Programming in matlab

Introducction to Synthetic Biology

E Navarro
A Montagud
P Fernandez de Cordoba
JF Urchueguía

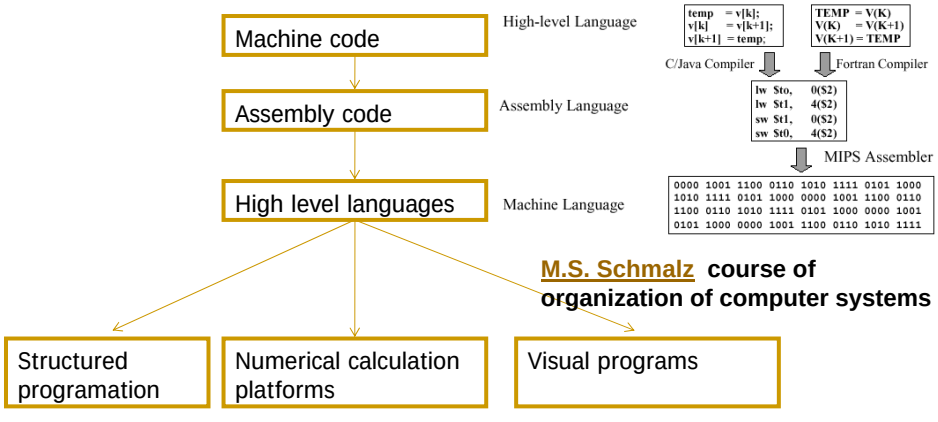
InterTech

Overview

- Introduction to programming languages
- Basic matlab operations
- Programming in matlab
- Solving ODE with matlab

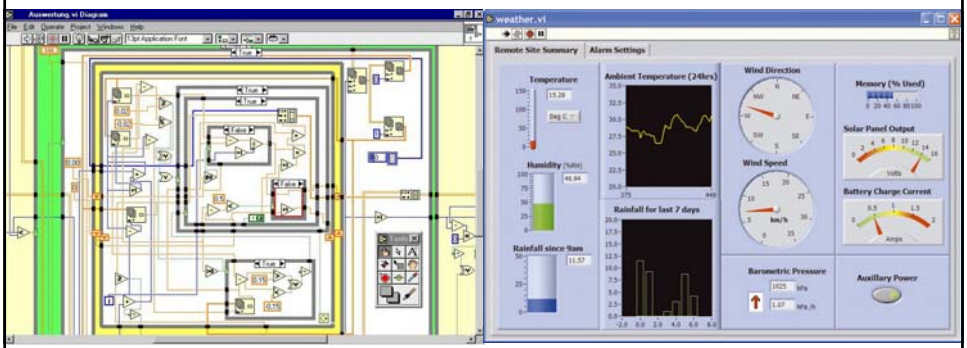
Programming languages

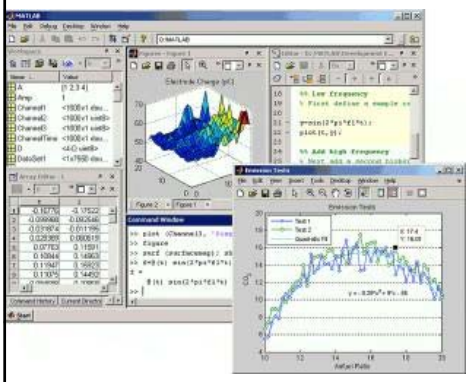
- They are designed to allow comunication with devices, allowing them to perform different opperations.
- First programming languages were predate to the first computers. They were punch cards according the desired function.
- After the invention of computers and the use of them to perform mathematical analysis, a lot of development has been obtained to make easier the programming for the user

