CGN 3421 - Computer Methods in Civil Engineering

Class Notes #1

What is Mathcad? According to the manual.....
“Mathcad is the industry standard calculation software for technical professionals, educators, and college students. Mathcad is as versatile and powerful as programming languages, yet it’s as easy to learn as a spreadsheet. Plus, it is fully wired to take advantage of the Internet and other applications you use every day”

Let’s talk about a very powerful math software, Mathcad that started a revolution in computational analysis. In the late 1980’s, a company called Wolfram created a computer program called Mathematica. This was the first computer code that could solve algebraic and calculus equations symbolically. That is, if I had an equation that said x*y = z, Mathematica could tell me that y = z / x, without ever needing me to assign numbers to x, y, or z. It also was able to solve integrals, differential equations, and derivatives symbolically. This was an incredible advance, and opened the doors to a whole new world of programming, numerical methods, pure mathematics, engineering, and science.

Since then, a competing code called Maple was developed and sold itself to other software companies to include in their programs.

The end result: Mathcad uses Maple as a solving engine in the background (you don’t see it) to solve problems symbolically.

What version of Mathcad we are going to use?

Old Versions of Mathcad
Mathcad releases new versions of their product at a fairly quick pace. As a result, you will probably come across more than one version, including: Mathcad 8, Mathcad 2000, Mathcad 2001i, Mathcad 11, 12, and 13 (latest). From 2000 or Mathcad 12, you can use Save As to save your worksheet in an older format if desired. This is necessary, for example, if you have Mathcad 11 for your personal machine. Newer versions of Mathcad CAN open worksheets made on older versions (Mathcad 12 can open a Mathcad 11 worksheet), but older versions CAN’T open worksheets from newer versions (Mathcad 11 can not open a Mathcad 12 worksheet). So if you create a worksheet at home (Mathcad 13), and try to open it at the Circa lab (Mathcad 12), you won’t be successful. However, you can avoid this using the Save As older versions feature. That is, you can create a program in Mathcad 13 environment but save it in Mathcad 12 format and thus you can open it in Mathcad 12.
**Homework Format and Template:**

We’ll each develop a simple template to use to start any new homework. The idea of a template is to layout the basic structure of what goes into a homework assignment, minus the actual input, calculations, and output necessary for a specific assignment.

**To create template:**
Simply start a new worksheet and type in what you want on the template. Then use Save As and pick the Mathcad Template option.

**To use template:**

From Mathcad, open the template through the File --> Open command

Change the Files of Type box at the bottom to Mathcad Template.

Before adding any information to the template, save it as a standard worksheet of desired name by using File --> Save As.

Note that the Save as Type box at the bottom automatically is set to standard Mathcad worksheet format.

Choose a name and save. Now the original template is preserved for the next use, and you can begin work on your new assignment.

The basic structure and ingredients for every homework is given on the next page. We’ll base our template on this.
The Homework Template Layout

Computer Methods in Civil Engineering

CGN 3421
Date, 2005
HW #
Your name

Problem #: 

Problem Statement: 

Pseudocode: 

Observations: 

________________________________________________________________________

Page Break
________________________________________________________________________

ORIGIN= 1

INPUT SECTION

CALCULATIONS SECTION

OUTPUT SECTION
Title Page:

**Info:** Name, Course, Assignment Number and Title.

**Summary/Observations:** A brief summary (description) of the assignment. What did the assignment ask for? What did you learn? Did you discover anything easy/difficult with the assignment? What would be one important concept either programming-wise or theory-wise in this homework assignment? It doesn’t have to be long, just clear and concise.

**Pseudocode:** Pseudocode is a detailed description of the steps needed to solve the problem from beginning to end.

Work Page(s):

**Standard Code:** Any Mathcad statements needed in every worksheet. (e.g. ORIGIN=1)

**Input Section:** Scalars, vectors, matrices needed to begin calculations. Includes both permanent assignments and user prompted information as well as data file loading.

**Calculation Section:** Manipulations of the input information

**Output Section:** This should include all your results (vectors, graphs, etc.). Clearly label what each piece of output is.

