



Instituto Jorge Sabato, 25 años.  
Comisión Nacional de Energía atómica.

## Modelización de Materiales 2018

# Herramientas



# Problema Ingenieril o Físico

## Preproceso

“dibujo” del problema

recinto de validez

Modelo Físico

Condiciones de contorno

## Preproceso

Matricialización

Lectura de datos

Resolución

Escritura de resultados

## Post Proceso

Mediciones Ingenieriles

Información Gráfica

Interpretación de  
resultados



# Problema Ingenieril o Físico

## Preproceso

“dibujo” del problema  
recinto de validez

Modelo Físico  
Condiciones de contorno

## Preproceso

Matricialización

Lectura de datos

Resolución

Escritura de resultados

## Post Proceso

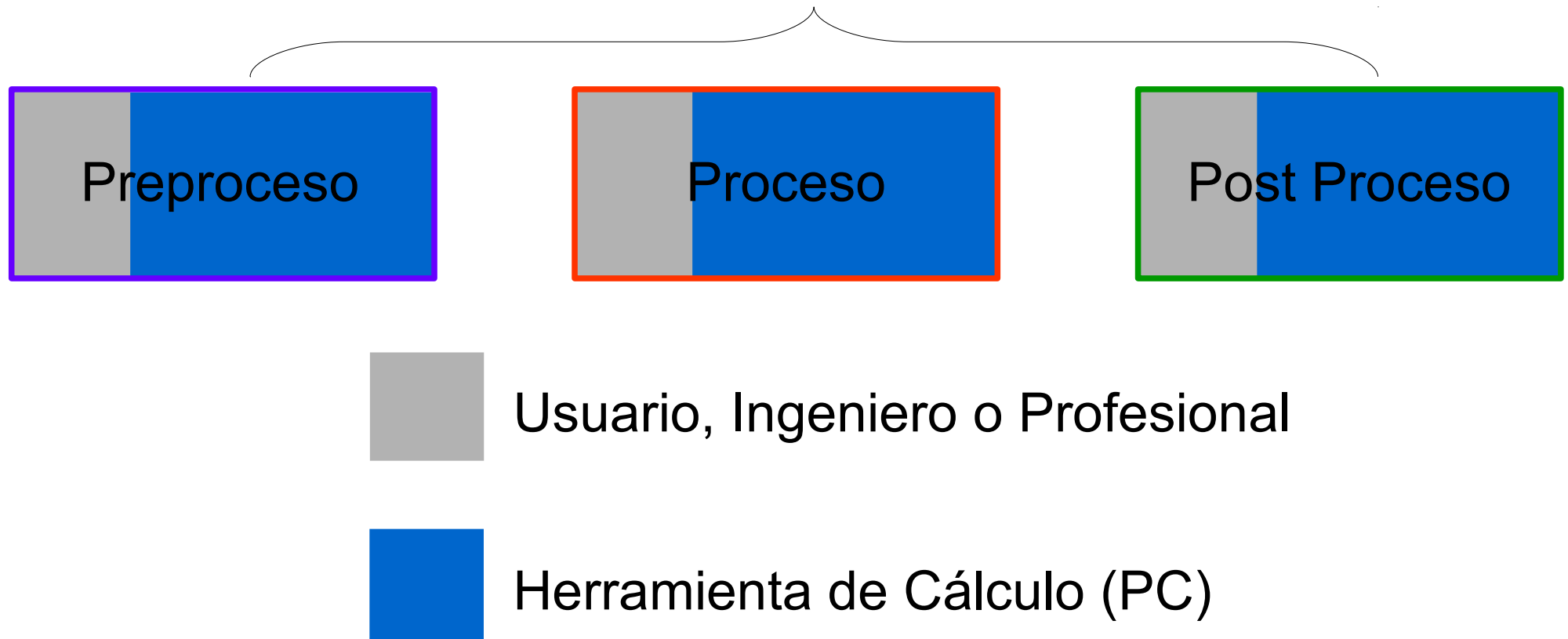
Mediciones Ingenieriles

Información Gráfica

Interpretación de  
resultados



# Problema Ingenieril o Físico





# Uso de Herramientas

Preproceso

Proceso

Post Proceso

Modelo Físico  
Condicion de  
Contorno

Matricialización

Interpretación de  
resultados

Lectura de datos  
Resolución  
Escritura de  
resultados

Mediciones

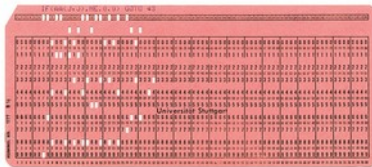
Dibujo  
Discretización

Gráficos





# Herramientas Alternativas



Fortran



Scilab



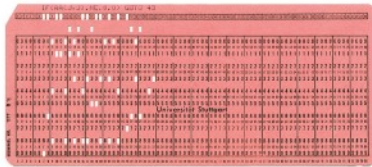
Python / Numpy / SciPy



Octave



# Herramientas Alternativas



Fortran



Scilab



Python / Numpy / SciPy



Octave



Instituto Jorge Sabato, 25 años.  
Comisión Nacional de Energía atómica.

