Lecture CH2-2.2.3

HW: Finish CH2-2.2.3

Lab next week is MS-Excel HW

**Lecture – Matlab**

In scope – variables defined and accessible

Scalars, vectors, and matrices

Scalar – single number

Vector – linear group of numbers arranged vert or horiz

Row = [10 120 30 40 50]

Column = Down

Matrix – rectangular or square group of numbers

Matlab (actually MATLAB) means MATrix LABratory. It’s a language that specializes in matrices.

Everything in Matlab is a matrix.

A scalar is a 1x1 matrix.

A vector is a 1xN or Nx1 matrix.

Variables

Normally you store numbers in variables and use the variables.

Variables save on typing

Variables store the state of our program.

Your programs will then manipulate the state, and make decisions based on the state.

Matlab demo

Semicolon suppresses output

And separates rows in a matrix

Definition – Communicate information clearly and effectively through graphical means.

Line chart

Bar graph

Pie chart

Charts in Excel…

Matlab

linspace = linear spacing of numbers.

>> theta = linspace(0, 6.4, 33);

This is the domain.

>> y=sin(theta);

>> plot(theta,y)

Same as:

>> plot(theta,sin(theta))

Variable

A box in which you can store 1 thing.

Ex.

>> A=10

Vector

Grouping of variables

It’s like a row of boxes.

Each box can contain one scalar.

The boxes are numbered, or indexed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 93 | 95 | 97 | 90 | 85 |

Evaluate at the command line.

>> \_

Assignment Statement

>> Temps = [93 95 97 90 85];

Take the value of the thing on the right and store it in the variable on the left.

>> mo=93 ; tu=95 ; we=97 ; th=90 ; fr=85;

>> Temps = [mo tu we th fr];

Temps =

93 95 97 90 85

What happens when we type this next?

>> th = 91;

Temps still equals the th=90 value

* + - * An assignment statement uses the value of the right hand side.
      * Compute a new value for the RHS by plugging in all the values, doing all the arithmetic, or whatever you find on the RHS.
      * THEN store the result in the variable.
      * Ex.

Original RHS: [mo tu we th fr]

New RHS: [93 95 97 90 85]

* + - * + Did not store the variable, instead stored the numbers corresponding to the variable at the time of the assignment.

Before or after matrix = error

>> Temps = [93 95 97 91 85];

>> Temps(0)  
Gives Error

>> Temps(7)  
Gives Error

Assign a value to a vector location.

What kind of statement would I use?

Assignment statement….

>> Temps(4) = 91

Temps =

93 95 97 91 85

Square root

>> sqrt(100)

ans = 10

Function name is sqrt

We give it an argument, 100

Argument appears between parentheses

>> x = 9;

>> sqrt(x)

ans =

3

>> y = sqrt(x);

>> y

y =

3

Cascade function

Cascade function calls returns a value

That value is for you to use in any way.

>> sqrt(sqrt(10))

ans =

1.7783

Use the return value of one function call as the argument to another function call.

>> sqrt(10)

ans =

3.1623

>> sqrt(ans)

ans =

1.7783

Matrix/vector argument

The sqrt function is vectorized: it can take a vector as an argument.

>> sqrt([4 9 16])

ans =

2 3 4

Notice that you must supply the square brackets in the arguments, there are none in the return.

Rem function

>> rem(10, 3)

Two arguments separated by commas

ans =

1

Size function

>> d = [1 2 3 ; 4 5 6];

>> size(d)

ans =

2 3

Multiple return values

>> [r c] = size(d);

>> r

r =

2

>> c

r =

3

Vector:

[1 2 3 4 5]

True and false

Logical or Boolean values:

1 and 0 = true and false

>> 5 < 10

ans =

1

>> 5 == 10

ans =

0

Vectorized operators

>> [0 5 10 15 20] < 10

ans =

1 1 0 0 0

The find function

>> find ([1 0 1 0])

ans =

1 3

>> find ([1 1 0 0 0])

ans =

1 2

>> find ([0 5 10 15 20] < 10)

ans =

1 2

>> find(10:15 ~= 11)

find([10 11 12 13 14 15] ~= 11)

find([1 0 1 1 1 1])

ans =

1 3 4 5 6

What’s a function?

A function is a named block of code.

It takes arguments as input.

It gives you a return value as output.

Computer scientists also call functions abstractions.

Many functions return a value:

>> A = [10 20 30];

>> a = length(A)

a =

3

Return values

Some functions don’t return anything:

>> disp(‘Hello’)

Hello

Displays character string… not variables.

>> a=disp(‘Hello’)

Arguments

These are the values that you supply when you call the function.

They are plugged into the unknowns inside the function.

Example: **y=mx+b**

**y** is the return value

**m, x,** and **b** are the unknowns

You supply the values (arguments) for the unknowns.

