

CGN 3421 - Computer Methods

Class web site:

www.ce.ufl.edu/~kgurl

Class text books: Recommended as a reference

'Numerical Methods for Engineers', Chapra and Canale
Fourth Edition, McGraw-Hill

Class software: Required purchase from Tech Hub

Mathcad 2000 (Circa labs and 200 Weil lab)

Mathcad 2001 (current version available for purchase, differences from 2000 minimal for our purposes)

Class Outline

Learn fundamentals of programming skills in general and Mathcad in particular ~ 5 or 6 weeks
Use Mathcad as a medium to investigate a variety of numerical methods of problem solving. The context will generally be civil engineering-type problems.

A more detailed list of topics covered can be found in the syllabus, available at the class web site:
[www.ce.ufl.edu/~kgurl -->classes/syllabi](http://www.ce.ufl.edu/~kgurl-->classes/syllabi)

The Programming Process

- 1) Read problem statement
- 2) Identify given input and variable input
- 3) Identify required output
- 4) Map a path from starting point to ending point
- 5) Write a pseudo-code in words to describe the process
- 6) Solve a simple example problem by hand or calculator or spreadsheet
- 7) Write the Mathcad worksheet (program)
- 8) Test the code with the example problem from step 6, verify the result
- 9) Be sure to test each of the desired features of the program. Never assume the answer out is correct

Example: Write a program to calculate the mass of a hollow sphere. The outer diameter is 10.2, the density is 1.25, the thickness is variable (the program user decides)

Apply the Programming Process:

- 2) given input: diameter, density variable input: thickness
- 3) output: mass
- 4) Map a path:
- 5) Write a pseudo-code

gather all necessary input (diameter, density, thickness)
 calculate outer and inner radius
 calculate volume of the solid material
 multiply volume by density to get mass
 display result

- 6) Solve a simple example problem by hand or calculator or spreadsheet
- 7) Write the Mathcad worksheet (program)

Please enter a value for the thickness of the skin of the sphere

thickness := 2

These variables will remain constant

Density := 1.25 Diameter := 10.2

Calculate outer and inner radius

$$\text{Router} := \frac{\text{Diameter}}{2}$$

Rinner := Router - thickness

$$\text{Volume} := \frac{4}{3} \cdot \pi \cdot (\text{Router}^3 - \text{Rinner}^3)$$

Volume = 430.859

Now test the code

I'll use a calculator to run an example and compare with my results

NOTE: Mathcad knows the value of π , unless it has been redefined as a new variable

Equal Signs

There are 5 types of equalities that can be used in Mathcad, each has a specific purpose and meaning. For now we need to discuss two types

$:=$ is used to assign a variable to a number or equation. As in $C := 10$. This sign is created by typing ‘:=’

$=$ is used to display the contents of a variable that has already been assigned. It’s used to display output $C := A + B$ calculates and assigns a value to C, $C =$ displays the result

The other 3 types are for assignments inside a function, comparison equal signs, and global equal signs. Each looks different. We will discuss them all as we progress.

Order Matters

Mathcad evaluates the user entered equations and variable assignments from left to right, top to bottom. This is significant in terms of what order you can enter values for variables that are used in other expressions. I can’t say $C := A + B$ until I’ve already defined (assigned values to) A and B. So $A := 6$ and $B := 5$ must appear above and to the left of $C := A + B$.

Here are some examples of the concept. Note that a variable name is made red by Mathcad when it does not have a value for that variable, and it is being used to define another variable (it appears on the right hand side (r.h.s.) of the $:=$ sign

doesn't work	doesn't work	doesn't work	works
$C := A + B$	$C := A + B$	$A := 6$	$A := 6$
$B := 5 + A$	$A := 6$	$B := 5 + A$	$B := 5 + A$
$A := 6$	$B := 5 + A$	$C := A + B$	$C := A + B$
			$C = 17$

Units

Note that the sphere mass example did not include any units for length, mass, volume, etc. Traditional programming languages don’t keep track of units. That is, its the programmer / user’s responsibility to use consistent units, and to know the units of the resultant calculations. A recent mission to Mars was ruined because someone didn’t use the right units, and didn’t think to check... One of the very nice features of Mathcad is that it allows you to assign units for your input variables. These propagate through the calculations and the resultant is automatically displayed in consistent units. Mathcad will also convert units. You can give inputs in metric units, and ask for the result in S.I. Most common units are already stored in Mathcad (try Insert -> Unit to see the options), but you can always define your own.

