Métodos Numéricos

Prof.: Ana Lucía Dai Pra daipra@fi.mdp.edu.ar

Prof.: Marcel Brun mbrun@fi.mdp.edu.ar
Ay.1ra: Florencia Montini florenciamontini@fi.mdp.edu.ar
Inti Pagnuco intipagnuco@gmail.com
Lisandro Escalada lisandroescalada@gmail.com
Ay. 2da: Armando Pezzente apezzente @fi.mdp.edu.ar
Ay. Adscr: Rodrigo Russo

www3.fi.mdp.edu.ar/metodos

Horarios

Teoría: Aula 12
- martes de 10 a 11 hs.

Práctica: En salas de computadoras
- Jueves de 9:30 a 12:30 hs. (Ing. Industrial)
- Lunes de 15 a 18 hs. (Ing. Electrónica e Ing. en Computación)

Día	Clase
Mar. 8/3	Teoría 1
Jue. 10/3	Práctica 1
Mar. 15/3	Teoria 2
Jue. 17/3	Práctica 2
Mar. 22/3	Teoria 3
Jue. 24/3	Feriado
Mar. 29/3	Teoría 3
Jue. 31/3	Práctica 3
Mar. 5/4	Teoria 4
Jue. 7/4	Práctica 3
Mar. 11/4	Teoría 4
Jue. 14/4	Práctica 3-4
Mar. 19/4	Teoría 5
Jue. 21/4	Práctica 4
Mar. 26/4	Parcial
Jue 28/4	Práctica 5
Mar. 3/5	Teoría 5
Jue 5/5	Práctica 5
Mar. 10/5	Teoría 5
Jue. 12/5	Práctica 5
Mar. 17/5	Teoría 6
Jue. 19/5	Práctica 5-6
Mar. 24/5	Teoría 7
Jue. 26/5	Práctica 6
Mar. 31/6	Teoria 7
Jue. 2/6	Práctica 7
Mar. 7/6	Repaso
Jue 9/6	Práctica 7
Mar. 14/6	Parcial
Jue 16/6 y Mar. 21/6	Repaso
Jue 23/6	Recup.

Cronograma 1er cuatrimestre 2016

Unidad	Descripción
1	**Introducción, Matlab**
2	**Error, representación de números en Punto flotante.**
3	**Solución de ecuaciones no lineales**, Criterios aprox., métodos cerrados, métodos abiertos, raíces de polinomios, **Sistemas de ecuaciones no lineales**.
4	**Solución sist. ecuaciones lineales**, Eliminación Gaussiana, LU, Condición, normas, métodos indirectos
5	**Interpolación y aproximación polinomial**, Interpolación por partes, Aprox. por Mínimos cuadrados,
6	**Integración**, Métodos Newton-Cotes, Romberg, Integr. Adaptativa.
7	**Solución Ecuaciones Diferenciales Ordinarias (EDO),** Métodos Euler, Heun, Taylor Métodos Runge-Kutta (RK) Sistemas EDO. **Ecuaciones Diferenciales Parciales (EDP)**

Régimen de cursada

Dos parciales, con posibilidad de un recuperatorio.

Posible evaluación en computadora.

Promoción:
Promedio parciales >= 7, cada parcial con nota >=5.

Habilitación:
Promedio parciales >= 5, cada parcial con nota >=4.

Cada parcial y totalizador consta de 3 ejercicios de los cuales deben aprobar 2.

Bibliografía
(disponible en biblioteca de la Facultad)

•*Métodos numéricos: con MATLAB*. Mathews, John H. 1999 (10391; 10414; 10639; 12158)

•*Métodos numéricos para ingenieros*. Chapra, Steven C. 2003. (10615; 9931) , 2007 (11233; 12027) 2011(12283).

•*Métodos numéricos: aplicados a la ingeniería*. Nieves Hurtado. 2003. (10510).

• *Numerical methods*. R. L. Burden y J. Douglas Faires. 2005. (11323).