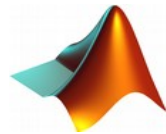
# Modelización de Materiales 2018

## Matlab

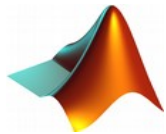




# Matlab: Introducción



MATrix LABoratory

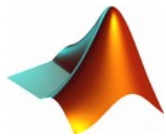


Multiplataforma (JVM)



MathWorks

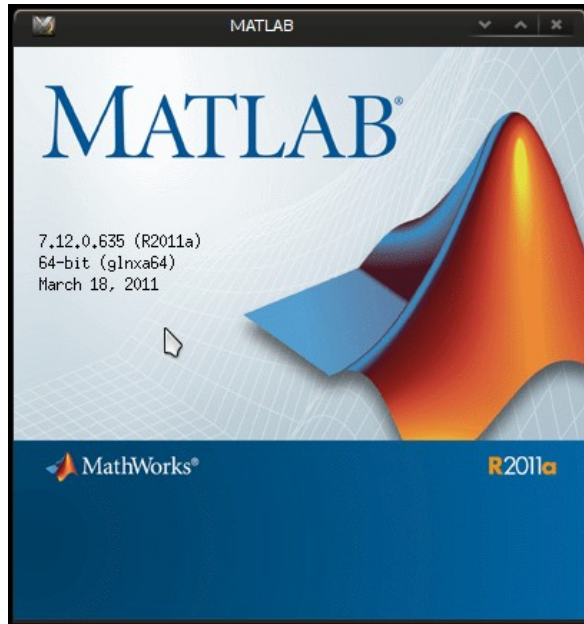
<http://www.mathworks.com/products/matlab/>



Lenguaje de programación /  
consola programable /  
Scripts

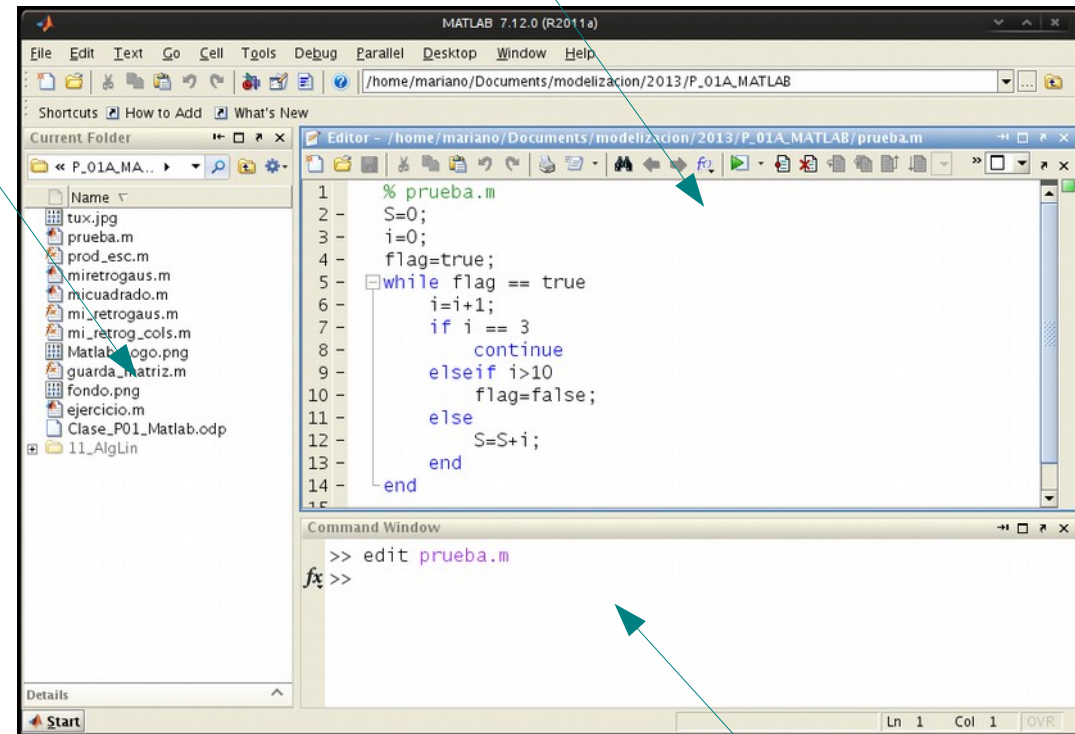


# Escritorio y consola



Panel Lateral

Editor



Ventana de Commandos



# Línea de Comandos

Asignación  
de variables

```
>> A = [ 1 2 3 ; 4 5 6 ; 7 8 9 ];
```

```
>> A
```

```
A =
```