Let's go to the Mathcad worksheet to play around with units...

Please enter a value for the thickness of the skin of the sphere

thickness := 2m

These variables will remain constant

Density := $1.25 \frac{\text{kg}}{\text{m}^3}$ Diameter := 10.2m

Calculate outer and inner radius

Router := $\frac{\text{Diameter}}{2}$ Rinner := Router - thickness

Volume := $\frac{4}{3} \cdot \pi \cdot (\text{Router}^3 - \text{Rinner}^3)$ Volume = 430.859m³

There are some tricks to using units properly. We'll discuss when the proper context comes up.

Programming Ingredients

The following list breaks down programming into 9 basic topics. We'll discuss each in detail and learn them all in the next few weeks. Once we understand the basics, even the most complicated programming tasks are just the proper combination of basic concepts.

- 1) Data Types
- 2) Variables / Arrays
- 3) Input / Output
- 4) Assignment Statements
- 5) Sequential Execution
- 6) Branching
- 7) Loops
- 8) Subprograms / functions
- 9) Built in functions

Operators

Order of Operations

1) Data Types

Types of information that can be stored and manipulated

- integers - whole numbers

`x := 5`

- real - decimal numbers

`x := 5.563`

- characters - non-numerical data (text)

`message := "GO IRISH"`

- complex - real and imaginary parts

`x = 5 + 4i`

- logical - binary on/off switch

`x = 1 or 0`

2) Variables / Arrays

Named locations for storing data types. We can represent quantities with whatever names we like.

CONSTANT

- numbers and words entered explicitly

VARIABLE

- Individual location to store a single piece of information
- Information can be any data type
- Stored information can be changed
- No limit on the number of variables you can create (practically speaking)

example: x is a variable assigned the constant 5, then 15

`x := 5`

`x := 15`

what is the value of x ?

What kinds of symbols can I use in a variable name?

Variable and function names can consist of:

- Upper and lowercase [letters](#)
- [Numbers](#) 0 through 9
- The [infinity](#) symbol
- The prime symbol ` (**keystroke Ctrl+F7**)
- [Greek](#) letters (**letter, then Ctrl+g**)
- Underscores (_)
- Percent symbols (%)
- Subscripts (the keystroke '[')

Variable Restrictions

- A name cannot start with one of the [numbers](#) 0 through 9, an underscore (_), a prime symbol ` , or a percent (%).
- The [infinity](#) symbol can only appear as the first character in a name.
- All characters in a name must be in the same font, have the same pointsize, and be in the same style.
- Mathcad does not distinguish between variable names and function names. Thus, if you define $f(x)$, and later you define the variable f , you will find that you cannot use $f(x)$ anywhere below the definition for f .
- Certain names are already used by Mathcad for built-in [constants](#), [units](#), and [functions](#). Although you can redefine these names, keep in mind that their built-in meanings will no longer exist after the definition. For example, if you define a variable called **mean**, the built-in function **mean(v)** can no longer be used.
- Mathcad distinguishes between uppercase and lowercase letters. For example, *diam* is a different variable from *DIAM*. A name cannot start with one of the numbers 0 through 9, an underscore (_), a prime symbol ` , or a percent (%).
- Mathcad distinguishes between names in different fonts. For example, AREA is a different variable than *AREA*.

