Numerical Methods Practical

Basic Concepts

and

Practical Number: 2

To find the absolute value of an integer

By

- Dr K. Priyanka
- Ms Deepti

Basic Concepts

Data Types

- Mathematica is not a type of language in strict sense. i.e. before defining a variable we don’t have to specify the type of variable.

For Example:

x = 0.2

0.2

x = 5

5

x + “ COVID- 19”

5+COVID- 19

- Notice that we are not specifying anywhere that x is a real variable or integer variable or string.
- While applying operators Mathematica doesn’t check the type of variable.

- Mathematica achieve the concept of variable type through the concept of **Head.**

For Example:

x = 2.3

Head [x]

Real

x = 1;

Head [x]

Integer

z = 3 + 4i;

Head [z]

Complex

x = { 1, 2, 3, 4, 5 };

Head[x]

List

x = 22/7

Head[x]

Rational

x = “Winner”

Head[x]

String

- So, each variable has “head” associated with it, which can be accessed through the function Head [ ] and value of “head” can be treated as variable type.

Constants and Variables

- Constants and variables are the building blocks of a programming language.
- Constant means it does not change its value. For example:

```
5=2
Set::setraw: Cannot assign to raw object 5.>>
```

- Variables are those symbols or combination of symbols that can change their value.

- It is usually a good idea to use variable names that begin with a lower case letter.
- There are different ways of defining variables.
- First form is **Direct Assignment,** which is done by ":=" symbol and it creates a global definition. For example:

Clear[x];

y:= 3+5x

y

3+5x

- To see all definations associated with a symbol, we enter “?” followed by the symbols name.

```
?y
Global`y
y:=3+5x
```

- If we change the value of x to 3, and check the value of y, it will be updated according to x.

```
x=3;
Print[“y=  ”,y];
y=18
?y
Global`y
y:=3+5x
```

- Second form of assignment is assignment with **Evaluation**, which is done with the “=” symbol.

z=x+y

21

?z

z=21

x=2;

Print[“z = ”,z];

z=21

Print[“y= ”, y];

y=13

- Change in value of x or y did not result in corresponding change in z but there is a change in the value of y.

Defining variable by **Rule Table**

- Syntax

{a -> 2, b -> 3, c - > 4}

{a->2, b->3, c->4}

- This way of defining in a rule table is not used until it is applied with the expression having **Replace All** "/." operator. For example:

a + b + c

a+b+c

a + b + c /. {a -> 2, b -> 3, c - > 4}

9

Relational Operators

- Operators used to check relation between the variables and expressions

x == y : equal, checks whether x and y are equal.

x != y : unequal, checks whether x and y are unequal.

x > y : checks whether x is greater than y

x >= y or x ≥ y : checks whether x is greater than equal to y

x < y : checks whether x is less than y

x <= y or x ≤ y : checks whether x is less than equal to y

x ==y==z : check whether x, y and z are all equal

x!=y!=z : checks wkhether x,y and z are all unequal

- A relational expression is either true or false. Mathematica provides logical constants: True and False

```
6>6
False
(2+3)==5
True
```

Logical Operators

- Logical And (&&) evaluates its arguments in order and return False immediately if any of the argument is False and return True if all arguments are True.

5 > 4 && 10> 5

True

5 > 4 && 10 < 5

False

- Or (II) evaluates its arguments in order and return True immediately if any of the argument is True and return False if all arguments are False.

5>4 II 10>12

True

10>12 II 4>5

False

- !expr: returns True if expr is False and returns False if expr is True.

x = 5;

!(x>4)

False

!(x < 4)

True

Control Statements and loops

- If Condition

Syntax: If [Condition, t, f] gives/evaluate t if the condition evaluates to True and f if the condition evaluates to False.

If [Condition, t] gives/evaluates t if the condition evaluates to True and will not do anything if the condition evaluates to False

Practical Number: 2
To find the absolute value of an integer

```
absFun [ x_ ] = If [ x≥ 0, x, -x];
Print [“Abs( - 4 ) = ”, absFun[ - 4]];
Print [“Abs( 5 ) = ”, absFun[ 5]];
Abs (- 4) = 4
Abs (5) =5
```