1	2	3
4	5	6
7	8	9

Transponer

```
>> A'
```

```
ans =
```

1	4	7
2	5	8
3	6	9

prompt

```
>>
```



# Indexación de vectores

Rango de filas,  
Todas las columnas

```
>> A(1:2,:)
ans =
```

1	2	3
4	5	6

Vector de índices

```
>> A([1 3],:)
```

```
ans =
```

1	2	3
7	8	9

```
>> A([1 3],[2 3])
```

```
ans =
```

2	3
8	9

```
>> |
```



# Control de Flujo del programa

for

```
Editor - /home/mariano/Documents/modelizacion/2013/P_01A_MATLAB/prueba.m*
1 % Modelización de Materiales 2013
2 % Introducción a Herramientas de cálculo.
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 S=0;
5 for i = 1:10
6     S=S+i;
7 end
8
9 disp(S)
```

while

```
Editor - /home/mariano/Documents/modelizacion/2013/P_01A_MATLAB/prueba.m
1 % Modelización 2013
2 % Introducción a Herramientas de cálculo.
3 S=0;
4 i=0;
5 while i<10
6     i=i+1;
7     S=S+i;
8 end
9 disp(S)
10
```



# Control de Flujo del Programa

## if

```
Editor - /home/mariano/Documents/modelizaci...
1 % prueba.m
2 S=0;
3 i=0;
4 flag=true;
5 while flag == true
6     i=i+1;
7     S=S+i;
8
9     if i == 10
10         flag=false;
11     end
12
13 end
14
15 disp(S);
```

```
Editor - /home/mariano/Documents/modelizaci...
1 % prueba.m
2 S=0;
3 i=0;
4 flag=true;
5 while flag == true
6     i=i+1;
7     S=S+i;
8
9     if i == 10
10         break
11     end
12
13 end
14
15 disp(S);
```

```
Command Window
>> prueba
55

fx >>
```

```
Editor - /home/mariano/Docume...
1 % prueba.m
2 S=0;
3 i=0;
4 flag=true;
5 while flag == true
6     i=i+1;
7     if i == 3
8         continue
9     elseif i>10
10         flag=false;
11     else
12         S=S+i;
13     end
14 end
15
16 disp(S);
```

```
Command Window
>> prueba
52

fx >>
```



# Funciones

```
Command Window
>> edit prod_esc.m
fx >> |
```

```
Editor - /home/mariano/Documents/modelizacion/2013/P_01A_MA...
1 % modelizacion de materiales y
2 % procesos 2013
3 % Funciones
4 function [ y , z ] = prod_esc(x1,x2)
5
6 y = x1 * x2';
7
8 norm1 = sqrt( x1*x1' );
9 norm2 = norm(x2);
10
11 z = y / ( norm1 * norm2 );
```

```
Command Window
>> clear all
>> a = [ 1 1 ] ;
>> b = [ 0 1 ] ;
>> [ adotb , cosab ] = prod_esc(a,b)

y =

     1

adotb =

     1

cosab =

     0.7071

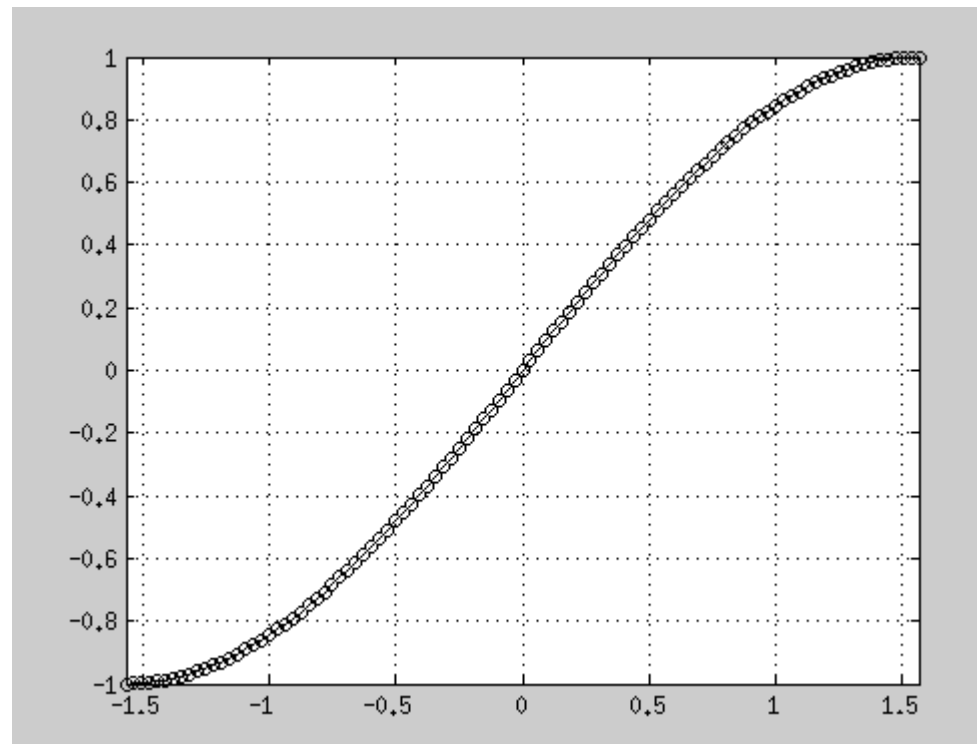
fx >> |
```



