Simulation in Materials Engineering

T1: Fundamentals of programming

J. Segurado

Departamento de Ciencia de Materiales
Polytechnic University of Madrid

September 5, 2018

Outline

1. Programming languages
2. Real numbers
3. MATLAB and OCTAVE frameworks

Programming

Computer Programming is a fundamental tool in Engineering and Science

But, what's programming and what's it used for?

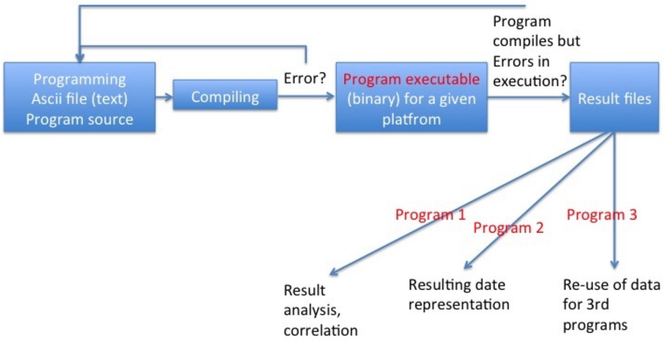
Programming is the act of entering instructions for the computer to perform. An ordered set of instructions to perform a task is a program. All programs use basic instructions as building blocks, as example

- Do this; then do that.
- If this condition is true, perform this action; otherwise, do that action.
- Do this action that number of times.
- Keep doing that until this condition is true.

These building blocks are combined and organized to form a program

Programming languages

Classically, programming is done using *compiled* languages, in the case of technical & scientific programming. FORTRAN, C, C++.



Scripting languages and MATLAB

- Alternative: scripting languages, do not requiere compilation and may integrate many of the tools in the same envirnoment.
- MATLAB (MATrix LABoratory) is an integrated environment for programming and visualization in scientific computing.

Advantages

- High level, Simple
- Many functions, algorithms, methods already implemented

Disadvantages

- Commercial (and expensive) framework
- Interpreted language: "Not" possible to generate executables to be run without MATLAB

OCTAVE

Alternative: GNU OCTAVE. A free implementation of a framework *almost* equal to MATLAB www.gnu.org/software/octave/ Current version (2018) is 4.4, and includes a Graphical User Interface very similar to MATLAB. Builts exist for several platforms:

- Mac OS-X, http://wiki.octave.org/Octave_for_MacOS_X
  - A binary built of Octave 4.03 can be installed for IOSX > 10.9.1, see previous web page
  - Install current Octave version using a Virtual Machine through Vagram, see instructions in http://deepneural.blogspot.fr/p/instructions-1_10.html
  - Compile original source files, only recommended for experts http://wiki.octave.org/Octave_for_Mac
- Windows: An official windows installer is available for version 4.0 https://ftp.gnu.org/gnu/octave/windows/octave-4.0.0_0-installer.exe
- Linux: Octave is included in repositories of most common Linux distributions. For more information and latest version http://www.gnu.org/software/octave/download.html

Working with numbers

The different sets of numbers defined in maths have to be represented in the computer, but this representation cannot be exact.

- The infinite Set $\mathbb{Z}$ includes the integer numbers. Countable magnitudes can be represented with $\mathbb{Z}$. Integers are "discrete" and can be exactly represented in a computer using bits. The only limit is the maximum number to be represented. `int8`, `int16`, `int32` are the three type of integers in matlab,
- The Set $\mathbb{R}$ of real numbers, of infinite size, is used to express physical magnitudes in general.
- Computers cannot use the set $\mathbb{R}$ and use the finite dimension subset $\mathbb{F}$ instead
- The numbers of the set $\mathbb{F}$ are called *floating point numbers*
- A computer substitutes each real x by its corresponding floating point $fl(x)$ by truncation. In matlab, floating points have double precision by default and are called `double`.

