Practice Session – Introduction to Octave

Energie eolienne - Wind Energy (U-PEC)

Renewable Energies and Energy Efficiency of Sustainable Buildings

Emilio Gómez Lázaro (emilio.gomez@uclm.es)[1]
Andrés Honrubia Escribano (andres.honrubia@uclm.es)[1]
Estefanía Artigao Andicoberry (estefania.Artigao@uclm.es)[1]

[1]Dpto. de Ingeniería Eléctrica, Electrónica, Automática y Comunicaciones
ETS de Ingeniería Industrial. Instituto de Investigación de Energías Renovables

Academic year 2023/2024

Octave Session

- Index
  - Installation manual
  - About Octave
  - Octave start screen
  - Before getting started….
  - Data types and structures
  - Operators & predefined functions
  - Coding: loops
  - Coding .m files: scripts and functions
  - Inputs and outputs: plotting, showing, opening and saving data

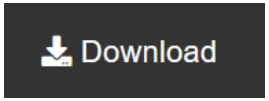
  - **Octave Practice**

Octave Session

GNU Octave

Scientific Programming Language

Installation manual

1. Visit: https://www.gnu.org/software/octave/index
2. Click on: 

3. Choose your Operating System. In case of Windows:

   - Windows-64 (recommended)
     - octave-9.1.0-w64-installer.exe (~ 380 MB) [signature]