•*Numerical Methods, Software, and Analysis* Rice, John R 1993 (8145).

•*Numerical methods using Matlab.* Penny, John Dr. 1999. (10787; 11322; 11321).

•Cualquier texto de Métodos Numéricos o Análisis Numérico.

Métodos Numéricos

Necesidad de transformar un problema matemático en numérico y resolverlo

Método analítico: Permite la obtención de resultados analíticos y exactos, pero a veces limitados. (Procesamiento simbólico).

Método numérico: Procedimiento (o algoritmo) mediante el cual se obtiene, (casi siempre de manera aproximada), la solución ***numérica*** a ciertos problemas, realizando evaluaciones de funciones y operaciones aritméticas elementales.

Los métodos numéricos son técnicas mediante las cuales es posible formular problemas matemáticos de tal forma que puedan resolverse usando operaciones aritméticas, Aunque hay muchos tipos de métodos numéricos, comparten una característica común: invariablemente se debe realizar un buen número de tediosos cálculos aritméticos. No es raro que con el desarrollo de computadoras digitales, eficientes y rápidas, el papel de los métodos numéricos en la solución de problemas de ingeniería haya aumentado en forma considerable en los últimos años.

• STEVEN C. CHAPRA, RAYMOND P. CANALE, Métodos Numéricos para Ingenieros con Aplicaciones en Computadoras Personales, McGraw Hill, México, 1987. Prefacio.

……

Han pasado veinte años desde que se publicó la primera edición de este libro. Durante ese periodo, nuestro escepticismo acerca de que los métodos numéricos y las computadoras tendrían un papel prominente en el currículo de la ingeniería - particularmente en sus etapas tempranas - ha sido rebasado por mucho. Así, esta nueva edición aún se basa en la premisa fundamental de que debe darse a los estudiantes de ingeniería una introducción profunda y temprana a los métodos numéricos

Los métodos numéricos nos vuelven aptos para entender esquemas numéricos a fin de resolver problemas matemáticos, de ingeniería y científicos en una computadora, reducir esquemas numéricos básicos, escribir programas y resolverlos en una computadora y usar correctamente el software existente para dichos métodos y no solo aumenta nuestra habilidad para el uso de computadoras sino que también amplia la pericia matemática y la comprensión de los principios científicos básicos.

• STEVEN C.CHAPRA, RAYMOND P. CANALE, Métodos Numéricos para Ingenieros con Aplicaciones en Computadoras Personales, McGraw Hill, México, 2007. Prefacio

Soluciones a tipos de problemas que analizaremos mediante Métodos Numéricos

- Ecuaciones no lineales de una variable
- Sistemas de ecuaciones lineales y no lineales
- Interpolación y aproximación polinomial
- Integración
- Ecuaciones diferenciales ordinarias y parciales

Aritmética finita – Error
Programación

Importancia de conocer los Métodos Numéricos

- Herramientas poderosas en la resolución de problemas que son imposibles o muy difíciles de resolver analíticamente.
- Aunque muchos paquetes de software disponen de herramientas de métodos numéricos, el uso inteligente de estos programas depende del conocimiento de la teoría básica en que se basan los métodos.
- En los paquetes de software disponibles, no necesariamente están todos los métodos implementados,
- El problema a resolver puede formar parte de un programa de computación y necesita resolverse exclusivamente por métodos numéricos.

Cálculo simbólico (Cálculo formal o álgebra computacional)

- Programas que manipulan expresiones algebraicas no numéricas generando soluciones en modo exacto.
- Implementa las modalidades del cálculo analítico.
- Los lenguajes de programación basados en cálculo simbólico son la interfase natural entre el código maquina y el lenguaje de las matemáticas.
- Estos programas combinan perfectamente herramientas de manipulación simbólica con técnicas que permiten efectuar cálculos numéricos de precisión arbitraria.