Representation of floating points

OCTAVE/MATLAB exercise

Lets consider a rational number, i.e. $x = 1/7$ which corresponds to $0.\overline{142857}$

```
>>1/7
ans = 0.14286
```

Try different representations of the number
`>>format long` gives `0.142857142857143`
`>>format long e` gives `0.142857142857143E-001`
`>>format short e` gives `0.14286E-001` See in the "variable explorer" that in all the cases, the type of variable is `double`

Representation of floating points

- As seen in the example real numbers (x) are truncated and the format indicates the computer the *precision* of the truncation
- The actual floating point number ($fl(x)$) stored by the computer can be written as
  $fl(x) = (-1)^s \cdot (0.a_1 a_2 a_3 ... a_t) \cdot \beta^e = (-1)^s \cdot m \cdot \beta^{e-t}, a_1 \neq 0$
  - $s = 0$ or 1, to define the sign
  - $\beta \geq 2$ is a natural number called *base*
  - m is an integer called *mantissa* of length t
  - e is an integer called exponent
- The precision of the floating point is given by t, and depends on the machine and/or program

Representation of floating points

- The set $\mathbb{F}$ is characterized by β, t and the range of e, $L \leq e \leq U$, $\mathbb{F} = \mathbb{F}(\beta, t, L, U)$
- In MATLAB $\mathbb{F} = \mathbb{F}(2, 53, -1021, 1024)$, corresponding to $t = 15$ in decimal base
- The relative rounding error of using a floating point instead of the real number corresponds to

$$\frac{|x - fl(x)|}{|x|} \leq \frac{1}{2}\epsilon_m = \frac{1}{2}\beta^{1-t} \qquad (1)$$

  and in MATLAB ϵ_m can be obtained by `>>eps` being $\epsilon_m = 2^{-52} \approx 2.22 \cdot 10^{-16}$ (double precision)
- The maximum and minimum positive floating point values are $x_{min} = \beta^{L-1}$ and $x_{max} = \beta^{U}(1 - \beta^{-t})$, obtained using `realmin` [1] and `realmax`

[1] MATLAB and OCTAVE provide a false value of *realmin* for 64 bit processors.

Operations with floating numbers

- The zero does not exists in $\mathbb{F}$ but if an operation gives a number smaller than x_{min} is treated as zero (underflow)
- If an operation gives a number bigger than x_{max} MATLAB provides `Inf` (Overflow)
- Associative property is lost when they involve under and overflow, or number with similar absolute values and contrary sign
- Indetermination as $0/0$ or ∞/∞ gives `NaN`

OCTAVE/MATLAB exercise

Let x be a small number, i.e. $1 \cdot 10^{-15}$, make ((1+x)-1)/x and calculate relative error with the real result.
Make the same with $x = 1 \cdot 10^{-14}$

MATLAB and OCTAVE frameworks

- MATLAB has a graphical environment with menus, windows and toolboxes. OCTAVE originally works in a text terminal. In last versions (>3.6) a graphical environment similar to MATLAB is provided (default from version 4.0)
- Variable treatment, basic programming, functions, etc have the same syntax in both programs and the examples and exercises will be compatible with both.
- Operations, programs, are written after the prompt ($>>$)

Use of the terminal

Matlab and Octave graphical environments include several elements

- Editor. Is a text editor to write a script before executing it. Any editor (notepad, gedit...) can be used for this purpose but the built-in editors provide enhancements to recognize functions, mark typing errors, etc
- Terminal. Is the engine, this is really the important part of the code. You type instructions there and you get the result. Instructions can be operations, call to functions...
- Utility windows: Variable explorer, menus, file manager.

Instructions to the terminal are typed after the prompt, usually $>>$ The first "Code" in matlab, equivalent to typical first C code is to ask the computer to write a message. This is done by

```
>>disp("Hello")
"Hello"
```

The first use if matlab is normally using the terminal as a calculator

Variables

- Variable type has not to be declared (double, integer, character), MATLAB declares it **automatically** and **dynamically** when initialized
- A variable is initialized by the operator $=$, examples:

```
>>a='simulation'
ans = simulation
>>a=3E-2
ans = 0.003
>>a=3
ans=3
```

- Variables cannot start with a number, bracket, point and cannot contain operator signs (i.e. $+, \&$)
- Variables are case sensitive! $a \neq A$

Algebraical operations and functions

- Addition, subtraction, multiplication, division and exponentiation of variables are done by standard symbols `+,-,*,^`
- Operations can be joined using (), order of operations are `(),^,*,/,+,-`
- Typical functions are implemented in MATLAB, `sin(x), cos(x), tan(x), atan(x), exp(x), log(x), log10(x), sinh(x), cosh(x), tanh(x),abs(x)`

OCTAVE/MATLAB exercise

Define x to be an integer, operate $x * 2$
Make the same being x a real and a character string.
Define x to be an integer, calculate $\frac{x+e^x}{x}$
Make the same being x a real and a character string.

