# Curso de Octave

29 de abril de 2015

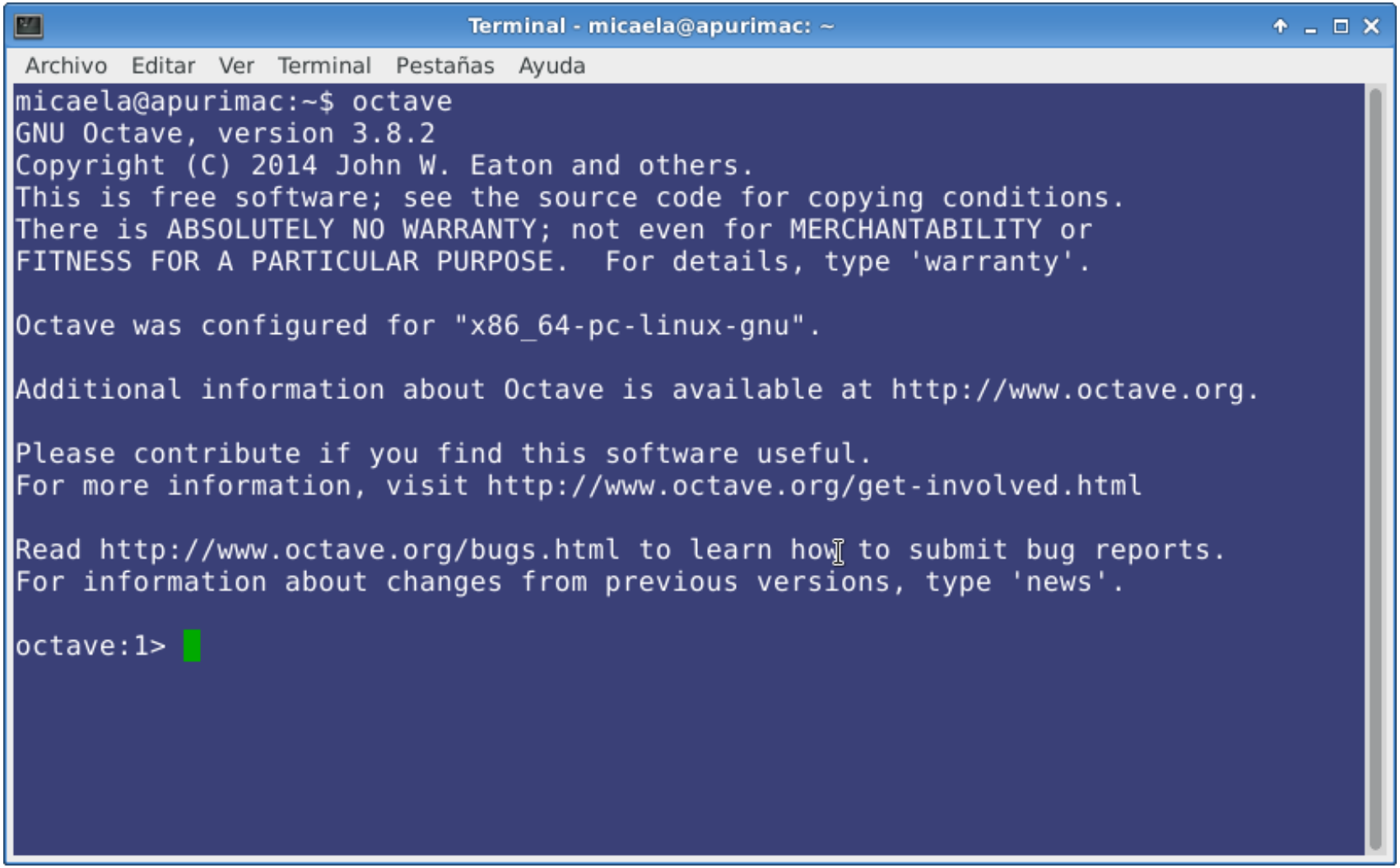
# Parte I

# Primera clase

# Introducción

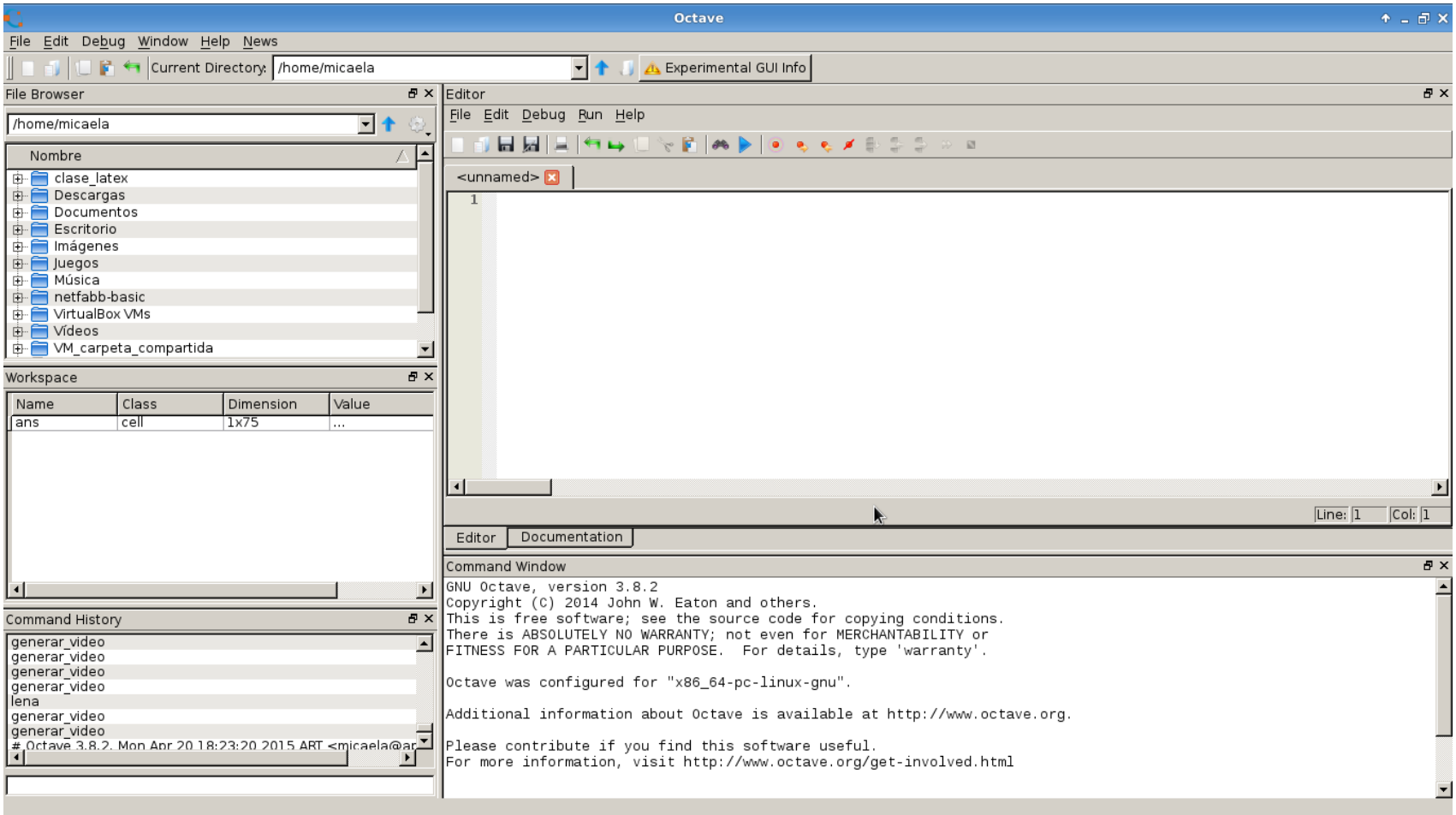
- Lenguaje orientado a análisis numérico
- Software libre
- Multiplataforma *(Microsoft Windows, Mac OS X, GNU/Linux y otros sistemas Unix)*
- Lenguaje interpretado

