

Introducción a Matlab

Dr. Humberto Madrid de la Vega

Universidad Autónoma de Coahuila

M.C. Irma Delia García Calvillo

Universidad Autónoma de Nuevo León y

Universidad Autónoma de Coahuila

Abril 2007

Contents

1	Vectores y graficación	1
1.1	Ejercicio	3
1.2	Maquillaje de la gráfica	4
1.3	Más vectores	5
2	Un poco de programación	9
2.1	Mínimos Cuadrados	12
3	Graficación de funciones en dos variables	12
3.1	Curvas de nivel	16
3.2	Campo direccional	19
4	Manipulación de matrices	21
4.1	Variables e instrucciones	22
4.2	Dimensiones variables	24
4.3	Operaciones con Matrices	24
4.4	Algunas matrices especiales	27
4.4.1	Matriz identidad	27
4.4.2	Matriz de ceros	27
4.4.3	Matriz de unos	28
4.4.4	Matriz aleatoria	28
4.5	Operaciones elemento a elemento	28
4.6	Subíndices	29
4.7	Matrices vacías	31
4.8	Almacenar y recuperar variables	32
4.9	Funciones matemáticas elementales	33
4.10	Formato racional	33
5	Ejercicios	34

El paquete *Matlab* se ha convertido en una de las principales herramientas en el ámbito de la computación científica. Su aplicabilidad va desde la enseñanza, la investigación científica y hasta la producción en la industria.

Estas breves notas no son propiamente un curso sobre *Matlab*. Intentan dar una idea sobre la importancia que puede tener este paquete en el terreno del cómputo científico. Está dirigido a personas que no está familiarizadas con el paquete e intentan que el lector en unas cuantas horas pueda apreciar las grandes ventajas de esta poderosa herramienta computacional.

Nos hemos concentrado en ejemplos sencillos de corte científico que ilustran el poder de experimentación y visualización que proporciona *Matlab*.

En cuanto a la graficación, los ejemplos ilustran las capacidades en dos dimensiones y una introducción a la graficación en tres dimensiones. Dejamos al lector que navegue por el *demo* de *Matlab* para que vea las potencialidades de esta poderosa herramienta científica.

1 Vectores y graficación

Consideremos el problema de graficar $f(x)$ en $[a, b]$. Supongamos que $f(x) = x^2$ y $[a, b] = [-2, 2]$, lo que necesitamos para graficar es una tabla de valores, por ejemplo

x	y=f(x)
-2	4
-1	1
0	0
1	1
2	4

Podemos pensar a cada columna de la tabla como un vector. En *Matlab* estos vectores se definen de la siguiente forma

$$\begin{aligned} x &= [-2 \ -1 \ 0 \ 1 \ 2] \\ y &= [4 \ 1 \ 0 \ 1 \ 4] \end{aligned}$$

y para graficar se utiliza la instrucción

plot(x,y)

Con esta instrucción se unen con líneas los puntos (x_i, y_i) a (x_{i+1}, y_{i+1}) para $i = 1, \dots, 4$, la gráfica que se obtiene es la figura 1. Si sólo se desea marcar los puntos de la gráfica se utiliza la instrucción

plot(x,y,'o')

Si se desea marcar los puntos y unir las líneas, se utiliza

plot(x,y,'o',x,y)

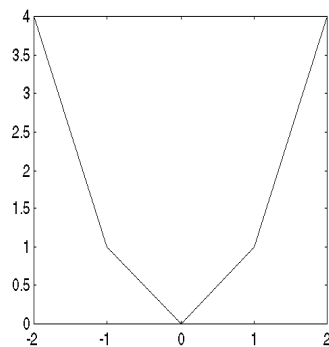


Figure 1: Gráfica de la función $f(x) = x^2$

o bien

plot(x,y,x,y,'o')

la gráfica resultante se aprecia en la figura 2.

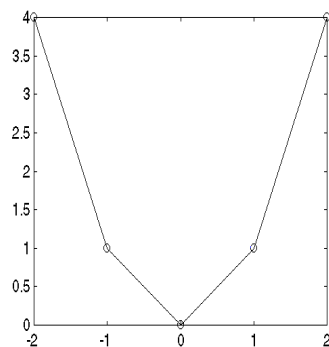


Figure 2: Gráfica marcando y uniendo puntos

Se tienen otras variantes, el comando

help plot

describe cada una de ellas.