4. Download the installer.exe, execute it and follow the different steps of the installation process.

About Octave

GNU Octave

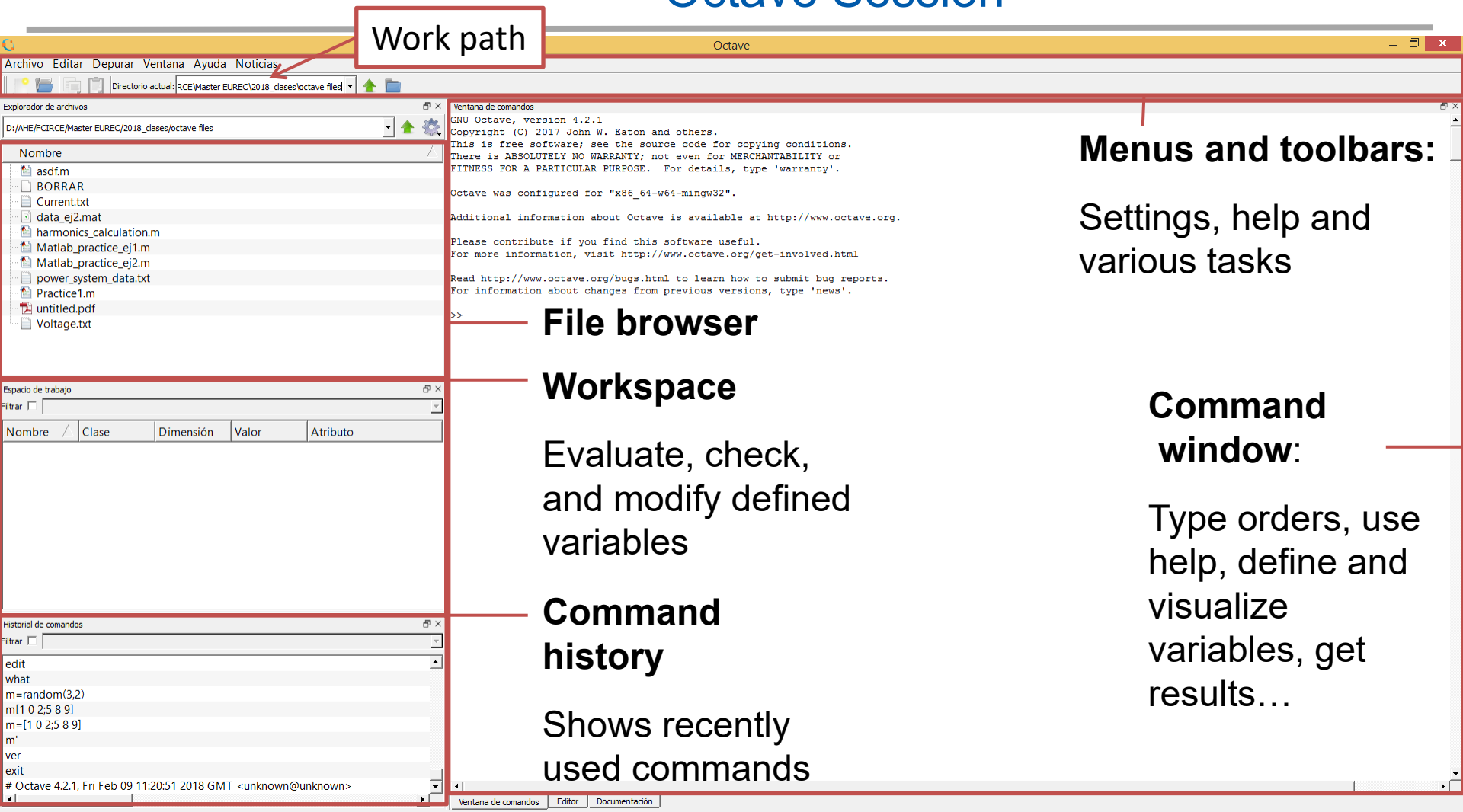
Scientific Programming Language

- Open-source interactive software system for numerical computations and graphics.
- Particularly designed for matrix computations.

```
>> ver
```

- Has its own native language, but it is compatible with other languages and formats, like C++ or Excel macros.
- Huge amount of predefined functions included in the basic software.
- Applications: computation and calculus, algorithm development, modeling, simulating and prototyping, data analysis, data selection, exploration and visualization.

Octave Session



Menus and toolbars:

Settings, help and various tasks

File browser

Workspace

Evaluate, check, and modify defined variables

Command history

Shows recently used commands

Command window:

Type orders, use help, define and visualize variables, get results…

Octave Session

Before getting started…

- **Errors:** they appear on the command window.

```
>> asdfasdfa1341234
error: 'asdfasdfa1341234' undefined near line 1 column 1

>> a=1; a(3)
error: a(3): out of bound 1

>> a='home'
a = home
>> sin(a)
error: sin: argument must be numeric
```

Octave Session

Before getting started…

- **Suppress command output:** semicolon at the end of the sentence.

```
>> a=1
a =
     1

>> b=2;
>> b
b =
     2
```

- **Clear command window:**

```
>> clc
```

- **Clear workspace (all variables):**

```
>> clear all
```

- **Stop ongoing process (useful if Octave gets stuck or to exit loops):** control + c

- **Close session:**

```
>> exit
```

Before getting started…

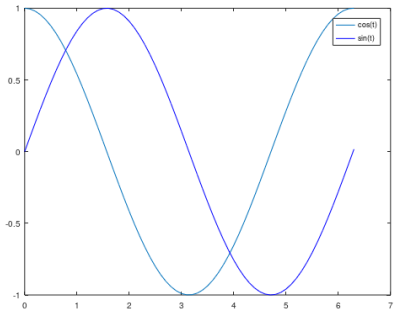
- **Octave help:** extremely useful! Access help typing *help* function:

```
>> help plot
'plot' is a function from the file C:\Octave\OCTAVE~1.0\mingw64\share\octave\5.1.0\m\plot\draw\plot.m

 -- plot (Y)
 -- plot (X, Y)
 -- plot (X, Y, FMT)
 -- plot (..., PROPERTY, VALUE, ...)
 -- plot (X1, Y1, ..., XN, YN)
 -- plot (HAX, ...)
 -- H = plot (...)
     Produce 2-D plots.

     Many different combinations of arguments are possible.  The
     simplest form is

          plot (Y)
```

Function description

Examples:

```
     t = 0:0.1:6.3;
     plot (t, cos(t), "-;cos(t);", t, sin(t), "-b;sin(t);");

This will plot the cosine and sine functions and label them
accordingly in the legend.
```



Octave Session

Data types & structures

- **Numbers:**
  - Floating point: 22.6453 6.05e23
  - Integer 8 -50
  - Complex: i 2.2e-8+ 4e-8j

```
>> help format
```

- **Char:** Array of characters, size 1xN.

Single quote marks!

```
>> 'char_array2343134'
```

Name	Value	Size	Class
ans	'char_arra...	1x17	char

- **Variables:** Not declaring needed, only definition!

```
>> this_variable_is_an_example235123=22
this_variable_is_an_example235123 =
    22
```

Numbers and characters can be used in variable names, but do not include Octave operators!!!

```
>> this_variable_is_an_example235123^^=22
parse error:

  syntax error

>>> this_variable_is_an_example235123^^=22
                                      ^
```