# Octave

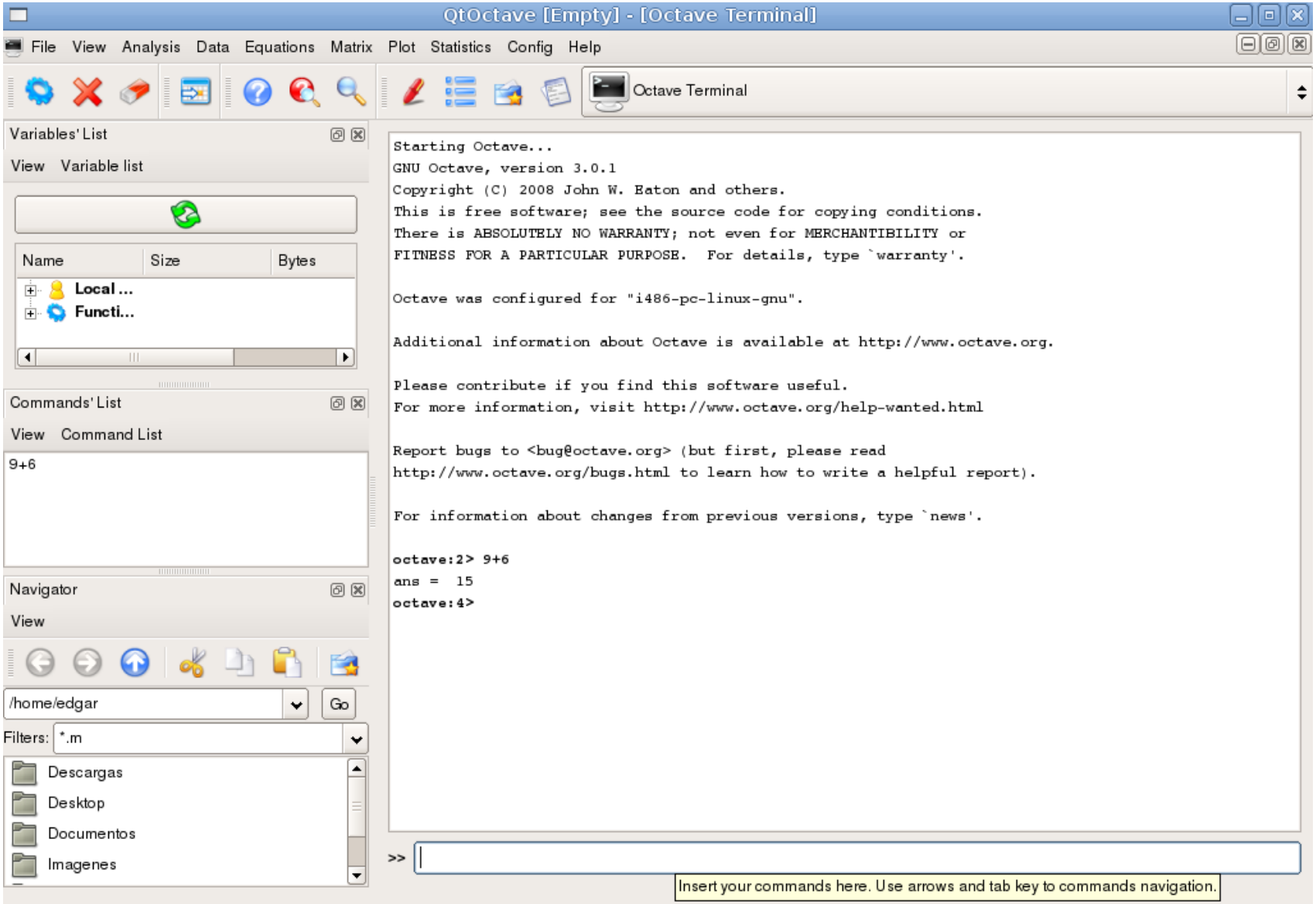

Terminal - micaela@apurimac: ~
Archivo Editar Ver Terminal Pestañas Ayuda
micaela@apurimac:~$ octave
GNU Octave, version 3.8.2
Copyright (C) 2014 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.
Octave was configured for "x86_64-pc-linux-gnu".
Additional information about Octave is available at http://www.octave.org.
Please contribute if you find this software useful.
For more information, visit http://www.octave.org/get-involved.html
Read http://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.
octave:1>

# Octave GUI

```
$ octave --force-gui
```



# QTOctave


QtOctave [Empty] - [Octave Terminal]
File View Analysis Data Equations Matrix Plot Statistics Config Help
Octave Terminal
Variables' List
View Variable list
Name Size Bytes
Local ...
Functi...
Commands' List
View Command List
9+6
Navigator
View
/home/edgar
Go
Filters: *.m
Descargas
Desktop
Documentos
Imagenes
Starting Octave...
GNU Octave, version 3.0.1
Copyright (C) 2008 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTIBILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type `warranty'.
Octave was configured for "i486-pc-linux-gnu".
Additional information about Octave is available at http://www.octave.org.
Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html
Report bugs to <bug@octave.org> (but first, please read
http://www.octave.org/bugs.html to learn how to write a helpful report).
For information about changes from previous versions, type `news'.
octave:2> 9+6
ans = 15
octave:4>
>>
Insert your commands here. Use arrows and tab key to commands navigation.