Matrices and vectors

- A matrix **A** is a set of elements a_{ij} distributed in m rows and n columns

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

- A matrix in MATLAB is delimited by [and],the elements of a row are separated by spaces and rows by ; example,the matrix **A** of entire numbers

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & 5 \\ 4 & 1 & 0 \end{bmatrix}$$

is declared using

```
>> A = [ 2 3 5; 4 1 0]
A =
        2     3     5
        1     4     0
```

Operations with matrices

- Addition. If $\mathbf{A}_{m\times n} = [a_{ij}]$ and $\mathbf{B}_{m\times n} = [b_{ij}]$, the addition of **A** and **B** is a new matrix $\mathbf{A}+\mathbf{B}_{m\times n} = [a_{ij} + b_{ij}]$
  In MATLAB `>>A+B`
- The product of matrix **A** by a real number λ is a new matrix $\lambda\mathbf{A} = [\lambda a_{ij}]$
  In MATLAB `>> a*A`
- Let $\mathbf{A}_{m\times p}$ and $\mathbf{B}_{p\times n}$ then the product of **A** and **B** is a new matrix $\mathbf{AB} = \mathbf{C}_{m\times n}$, being

$$c_{ij} = \sum_{p}^{k=1} a_{ik} b_{kj}$$

  In MATLAB `>>A*B`
- If $\mathbf{A}_{m\times n} = [a_{ij}]$, then the transpose of **A** is $\mathbf{A}^T_{n\times m} = [a_{ji}]$. Property: $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$. In MATLAB, the operation is done by `transpose(A)`

Ranges and submatrix extraction

A range is a sequence of numbers that can be generated using the range operator, namely, colon(:) and very useful to create and manipulate matrices and vectors

- To create a vector *m* containing a list of integers from i to j `>m=i:j`
- To create a vector *m* containing a list of integers from i to j by increment k `>m=i:k:j`

In MATLAB/Octave is very easy to add/delete/extract rows or columns from a given matrix using range operator (:). Let $\mathbf{A}_{m\times n}$ be a matrix, then

- To extract the row *i* `>>A(i,:)`
- To extract the column *j* `>>A(:,j)`
- To extract a submatrix `>>A(m,n)`, being *m* and *n* vectors of integers, i.e $m = [1\ 3\ 5]$ and $n = 1$

Generation of special matrices

- Unit square matrix of size $n \times n$ `>>eye(n)`
- 0 matrix of size $n \times m$ `>>zeros(n,m)`
- Matrix with random coefficientes [0,1] of size $n \times m$ `>>rand(n,m)`

OCTAVE/MATLAB exercise

- Create a random matrix **A** with elements [0,5.] of size 10×10
- Create a vector **m** containing the odd numers between 0 and 5
- Extract the row and the column 3 of **A**
- Create a new matrix containing only odd rows and columns. Use **m**
- Add the corresponding unit matrix to the last one

Determinant of square matrices

- Determinant of a matrix $\mathbf{A}_{n\times n} = [A_{ij}]$, defined by *Laplace rule* (recursive)

$$\det(\mathbf{A}) = \begin{cases} a_{11} & \text{if } n = 1 \\ \sum_{j=1}^{n} \Delta_{ij} a_{ij}, \text{for any i}, 1 \leq \text{i} \leq \text{n} & \text{if } n > 1 \end{cases}$$