Octave Session

Data types & structures

- **Constants:** pi, NaN, i or j, inf…
- **Array:** NxM array, all data within the array must have the same data type.
  Every row should have the same number of elements. Data types: numbers, chars…
- **Cell array:** NxM array, data within the array can have different data type.
  Elements may have different dimensions
- **Other:** Structures, objects, handles…

```
>> matrix=[ 1 2; 3 4]
>> matrix
>> matrix(1,2)
>> matrix(1)
>> matrix(1,:)

>> string='abcdefgh'
>> string(4:6)
```

```
>> vector_test = [1:2:10]

vector_test =

     1     3     5     7     9

>> vector_test = linspace(1,10,5)

vector_test =

    1.0000    3.2500    5.5000    7.7500   10.0000
```

```
>> new_cell=cell(1,2);
>> new_cell{1}=string;
>> new_cell{2}=matrix;
>> new_cell{2}(1)
>> new_cell{1,1}(3:5)
>> new_new_cell=cell(1,2);
>> new_new_cell{1}=new_cell;
>> new_new_cell{1}{2}(2,:)=[-52 1]
```

Operators & predefined functions

- Bear in mind the input data type on operators and functions, and all the inputs needed to use them

```
Arithmetic operators.
  plus       Plus                                +
  uplus      Unary plus                          +
  minus      Minus                               -
  uminus     Unary minus                         -
  mtimes     Matrix multiply                     *
  times      Array multiply                      .*
  mpower     Matrix power                        ^
  power      Array power                         .^
  mldivide   Backslash or left matrix divide     \
  mrdivide   Slash or right matrix divide        /
  ldivide    Left array divide                   .\
  rdivide    Right array divide                  ./
  kron       Kronecker tensor product            kron
```

```
Relational operators.
  eq         Equal                               ==
  ne         Not equal                           ~=
  lt         Less than                           <
  gt         Greater than                        >
  le         Less than or equal                  <=
  ge         Greater than or equal               >=
```

```
Logical operators.
             Short-circuit logical AND           &&
             Short-circuit logical OR            ||
  and        Element-wise logical AND            &
  or         Element-wise logical OR             |
  not        Logical NOT                         ~
  xor        Logical EXCLUSIVE OR
  any        True if any element of vector is nonzero
  all        True if all elements of vector are nonzero
```

```
Special characters.
  colon      Colon                               :
  paren      Parentheses and subscripting        ( )
  paren      Brackets                            [ ]
  paren      Braces and subscripting             { }
  punct      Function handle creation            @
  punct      Decimal point                       .
  punct      Structure field access              .
  punct      Parent directory                    ..
  punct      Continuation                        ...
  punct      Separator                           ,
  punct      Semicolon                           ;
  punct      Comment                             %
  punct      Invoke operating system command     !
  punct      Assignment                          =
  punct      Quote                               '
  transpose  Transpose                           .'
  ctranspose Complex conjugate transpose         '
  horzcat    Horizontal concatenation            [,]
  vertcat    Vertical concatenation              [;]
  subsasgn   Subscripted assignment              ( ),{ },.
  subsref    Subscripted reference               ( ),{ },.
  subsindex  Subscript index
```

```
Bitwise operators.
  bitand     Bit-wise AND.
  bitcmp     Complement bits.
  bitor      Bit-wise OR.
  bitmax     Maximum floating point integer.
  bitxor     Bit-wise XOR.
  bitset     Set bit.
  bitget     Get bit.
  bitshift   Bit-wise shift.
```

```
Set operators.
  union      Set union.
  unique     Set unique.
  intersect  Set intersection.
  setdiff    Set difference.
  setxor     Set exclusive-or.
  ismember   True for set member.
```