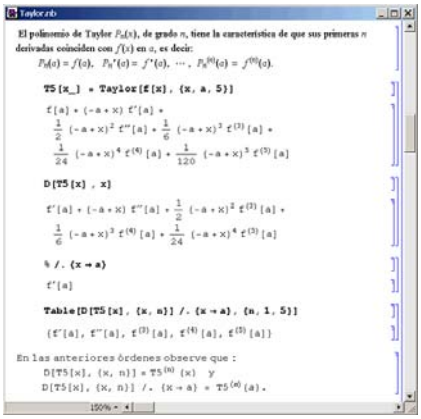


High level programming

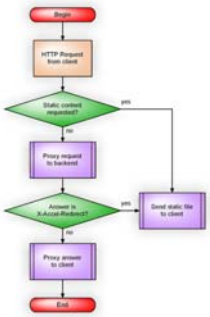
```
1         PROGRAM PRINCIPAL
2           PARAMETER (TAMMÁX=99)
3           REAL A(TAMMAX)
4     10    READ (5,100,END=999) K
5    100    FORMAT(I5)
6             IF (K.LE.O.OR K.GT.TAMMÁX) STOP
7             READ *,(A(I),I=1,K)
8             PRINT *,(A(I),I=1,K)
9             PRINT * ,, 'SUMA=', SUM(A,K)
10            GO TO 10
11    99    PRINT * ,, "Todo listo"
12          STOP
13          END
14  SUBPROGRAMA DE SUMATORIA EN C
15        FUNCTION SUM(V,N)
16          REAL :: V(N) ! Declaración de estilo nuevo
17          SUM = 0.0
18          DO 20 I = 1,N
19            SUM = SUM + V(I)
20    20      CONTINUE
21          RETURN
22          END
```







- Algorithm: Previous clear scheme of flow and operations in order to develop a task



Matlab

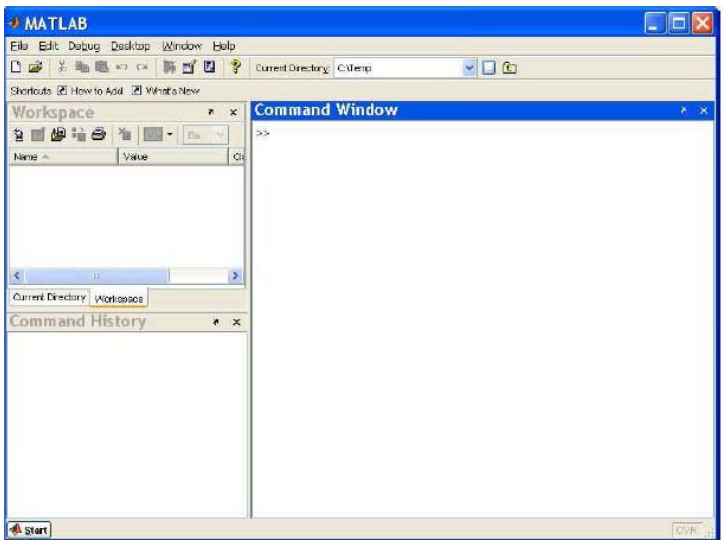
- Introduction.
- Help
- Matrix operations
- Matrix functions.
- Graphycs.
- Programming.
- Numerical analysis

- ¿What is Matlab?, MATrix LABoratory
  - MATLAB is a program for performing numerical calculations with matrix and vectors.
  - As a particular case it can perform operations with escalars (reals and complex).
  - It can also perform graphical representations of the results and nowadays have many useful applications.

Matlab have several packages not included in the standard version which extend its capabilities:

- Optimization toolbox
- Statistical toolbox
- Simbiology
- Simbolic math toolbox
- And the like...

http://www.mathworks.com/products/product_listing/index.html



The basic Matlab elements are: constants, variables, operations, expresions and functions.

Numerical constants:

- Integers: 2 35 -48
- Reals: 2. -35.2 48.45
  - 16 significatives numbers
  - [2.2250e-308 1.7e+308].
- Complex numbers: 2+3i 4*j i,j=$(-1)^{½}$

Elementary operators:

addition: + product: * Exponential: ^

Substraction: - Division: /

Variable: it is the brand which identifies a part of the memory.