Supóngase ahora que se desea graficar $y = \text{sen}(x)$ en $[0, 2\pi]$. Una forma de hacerlo es primero generar un vector de abscisas con puntos igualmente espaciados en el intervalo a graficar, evaluar la función en las abscisas generadas y graficar ambos vectores. Para generar un vector con puntos igualmente espaciados en el intervalo $[0, 2\pi]$ se puede utilizar la instrucción

x=0:0.1:2*pi

el cual genera un vector con un espaciado uniforme de 0.1 entre sus componentes, partiendo de 0 hasta 2π . Para calcular las ordenadas se podría utilizar una instrucción *for* (la cual veremos más adelante), pero en *Matlab* es más fácil, simplemente la instrucción

y=sin(x)

genera las correspondientes ordenadas. Como $\mathbf{x}$ es un vector, entonces $\mathbf{sin}(\mathbf{x})$ es también otro vector cuyas componentes son la función seno aplicada a cada una de las componentes del vector $\mathbf{x}$, esto es, $y(i) = \sin(x(i))$. Para graficar se utiliza nuevamente el comando **plot(x,y)**. La gráfica resultante es la figura 3.

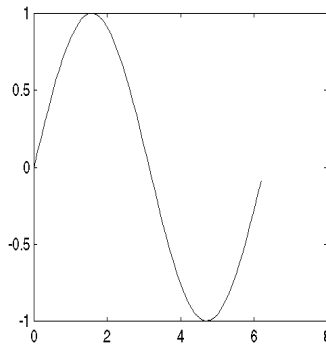


Figure 3: Gráfica de la función $\text{seno}(x)$ $[0, 2\pi]$

Una mejor idea para generar n puntos igualmente espaciados en $[0, 2\pi]$, por ejemplo para $n = 50$, es usar el comando

x=linspace(0,2*pi,50)

Esta instrucción genera 50 puntos igualmente espaciados entre 0 y 2π . La generación del vector $\mathbf{y}$ al igual que la gráfica se realizan de la misma forma que en el ejemplo anterior.

1.1 Ejercicio

Grafique **y=cos(x)** para $x \in [-2\pi, 2\pi]$ con 10, 20, 50 y 100 puntos.

1.2 Maquillaje de la gráfica

Se pueden poner accesorios a las gráficas, fijar la ventana de graficación, título, etiquetas en los ejes, zooms, entre otros. Con la instrucción `plot(x,y)` *Matlab* automáticamente genera la ventana de graficación, el rango en el eje x y en el eje y , pero se puede especificar los rangos de valores de x y de y , con la instrucción

```
axis([xmin xmax ymin ymax])
```

Es decir, se ajustan los ejes de manera que el eje x varía de $xmin$ a $xmax$ y el eje y varía de $ymin$ a $ymax$. Por ejemplo, teclee las instrucciones

```
x=linspace(0,2*pi,100);  
y=sin(x);  
plot(x,y), axis([0 6.3 -1.2 1.2])
```

observe la grafica resultante y compare con la figura 3.

Ejercicio. Graficar un círculo. Una forma de graficar un círculo es la siguiente:

```
t=linspace(0,2*pi,100);  
x=cos(t);  
y=sin(t);  
plot(x,y)
```

Teóricamente esto es un círculo, pero las escalas en los ejes no son iguales. Intente ahora con

```
plot(x,y), axis('square')
```

y luego con

```
plot(x,y), axis('equal')
```

También se pueden poner letreros a las gráficas. Para los títulos se utiliza la instrucción `title`. Por ejemplo, para los datos de la gráfica de la función seno, teclear

```
plot(x,y), axis([0 6.28 -1.2 1.2]), title('Grafica del seno')
```

Las etiquetas en los ejes se escriben con `xlabel`, `ylabel`, las cuales pueden ponerse al igual que el título, después de la instrucción `plot`. La instrucción `grid` genera una cuadrícula en la ventana de graficación, también va después de la instrucción `plot`. Si se desea poner varios accesorios a la gráfica, como las instrucciones van después del `plot`, no caben todas en una línea, entonces lo conveniente es crear un programa. Se puede poner toda esta información en un archivo de la siguiente manera: del Menú *File* escoger *New* y *M-file*, esto invoca al editor y luego hay que teclear todas las instrucciones y guardar el archivo poniéndole un nombre con extensión *m*. Por ejemplo, generar un programa de nombre *grafsen.m* que contenga las instrucciones

```

x=linspace(0,2*pi,100);
y=sin(x);
plot(x,y);
axis([0 6.3 -1.2 1.2]);
grid title('Grafica del seno');
xlabel('Etiqueta eje x') ylabel('Etiqueta eje y')

```

Al teclear desde *Matlab* la instrucción **grafsen** se ejecutarán todas las instrucciones anteriores y el resultado es la gráfica que aparece en la figura 4. La instrucción **type grafsen**, despliega el contenido del archivo *grafsen.m*.