# Entorno de trabajo

- Interfaz orientada a línea de comandos
- Ejecución de scripts (archivos con extensión *.m*)

# Modo calculadora

- Operadores aritméticos (+, -, *, /)

# Modo calculadora

- Operadores aritméticos (+, -, *, /)
- Potenciación (^)

# Modo calculadora

- Operadores aritméticos (+, -, *, /)
- Potenciación (^)
- Reglas de precedencia

# Modo calculadora

- Operadores aritméticos (+, -, *, /)
- Potenciación (^)
- Reglas de precedencia
- Funciones matemáticas (sin, cos, log, exp, etc.)

# Modo calculadora

- Operadores aritméticos (+, -, *, /)
- Potenciación (^)
- Reglas de precedencia
- Funciones matemáticas (sin, cos, log, exp, etc.)
- Números reales o imaginarios (i, j)

# Modo calculadora

- Operadores aritméticos (+, -, *, /)
- Potenciación (^)
- Reglas de precedencia
- Funciones matemáticas (sin, cos, log, exp, etc.)
- Números reales o imaginarios (i, j)
- Constantes matemáticas (e, pi)

# Modo calculadora

- Operadores aritméticos (+, -, *, /)
- Potenciación (^)
- Reglas de precedencia
- Funciones matemáticas (sin, cos, log, exp, etc.)
- Números reales o imaginarios (i, j)
- Constantes matemáticas (e, pi)
- Notación científica (1e10, 1e-10)

# Comandos generales

; No imprime el resultado del comando.

# Comandos generales

; No imprime el resultado del comando.

# o % Comentario hasta el final de la línea.

# Comandos generales

; No imprime el resultado del comando.

# o % Comentario hasta el final de la línea.

help Breve descripción de los operadores y funciones.

# Comandos generales

; No imprime el resultado del comando.

# o % Comentario hasta el final de la línea.

help Breve descripción de los operadores y funciones.

lookfor Busca una cadena dentro de la ayuda de todas las funciones disponibles.

# Comandos generales

; No imprime el resultado del comando.

# o % Comentario hasta el final de la línea.

help Breve descripción de los operadores y funciones.

lookfor Busca una cadena dentro de la ayuda de todas las funciones disponibles.

clc Limpia la pantalla.

# Comandos generales

; No imprime el resultado del comando.

# o % Comentario hasta el final de la línea.

help Breve descripción de los operadores y funciones.

lookfor Busca una cadena dentro de la ayuda de todas las funciones disponibles.

clc Limpia la pantalla.

cd Cambia el directorio de trabajo.

# Comandos generales

; No imprime el resultado del comando.

# o % Comentario hasta el final de la línea.

help Breve descripción de los operadores y funciones.

lookfor Busca una cadena dentro de la ayuda de todas las funciones disponibles.

clc Limpia la pantalla.

cd Cambia el directorio de trabajo.

ls Lista el contenido del directorio de trabajo.

# Comandos generales

; No imprime el resultado del comando.

# o % Comentario hasta el final de la línea.

help Breve descripción de los operadores y funciones.

lookfor Busca una cadena dentro de la ayuda de todas las funciones disponibles.

clc Limpia la pantalla.

cd Cambia el directorio de trabajo.

ls Lista el contenido del directorio de trabajo.

pwd Muestra la ruta del directorio de trabajo donde nos encontramos.

# Tipos de datos

Las matrices son el elemento básico.
Los vectores y los escalares son casos particulares de las mismas.

Vector fila Elementos separados por espacios o ‘,’

Vector columna Elementos separados por ‘;’

# Operaciones con matrices

- Los operadores matemáticos operan según las reglas del álgebra matricial.

# Operaciones con matrices

- Los operadores matemáticos operan según las reglas del álgebra matricial.
- Los operadores matemáticos precedido por '.', operan elemento a elemento de la matriz.

# Definición de variables

## Variable

Identificador que permite almacenar datos.