  being Δ_{ij} a *minor*, defined as $\Delta_{ij} = (-1)^{i+j}\det(\mathbf{A}_{ij})$ and $\mathbf{A}_{ij}$ is the square matrix of $(n-1)\times(n-1)$ dimensions, obtained by eliminating $i-$th row and the $j-$th column to the matrix $\mathbf{A}$
- The rule involves for a general matrix a number of $2n!$ operations.
- Example, for $n = 2$, $\det(A) = a_{11}a_{22} - a_{12}a_{21}$
- For $n = 3$, $det(A) = a_{11}a_{22}a_{33} + a_{31}a_{12}a_{23} + a_{21}a_{13}a_{32} - a_{11}a_{23}a_{32} + a_{21}a_{12}a_{33} + a_{31}a_{13}a_{22}$

Determinant of square matrices

Some properties of the determinants:

- If any row or column is 0 or a linear combination of others then $\det = 0$ (singular matrix).
- $\det(\mathbf{AB}) = \det(\mathbf{A})\det(\mathbf{B})$
- $\det(\mathbf{A}^T) = \det(\mathbf{A})$
- If $\mathbf{A}_{n\times n}$ is *triangular* then $\det(\mathbf{A}) = a_{11}a_{22}\cdots a_{nn}$

The command for computing the determinant of a square matrix **A** in MATLAB is `det(A)`. It is not based on Laplace rule and is much more efficient

OCTAVE/MATLAB exercise

Define a scalar a and two non-singular matrices $\mathbf{A}_{3\times 3}$ and $\mathbf{B}_{3\times 3}$, and calculate $\det(\mathbf{A})$, $\det(\mathbf{B})$,$\det(\mathbf{AB})$, $\det(a\mathbf{A})$. What is the relation between $\det(\mathbf{A})$ and $\det(a\mathbf{A})$?

Inverse of a square matrix

A n-by-n matrix $\mathbf{A}$ is called invertible or nonsingular or nondegenerate, if there exists an n-by-n matrix $\mathbf{B}$ such that $\mathbf{AB} = \mathbf{BA} = \mathbf{I}_n^*$. If this is the case, $\mathbf{B}$ is uniquely determined by $\mathbf{A}$ and is called the inverse of $\mathbf{A}$, denoted by $\mathbf{A}^{-1}$

- A square matrix is singular if and only if its determinant is 0
- If $\mathbf{A}_{n\times n}$ and $\mathbf{B}_{n\times n}$ are nonsingular then, $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$
- $\det(\mathbf{A}^{-1}) = \det(\mathbf{A})^{-1}$

* $\mathbf{I}_n$ is a unit matrix, that is defined as a diagonal matrix with all its terms equal to 1.

$$\mathbf{I}_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverse of a square matrix

In MATLAB the inverse of a matrix **A** can be obtained using `inv(A)`

OCTAVE/MATLAB exercise

- Define a scalar a and two non-singular matrices $\mathbf{A}_{3\times 3}$ and $\mathbf{B}_{3\times 3}$, and calculate $\mathbf{A}^{-1}$, $\mathbf{B}^{-1}$,$(\mathbf{AB})^{-1}$, $(a\mathbf{A})^{-1}$. What is the relation between $\mathbf{A}^{-1}$ and $(a\mathbf{A})^{-1}$?
- Calculate $\mathbf{A}^{-1}\mathbf{A}$ and $\mathbf{A}\mathbf{A}^{-1}$ and check the solution
- Introduce the matrix **A**, `>>A=[1E30 0 1 ; -1 1E-9 0; 0 0 2]` Obtain the matrix $\mathbf{B} = \mathbf{A}^{-1}$. Finally calculate $\mathbf{B}^{-1}$. Is logical the result?

Programming using MATLAB

MATLAB and OCTAVE provide an easy method to program interpreted applications using all the commands and functions implemented in MATLAB.

Relational operators

Let *a* and *b* be two variables, the comparison between a and b is a relational operator that returns a *logical value* of 1 if *true* and 0 if *false*

- Equal `a==b`
- Greater `a>b`, greater than `a>=b`
- Lower `a<b`, lower than `a<=b`
- Different than `a!=b` and also `a~=b`

OCTAVE/MATLAB exercise

- Define *a* and *b* as 2 different reals. Calculate `a==b; a>b; a=>b; a<b; a=<b; a!=b`
- Do the same being *a* and *b* as 2 matrices $m \times n$
- Do the same defining *a* and *b* as character string.