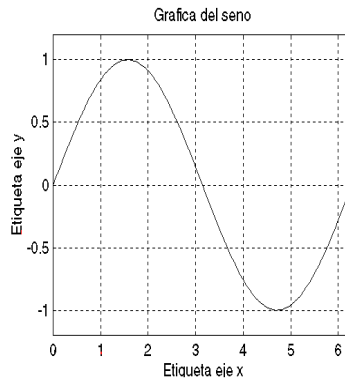


Figure 4: Títulos y etiquetas en la gráfica

También se pueden hacer acercamientos o *zooms* a la gráfica actual. Con la instrucción

```
zoom on
```

se activa el comando de acercamiento, se sitúa el cursor en la gráfica y sosteniendo oprimido el botón izquierdo del ratón se genera un rectángulo que se puede hacer del tamaño que se requiera y al soltar el mouse aparecerá la gráfica restringida al rectángulo que se generará. Con el comando

```
zoom off
```

se desactiva el *zoom*. Como ejercicio ejecute **grafsen**, **zoom on**.

Ejercicio. Modifique el archivo *grafsen.m* para graficar la función $y=\cos(x)$ adecuando los ejes para la ventana de graficación.

1.3 Más vectores

Supongamos ahora que se desea graficar $f(x) = 3x^3 + 2x - 3$ en el intervalo $[-2, 3]$ usando 50 puntos. La instrucción

```
x=linspace(-2,3,50)
```

genera las abscisas. Ahora, cómo encontramos $y = f(x)$?. El problema es que x es un vector, no un escalar, así que necesitamos realizar operaciones entre vectores. En *Matlab* si $\mathbf{x}$ es un vector y a un escalar entonces

```
a + x
```

es un vector que a cada componente de $\mathbf{x}$ le suma el escalar a .

```
a*x
```

también es un vector que a cada componente de $\mathbf{x}$ la multiplica por el escalar a , y si $\mathbf{z}$ es otro vector de la misma dimensión de $\mathbf{x}$, entonces

```
x + z
```

en un vector de la misma dimensión de $\mathbf{x}$ donde la i -ésima componente de $\mathbf{x} + \mathbf{z}$ viene dada por $x(i) + z(i)$. Dados dos vectores de igual dimensión, $\mathbf{x}$ y $\mathbf{z}$, a menudo es conveniente generar un vector $\mathbf{w}$ tal que $w(i) = x(i)*z(i)$, esto se logra con la multiplicación elemento a elemento, la cual se define anteponiendo un punto antes del operador, en este caso tenemos que

```
w = x.*z
```

Por ejemplo, $\mathbf{x}.*\mathbf{x}$ es un vector cuyas componentes son $x(i)^2$. Otras operaciones elemento a elemento disponibles son

```
.*      ./      .^
```

Entonces para graficar $f(x) = 3x^3 + 2x - 3$, con el vector $\mathbf{x}$ generado anteriormente, tenemos que

```
y=3*x.^3+2*x-3
```

genera el vector de ordenadas. Entonces la gráfica deseada se obtiene con las instrucciones

```
x=linspace(-2,3,50)
y=3*x.^3+2*x-3
plot(x,y)
```

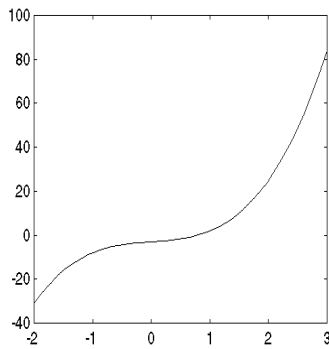


Figure 5: Gráfica del polinomio cúbico

cuya gráfica aparece en la figura 5.

Ahora deseamos graficar la función $f(x) = e^x \cos(x)$. Las instrucciones son las siguientes

```
x=linspace(-5,5,50)
y=exp(x) .* cos(x)
plot(x,y)
```

La figura 6, muestra la salida de estas instrucciones.