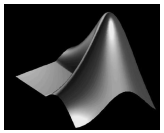
Necesidad de adoptar un lenguaje de programación

Adoptaremos como lenguaje de trabajo al **Matlab.**

Razones:

- Facilidad de uso. Visualización, cómputo, programación.
- Lenguaje técnico, adoptado por universidades, empresas e industrias.
- Herramientas actualizadas
- Utilización en el resto de la carrera.
- Facilidad de extensión (agregado y modificación de sus funciones)

MATLAB



- Es un sistema interactivo,
- Tipo de dato básico: matriz (arreglo), ***Mat**rix **lab**oratory*
- Está basado en el uso de funciones. Posee conjuntos de funciones para problemas específicos: *toolboxes* (caja de herramientas),
- Considera problemas de distintas áreas técnicas: procesamiento de señales, procesamiento de imágenes, sistemas de control, simulación, redes neuronales, lógica difusa, algoritmos genéticos, estadística, etc.
- La mayoría de las funciones están abiertas al usuario.

Tipos de datos

El tipo de dato básico es el arreglo, el cual no necesita ser dimensionado, puede estar integrado por siete tipos de datos.

double - Utilizado por defecto para todos los cálculos, 64 bits.

single - 32 bits.

char -16 bits, el arreglo constituye un *string*.

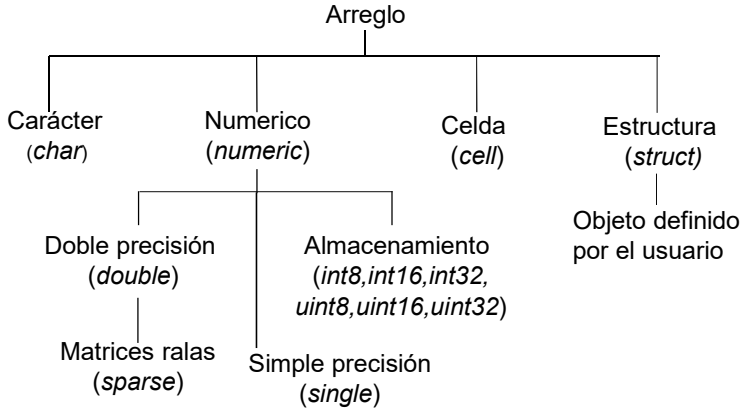
sparse - Matrices ralas, sólo 2D.

storage - Enteros con signo y sin signo, no admiten operaciones matemáticas.

cell - Arreglos compuestos de distintas clases de datos u otros arreglos.

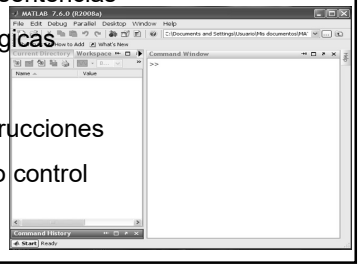
struct - Objeto definido por el usuario.

Tipo de datos

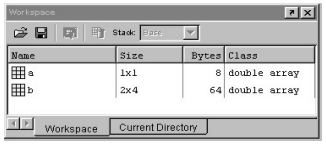


Comandos MATLAB

- Toda instrucción dada a continuación del "*prompt* " >> es considerada un comando.
- Los comandos comprenden:
  - Instrucciones de asignación o sentencias
  - Operaciones matemáticas y lógicas
  - Llamadas a funciones
  - Referencias a conjunto de instrucciones
  - Comandos de edición, ayuda o control
  - ...



Espacio o memoria de trabajo (Workspace)



Comandos:

who permite ver los nombres de todas las variables en uso.

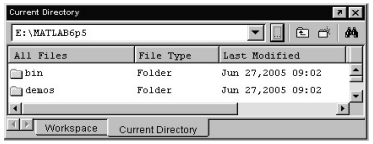
whos muestra los nombres de variables con sus respectivos tamaños, tipos y total de espacio utilizado.

clear *[variables]* borra variables.

save archivo *[variables]* guarda el *workspace* o parte de él en un archivo.*mat* (archivo binario). Hay opciones para otros tipos de archivo.

load archivo lee archivo *.mat* y lo guarda en el *workspace*.