Programming using MATLAB

Terminal Input/output

Display a text MATLAB/OCTAVE displays a text or value of a variable by using the command `disp()`

- Let a be an initialized variable, `disp(a)` returns its value
- `disp('THIS IS A TEXT')` returns the string between ' '
- Formatted output can be done using fprintf:

```
age = 21;student='Juan';
fprintf ('Student %s is %d years old.\n',
        student,pct);
```

- The arguments within `printf` are printed in specific formats:
  - `%d` for entire decimal number
  - `%s` for string of characters
  - `%f` and `%e` are used for floating numbers in standard fixed point notation or engineering notation respectively
  - `\n` indicates new line

Programming using MATLAB

Terminal Input/output

Read a value from the terminal The command `var=input('TEXT')` displays the TEXT and waits until the entry of a value through the keyboard. Value is saved on variable *var*
Example: `age=input('How old are you')`

OCTAVE/MATLAB exercise

- Use `input` to get an integer number saved in variable *a*
- Generate a random number in a varibale *b*
- Define a variable *c* containing the product $a \times b$
- Write using `printf` the three numbers in its corresponding format within a sentence like *given value is a, random number is b and product is c*

Programming using MATLAB

File Input/output

In many cases, inputs for a program are stored in an ascii file, i.e. points of a stress strain curve. In addition, many times is necesary to output program results to an external file.
The process to open, manipulate and close a file

```
filename = 'myfile.txt';
fid = fopen (filename, 'mode');
# Do the actual I/O here...
fclose (fid);
```

where `mode` is substituted by write `w` or read `r`

- for writting in a file `fprintf(fid,...)` is used with same conventions as `printf`
- for reading from a file `fscanf (fid, template)` is used, where template indicates the format to be read

Programming using MATLAB

Logical operators

Logical operators The symbols `&` , `|` and `~` are the logical operators AND, OR, and NOT. These operators are used in conditional statements. Logical operations return a logical variable 1 (true) or 0 (false), as appropriate. Let $cond1$ and $cond2$ be two conditions, i.e. $cond1 = a == b$, and $cond2 = c >= 2$ then the logical operators relate both conditions to give a new condition.

OCTAVE/MATLAB exercise

- Define a and b to be 0 or 1 (4 cases) and check `a&b`, `a|b` in all the cases
- Define three reals a, b, c, and two relational conditions, i.e. `cond1=a>=b`
- Check the result of expr1 AND expr2

Programming using MATLAB

Conditions

In programming is very useful that as a result of a given condition different expressions are run. This is done using the conditional *if* – *elseif* – *else* statements

```
if cond1
  statement1
else if cond2
  statement2
.
.
else
  statementN
end
```

Programming using MATLAB
Conditions

OCTAVE/MATLAB exercise

- Let $ax^2 + bx + c = 0$ be a second order equation in x with real coefficients a, b, c. Define a procedure to obtain the values of x for any possible number of real roots (2,1,or 0)
- Use the sequence to solve different cases: $2x^2 + x - 1 = 0$, $2x^2 + x - 1 = 0$, $3x^2 + x - 1 = 0$
- Define the polynomial function as a vector of the coefficients using `p= [ a b c]`
- Solve the equation using the MATLAB/OCTAVE command `roots(p)` and compare with the programmed sequence

Programming using MATLAB

Loops

- A loop is a sequence of statements which is specified once but which may be carried out several times in succession.
- The code "inside" (XXX) the loop is obeyed a specified number of times, or once for each of a collection of items, or until some condition is met, or indefinitely.

  for loop

  ```
  for var=val1:val2
  XXX
  end
  for var=MATRIX
  XXX
  end
  ```

Programming using MATLAB

Loops

while loop

```
while (condition)
  XXX
end
```

do/until loop

```
do
 XXX
until (condition)
```

Programming using MATLAB

Loops

OCTAVE/MATLAB exercise

The *e* number is a real constant that appears many times in mathematic and physics. Its value is $e \approx 2,718281828459045235360287471352....$ It can be defined as the value of an infinite series