Matlab makes differences between capitals and no capital letters

To define a variable:

>>a=3.2 ; b=53.65

To see the defined variables:

>> who

To remove a variable

>> clear variable1 variable2

If a variable is between ‘’ it is a text variable

>>c=‘matlab is nice’

Numerical expresions: A set of functions and variables already related by aritmetic operators.

Format: by default matlab uses short format but

- >> format long (14 cifras significativas)
- >> format short (5 cifras significativas)
- >> format short e (notación exponencial)
- >> format long e (notación exponencial)
- >> format rat (aproximación racional)

Predefined variables:

i = $(-1)^{½}$ pi = π Inf= ∞ NaN= Undefined solution
date = date
rand = Random numbers between 0 and 1

Functions:

name(argument)

- sqrt (x) square roor
- abs(x) module of x
- conj(z) conjugated number
- real(z), imag(z)
- exp(x)
- sin(x) asin(x) [$-\pi/2$ $\pi/2$] cos(x) acos(x) [0 π] tan(x) atan(x) [$-\pi/2$ $\pi/2$] angle(z) log(x), log10(x)
- rats(x), rem(x,y), round(x), sign(x)

Help comands:

- help
- what files .m and .mat of the directory
- dir files of the present directory

Basic comands

- type File_name Show the file content
- delete File_name
- cd
- pwd show the directory
- clr clean the memory of the variables
- which File_name
- startup.m boot file.

Saving in a file the commands of a sesion
>> diary File_name
...
>> diary off

VECTORS Y MATRIX

- Basis of matlab (Matrix Laboratory).

```
» A=[1 3 5; 6 9 2; 4 8 7]

A =

     1     3     5
     6     9     2
     4     8     7

» det(A)

ans =

     5
```

```
» A^2+3*A

ans =

    42    79    61
    86   142    68
    92   164   106
```

File vectors: The elements are separated by spacebars or comma
>> v =[2 3 4]

Vectores columna: The elements are separated by intro or by ";"

>> w =[2;3;4;7;9;8]

>> length(vector name) Dimension of the vector.

Generating vectors:
• Specifying the increment of its components v=a:h:b;
• specifying it dimension linspace(a,b,n); The increment is given by k=(b-a)/(n-1)
• Components separated using a logarithmic logspace(a,b,n) 10^a and 10^b.
•By hand v(3)=4

Operating with scalars:
v+k
v-k
v*k
v/k
k./v
v.^k
k.^v

Operating with vectors:
v+w
v-w
v.*w
v./w
v.^w

scalar product of vectors
v*w

Some specific functions:

sum(v)
prod(v)
v` (files ↔ columns)
dot(v,w) scalar product
cross(v,w) vectorial product
max(v)

Matrix

To define a matrix it is not necessary to define its size

Example of matrix definition:
» A=[1 2 3; 4 5 6; 7 8 9]

The response is
A =
1 2 3
4 5 6
7 8 9

» Inv()
» A'

To have access to the vector elements we write x(3);
To have access to the elements of a matrix A(1,2) or A(4)

Matrix operators:
+
–
*
'
^
\
/
.* product element by element
./ y .\ división element by element
.^ power element by element

Functions with matrix:

- diag(A)
- sum(diag(A))
- diag(A,k)
- norm(A)
- det(A)
- full(A)
- find(a)
- inv(A)
- rank(A)
- eig(A), [v,D]=eig(A)

Generating matrix:

- zeros(n,m)
- ones(n,m)
- eye(n,m)
- rand(n,m)
- diag(v), diag(v,k)
- sparse(i,j,c,m,n)

POLYNOMIALS

The polynomials are represented by a file vector of dimension n+1 being n the degree of the polynomia

Having the polynomia

x^3+2x

it is represented by

>> pol1=[1 0 2 0]

>>root=roots(pol1) (give the root)

A polynomia can be rebuilt

```
>> p=poly(raices)
```

Example 1:

```
pol2=[2 4 0 1]; % Definition of the polynomia2x^3+4x^2+1
raices=roots(pol2) % root calculation
pol2_n=poly(raices) % rebuilt polynomia
```