Comandos de ayuda



path **y** ***cd*** muestra y cambia el directorio actual.

what *dir* muestra el contenido del directorio.

help *comando* Ayuda sobre el uso de un comando o sobre los comandos incorporados en una *toolbox*.

helpwin va a la ventana de ayuda

helpdesk **y** ***doc*** va a archivo *web*

lookfor *palabra* busca comandos en los que aparezca *´palabra´* en los comentarios de encabezamiento.

Ingreso de Matrices

- Los elementos ubicados dentro de una misma fila pueden ir separados por coma.
- Cada fila va separada por punto y coma

Ejemplo: Crear una matriz A de tamaño 3x3.

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

Un escalar es una matriz de 1x1 y un vector es una matriz de 1xn o nx1 si es un vector columna

Un ";" al final del comando anula la visualización de los resultados

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

Operaciones entre Matrices

+ **suma** (ambos operadores deben tener igual tamaño)
- **resta** (ambos operadores deben tener igual tamaño)
* **multiplicación** (un operador m x n y el otro n x p)
^ **potencia** (*a ^ 2*, es equivalente a *a * a*)
` **traspuesta**
\ **división izquierda** *x=A|b* es la solución de A*x=b
/ **división derecha** *x=b/A* es la solución de x*A=b

Operaciones elemento a elemento

Ambos operadores deben tener igual tamaño o uno de ellos debe ser un escalar.

.* **multiplicación**
.^ **potencia**
./ **división izquierda**
.\ **división derecha**

Ejemplo: [1,2,3,4] .* [1,2,3,4] = [1,4,9,16]

[1,2,3,4] .^ 2 = [1,4,9,16]

Matrices Complejas

Las letras ***i*** y ***j*** pueden ser utilizadas como unidad imaginaria (prestar atención si ***i*** y ***j*** no están siendo utilizadas como variables).

Ejemplos:

```
>> 2+6i;
>> A = [1 2; 3 4] + j*[5 6; 7 8]
A=
   1.0000 + 5.0000i   2.0000 + 6.0000i
   3.0000 + 7.0000i   4.0000 + 8.0000i
>> B = [1+5i 2+6i; 3+7i 4+8i]
B =
   1.0000 + 5.0000i   2.0000 + 6.0000i
   3.0000 + 7.0000i   4.0000 + 8.0000i
>> A+B
ans = 2.0000 +10.0000i   4.0000 +12.0000i
        6.0000 +14.0000i   8.0000 +16.0000i
>> A*B
ans = -60.0000 +42.0000i -68.0000 +56.0000i
        -76.0000 +74.0000i -84.0000 +96.0000i
```

Submatrices

$$A = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix} \qquad A(1:3,4) = \begin{pmatrix} d \\ h \\ l \end{pmatrix}$$

$$A(2,:) = \begin{pmatrix} e & f & g & h \end{pmatrix} \qquad A(:,[2\ 4]) = \begin{pmatrix} b & d \\ f & h \\ j & l \\ n & p \end{pmatrix}$$

Generación de Matrices especiales

Funcion	Utilidad
eye(n)	matriz identidad
zeros(m,n)	matriz de ceros
ones(m,n)	matriz de unos
diag(x)	crea una matriz con diagonal ´x´ o extrae la diagonal
triu(A)	obtiene la parte triangular superior de una matriz
tril(A)	obtiene la parte triangular inferior de una matriz
rand(m,n)	genera una matriz de valores aleatorios
hilb(n)	crea una matriz de Hilbert
magic(n)	crea un cuadrado magico

Ej:

```
>> a=[2 3; 4 5]
>> diag(a)
ans =
    2
    5
```

```
>> diag([1 2])
ans =
    1    0
    0    2
```

Sentencias, expresiones y variables

- MATLAB es una lenguaje intérprete, es decir, la expresiones que se ingresan son interpretadas y evaluadas.
- Las sentencias MATLAB son de la forma:

Variable = expresion

expresion

- Las expresiones están compuestas por operadores, funciones y nombres de variables.
- La evaluación de la expresión produce una matriz, la cual se muestra en pantalla y es asignada a una variable para un futuro uso. Si no se le asigna un nombre a la variable, se crea la variable ***ans*** (*answer*).