Octave Session

Operators & predefined functions

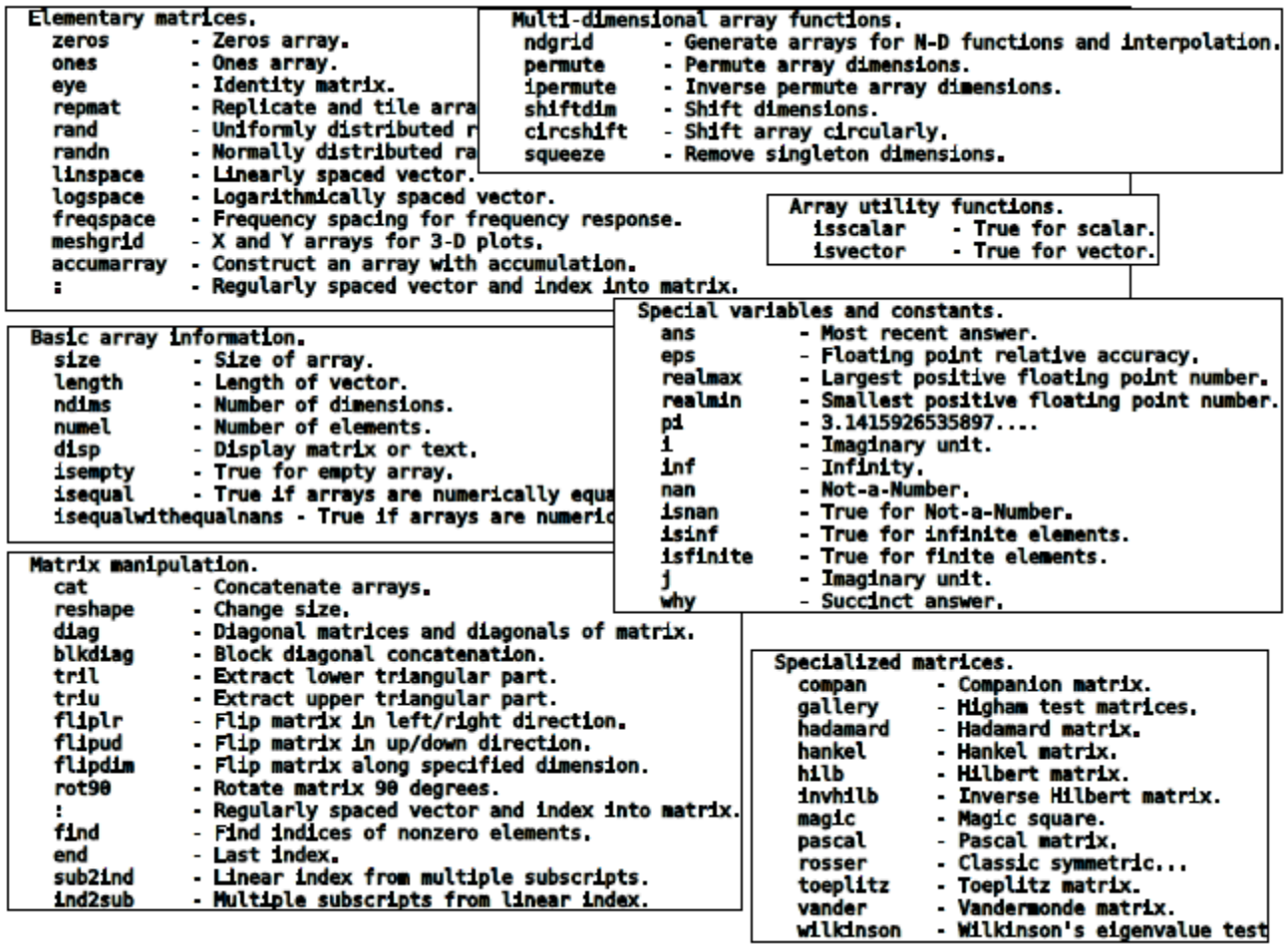
```
Trigonometric.
  sin         - Sine.
  sind        - Sine of argument in de
  sinh        - Hyperbolic sine.
  asin        - Inverse sine.
  asind       - Inverse sine, result i
  asinh       - Inverse hyperbolic sin
  cos         - Cosine.
  cosd        - Cosine of argument in
  cosh        - Hyperbolic cosine.
  acos        - Inverse cosine.
  acosd       - Inverse cosine, result
  acosh       - Inverse hyperbolic cos
  tan         - Tangent.
  tand        - Tangent of argument in
  tanh        - Hyperbolic tangent.
  atan        - Inverse tangent.
  atand       - Inverse tangent, resul
  atan2       - Four quadrant inverse
  atanh       - Inverse hyperbolic tan
  sec         - Secant.
  secd        - Secant of argument in
  sech        - Hyperbolic secant.
  asec        - Inverse secant.
  asecd       - Inverse secant, result
  asech       - Inverse hyperbolic sec
  csc         - Cosecant.
  cscd        - Cosecant of argument in degrees.
  csch        - Hyperbolic cosecant.
  acsc        - Inverse cosecant.
  acscd       - Inverse cosecant, resu
  acsch       - Inverse hyperbolic cos
  cot         - Cotangent.
  cotd        - Cotangent of argument
  coth        - Hyperbolic cotangent.
  acot        - Inverse cotangent.
  acotd       - Inverse cotangent, res
  acoth       - Inverse hyperbolic cotangent.
  hypot       - Square root of sum of squares.
```

```
Exponential.
  exp         - Exponential.
  expm1       - Compute exp(x)-1 accurately.
  log         - Natural logarithm.
  log1p       - Compute log(1+x) accurately.
  log10       - Common (base 10) logarithm.
  log2        - Base 2 logarithm and dissect floating point number.
  pow2        - Base 2 power and scale floating point number.
  realpow     - Power that will error out on complex result.
  reallog     - Natural logarithm of real number.
  realsqrt    - Square root of number greater than or equal to zero.
  sqrt        - Square root.
  nthroot     - Real n-th root of real numbers.
  nextpow2    - Next higher power of 2.
```

```
Complex.
  abs         - Absolute value.
  angle       - Phase angle.
  complex     - Construct complex data from real and imaginary parts
  conj        - Complex conjugate.
  imag        - Complex imaginary part.
  real        - Complex real part.
  unwrap      - Unwrap phase angle.
  isreal      - True for real array.
  cplxpair    - Sort numbers into complex conjugate pairs.
```

```
Rounding and remainder.
  fix         - Round towards zero.
  floor       - Round towards minus infinity.
  ceil        - Round towards plus infinity.
  round       - Round towards nearest integer.
  mod         - Modulus (signed remainder after division).
  rem         - Remainder after division.
  sign        - Signum.
```