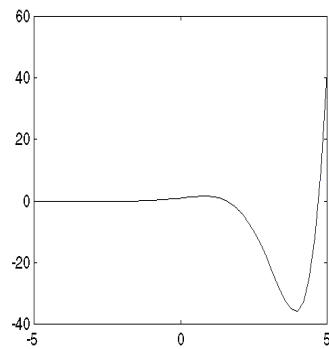


Figure 6: Gráfica de la función $e^x \cos(x)$

De igual manera para la llamada función “jorobas”, $f(x) = 1/((x-.3)^2+.01) + 1/((x-.9)^2+.04)$

- 6

```

x=linspace(0,2,50)
y=1./((x-.3).^2+.01) + 1./((x-.9).^2+.04) - 6
plot(x,y)

```

La gráfica es la figura 7.

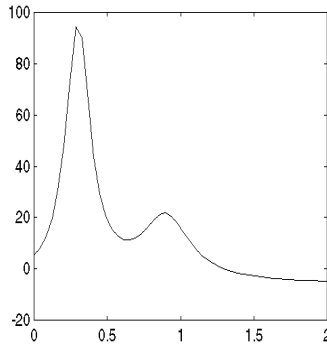


Figure 7: Gráfica de la función jorobas

Ejercicio

1. Introduzca

$$\mathbf{x}=[1 \ 2 \ 3], \quad \mathbf{y}=[4 \ 5 \ 6],$$

realice las siguientes operaciones elemento a elemento. En cada caso, exprese $z(i)$ en función de $x(i)$ y $y(i)$. Por ejemplo, si $\mathbf{z}=\mathbf{x}.*\mathbf{y}$ entonces $z(i) = x(i)*y(i)$.

```

z=x.*y  z=x./y
z=x.^2
z=x.^y
z=2.^[x y]
z=1./x
z=1./x.^2

```

2. Grafique $y = \sin^2(x)$ en $[0, 2\pi]$
3. Grafique $y = \cos^2(x)$ en $[0, 2\pi]$
4. Grafique $y = \sin^2(x) + \cos^2(x)$ en $[0, 2\pi]$

2 Un poco de programación

El desarrollo en series de $\arctan(x)$ proporciona un método para calcular el número pi. Veamos

$$\arctan(x) = x - x^3/3 + x^5/5 - x^7/7 + \dots \quad (1)$$

Recordando que $\tan(\pi/6) = 1/\sqrt{3}$ tenemos que $\pi/6 = \arctan(1/\sqrt{3})$, es decir, $\pi = 6\arctan(1/\sqrt{3})$

El programa **pi_1**, calcula una aproximación a pi usando n sumandos de la serie (1), grafica las aproximaciones para $i = 1 : n$ y también los errores de aproximación . Para ejecutarlo simplemente teclee el comando

pi_1

y siga las instrucciones. A continuación desplegamos el listado del programa **pi_1**

```
%
% Aproximacion a pi por el metodo de Gregory
%
clear
format short e
%
% n=numero de terminos de la sumatoria
%
n=input('Numero de terminos de la sumatoria ');
x=1/sqrt(3);
t=x;      % t = x^{(2k-1)} con su signo
atan(1)=t; % aqui se acumula la sumatoria
for i=2:n
    k=2*i-1;
    t=-t*(x^{2});
    atan(i)=atan(i-1)+t/k;
end
pi_aprox=6*atan;
plot(pi_aprox);
title('aproximacion a pi')
pause
error=abs((pi-pi_aprox)/pi);    % error relativo
plot(error)
title('Errores')
```

En el programa aparecen varias instrucciones nuevas:

- **%**
Se utiliza para comentarios, todo lo que está a la derecha de este signo será ignorado por *Matlab* al ejecutar el programa.
- **Clear**
Elimina de la memoria todas las variables utilizadas hasta este momento.
- **format short e**
Indica el formato en el que se desplegará los resultados, en este caso 5 dígitos y notación exponencial, *Matlab* siempre trabaja con aritmética de 16 dígitos, la instrucción **format** sólo se usa para desplegar los resultados. Use **help format** para ver todas las opciones.
- **Input**
Sirve para esperar un valor que será asignado a una variable, este valor es proporcionado por el usuario una vez que el programa es ejecutado.
- **for**
Indica un ciclo, que en este caso inicia en 2, en cada iteración aumenta el contador una unidad, hasta llegar al tope, en este caso n . La sintaxis de *for* es

```
for i=v
instrucciones
end
```

donde v es un vector, en este caso $v=2:n$.