Sentencias, expresiones y variables

Nota:

- Una sentencia puede ser continuada en la línea siguiente colocando '**...**'
- Varias sentencias se pueden colocar en una misma línea usando '**,**' o '**;**'
- Si una sentencia es terminada con '**;**', el resultado de la operación no se muestra en pantalla.
- MATLAB es sensible a las mayúsculas y minúsculas. ("*var*" es una variable distinta de "*Var*" , "*VAR*" o "*vAr*")

Estructuras de Control

CICLO

for *variable = valor inicial : {incremento :} valor final*
sentencias
end

CICLO CONDICIONAL

while *condición*
sentencias
end

CONDICION

if *condición*
sentencias 1
{**else**
sentencias 2 }
end

Operadores relacionales y lógicos

<	menor que	&	y
>	mayor que	|	o
<=	menor que o igual	~	no (alt 126)
>=	mayor que o igual		
==	igual		
~=	no igual		

Ciclos implícitos

```
>>for i =1:10
   a(i) = i
end
```

```
>>a = 1:10
```

```
>>c(a) = 5
```

```
>>for i =1:10
   c(i) = cos(i)
end
```

```
>>c = cos(1:10)
```

```
>>c = cos(a)
```

```
for i =1:10
   if c(i) > 0.5
      c(i)
   end
end
```

```
>>j=(find (c > 0.5));
c(j)
```

```
>>c(find(c>0.5))
```

```
>>c(c>0.5)
```

Cálculo matricial

Tic, *sentencias*, toc
muestra el tiempo empleado

Funciones Escalares

Función	Utilidad
sin	Seno
cos	Coseno
tan	Tangente
asin	Arcseno
acos	Arcoseno
atan	Arcotangente
exp	Exponencial
log	Logaritmo natural
rem	Resto
abs	Valor absoluto
mod	Módulo
sqrt	Raíz cuadrada
sign	Signo
round	Redondeo
floor	Truncar
ceil	Aproximar hacia arriba

Se aplican a cada elemento de la matriz

Ej:

```
>> a=[2 3; 4 5]
>> sin(a)

ans =

    0.9093    0.1411
   -0.7568   -0.9589
```

Funciones Vectoriales

Función	Utilidad
max	Máximo
min	Mínimo
sort	Ordenar
sum	Suma
prod	Producto
median	Mediana
mean	Media
std	Desviación estandar
any	Alguno
all	Todos

Devuelven un único valor por vector

Ej:

```
>> a=[2 3; 4 5];
>> max(a)

ans =

    4    5
>> m=max(max(a))

m =

    5
```

Gráficos

Se grafican en una ventana aparte.

Esta ventana posee menúes para todas las transformaciones necesarias. (pueden variar según la versión)

Comandos:

figure: Crea una ventana de figura y la hace figura corriente. La primera se identifica como figura 1, incrementándose el número en sucesivas figuras.

figure n: Crea nuevas ventanas que identifica con el número *n* o toma la figura *n* como corriente.

hold on: Se mantiene la misma figura, superponiendo los gráficos

hold off: Desactiva el comando anterior.