- El nombre de la variable solo puede tener...
  - Letras
  - Números
  - Guión bajo
- No puede empezar con números
- Sensible a mayúsculas y minúsculas

# Listar variables y algo más

- `who`

# Listar variables y algo más

- `who`
- `whos`

# Listar variables y algo más

- `who`
- `whos`
- `clear`

# Listar variables y algo más

- `who`
- `whos`
- `clear`
- `size(...)`

# Listar variables y algo más

- `who`
- `whos`
- `clear`
- `size(...)`
- `rows(...)`

# Listar variables y algo más

- `who`
- `whos`
- `clear`
- `size(...)`
- `rows(...)`
- `columns(...)`

# Listar variables y algo más

- `who`
- `whos`
- `clear`
- `size(...)`
- `rows(...)`
- `columns(...)`
- `length(...)`

# Secuencias

- `numero_inicial:numero_final`

# Secuencias

- `numero_inicial:numero_final`
- `numero_inicial:paso:numero_final`

# Secuencias

- `numero_inicial:numero_final`
- `numero_inicial:paso:numero_final`
- `linspace(BASE,LIMIT,N)`
  N elementos linealmente espaciados entre BASE y LIMIT.

# Secuencias

- `numero_inicial:numero_final`
- `numero_inicial:paso:numero_final`
- `linspace(BASE,LIMIT,N)`
  N elementos linealmente espaciados entre BASE y LIMIT.
- `logspace(A,B,N)`
  N elementos logaritmicamente espaciados de $10^A$ a $10^B$.

# Funciones especiales

- `ones(...)`

# Funciones especiales

- `ones(...)`
- `zeros(...)`

# Funciones especiales

- `ones(...)`
- `zeros(...)`
- `rand(...)`

# Funciones especiales

- `ones(...)`
- `zeros(...)`
- `rand(...)`
- `eye(...)`

# Funciones especiales

- `ones(...)`
- `zeros(...)`
- `rand(...)`
- `eye(...)`
- `diag(...)`

# Referenciar y/o modificar elementos de una matriz

- Subíndice expresado entre paréntesis
- Los subíndices comienzan en 1
- Rango mediante el operador ‘:’
- Comando `end`

# Otras funciones útiles

- `min(...)`, `max(...)`
- `reshape(...)`, `Transponer '`
- `norm(...)`, `inv(...)`, `det(...)`,`eig(...)`
- `ceil(...)`, `floor(...)`

# Operaciones con polinomios

## Polinomios

$$p(x) = 2x^3 + x^2 - 5$$

$$P = [2, 1, 0, -5]$$

- `roots(P)`
- `polyval(P,X)`

# Operaciones lógicas

Al efectuar operaciones lógicas el resultado será 1 para verdadero y 0 para falso.
Para matrices, las operaciones se realizan elemento a elemento.

## Operadores de comparación

Menor $<$

Menor o igual $<=$

Igual $==$

Mayor $>$

Mayor o igual $>=$

Distinto $!=$ o $\sim=$

# Operaciones lógicas

## Operadores booleanos

AND &

OR |

NOT ! o ~

Cualquier elemento distinto de 0 es considerado verdadero

# Búsquedas

## find()

Devuelve los índices de los elementos de una matriz distintos de cero.

# ¡Ejercicio!

# Ejercicio 1

## Código

```
close all
clear all
clc
```

# Ejercicio 1

## Código

```
close all
clear all
clc

% Leemos la imagen con la funcion imread()
```

# Ejercicio 1

## Código

```
close all
clear all
clc

% Leemos la imagen con la funcion imread()
```

# Ejercicio 1

## Código

```
close all
clear all
clc

% Leemos la imagen con la funcion imread()
I = imread('horus.pbm');
```

# Ejercicio 1

## Código

```
close all
clear all
clc

% Leemos la imagen con la funcion imread()
I = imread('horus.pbm');

% Graficamos la imagen leida con la funcion imshow()
```

# Ejercicio 1

## Código

```
close all
clear all
clc

% Leemos la imagen con la funcion imread()
I = imread('horus.pbm');

% Graficamos la imagen leida con la funcion imshow()
```

# Ejercicio 1

## Código

```
close all
clear all
clc

% Leemos la imagen con la funcion imread()
I = imread('horus.pbm');

% Graficamos la imagen leida con la funcion imshow()
figure
imshow(I);
```