# Gráficos

Command Window

```
>> x=[ -pi/2 : pi/100 : pi / 2 ] ;  
>> y = sin(x);  
>> plot(x,y,'o-k');  
>> grid on  
>> xlim( [ -pi/2 pi/2 ] );  
fx >>
```







# Entrada – Salida (abreviado)

Abrir archivo,  
reescribir o agregar:

Guardar un tipo de  
Datos en el medio de  
Una linea:

Concatenar

Imprimir en archivo  
Con retorno de carro!

```
1 %Modelizacion de Materiales y Procesos 2013
2 function fid=guarda_matriz(filename,step,matriz)
3
4 if step == 1
5     fid = fopen(filename,'w');
6 else
7     fid = fopen(filename,'a');
8 end
9 line='';
10
11 [ n m ]=size(matriz);
12
13 fprintf(fid,'\n \n Matriz de %d x %d en el paso %d \n',n,n,step);
14 fprintf(fid,'===== \n');
15
16 for i = 1:n
17     for j=1:m
18         line=[line,num2str(matriz(i,j),' %10.6f ')];
19     end
20     fprintf(fid,[line,'\n']);
21 end
22 fclose(fid);
23 return;
24
```



# Documentación

```
Editor - prod_escm Command Window
>> help quiver
QUIVER Quiver plot.
QUIVER(X,Y,U,V) plots velocity vectors as arrows with components (u,v)
at the points (x,y). The matrices X,Y,U,V must all be the same size
and contain corresponding position and velocity components (X and Y
can also be vectors to specify a uniform grid). QUIVER automatically
scales the arrows to fit within the grid.

QUIVER(U,V) plots velocity vectors at equally spaced points in
the x-y plane.

QUIVER(U,V,S) or QUIVER(X,Y,U,V,S) automatically scales the
arrows to fit within the grid and then stretches them by S. Use
S=0 to plot the arrows without the automatic scaling.

QUIVER(...,LINESPEC) uses the plot linestyle specified for
the velocity vectors. Any marker in LINESPEC is drawn at the base
instead of an arrow on the tip. Use a marker of '.' to specify
no marker at all. See PLOT for other possibilities.

QUIVER(...,'filled') fills any markers specified.

QUIVER(AX,...) plots into AX instead of GCA.

H = QUIVER(...) returns a quivergroup handle.

Example:
[x,y] = meshgrid(-2:2:2,-1:15:1);
z = x .* exp(-x.^2 - y.^2); [px,py] = gradient(z,2,15);
contour(x,y,z), hold on
quiver(x,y,px,py), hold off, axis image


See also feather, quiver3, plot.

Reference page in Help browser
doc quiver

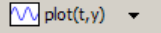
fx >> |
```

```
Command Window
>> doc quiver
fx >>
```

**quiver**  
Quiver or velocity plot



**Alternatives**

To graph selected variables, use the Plot Selector  in the Workspace Browser, or use the Figure Palette Plot Catalog. Manipulate graphs in *plot edit* mode with the Property Editor. For details, see [Plotting Tools — Interactive Plotting](#) in the MATLAB Graphics documentation and [Creating Graphics from the Workspace Browser](#) in the MATLAB Desktop Tools documentation.

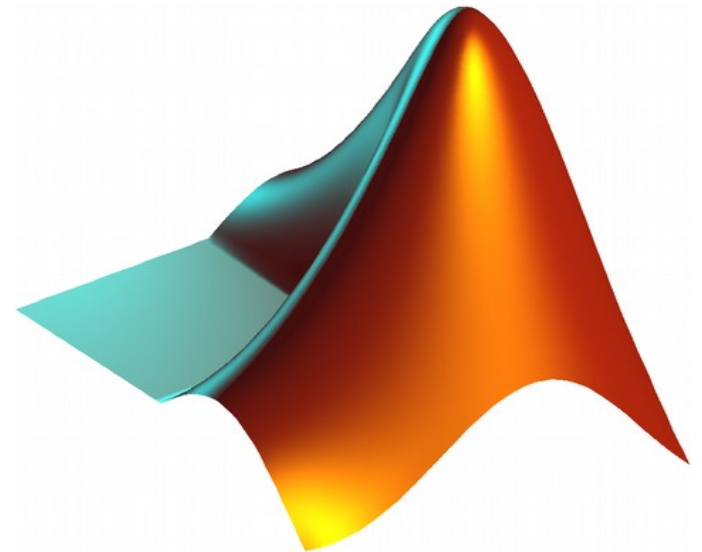
**Syntax**

```
quiver(x,y,u,v)
quiver(u,v)
quiver(...,scale)
quiver(...,LineStyle)
quiver(...,LineStyle,'filled')
quiver(...,'PropertyName',PropertyValue,...)
quiver(axes_handle,...)
h = quiver(...)
```

**Description**



# Ejercicio





Instituto Jorge Sabato, 25 años.  
Comisión Nacional de Energía atómica.