Operators & predefined functions



```
Elementary matrices.
  zeros        - Zeros array.
  ones         - Ones array.
  eye          - Identity matrix.
  repmat       - Replicate and tile arra
  rand         - Uniformly distributed r
  randn        - Normally distributed ra
  linspace     - Linearly spaced vector.
  logspace     - Logarithmically spaced vector.
  freqspace    - Frequency spacing for frequency response.
  meshgrid     - X and Y arrays for 3-D plots.
  accumarray   - Construct an array with accumulation.
  :            - Regularly spaced vector and index into matrix.

Multi-dimensional array functions.
  ndgrid       - Generate arrays for N-D functions and interpolation.
  permute      - Permute array dimensions.
  ipermute     - Inverse permute array dimensions.
  shiftdim     - Shift dimensions.
  circshift    - Shift array circularly.
  squeeze      - Remove singleton dimensions.

Array utility functions.
  isscalar     - True for scalar.
  isvector     - True for vector.

Basic array information.
  size         - Size of array.
  length       - Length of vector.
  ndims        - Number of dimensions.
  numel        - Number of elements.
  disp         - Display matrix or text.
  isempty      - True for empty array.
  isequal      - True if arrays are numerically equ
  isequalwithequalnans - True if arrays are numeric

Special variables and constants.
  ans          - Most recent answer.
  eps          - Floating point relative accuracy.
  realmax      - Largest positive floating point number.
  realmin      - Smallest positive floating point number.
  pi           - 3.1415926535897....
  i            - Imaginary unit.
  inf          - Infinity.
  nan          - Not-a-Number.
  isnan        - True for Not-a-Number.
  isinf        - True for infinite elements.
  isfinite     - True for finite elements.
  j            - Imaginary unit.
  why          - Succinct answer.

Matrix manipulation.
  cat          - Concatenate arrays.
  reshape      - Change size.
  diag         - Diagonal matrices and diagonals of matrix.
  blkdiag      - Block diagonal concatenation.
  tril         - Extract lower triangular part.
  triu         - Extract upper triangular part.
  fliplr       - Flip matrix in left/right direction.
  flipud       - Flip matrix in up/down direction.
  flipdim      - Flip matrix along specified dimension.
  rot90        - Rotate matrix 90 degrees.
  :            - Regularly spaced vector and index into matrix.
  find         - Find indices of nonzero elements.
  end          - Last index.
  sub2ind      - Linear index from multiple subscripts.
  ind2sub      - Multiple subscripts from linear index.

Specialized matrices.
  compan       - Companion matrix.
  gallery      - Higham test matrices.
  hadamard     - Hadamard matrix.
  hankel       - Hankel matrix.
  hilb         - Hilbert matrix.
  invhilb      - Inverse Hilbert matrix.
  magic        - Magic square.
  pascal       - Pascal matrix.
  rosser       - Classic symmetric...
  toeplitz     - Toeplitz matrix.
  vander       - Vandermonde matrix.
  wilkinson    - Wilkinson's eigenvalue test
```