Example 2:

```
A=[1 2 3 ; 2 3 4; 4 2 5]; p=poly(A) % Char pol
roots(p) % eigenvalues of A
```

To calculate the value of a polynomia:

```
>>y=polyval(p,x)
```

To multiply or divide two polynomia:

```
>>conv(p1,p2)
>>deconv(p1,p2)
>>[p4,r] = deconv(p3,p2)
>>[r,p,k] =residue(p1,p2)
```

$$\frac{p1(x)}{p2(x)} = \frac{r(1)}{x - p(1)} + \cdots + \frac{r(n)}{x - p(n)} + k(x)$$

```
>>polyder(p)
```

Linear Equations

Ax=b

Compatibility of a system:

>>rank(A)-rank([A,b])

Example:
x=ones(4,4);v=[2 2 2 2];y=x+diag(v)
rank(y)

3	1	1	1
1	3	1	1
1	1	3	1
1	1	1	3

Ax=b

If we have a square matrix:

>>Ainv=inv(A)
>>x=Ainv*b
Or
>>x=A\b

GRAPHYCS

These functions represent a set of data:

- plot(a,b,M)
- loglog(a,b,M)
- semilogx(a,b,M)
- semilogy(a,b,M)

To plot functions introduced as key-character chain :
•Fplot(f,[a b],M)

To represent symbolic expressions in a plane:
•Ezplot(f,[a b])

Numerical Analysis

To develop the numerical analysis of functions, matlab has predefined several functions and packages that can be used in an easy way.

Local minimum of a function

min=fminbnd('funcion',a,b,opciones)

Example:
Calculus of the local $f(x)=3x^4-4x^3$ in [-1 2]

fminbnd('3*x^4-4*x^3',-1,2)

fminbnd('-(3*x^4-4*x^3)',-1,2)

fminbnd('-(3*x^4-4*x^3)‘,0,2)

Calculating the minimum of a several variable function

min=fminsearch(‘funcion’,x0)

Here we give a point in which we want to minimize the function

Example

Minimize the function f(x)=sin(xy) around [0,0]

fminsearch(‘sin(x(1)*x(2))’,[0,0])

To calculate the zeros of a function

root=fzero(‘funcion’,x0)

We should give the inital point x0 to begin iterations

example:

sen(x)-2cos(2x)+$x^2=\pi^2$-2.

suppose that the function changes its sign at this interval[0,10]

x=0;eval(‘sin(x)-2*cos(2*x)+x^2-pi^2+2’)

x=10;eval(‘sin(x)-2*cos(2*x)+x^2-pi^2+2’)

x0=5

x=fzero(‘sin(x)-2*cos(2*x)+x^2-pi^2+2’,5)

eval(‘sin(x)-2*cos(2*x)+x^2-pi^2+2’)

Intregration:

.-Integration of unidimensional functions

quad

quadl

.-Double integration

dblquad

Example

```
ia=quad('sin(x)+1',0,2*pi); %(6.2832)

xmin=pi;xmax=2*pi;
ymin=0;ymax=pi;
result=dblquad('y*sin(x)+x*cos(y)',xmin,xmax,ymin,ymax)
%-9.8698
 ia=quad('sin(x).^2.*cos(x).^2',0,4*pi) %1.0051e-030
 ial=quadl('sin(x).^2.*cos(x).^2',0,4*pi) %1.5708
```

Solving differential equations

Solving problems of initial values for differential equations.

[T,Y]=solver('F',tspan,y0)

.- solver Algorithm used to solve the diferential equation: ode45, ode23, ode113, ode15s,ode23s.

.-F string contains the name of the file with the differential equation.