Modelización de Materiales 2018

# Herramientas



## Problema Ingenieril o Físico

### Preproceso

“dibujo” del problema

recinto de validez

Modelo Físico

Condiciones de contorno

### Preproceso

Matricialización

Lectura de datos

Resolución

Escritura de resultados

### Post Proceso

Mediciones Ingenieriles

Información Gráfica

Interpretación de  
resultados



## Problema Ingenieril o Físico

### Preproceso

“dibujo” del problema

recinto de validez

Modelo Físico

Condiciones de contorno

### Preproceso

Matricialización

Lectura de datos

Resolución

Escritura de resultados

### Post Proceso

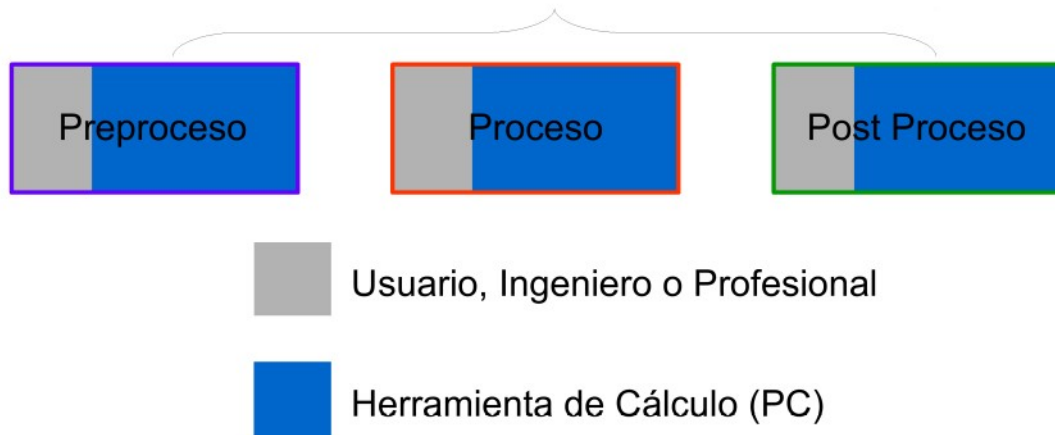
Mediciones Ingenieriles

Información Gráfica

Interpretación de  
resultados

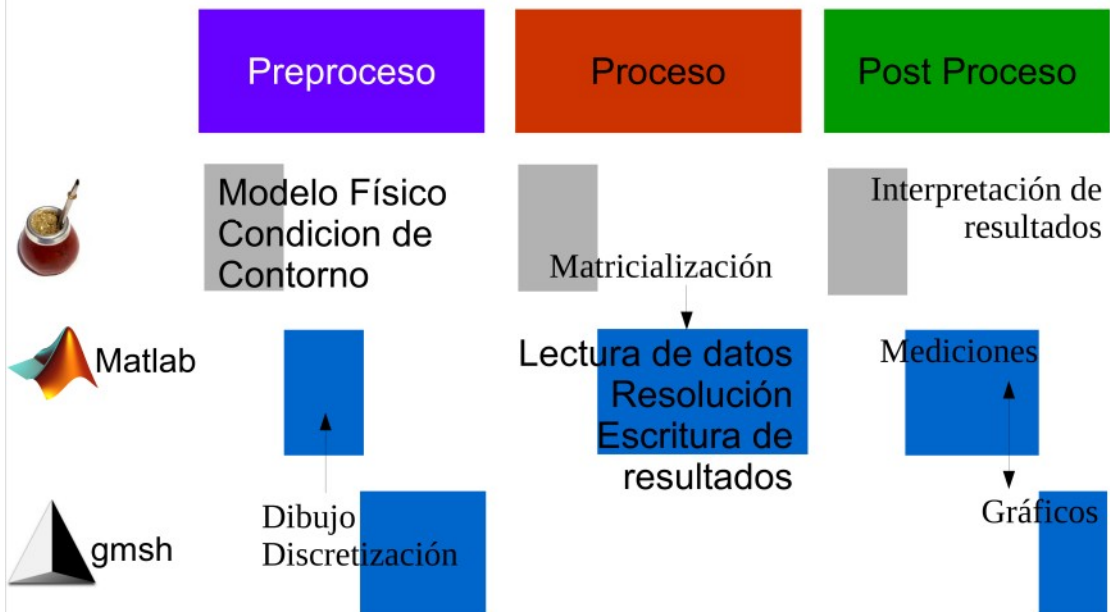


## Problema Ingenieril o Físico





## Uso de Herramientas



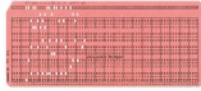




## Herramientas Alternativas



Matlab



Fortran



Scilab



Python / Numpy / SciPy



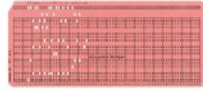
Octave

Bien entonces, las herramientas de cómputo que vamos a usar oficialmente es el matlab, las clases van a estar orientadas a esto.