Coding: loops

- **4 main types of loop:** Different reserved words for each type of loop!

```
IF expression
   statements
ELSEIF expression
   statements
ELSE
   statements
END
```

```
FOR variable = expr
   statement
   ...
   statement
END

WHILE expression
   statements
END
```

```
SWITCH switch_expr
  CASE case_expr,
    statement, ..., statement
  CASE {case_expr1, case_expr2, case_expr3,...}
    statement, ..., statement
   ...
  OTHERWISE,
    statement, ..., statement
END
```

```
TRY
  statement
  ...
  statement
CATCH
  statement
  ...
  statement
END
```

```
break
continue
return
```

```
>> start=0;
>> if (1>2)&& (start==0)
start=start+1;
else start=start-1;
end
```

Coding .m files: scripts and functions

.m file: text file which has code written in Octave language. Once created, they can be invoked from the command window if they are in the current Work path. Two different types:

- **Script:** Contains Octave statements, works using workspace variables and has no output or inputs. Has the same effect as typing the code content.

my_script.m

```
1 -    N=32
2 -  for R = 1:N
3 -                for C = 1:N
4 -                    A(R,C) = 1/(R+C-1);
5 -                end
6 -  end
7
8
```

```
>> my_script
>> A
```

Octave Session

Coding .m files: scripts and functions

- **Function:** Contains Octave statements, that have outputs and inputs. They work with local variables, that are not stored at the end of the execution if they are not an output.
- Syntax:

```
function [out1, out2, ...] = funname(in1, in2, ...)
                (%insert code here)
end
```

The name of the function must have the same name in the .m file.

sine.m*

```
1   function out=sine(input)
2   %we only calculate the sine of the input array,
3   %and we add the input, taking this as the output
4
5   intermediate=sin(input);
6   out=intermediate+input;
7
8   end
```

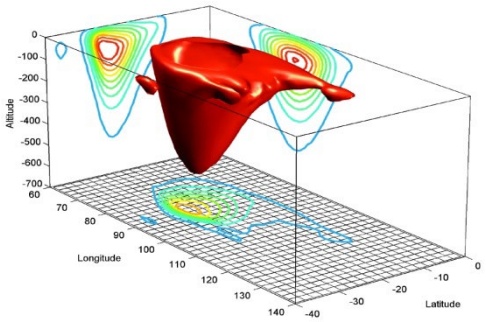
```
>> clear all
>> store_here=sine(0:0.1:pi)
```

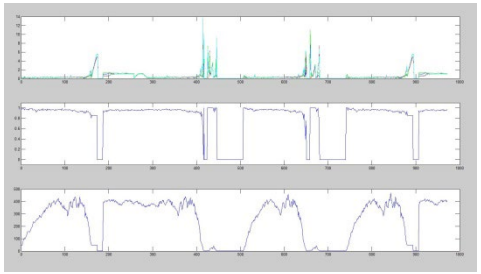
Octave Session

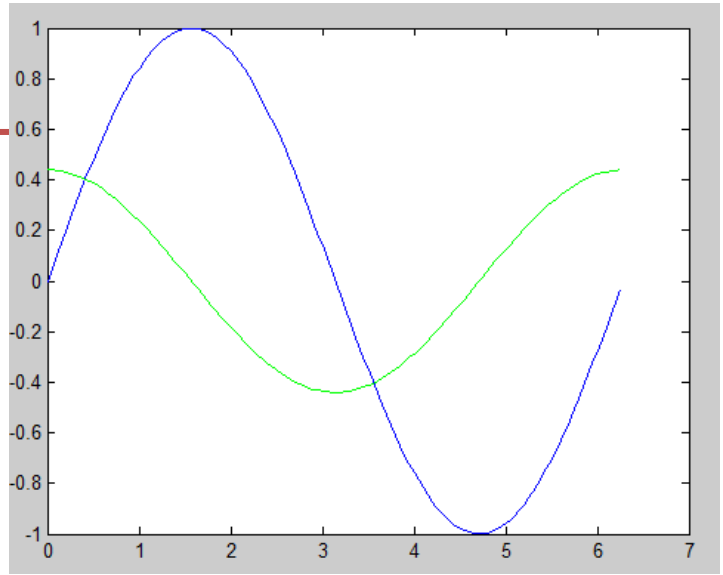
Inputs and outputs: plotting, showing, opening and saving data

- **Plotting:** From an existing array or expression, Octave can generate customizable graphs: title, axis, labels, color, background, line thickness, markers…

```
>> help plot

>> x=(0:0.05:2*pi);
>> plot(x,sin(x))
>> hold on;
>> plot (x,0.44*cos(x), 'color','g')
```







Octave Session

Inputs and outputs: plotting, showing, opening and saving data

- **Showing data on the command window and user interaction:** Visualization format, settings, and other user interaction features can be controlled:

```
format
echo
diary
pause
disp
input
```

```
>> reply = input('Do you want more? Y/N [Y]:','s');
>> help format
```