.-tspan time integration vector [t0 tfinal]

.-y0 column vector of the initial conditions.

function	type	method
ode45	Non-stiff dif ec	Medium order
ode23	Non-stiff dif ec	Low order
ode113	Non-stiff dif ec	Variable order
ode15s	stiff dif ec	Variable order
ode23s	stiff dif ec	Low order

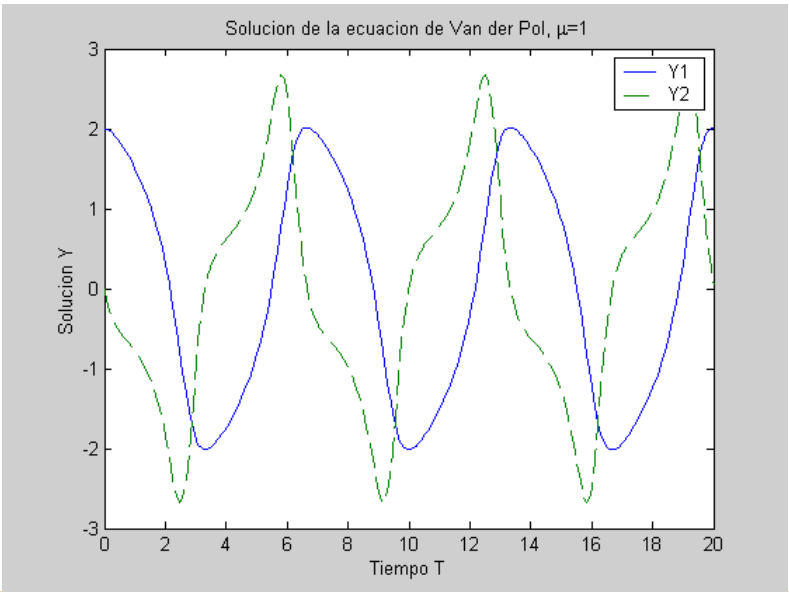
Example

$$y_1'' - \mu\left(1 - y_1^2\right)y_1' + y_1 = 0$$
$$\mu > 0$$

Rewriting the system:

$$y_1'' = y_2$$
$$y_2' = \mu\left(1 - y_1^2\right)y_2 - y_1$$

```
function dy=vdp1(t,y)
dy=[y(2); (1-y(1)^2)*y(2)-y(1)];
Call solver
[T,Y]=ode45('vdp1',[0 20],[2;0]);
plot(T,Y(:,1),'-',T,Y(:,2),'--')
title('Solucion de la ecuacion de Van der Pol, \mu=1')
xlabel('Tiempo T')
ylabel('Solucion Y')
legend('Y1','Y2')
```



Solving ODE analitically

R=Dsolve('eq1,eq2,...', 'cond1,cond2,...','v')

•Symbolically it solves ODE(s) specified by eq1, eq2,... using v as the independent variable and the boundary and/or initial condition(s) specified by cond1,cond2,....

•If dsolve cannot find an explicit solution, it attempts to find an implicit solution. When dsolve returns an implicit solution, it issues a warning. If dsolve cannot find either an explicit or an implicit solution, returns the empty sym. A numeric solution with ode23 or ode45 functions.

Examples

```
>>dsolve('Dx = -a*x')
C1*exp(-a*t)
```

```
>>dsolve('Dy = a*y', 'y(0) = b')
b*exp(a*t)
```

```
>> r=dsolve('Dx = y', 'Dy = -x','x(0)=0','y(0)=1','t')
r =
    x: [1x1 sym]
    y: [1x1 sym]
>> a=r.x
 a =sin(t)
```

ezplot(function, [a b]) Easy to be used for plotting

```
sol = dsolve('Dy=t*y^2','y(-2)=y0','t')
for y0=-0.4:0.2:2
        ezplot( subs(sol,'y0',y0) , [-2 2])
        hold on
end
hold off
axis tight
```

PROGRAMMING

All the comand files in matlab have the extension .m:

•Function files: The first line is executable and begin with the word function

•Program files: Comand sequence.