Sin Embargo, Pueden elegir entre otras alternativas. Por ejemplo, históricamente siempre hay gente que elije tener una experiencia en Linux / Fortran , el año pasado alguien hizo todas las prácticas en Octave (versión libre de Matlab ) , pero también se puede pensar en cosas nuevas como Python o Scilab, incluso c, c++ , o cualquier otra cosa que nos quieran enseñar.



## Herramientas Alternativas



Fortran



Scilab



Python / Numpy / SciPy



Octave

Sin embargo, de nuevo, la herramienta oficial de la materia es Matlab, y las clases van a mostrar ejemplos en este lenguaje. Sin embargo, por supuesto cualquier herramienta que elijan va a tener soporte de nuestra parte, siempre que esté a nuestro alcance. usar oficialmente es el matlab, las clases van a estar orientadas a esto.



Instituto Jorge Sabato, 25 años.  
Comisión Nacional de Energía atómica.

## Modelización de Materiales 2018

# Matlab



## Matlab: Introducción



MATrix LABoratory



Multiplataforma (JVM)



MathWorks

<http://www.mathworks.com/products/matlab/>

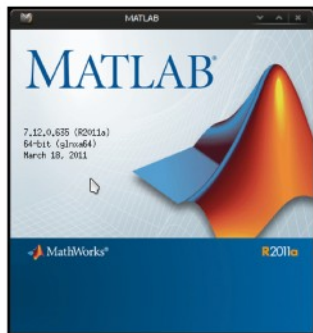


Lenguaje de programación /  
consola programable /  
Scripts

Matlab es un acrónimo de Matrix Laboratory, lo cual nos viene como anillo al dedo porque como vamos a ver la filosofía de esta materia es que las ecuaciones diferenciales se van a expresar en forma matricial. La interfaz está basada en java de manera que es multiplataforma. El programa ofrece un lenguaje de programación no compilable, es decir que los programas se pueden correr directamente desde la ventanita (rapidamente) , pero a demas ofrece la posibilidad de guardar la secuencia de comandos en scripts, para repetir ejecucion y compartir programas.

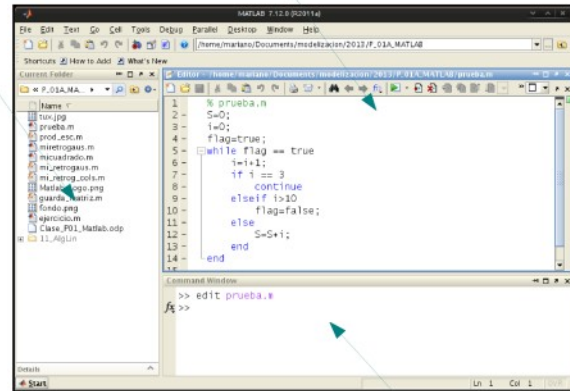


## Escritorio y consola



Panel Lateral

Editor



Ventana de Commandos

Cuando abrimos matlab tenemos una vista mas o menos así: un panel lateral configurable, pero principalmente una línea de comandos, y opcionalmente un editor.



## Línea de Comandos

Asignación de variables → 

```
>> A = [ 1 2 3 ; 4 5 6 ; 7 8 9 ];
```

```
>> A
```

```
A =
```

1	2	3
4	5	6
7	8	9

Transponer → 

```
>> A'
```

```
ans =
```

1	4	7
2	5	8
3	6	9

prompt → 

```
>>
```

Conceptos básicos, así se escribe una matriz, sobre la cual se pueden hacer todo tipo de operaciones por ejemplo transponer.



## Indexación de vectores

Rango de filas,  
Todas las columnas

```
>> A(1:2,:)
ans =
     1     2     3
     4     5     6
```

Vector de índices

```
>> A([1 3],:)
ans =
     1     2     3
     7     8     9
```

```
>> A([1 3],[2 3])
ans =
     2     3
     8     9
```

```
>> |
```

Una cosa que se puede hacer sobre las matrices es tomar alguna submatriz o corte ( slice ) de esta manera, simplemente haciendo uso de vectores de índices, que pueden ser una lista de índices o un rango de índices. Fijemse que implícitamente estamos viendo esta variable “ans” que guarda el resultado de la última operación, siempre que no se halla indicado otra variable para guardar el resultado.