**Hand Verification:** A sample problem by hand or other means independent of Mathcad. You need not verify every calculation, just samples to show your Mathcad results work.
The Programming Process

Input Phase
1) Read problem statement carefully (at least twice)
   (If you take a wrong step at the first, then it will be very difficult to correct later)
   2) Identify given input (which is variable (user-defined) input or constant input, etc.)
   3) Identify required output

Design Phase
4) Map a path from starting point to ending point (e.g., a flow chart)
   5) Write a pseudo-code in words to describe the process
6) Solve a simple example problem by hand calculation or spreadsheet
   (If you decided to skip this step, you would have to suffer many hours or sleepless
   nights to find a logical bug, small enough to make it extremely difficult to find yet
   very tedious – period)

Programming Phase
7) Write the Mathcad worksheet (programming starts)

Validation Phase
8) Test your Mathcad program with the example problem from step 6 and verify the
   result
9) Be sure to test each of the desired features of the program. Never assume the
   answer out is correct

The Basic Features of Mathcad
(Before we learn a good programming procedure, first we are introduced to only
two of the basic features of Mathcad)

1. 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.
2. 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

<table>
<thead>
<tr>
<th>Wrong</th>
<th>Wrong</th>
<th>Wrong</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>C := A + B</td>
<td>C := A + B</td>
<td>A := 6</td>
<td>A := 6</td>
</tr>
<tr>
<td>B := 5 + A</td>
<td>A := 6</td>
<td>B := 5 + A</td>
<td>B := 5 + A</td>
</tr>
<tr>
<td>A := 6</td>
<td>C := A + B</td>
<td>C := A + B</td>
<td>C := A + B</td>
</tr>
<tr>
<td>B := 5 + A</td>
<td></td>
<td></td>
<td>C = 17</td>
</tr>
</tbody>
</table>

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 for 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 := “Gator00”

complex - real and imaginary parts
x := 5 + 4i

logical - binary on/off switch
x := 1 or 0

2) Constants/Variables / Arrays
Named locations for storing data types. We can represent quantities with whatever names we like.

A. CONSTANT
   numbers and words entered explicitly

B. 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?

Let’s combine all the programming components that you learn so far.

Example for the Programming Process:

Write a program to calculate the mass of a hollow sphere. The outer diameter is 10.2, the density is 1.25, the thickness would be input by user (the program user decides)

2) given input: diameter, density    user 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

- Router := Diameter / 2

- Rinner := Router – thickness

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

- Volume = 430.859

what happens if capitalization is wrong in a variable (not consistent)?

Now test the code
I’ll use a calculator to run an example and compare with my results

Mathcad knows the value of “Volume”. If you mis-typed it as “volume”, then you define a totally different variable rather than “Volume” It is very important to be consistent with assigning and using a variable name.

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 `.`) Yes. it’s a dot.

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 point size, 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.
Mathcad distinguishes between names in different fonts. For example, AREA is a different variable than AREA.

Variables with Units

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.

Calculations in Mathcad can be done on a unitless basis or by associating defined units with parameters in the calculations. One advantage of using units is that unit conversions are automatically performed by Mathcad. For example, if one variable is defined as having units of feet and another inches and the two values are added, a unit conversion will automatically takes place. The result of the calculation can be displayed in feet or inches, etc.
\( x := 10 \text{ft} \quad y := 12 \text{in} \quad z := x + y \quad z = 11 \text{ft} \quad z = 132 \text{in} \)

You can also define custom units that are not predefined in Mathcad. This is often useful in engineering where we use units like kips, ksi, etc. The result of the calculation can be displayed in feet or inches, etc.

\[
\begin{align*}
\text{kip} & := 1000 \text{lbf} \\
\text{ksi} & := \frac{1 \text{kip}}{1 \text{in}^2} \\
w & := 3 \frac{\text{kip}}{\text{ft}} \\
L & := 8 \text{ft}
\end{align*}
\]

\[
\begin{align*}
Mo & := \frac{wL^2}{8} \\
Mo & = 24 \text{kip ft} \\
Mo & = 288 \text{kip in}
\end{align*}
\]

Another advantage of using units is that Mathcad will not allow you to perform a calculation that has inconsistent units. Thus, many errors can be caught using this unit checking mechanism. The following simple example illustrates this, however, the same checking will be performed no matter how complex and large the expression becomes.