The unknowns are known as *parameters* in computer science.

>> a = length(A) *A is the argument*

>> disp(‘Hello’) *‘Hello’ is the argument*

But:

Programming language designers can’t predict all your programming needs.

Languages include only the most useful functions.

Languages provide a way to create your own functions.

Defining a function

Format:

**Function** *returnVar = name* *(parameters)*

*body*

**end**

The function can be any legal identifier.

Just like naming a variable

Parameters are optional. They are variables separated by commas.

returnVar is optional. It is a variable

end is option, but encouraged.

Example:

*function result = doubler( x )*

*Result = x \* 2;*

*fprintf(‘doubler is returning %g\n’, result)*

*end*

result = return value

the parameter **x** and the return variable **result** exist only inside this function

Output Input

result = doubler( x )

Edit and save

You can’t create new functions at the command line.

Edit and then save that function in a file called **doubler**

***Look for this part of notes online!!!***

Two phases to a function’s life

1. When you define the function
2. When you call (invoke/run/execute) the function

Example

>> x = doubler( 10 )

We say that we apply a function to an argument.

We apply doubler to the number 10

Constant Functions

Matlab does not have the notion of the constant value like other languages do.

Instead a function is used.

The function simply returns a value.

Functions for constants

function value = pi

value = 3.14159;

end

Priorities:

1. Variable
2. Local Function
3. Built in Function

A parameter is an unknown value in a function.

It is a variable.

An argument is a value supplied to the parameter.

When a function is called, a copy of the argument is placed in the parameter variable.

Call by value

Arguments are *copied* into the parameters.

This improves the program safety.

Reduces efficiency.

Book -> Friend -> changes book

Make copy first and the original is unchanged…

function mystery(x)

x(1) = 0;

end

Rand

Number between 0 and 1, does not include 0 or 1.

>>rand

ans =

>>rand(1,4)

1x4 vector with random numbers

Summary of Random Ranges

0 to 1 rand

0 to n rand \* n

m to n rand \* (n-m) + m

dayN = input(‘Enter a numner: ‘);

if dayN == 1

disp(‘Sunday’)

end

if dayN == 2

disp(‘Monday’)

end

for variable = vector

body

end

*for* each variable in the *vector*, assign it to the *variable* and do the *body*

while condition

body

end

V = [2 4 6 8];

For x = 1:length(V)

disp(x)

end

find([0 0 0 0 0 0 0 0])

ans = Empty matrix 1 by 0

size(ans)

ans = 1 0

zeroes(1,4)

ans =

0 0 0 0

ones(2, 8)

ans =

1 1 1 1

Midterm – What gets output for size, >, < and other funtions

The alphabet character string numbers

* Parallel vectors form a table

|  |  |
| --- | --- |
| X | Y |
| 100 | 35 |
| 47 | -80 |
| -12 | 15 |

* X = [100 47 -12];
* Y = [35 -80 15];
* Dx = [0.0 1.1 -3.6];
* Dy = [-2.4 -3.8 0.7];
* Mass = [77.3 120.9 61.4];
* Matricies
* A = [4 5 5; 3 3 3; 1 2 4]
* A =

4 5 5

3 3 3

1 2 4

* A(1,1)
* ans =

4

* A(3,1)
* ans =

1

* A(1,1:3)

ans =

4 5 5

* A(:,2)
* ans =

5

3

2

* 0-D = Scalar
* 1-D = Vector
* 2-D = Matrix
* Vector + Vector
* [1 2 3] + [1 2 3] = [2 4 6]
* [1 2 3] + [1 2 3 4] = Err
* Vector + Matrix
* [1 2 3; 1 2 3] + [1 2 3] = Err

Vector + Vector

1:5 + 6:10 = 1

[1:5] + [6:10] = [7 9 11 13 15]

* Volume of a flat cylinder

depth = a

radius = z

* + area = pi\*z^2
  + volume = area\*s
* ITE C80 1PM-3PM
* 2 Note Sheets
* Built in functions:
  + sqrt rem min max length size floor ceil
* Ranges (colon operator)
* Row vectors, column vectors, 2-D matricies
* Ranges (colon operator)
* Index a vector using numbers, logicals
* Setting an element in a vector
* Functions: zeros ones
* Plotting – relation to vector
* User-defined functions
* Create, define, call, global, multiple return values
* User controlled I/O
* Know how input works, disp, fprintf (%g, %d, %c, %s)
* Relational operators <<= > >= == ~= ~
* And, or
* Find
* If , else, elseif
* Switch
* For
* While
* Break
* Know multidimensional matricies/arrays
* Cell arrays: a(n) a{n}
* Structures, structure arrays
* Fopen
* Fprintf
* Fgetl
* Fclose