Funciones para graficar

plot Abre una ventana de gráfico si no existe y grafica una serie de puntos (*x,y*), que pueden estar unidos por una línea.

plot3 Crea una figura si no existe y grafica una serie de puntos (*x,y,z*)

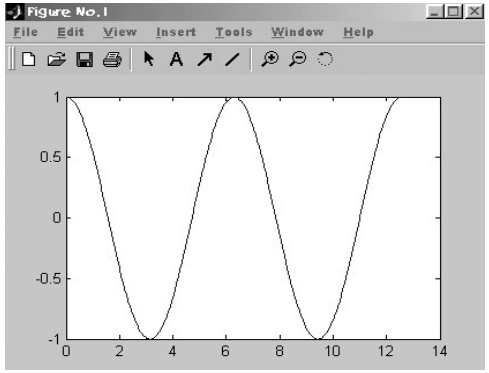
surf y ***mesh*** Grafican superficies

subplot Divide la ventana de gráficos en una *matriz de gráficos.*

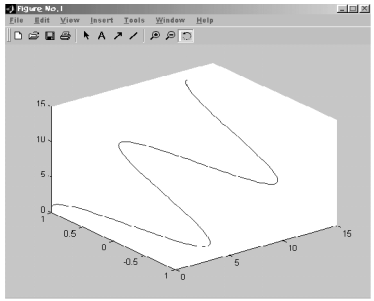
fplot Grafica una función dentro de un intervalo

Ejemplo 1.

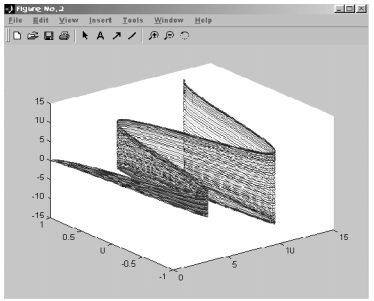
```
x = 0 : pi/50: pi*4;
y = cos(x);
plot(x,y)
```



Ejemplo 2.



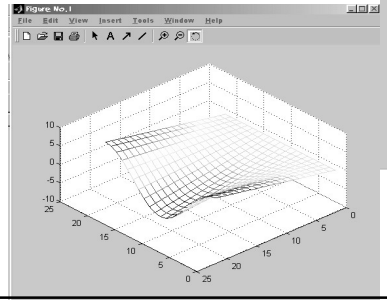
```
z = x+y;
Plot3 (x,y,z)
```

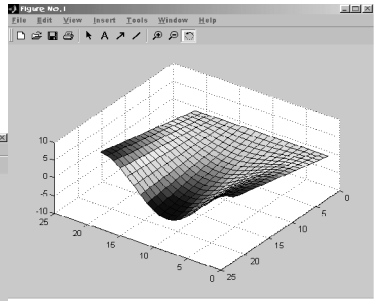


```
w = x'*y;
plot3 (x,y,w)
```

Ejemplo 3.

```
x = 0 : pi /10 : pi*2;
y =cos(x);
z = x‘ * y;
surf(z)
```





mesh(z)

Gráficos

Otros comandos para gráficos, equivalentes a los de menúes de figuras

title Título del gráfico.
xlabel Etiqueta del eje x
ylabel Etiqueta del eje y
gtext Coloca un texto en el gráfico usando el mouse
text Posiciona el texto en coordenadas especificadas.
grid Coloca una grilla al gráfico.
axis Cambia la característica de los ejes

Cálculo simbólico- Ejemplos

- **Ejemplo numérico:** derivada de **'cos(x)+x^2'**

```
>> x =1:10
x=    1    2    3    4    5    6    7    8    9   10
>> f=cos(x)+x.^2
f =   1.5403   3.5839   8.0100  15.3464  25.2837  36.9602  49.7539  63.8545
   80.0889  99.1609
>> diff(f)
ans=   2.0436   4.4262   7.3363   9.9373  11.6765  12.7937  14.1006
   16.2344  19.0721
```

- **Ejemplo simbólico:** derivada e integral de **'cos(x)+x^2'**

```
>> df=diff('cos(x)+x^2'), if=int(cos(x)+x^2)
df = -sin(x)+2*x
If=sin(x)+1/3*x^3
>> subs(df,x)
ans =    1.1585   3.0907   5.8589   8.7568  10.9589  12.2794  13.3430
   15.0106  17.5879  20.5440
```