Input output (IO) comands

•input: introduces data from keyboard

variable=input('mensaje a pantalla');

•disp: shows text

disp('El algoritmo no ha convergido')

•menu: creates an option menu

option=menu('title','option1',...'optionp')

•error('condition','message'): gives information about a problem in the program execution

Programing functions

First line represents the execution line

function arg_salida=nombre_funcion(arg_entrada)

Then we can add all the commands that we want according to the purpose of the function

Return: gives back the control to the main program

Definition of a function which calculates de temperature knowing the pressure and the volume for an ideal gas

```
function t=gases(p,v,n)
%t=gases(p,v,n)
%funcion que considra la ley de los gases ideales
%Argumentos de entrada:
%presion p (atmosferas)
%volumen v (litros)
%número de moles n
%Argumentos de salida:
%temperatura t (grados kelvin)
%R=0.0821 atm.litro/mol.grado
R=0.0821
t=p*v/(n*R);
```

Saved as gases.m

temp=gases(20,10,10)

-The m-file can be executed only after >> or in another file
-The m-file does not use input arguments
-The variables of a function are locals and the variables of a file are global.

global variable
echo writes each comand of the file in the screen
pause stops the execution of a program, unless the user push a key.
keyboard same utility although it allows to introduce commands during the execution of the program.

Bucles. Nested Bucles

for k=n1:incre:n2
end

for k=vector_columna
end

Break: breaks the execution sequence

Condicional control structures

```
if
end
```

```
if
else
end
```

```
if
elseif
end
```

```
if
elseif
else
end
```

```
while
end
```

Logical and order operators

- <, <=, >, >=
- ==, ~=
- OR: | AND: & NOT: ~

The result of this operations will be 1 or 0

Other functions:

xor(x,y): "or" 0 if x or y are different from 0

any(x): True if any element of a vector is a nonzero number or is logical 1 (TRUE)..

all(x): True if all elements of a vector are nonzero.

isempty(x): 1 if the matrix x is empty.

isequal(x1,x2,...,xn): 1 if all the matrix are identical.

```
switch expresion
        case expresion_0
                comandos_0 que deben ejecutarse
        case expresion_1
                comandos_1 que deben ejecutarse
        case expresion_2
                comandos_2 que deben ejecutarse
        otherwise
                comandos escoba que deben ejecutarse
end
```

IO to external files
fread and fscanf to read
fprintf and fwrite to write
The general procedure is:

-Open the file that you want to read or write.
-Put the write or read pointer on the desired position.
-Read or write variables.
-Close the file.

-Open the file fopen

ident=fopen('nombre de fichero',type of file)

ident saves the identification number, if it is -1 the file can not be opened

-Close the file fclose

fclose(ident)
fclose('all')
verif=fclose('all')

-Putting the pointer.

1.-Put the pointer in the beginning of the file:
frewind(ident)
2.-Put the pointer inside a file
test=fseek(ident,posi,'origen')

This sentece put the pointer of the file ident in the possition indicated by posi:

.-If posi>0 move posi bytes upwards.
.-If posi<0 mueve posi bytes downwards.
.-If posi=0 no movement.

The variable origen shows from which position the pointer begins to move:

.-'bof': Beginning of the file.
.-'cof': Possition that if have now.
.-'eof': End of the file.

fseek turns back the test variable(-1 if something go wrong).
To know the pointer position: posi=ftell(ident)

Reading formated data:

[data,counter]=fscanf(ident,'format',how_many)

1.-Read data from the file ident.
2.-The data are saved in data.
3.-how_many shows how many data are going to be read.

.-k
.-[n,m] : The data are stored column to column
.-inf : all the data of the file.

4.-counter shows how many data have been read
5.-format: format to read.

