choice for high performance numerical computations on supercomputers and the like.

• Fortran stores matrix data in memory down the columns rather than across the rows. Hence the Matlab ethos of working down columns instead of across rows.