\[
\begin{align*}
x & := 10 \text{ft} \quad y := 10 \text{kip} \quad x + y = 1 \\
\text{values having different units cannot be added together. Error message is given “The units in this expression do not match”}
\end{align*}
\]

**Example** of using units (SI):

These variables will remain constant

\[
\begin{align*}
\text{Density} & := 1.25 \frac{\text{kg}}{\text{m}^3} \\
\text{Diameter} & := 10.2 \text{m}
\end{align*}
\]

Calculate outer and inner radius

\[
\begin{align*}
\text{Router} & := \frac{\text{Diameter}}{2} \\
\text{Rinner} & := \text{Router} - \text{thickness}
\end{align*}
\]

\[
\begin{align*}
\text{Volume} & := \frac{4}{3} \pi \left( \text{Router}^3 - \text{Rinner}^3 \right)
\end{align*}
\]

\[
\begin{align*}
\text{Volume} & = 430.859 \text{m}^3
\end{align*}
\]

**C. ARRAYS**

More than one storage space under a single name

Referenced by a number
e.g. Using 3 individual variables to store student grades

\[
\begin{align*}
\text{grade1} & := 23 \\
\text{grade2} & := 25 \\
\text{grade3} & := 18
\end{align*}
\]

Using a single array (grade) to store student grades

\[
\text{grade} = \begin{pmatrix} 
0 \\
23 \\
25 \\
18
\end{pmatrix}
\]

Why there are four values although we only entered 3 values? We’ll see later (i.e., as a built-in variable, an origin of the array (grade) is defaulted equal to 0). But we can change it by re-defining it as 1.

**Array types (a first taste)**

**scalar** - single piece of information (no index)

\( x := 8 \)

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

\( x := \text{Ctrl+M} \text{ pick # rows and 1 column, enter values} \)

\[
x := \begin{pmatrix} 
4 \\
16 \\
2 \\
3
\end{pmatrix}
\]

1 column, 3 rows

\( x_2 = 16 \)

**2-D array** - multiple rows and columns

\( x := \text{Ctrl+M} \text{ pick # rows and columns, enter values} \)

\[
x := \begin{pmatrix} 
2 & 4.5 & 9.8 \\
10.3 & 35.6 & 46
\end{pmatrix}
\]

2 rows, 3 columns

\( x_{1,3} = 9.8 \)
3) Input / Output

Input - methods of including data needed to begin or continue

   enter as constant

weight := 185

load from a separate saved file (excel, 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 MS-Word, etc.

Load or read an outside data file

To read data from a data file using the File Read/Write component:

a) Click in a blank spot of your worksheet.
b) Choose Component from the Insert menu.
c) Choose File Read or Write from the list and click “Next.” This launches the first part of the File Read or Write Setup Wizard.
d) Choose “Read from a file” and press “Next” to go to the second page of the Wizard.
e) From the File Format drop-down list in this Wizard, choose the type of file you want to read.
f) Type the path to the data file you want to read or use the “Browse” button to locate it.
g) Press “Finish.” You’ll see the File Read or Write component icon and the path to the data file.
h) In the placeholder that appears to the left, enter the name of the Mathcad variable to which the data from the file will be assigned.

4) Assignment statements

Assign information to a variable using : =

   ** constants **
x := 14.5

   ** expressions **
\[ x := 14.5 - 6 + 18.0/3.0 \]

** equations **
\[ y := 5 \]
\[ x := 2.4 \]
\[ z := (y - 1)/2 + y^3 - \sin(x) \]

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 judgment based on the IQ score

\[ \text{IQ} := 40 \]

\[ \text{Words} := \begin{cases} 
"\text{Go Vols}" & \text{if } \text{IQ} < 80 \\
"\text{Go Gators}" & \text{otherwise} 
\end{cases} \]

\[ \text{Words} = "\text{Go Vols}" \]

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 (using a built-in function; \text{ceil}(x); returns the greatest integer) and the other; \text{floor}(x))
fact(x) := |
s ← 1
x ← ceil(x)
for i ∈ 1..x
s ← s·i

s

fact(2.5) = 6

Did you notice *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.

This enables us to
    Break large problems into several smaller ones
    Eliminates re-writing statements over and over
Make your program accessible by any future main program (Modules)
Create our own functions
Utilize hundreds of Mathcad 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

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

mean(x) = 6.667

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