- **Showing messages on the command window:** I can include workspace variables

```
>> help fprintf
>> fprintf('This is a test to check how fprintf works');
>> fprintf('If I add inverted slash at the end the spacer jumps to the next line \n');

>> to_be_inserted=20;
>> fprintf('The variable is: %6.2f\n',to_be_inserted);
The variable is:  20.00
```

Inputs and outputs: plotting, showing, opening and saving data

- **Saving data:** We can save an existing variable from our workspace in a new file, in many different formats. We use command *save.*

```
>> help save
```

- **Loading data:** We can open an existing file located on the current path in many formats. We use command *load.*

```
>> help load
```

Saving and loading *on Octave* format

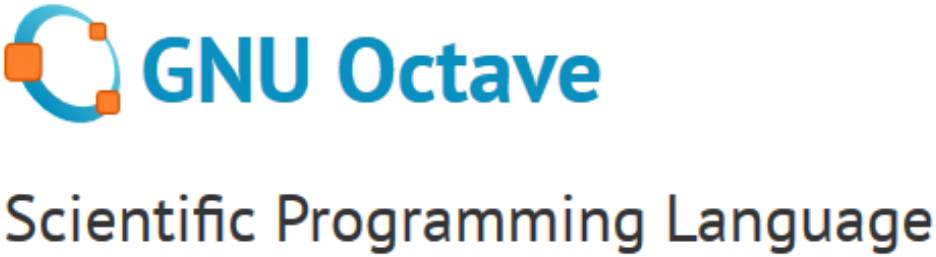
```
>> mat2=magic(6);
>> save('output','mat2')
>> clear all
>> load output
>> mat2
```