# Ejercicio 1

## Código

```
close all
clear all
clc

% Leemos la imagen con la funcion imread()
I = imread('horus.pbm');

% Graficamos la imagen leida con la funcion imshow()
figure
imshow(I);

% Buscamos en I cierto tipo de datos
```

# Ejercicio 1

## Código

```
close all
clear all
clc

% Leemos la imagen con la funcion imread()
I = imread('horus.pbm');

% Graficamos la imagen leida con la funcion imshow()
figure
imshow(I);

% Buscamos en I cierto tipo de datos
```

# Ejercicio 1

## Código

```
close all
clear all
clc

% Leemos la imagen con la funcion imread()
I = imread('horus.pbm');

% Graficamos la imagen leida con la funcion imshow()
figure
imshow(I);

% Buscamos en I cierto tipo de datos
index0 = find(I==0);
```

# Ejercicio 1

## Código

```
close all
clear all
clc

% Leemos la imagen con la funcion imread()
I = imread('horus.pbm');

% Graficamos la imagen leida con la funcion imshow()
figure
imshow(I);

% Buscamos en I cierto tipo de datos
index0 = find(I==0);
index1 = find(I==1);
```

# Ejercicio 1

## Código

```
% Invertimos los valores de los datos
```

# Ejercicio 1

## Código

```
% Invertimos los valores de los datos
```

# Ejercicio 1

## Código

```
% Invertimos los valores de los datos
I(index0) = 1;
```

# Ejercicio 1

## Código

```
% Invertimos los valores de los datos
I(index0) = 1;
I(index1) = 0;
```

# Ejercicio 1

## Código

```
% Invertimos los valores de los datos
I(index0) = 1;
I(index1) = 0;

% Graficamos la imagen procesada
```

# Ejercicio 1

## Código

```
% Invertimos los valores de los datos
I(index0) = 1;
I(index1) = 0;

% Graficamos la imagen procesada
```

# Ejercicio 1

## Código

```
% Invertimos los valores de los datos
I(index0) = 1;
I(index1) = 0;

% Graficamos la imagen procesada
figure
imshow(I);
```

# Ejercicio 1

## Código

```
% Invertimos los valores de los datos
I(index0) = 1;
I(index1) = 0;

% Graficamos la imagen procesada
figure
imshow(I);

% Generamos un archivo con la nueva imagen
```

# Ejercicio 1

## Código

```
% Invertimos los valores de los datos
I(index0) = 1;
I(index1) = 0;

% Graficamos la imagen procesada
figure
imshow(I);

% Generamos un archivo con la nueva imagen
```

# Ejercicio 1

## Código

```
% Invertimos los valores de los datos
I(index0) = 1;
I(index1) = 0;

% Graficamos la imagen procesada
figure
imshow(I);

% Generamos un archivo con la nueva imagen
imwrite(I,'horus_invertido.pbm');
```

# Parte II

# Segunda clase

# Cadenas de caracteres

## *Strings*

Vector constante de caracteres. Delimitado por comillas simples o dobles.

- Se puede trabajar con los caracteres de las cadenas como si fuesen elementos de un vector.
- Se pueden utilizar como elemento de un vector.

# Salida de datos con formato

- `disp(...)`
- `num2str(...)`
- `printf(...)`

# Control de flujo

## Sentencia condicional

Es una instrucción o grupo de instrucciones que se pueden ejecutar o no en función del valor de una condición.

```
if (COND)
    ...
endif
```

```
if (COND)
    ...
else
    ...
endif
```

```
if (COND1)
    ...
elseif (COND2)
    ...
else
    ...
endif
```

# Control de flujo

## Bucle

Es una instrucción o grupo de instrucciones que se realizan repetidas veces, hasta que la condición asignada a dicho bucle deje de cumplirse.

```
while (COND)
    ...
endwhile
```

```
for i = 1:10
    ...
endfor
```

# Control de flujo

## break

Permite terminar de forma abrupta un bucle

```
while (i <= length(a))

   if (a(i) == 0)
      break;
   endif

   num/a(i);
   i++;

endwhile
```

# Control de flujo

- pause(...)
- tic/toc

# Gráficos bidimensionales

## Ventana de gráficos

El comando `figure` permite abrir múltiples ventanas de gráficos al mismo tiempo. Las ventanas abiertas se numeran con identificadores únicos, los cuales son enteros positivos.

- `figure()`
- `close()`

# Gráficos bidimensionales

Octave permite crear gráficos simples en dos dimensiones. Con ejes lineales...