ARRAYS

- More than one storage space under a single name
- Referenced by a number

e.g. Using 3 individual variables to store student grades

```
grade1 ::= 23
grade2 := 25
grade3 := 18
```

Using a single array (grade) to store student grades

```
grade1 := 23
grade2 := 25
grade3 := 18
```

```
grade =  $\begin{bmatrix} 0 \\ 23 \\ 25 \\ 18 \end{bmatrix}$ 
```

we'll see later why there are four values although we only entered 3 values

Array types

scalar - single piece of information (no index)

```
x := 8
```

vector (1-D array)- column of information (one index)

x := Ctrl+M pick # rows and 1 column, enter values

```
x :=  $\begin{bmatrix} 4 \\ 16 \\ 2 \\ 3 \end{bmatrix}$       1 column, 3 rows
```

```
x2 = 16
```

2-D array - multiple rows and columns

x := Ctrl+M pick # rows and columns, enter values

```
x :=  $\begin{bmatrix} 2 & 4.5 & 9.8 \\ 10.3 & 35.6 & 46 \end{bmatrix}$       2 rows, three columns
```

```
x1,3 = 9.8
```

3) Input / Output

Input - methods of including data needed to begin or continue

- enter as constants
weight := 185
- load from a separate saved file (excell, ascii text, and other forms can be handled in Mathcad)
use the insert => component command
see help -> import data
- output from variables
- visual display (plots and graphs) - export to M.S.Word, etc.

4) Assignment statements

Assign information to a variable using :=

- constants
x := 14.5
- expressions
x := 14.5 - 6 + 18.0/3.0
- equations
y := 5
x := 2.4
z := (y - 1)/2 + y^3 - sin(x)
- load or read an outside data file

5) Sequential Execution

Multiple assignment statements performed from top to bottom left to right order

see examples earlier in this lecture

6) Branching - Decision making

Set conditions for executing certain groups of statements or assigning values

e.g. assign a value judgement based on the mass value

```

mass := 10

text := | "too light"  if mass < 50
        | "okay"      otherwise

text = "too light"

```

7) Loops - Repeating statements

Loops allow us to re-use the same commands over and over without actually re-typing them. There are counted loops and conditional loops.

counted loops - repeat commands a set number of times

use a counted loop when we know before we get to the loop how many times we want to run the loop

e.g. write a function that calculates the factorial value for a given integer

```

fact(x) := | s ← 1
           | x ← ceil(x)
           | for i ∈ 1..x
           |   s ← s·i
           | s

fact(2.5) = 6

```

ANOTHER EQUAL SIGN

In the example above we see a third version of the equal sign. The arrow ← is a LOCAL equal sign that only applies within the function. It is not recognized outside the function. All the stuff to the right of the vertical line is within the function. More on this later.

Conditional loops - repeat if condition is true

Use a conditional loop when the number of times we want to run it is dependent on satisfying certain conditions that change as the loop iterates.

```
while (condition is true)
  instructions to be looped
```

e.g. while the value of mass is greater than 100, continually multiply mass by 0.9, and automatically stop when the value of mass falls below 100

```
mass := 200
out(x) := | while x > 100
          | x ← x·0.9
          | x
mass := out(mass)
mass = 95.659
```

8) Subprograms / functions

User-created smaller programs used by the main program when needed. Users can create any set of instructions that they wish to apply often, and assign that set of instructions a name. These functions do not need to appear in the same Mathcad worksheet. They can be in a different worksheet and called from the current sheet. In this manner, we can create whole suites of complicated functions, and not have to actually see them in the worksheet in which we use them.

- breaks large problems into several smaller ones
- Eliminates re-writing statements over and over
- Can be accessed by any future main program
- User can create his / her own functions
- Mathcad has hundreds of built in functions

9) Built in Functions

built-in (comes with Mathcad) smaller programs used by main program when needed same attributes as in 8) above

e.g. enter in several student grades and calculate the mean

$$x := \begin{pmatrix} 3 \\ 8 \\ 9 \end{pmatrix}$$

$$\text{mean}(x) = 6.667$$

mean is a built in function that calculates an average value of the data sent to it.

Operators

- + Addition
- Subtraction
- * Multiplication
- / Division
- ^ Power
- () Specify evaluation order

Order of Operations

() ^

* /

+ -