## Control de Flujo del programa

for

```
Editor - /home/mariano/Documents/modelizacion/2013/P_01A.MATLAB/prueba.m
1 % Modelización de Materiales 2013
2 % Introducción a Herramientas de cálculo.
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 S=0;
5 for i = 1:10
6     S=S+i;
7 end
8
9 disp(S)
```

while

```
Editor - /home/mariano/Documents/modelizacion/2013/P_01A.MATLAB/prueba.m
1 % Modelización 2013
2 % Introducción a Herramientas de cálculo.
3 S=0;
4 i=0;
5 while i<10
6     i=i+1;
7     S=S+i;
8 end
9 disp(S)
10
```

Como en todos los lenguajes de programación se puede hacer uso de los lazos (loops)

En for hay que indicar un rango de una variable sobre el cual se va a operar , entonces por ejemplo aquí dice: para i , que toma valores desde 1 hasta 10, hacer esta operación, el end indica que ya se indicaron todos los pasos .

En cambio while indica alguna condición lógica que marcará el fin del lazo. Por ejemplo aquí dice: Mientras i sea menor que 10, hacer todo esto.

Ver que en for se incrementa automáticamente i, en while hay que hacerlo a mano.





## Control de Flujo del Programa

if

The image displays three MATLAB script windows and two Command Windows, illustrating different flow control logic for a program named 'prueba.m'.

**Script 1 (Left):**

```
1 % prueba.m
2 S=0;
3 i=0;
4 flag=true;
5 while flag == true
6     i=i+1;
7     S=S+i;
8
9     if i == 10
10        flag=false;
11    end
12
13 end
14
15 disp(S);
```

**Command Window 1 (Bottom Left):**

```
>> prueba
55
fx >>
```

**Script 2 (Middle):**

```
1 % prueba.m
2 S=0;
3 i=0;
4 flag=true;
5 while flag == true
6     i=i+1;
7     S=S+i;
8
9     if i == 10
10        break
11    end
12
13 end
14
15 disp(S);
```

**Command Window 2 (Bottom Middle):**

```
>> prueba
55
fx >>
```

**Script 3 (Right):**

```
1 % prueba.m
2 S=0;
3 i=0;
4 flag=true;
5 while flag == true
6     i=i+1;
7     if i == 3
8         continue
9     elseif i>10
10        flag=false;
11    else
12        S=S+i;
13    end
14 end
15
16 disp(S);
```

**Command Window 3 (Bottom Right):**

```
>> prueba
52
fx >>
```

EN Matlab también hay condicionales, y cualquier problema se puede resolver con cantidad suficiente de if, for , while acomodados en forma mas o menos inteligente.

El if tiene esta estructura. Por ejemplo aquí:  
Si i es igual a 10, hacer esto. Fijarse en los comandoos break y continue, y se puede usar la estrcutrua if, elseif, else, end.



## Funciones

```
Command Window
>> edit prod_esc.m
fx >> |
```

```
Editor - /home/mariano/Documents/modelizacion/2013/P_01A_MA...
1 % modelizacion de materiales y
2 % procesos 2013
3 % Funciones
4 function [ y , z ] = prod_esc(x1,x2)
5
6 y = x1 * x2';
7
8 norm1 = sqrt( x1*x1' );
9 norm2 = norm(x2);
10
11 z = y / ( norm1 * norm2 );
```

```
Command Window
>> clear all
>> a = [ 1 1 ] ;
>> b = [ 0 1 ] ;
>> [ adotb , cosab ] = prod_esc(a,b)

y =

     1

adotb =

     1

cosab =

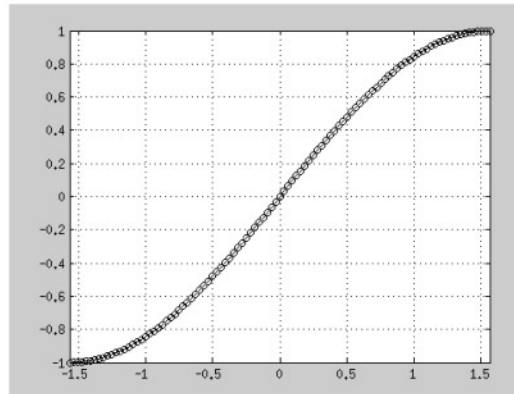
    0.7071

fx >> |
```

Las funciones en matlab son una especie de unidad operativa a la cual le doy una serie de parámetros, y me devuelve una serie de respuestas. Se declaran preferentemente en archivos separados. Cuando se ejecutan por ejemplo desde la línea de comandos, hay que darles los argumentos en el orden en el que la función los toma en la declaración, no necesariamente con el mismo nombre, y me va a devolver las variables de salida en el orden declarado, no necesariamente con el mismo nombre.



```
Command Window
>> x=[ -pi/2 : pi/100 : pi / 2 ] ;
>> y = sin(x);
>> plot(x,y,'o-k');
>> grid on
>> xlim( [ -pi/2 pi/2 ] );
fx >>
```



Bueno, gráficos:

El comando plot grafica siempre un vector en función de otro, pudiendo poner opciones de estilo como qué símbolo quieren , color y ese tipo de cosas.