- `plot()`
- `stem()`

# Gráficos bidimensionales

## Argumentos de formato

Estilo de línea '-','- -',':','-.'

Estilo de marcador '+','o','*','.','x','s','d', etc.

Color 'k','r','g','b','m','c','w'

# Gráficos bidimensionales

## Funciones adicionales

- `title()`
- `xlabel()`
- `ylabel()`
- `legend()`
- `axis()`
- `grid`

# Gráficos bidimensionales

## Gráficos múltiples

hold on Permite superponer gráficos en un mismo diagrama.

subplot(filas, columnas, indice) Permite mostrar diferentes gráficos en una misma ventana.

# Gráficos bidimensionales

## Imprimir a un archivo

*print()* permite imprimir un gráfico y guardarlo como un archivo de imágen.

```
print (h, filename, options)
```

- `h` es un manejador que permite seleccionar la figura que se desea imprimir (`h = figure (n)`)
- Múltiples formatos de salida disponibles (ps, eps, png, jpeg, gif, etc.)

# Funciones definidas por el usuario

## Declaración de una función

```
function [variable_retorno] = mi_funcion (arg1, arg2)

    ...

endfunction
```

## Invocación de una función

```
a = mi_funcion(b,c);
```

# Manejo de archivos

| | |
|---|---|
| save() | Guarda las variables y su valor en un archivo. |
| load() | Recupera el valor de variables desde un archivo. |
| dlmwrite() | Escribe una matriz de datos a un archivo. |
| dlmread() | Lee una matriz de datos desde un archivo. |

# ¡Ejercicio!

# Ejercicio 2

## Código

```
close all
clear all
clc
```

# Ejercicio 2

## Código

```
close all
clear all
clc

% Leemos el archivo de sonido
```

# Ejercicio 2

## Código

```
close all
clear all
clc

% Leemos el archivo de sonido
```

# Ejercicio 2

## Código

```
close all
clear all
clc

% Leemos el archivo de sonido
[y, Fs, bps] = wavread('timbales.wav');
```

# Ejercicio 2

## Código

```
close all
clear all
clc

% Leemos el archivo de sonido
[y, Fs, bps] = wavread('timbales.wav');

% Promediamos cada elemento de la señal
% con los K elementos vecinos
% La idea es simular un filtro pasa bajos
% Las frecuencias altas se atenúan
```

# Ejercicio 2

## Código

```
close all
clear all
clc

% Leemos el archivo de sonido
[y, Fs, bps] = wavread('timbales.wav');

% Promediamos cada elemento de la señal
% con los K elementos vecinos
% La idea es simular un filtro pasa bajos
% Las frecuencias altas se atenúan
```

# Ejercicio 2

## Código

```
close all
clear all
clc

% Leemos el archivo de sonido
[y, Fs, bps] = wavread('timbales.wav');

% Promediamos cada elemento de la señal
% con los K elementos vecinos
% La idea es simular un filtro pasa bajos
% Las frecuencias altas se atenúan
N = rows(y);
K = 10;
output = zeros(size(y));
```

# Ejercicio 2

## Código

```
for i = 1:N
   if(i+K) <= N
      output(i,:) = sum(y(i:i+K,:))*(1/K);
   else
      output(i,:) = sum(y(i:N,:))*(1/K);
   endif
endfor
```

# Ejercicio 2

## Código

```
for i = 1:N
   if(i+K) <= N
      output(i,:) = sum(y(i:i+K,:))*(1/K);
   else
      output(i,:) = sum(y(i:N,:))*(1/K);
   endif
endfor

% Generamos un archivo de sonido con la señal filtrada
```

# Ejercicio 2

## Código

```
for i = 1:N
   if(i+K) <= N
      output(i,:) = sum(y(i:i+K,:))*(1/K);
   else
      output(i,:) = sum(y(i:N,:))*(1/K);
   endif
endfor

% Generamos un archivo de sonido con la señal filtrada
```

# Ejercicio 2

## Código

```
for i = 1:N
   if(i+K) <= N
      output(i,:) = sum(y(i:i+K,:))*(1/K);
   else
      output(i,:) = sum(y(i:N,:))*(1/K);
   endif
endfor

% Generamos un archivo de sonido con la señal filtrada
wavwrite(output, Fs, bps, 'timbales_filtrado.wav')
```

# ¡Muchas gracias!

¿Preguntas?