Octave Practice

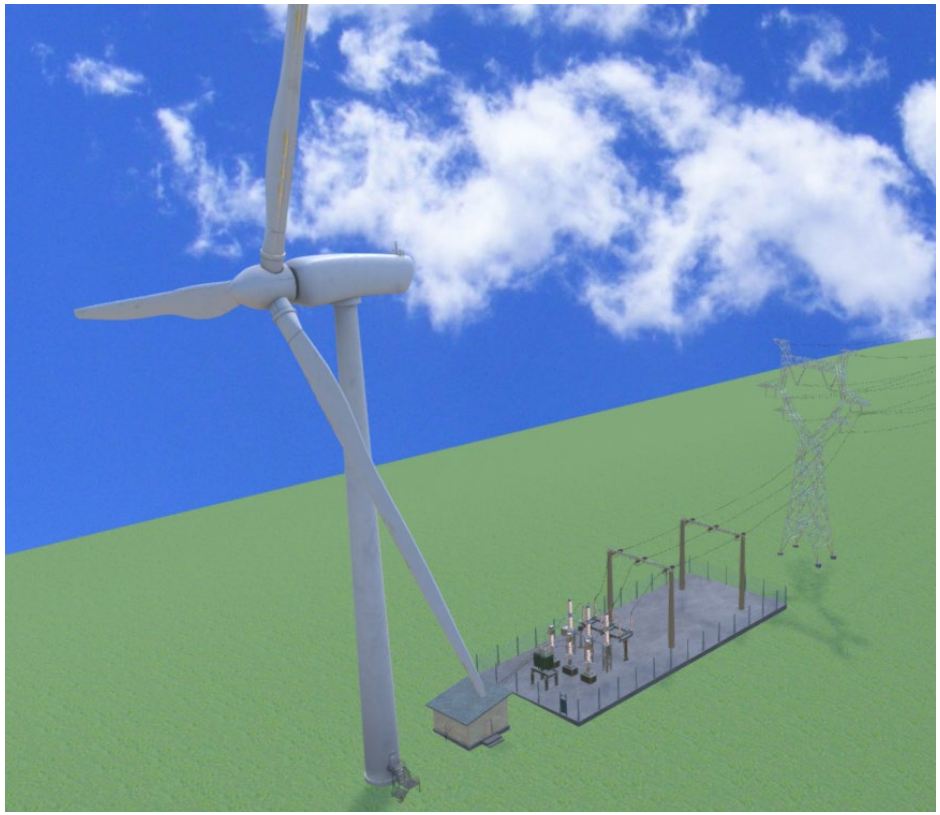
Scientific Programming Language

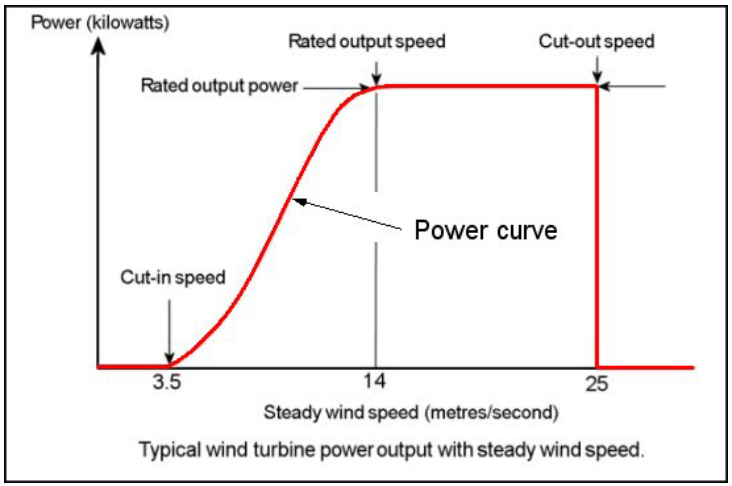
Octave Session

- Index
  - Practice 0: Wind energy



Short introduction:





Typical wind turbine power output with steady wind speed.

$$P_{wind} = C_p \cdot \frac{1}{2} \cdot \rho \cdot \pi \cdot R^2 \cdot V_{wind}^3$$

A change of 10% in wind speed causes a change of 33% in power generated

Data needed: External file (data_ej2.mat)

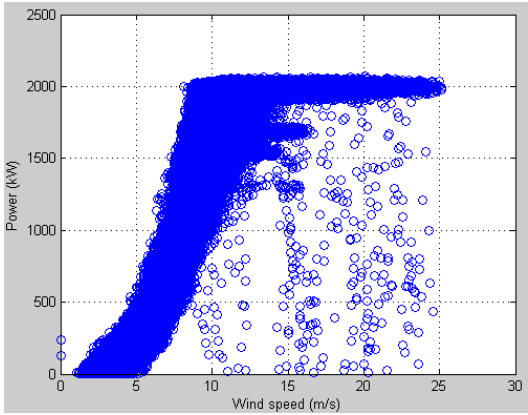
Column 1: time data. See *datenum* and *datevec* functions.

Column 2: power generated by wind turbine, WT (kW).

Column 3: wind speed in front of the WT (m/s).

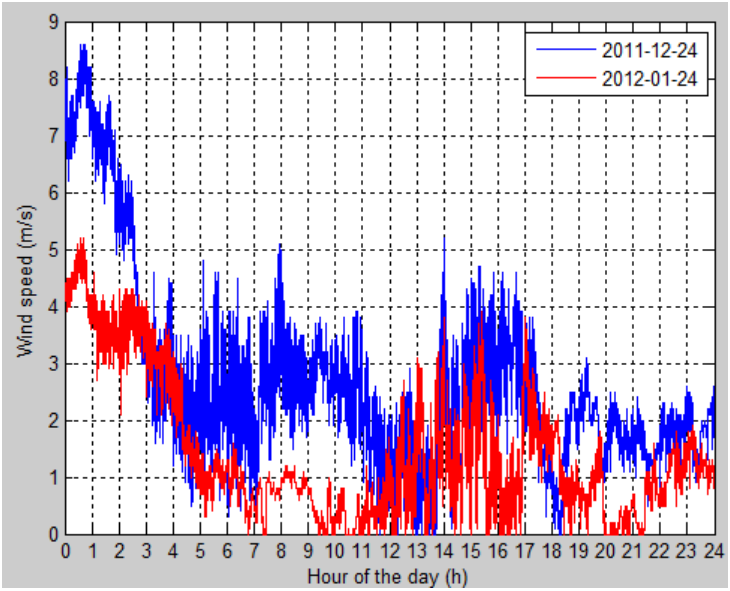
Tasks:

1. Load the data file, see *load* function.
2. When did the measurement campaign start? When did it finish?
3. Which is the sampling frequency? (samples per minute)
4. Plot the power curve of the WT: wind speed (horizontal) VS power (vertical).
5. Filter positive values of power, and build a new plot (see *find* function) →



Tasks:

6. Which is the nominal power of the WT? (see previous plots)
7. Which is the maximum wind speed recorded in the whole measurement campaign (see *max* function)? Which is the power generated by the WT at this wind speed value? Explain your results.
8. In only one figure, plot the wind speed recorded in December, 24th 2011 VS wind speed in January, 24th 2012.



Thank for your attention

Andrés Honrubia Escribano

andres.honrubia@uclm.es