## Entrada – Salida (abreviado)

Abrir archivo,  
reescribir o agregar:

Guardar un tipo de  
Datos en el medio de  
Una línea:

Concatenar

Imprimir en archivo  
Con retorno de carro!

```
1 %Modelización de Materiales y Procesos 2013
2 function fid=guarda_matriz(filename,step,matriz)
3
4 if step == 1
5     fid = fopen(filename,'w');
6 else
7     fid = fopen(filename,'a');
8 end
9 line='';
10
11 [ n m ]=size(matriz);
12
13 fprintf(fid,'\n \n Matriz de %d x %d en el paso %d \n',n,n,step);
14 fprintf(fid,'===== \n');
15
16 for i = 1:n
17     for j=1:m
18         line=[line,num2str(matriz(i,j),' %10.6f ')];
19     end
20     fprintf(fid,[line,'\n']);
21 end
22 fclose(fid);
23 return;
24
```

Hay un aspecto sobre el que vamos a trabajar mucho que es la interacción de matlab con otros programas, y para eso tenemos que aprender, tarde o temprano, a leer parámetros desde un archivo de texto y a escribir resultados en otro archivo de texto. Para eso hay cuatro comandos básicos. Los dos primeros son los comandos que abren o cierran los archivos: open y close. Una vez abierto el archivo, guardar algún contenido, línea por línea supngamos, para eso se usa el comando fprintf.

Para leer de un archivo se usa otro comando que es fgetl. Por ahora nos quedamos con esto y ya vamos a ver en otra presentación más detalle sobre I/O, ya que va a ser importante para nosotros.



# Documentación

```
Command Window
>> help quiver
QUIVER Quiver plot.
QUIVER(X,Y,U,V) plots velocity vectors as arrows with components (u,v)
at the points (x,y). The matrices X,Y,U,V must all be the same size
and contain corresponding position and velocity components (X and Y
can also be vectors to specify a uniform grid). QUIVER automatically
scales the arrows to fit within the grid.

QUIVER(U,V) plots velocity vectors at equally spaced points in
the x-y plane.

QUIVER(U,V,S) or QUIVER(X,Y,U,V,S) automatically scales the
arrows to fit within the grid and then stretches them by S. Use
S=0 to plot the arrows without the automatic scaling.

QUIVER(...,LINESPEC) uses the plot linestyle specified for
the velocity vectors. Any marker in LINESPEC is drawn at the base
instead of an arrow on the tip. Use a marker of '.' to specify
no marker at all. See PLOT for other possibilities.

QUIVER(...,'filled') fills any markers specified.

QUIVER(Ax,...) plots into AX instead of GCA.

H = QUIVER(...) returns a quivergroup handle.

Example:
[x,y] = meshgrid(21:212,-1:151);
z = x.*exp(-x/2 - y/2); [m,py] = gradient(z,2,15);
contour(x,y,z), hold on
quiver(x,y,m,py), hold off, axis image

See also feather, quiver3, plot.

Reference page in Help browser
doc quiver

fx >> |
```

```
Command Window
>> doc quiver
fx >>
```

The image shows the MATLAB Help browser window. The left pane contains a table of contents with the following structure:

- MATLAB
  - Getting Started
  - User's Guide
  - Functions
  - Desktop Tools and Development
  - Data Import and Export
  - Mathematics
  - Data Analysis
  - Programming and Data Types
  - Object-Oriented Programming
  - Graphics
    - Basic Plots and Graphs
    - Plotting Tools
    - Annotating Plots
    - Specialized Plotting
      - Area, Bar, and Pie Plots
      - Contour Plots
      - Direction and Velocity Plots
        - fx comet - 2-D comet
        - fx comet3 - 3-D comet
        - fx compass - Plot arrow
        - fx feather - Plot velocity
        - fx feather3 - 3-D quiver
        - fx quiver - 2-D quiver
        - fx quiver3 - 3-D quiver
      - Discrete Data Plots

The right pane displays the documentation for the **quiver** function. It includes a title 'quiver' with a subtitle 'Quiver or velocity plot', a small icon of a quiver plot, and sections for 'Alternatives', 'Syntax', and 'Description'. The 'Syntax' section lists the following function signatures:

```
quiver(x,y,u,v)
quiver(u,v)
quiver(...,scale)
quiver(...,LineStyle)
quiver(...,LineStyle,'filled')
quiver(...,'PropertyName',PropertyValue,...)
quiver(axes_handle,...)
h = quiver(...)
```



## Ejercicio