%d: decimals

%e: Exponential notation

%f: fix point notation

%g: neither considered nor significative 0

%s: character variable

Reading not formated data

data=fread(ident,how_many,'precision')

Writing in a file

Formated data:

counter=fprintf(ident,'format',data,controls)

\n new linw

\t goes forward to the nex tab position

Not formated data:

counter=fwrite(ident,data,'precision')

Example:

```
function root2=sole2(a,b,c)
%root2=sole2(a,b,c)
%solve an equation of second orden
%ax^2+bx+c=0, a~=0
%
if (nargin ~=3)
      error('the numer of arguments must be 3')
end
discri=b^2-4*a*c;
root2=[];
if(discri==0)
       root2=-b/(2*a);
       disp(['Double root2=',num2str(root2)])
       return
elseif(discri>0)
```

```
        root2(1)=(-b+sqrt(discri))/(2*a);
        root2(2)=(-b-sqrt(discri))/(2*a);
        disp(['Simple Real Roots=',num2str(root2)])
        return
    else
        root2(1)=(-b+sqrt(-discri)*i)/(2*a);
        root2(2)=(-b-sqrt(-discri)*i)/(2*a);
        disp(['Imaginary Roots=',num2str(root2)])
        return
    end
```

This function is executed:

```
root2=sole2(2,3,1);
root2=sole2(1,0,1);
```

Example 3:

Develop a program to multiply complex matrix without using the predefined matlab functions:

```
A=[1 2;3 4];
B=[2 3;4 5];
C=[3 4;5 6];
D=[4 5;6 7];
E=zeros(2,2);F=zeros(2,2);
for i=1:1:2;
   for j=1:1:2;
   E(i,j)=A(i,j)*C(i,j)-B(i,j)*D(i,j)
   F(i,j)=A(i,j)*D(i,j)+B(i,j)*C(i,j)
   end
end
fid = fopen('exp.txt','wt');

for i=1:1:2;
   for j=1:1:2;
    if (F(i,j)>0)
      fprintf(fid,'%i + %i i\t',E(i,j),F(i,j));

    else
      fprintf(fid,'%i %i i\t',E(i,j),F(i,j));
    end
    end
   fprintf(fid,'\n');
end

  fclose(fid)
```

Example 3:

Reading data

Name	Analysis	Algebra	Physics	Statistics
Fernando Gómez Pereira	3	6	5	7
Susana Rodríguez Pérez	7	4	3	1
Carlos Leis Álvarez	8	9	7	9
Arturo Gómez Álvarez	5	4	5	9
Silvia Tais Álvarez	10	9	10	9
Andrea Gallego Nimes	3	3	2	4
Alicia Caballero Leis	6	8	8	5
Antonio Fraga Gómez	5	7	6	5
Beatriz Machado Gómez	4	3	5	4
Laura Tobío Manzanal	7	8	5	9
Juan Rico Fraga	4	7	5	5
Andrés Pena Gómez	6	8	5	6
Luis Blanco Villa	8	6	6	4
Sandra Puentes Gallego	9	9	7	9
Isolina Prieto Gómez	5	5	6	6
Teresa Sieiro Gon	4	2	5	3
Ricardo López Amigo	8	6	2	9

Example 2:

```
ident=fopen('datos.m');
frewind(ident)
variables=[];
for i=1:5
   variable=fscanf(ident,'%s',1);
   long_v(i)=length(variables)+length(variable);
   variables=[variables variable];
   if(i==1)
      fprintf(1,'\t%s\t\t',variable)
   else
      fprintf(1,'\t%s\t',variable)
   end
end
nombres=[];
califica=[];
```

```
for i=1:17
   nombre=fscanf(ident,'%s,%c',1);
   apellido1=fscanf(ident,'%s,%c',1);
   apellido2=fscanf(ident,'%s,%c',1);
   l=3*(i-1)
   long_n(l+1)=length(nombres)+length(nombre);
   long_n(l+2)=long_n(l+1)+length(apellido1);
   long_n(l+3)=long_n(l+2)+length(apellido2);
   nombres=[nombres nombre apellido1 apellido2];
   califica=[califica; fscanf(ident,'%i',4)'];
   fprintf(1,'\n%9s %s %9s',nombre, apellido1, apellido2)
   fprintf(1,'\t%i\t\t%i\t\t%i\t\t%i',califica(i,:))
end
fclose(ident)
```

Bibliography

- Matlab manual. http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.shtml