Cálculo simbólico - Ejemplos

```
syms x
f = x^3-6*x^2+11*x-6,  g = (x-1)*(x-2)*(x-3),  h = -6+(11+(-6+x)*x)*x

C=collect (g), E=expand(h), F=factor(h), H=horner(f),S=solve(C)',Sx=subs(f,5)
C = x^3-6*x^2+11*x-6
E = x^3-6*x^2+11*x-6
F =(x-1)*(x-2)*(x-3)
H=-6+(11+(-6+x)*x)*x
S =[ 1, 2, 3]
Sx=24

pretty(f), pretty(g), pretty(h)
 3    2
x - 6 x  + 11 x - 6
(x - 1) (x - 2) (x - 3)
-6 + (11 + (-6 + x) x) x
```

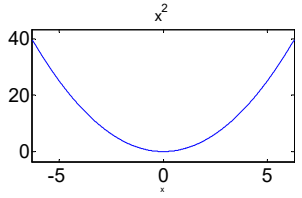
Gráficos (funciones simbólicas)

Realizan gráficos de '*f(x)*' donde *f* es un string o una expresión simbólica representando una expresión matemática, involucrando variables simbólicas (syms), por ej. '*x*' e '*y*'.

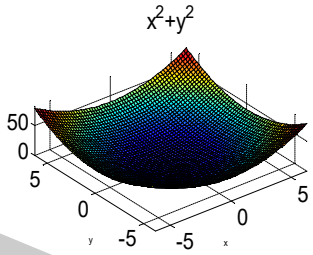
- `ezplot`. Grafica una función '*y=f(x)*' en el dominio elegido, si no se especifica el dominio asume uno por defecto.
- `ezplot3`. Grafica curvas en 3D.
- `ezsurf`. Grafica superficies.
- `ezmesh`. Grafica superficies grilladas.

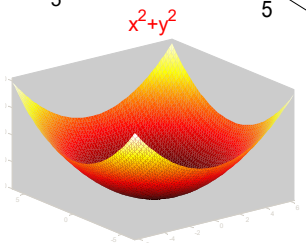
Ejemplos

ezplot('x^2')



ezsurf('x^2+y^2')





Archivos M

MATLAB puede ejecutar una secuencia de sentencias guardadas en archivos. Tales archivos son llamados "archivos M", pues estos poseen extensión ".m".

Existen dos tipos de archivos M: archivos ***script*** y ***funciones.***

- ***Script:*** Secuencia de comandos que se ejecutan al colocar el nombre del archivo como comando. Las variables del archivo crearán o modificarán las del espacio de trabajo
- ***Funciones:*** Extensión de las funciones MATLAB. Se pueden crear funciones específicas para el problema que se desea resolver. Las variables de un archivo de función son locales a la función.

Declaración de Archivos de Función

```
function salida = nombre_funcion(entrada)
% Explicación de lo que la función realiza
Sentencias…
salida = valor a retornar;
```

```
function [s1,..,sn]=nombre_funcion(e1,..,em)
% Explicación de lo que la función realiza
Sentencias…
s1 = valor a retornar;
...
sn = valor a retornar;
```

Algunos tipos de funciones pueden ser declaradas en línea ('inline')

Entrada/Salida de texto

- Asigna un string a una variable:
  variable = 'texto'
- Muestra un texto por pantalla:
  ***disp**('texto')*
- Muestra un mensaje de error y finaliza la función que se está ejecutando:
  ***error**('texto')*
- Entrada de un dato. Muestra el *texto* y el valor que se ingrese queda almacenado en *variable*:
  *variable = **input**('texto')*

Entrada/Salida de datos

- `fscanf`. Lee datos formateados desde un archivo.
- `fprintf`. Escribe datos formateados en un archivo.
- `fread`. Lee datos binarios desde un archivo.
- `fwrite`. Escribe datos binarios en un archivo
- `fopen`. Abre un archivo
- `fclose`. Cierra un archivo