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} \quad (2)$$

- Write a procedure using a "for loop" to obtain an approach of e with 2,10,100 and 1000 terms of the series
- Write a procedure using a "while" or "do" loop to obtain a "good" approach to *e*

Programming using MATLAB

Scripts and programs

- A program or a script is a sequence of instructions/operations structured to perform a more complex procedure
- In MATLAB and OCTAVE (interpreted languages) program consist on ascii files containing the sequence of operations in lines.
- A MATLAB/OCTAVE script SHOULD always end in `.m`, i.e. `example.m`
- Programs are run by writing the name of the script in the prompt `>>example`

OCTAVE/MATLAB exercise

Write a program that ask for the coefficients of a second order equations and provides the solution just in case they are real

Programming using MATLAB

Functions

- In programming is very usual that a given set of operations is repeated several times in the code, i.e. a second order polynomical equation that has to be solved several times
- For the sake of clarity, code brevity, etc, those sets of operations (small programs) can be written as *functions*
- Each functions is called by its name, and must be saved as a file `name.m`

```
function [out1 ,... , outn]= name(in1 ,..., inm)

definition of the operations to do with in1 to inm
in order to obtain out1 to outn


return
end
```

Programming using MATLAB

Functions

Examples: Program to obtain the determinant of a 2x2 matrix

```
function detFUNC=determinant2(A)
[n,m]=size(A);
if n==m
detFUNC=A(1,1)*A(2,2)-A(1,2)*A(2,1);
disp ('determinant computed, value='),disp(detFUNC);
else
disp ('ERROR: Matrix is not 2x2 ');
end

return
end
```

OCTAVE/MATLAB exercise

Write and run the previous determinant2 function. Try with and without ";"

Programming using MATLAB

Functions

OCTAVE/MATLAB exercise

A Fibonacci series consist on the sequence of numbers where each new number of the series consist on the addition of the preceding 2 values. The values of the terms depend on two initial values a_1, a_2, and $a_n = a_{n-1} + a_{n-2}$ Program a function that uses a_1, a_2 and n as input parameters and gives the value of the term a_n

Plots

Functions, time series, etc can be represented within the MATLAB/Octave framework, using `plot` command:

- Plot the elements of a vector Y as function of the index: `plot(Y)`
- Plot Y=Y(X), being X and Y vectors `plot(X,Y)`
- Plot functions, `t = 0:0.1:6.3; plot (t, cos(t))`
- In MATLAB and Octave > 6.3 or UPM-Octave, graphs can be personalized, modified and saved using GUI menus.

OCTAVE/MATLAB exercise

Plot the evolution of the fibonacci series as function of *n* until n=100

Computational costs

- The computational cost of an algorithm is the number of floating point operations that are required for its execution.
- The maximum number of floating point operations which the computer can execute in one second (flops) measures the speed of a computer. *Megaflop* $= 10^6$ *flops*, *Gigaflop* $= 10^9$ *flops* and *Teraflop* $= 10^{12}$ *flops*
- As example, a PC processor of year 2010, 6 core PC Intel Core i-7 980 XE reaches 109 GFlops

The efficiency of an algorithm is usually evaluated by the **order of magnitude** of the floating point operations needed as a function of a paramenter n that measures the size of the analyzed system $O(n)$, i.e. in a linear system n can be the number of unknowns.

Computational costs

Growth of number of operations for different algorithms

n	O(1)	$O(\log n)$	$O(n)$	$O(n \log n)$	$O(n^2)$	$O(n^3)$
1	1	1	1	1	1	1
2	1	1	2	2	4	8
4	1	2	4	8	16	64
100	1	5	100	500	1E4	1E6
1000	1	7	1000	7000	1E6	1E9

Usual computational methods:

- Molecular dynamics (evaluation of potential) with cutoffs $O(n)$
- Dislocation dynamics,evaluation of Peach-Koeler forces $O(n^2)$
- Finite elements $O(n^2 - n^3)$ (sparse matrix inversion)