- **pause**
Instrucción utilizada para indicar que los cálculos se detengan hasta que el usuario presione una tecla, en este caso con la gráfica en pantalla espera que se presione una tecla para continuar la ejecución del programa. *pause(n)* espera n segundos antes de continuar ejecutando el programa.

Una de las ventajas de *Matlab* es que se pueden escribir programas de una forma muy rápida y sencilla. Como otros lenguajes de programación, tiene estructuras de control como son las instrucciones **for**, **while**, **if-else**, etc. Estas instrucciones conjuntamente con su capacidad de graficación nos permiten realizar rápidamente experimentos computacionales. Esta es quizá la mayor fortaleza de *Matlab*. Frecuentemente nos interesa hacer un cierto cálculo con una precisión determinada. Por ejemplo, si queremos calcular π con 5 decimales, podemos escribir un programa que haga sólo los pasos necesarios para lograr tal aproximación. Aquí es donde la instrucción **while** es importante, porque no sabemos de antemano cuántos pasos tenemos que efectuar. En el programa **pi_2** se usa dicha instrucción en lugar de la instrucción **for**. La sintaxis para **while** es

```
while <condicion>

    instrucciones

end
```

El programa se ejecuta tecleando

pi_2

El listado del programa **pi_2.m** es el siguiente.

```
%
% Aproximacion a pi por el metodo de Gregory con 5 digitos
%
clear
format short e
x=1/sqrt(3);
t=x;           % t = x\^{ }(2k-1) con su signo
atan(1)=t;      % aqui se acumula la sumatoria
i=1;
pi_aprox(1) = 6*atan(1);
error(1) = abs((pi-pi_aprox(1))/pi);
while error > 1.e-5
    i=i+1;
    k=2*i-1;
    t=-t*(x\^{ }2);
    atan(i)=atan(i-1)+t/k;
    pi_aprox(i) = 6*atan(i);
    error(i) = abs((pi-pi_aprox(i))/pi);
end
plot(pi_aprox);
title('aproximacion a pi')
pause
plot(error)
title('Errores')
```

Matlab es capaz de realizar tareas sofisticadas con unas cuantas instrucciones. El programa **poligono.m** combina las instrucciones **for** y **subplot**. Use **type poligono** para desplegar el listado del programa. Ejecútelo.

Ejercicios. Conversiones de temperaturas. Los siguientes ejemplos generan tablas de temperaturas, use las siguientes relaciones entre temperaturas en grados Centígrados (T_C), grados Fahrenheit (T_F), grados Kelvin (T_K) y grados Rankin (T_R):

$$\begin{aligned}T_F &= T_R - 459.67 \\T_F &= (9/5)T_C + 32 \\T_R &= (9/5)T_K\end{aligned}$$

- a) Genere una tabla con las conversiones de Fahrenheit a Kelvin para valores de $0^\circ F$ a $200^\circ F$. Permita al usuario especificar el incremento en grados F entre cada línea.

- b) Genere una tabla con las conversiones de Centígrados a Rankin. Permita al usuario introducir la temperatura inicial y el incremento entre líneas. Imprima 25 líneas de la tabla.
- c) Genere una tabla con las conversiones de Centígrados a Fahrenheit. Permita al usuario introducir la temperatura inicial, el incremento entre líneas y el número de líneas de la tabla.

2.1 Mínimos Cuadrados

Un problema que se presenta con frecuencia es el de ajustar un conjunto de datos en el plano por medio de un polinomio. El método más usado es el de mínimos cuadrados. El programa contenido en **ajusta.m** realizar esta tarea. El archivo **censo.m** contiene los datos del censo de Estados Unidos de 1900 a 1990. Las componentes del vector **x** representan los años de los censos de 1900 a 1990 cada diez años y el vector **y** las correspondientes poblaciones. Ejecute **censo** y luego **ajusta**. *Matlab* tiene un programa de demostración muy interesante llamado **census**.

El programa **censumex** contiene los datos del censo de México de 1900 a 2000, ejecútelo.

3 Graficación de funciones en dos variables

Matlab cuenta con diferentes comandos que permiten generar las gráficas de funciones en dos variables. A continuación veremos algunos de los comandos y la forma en la que se utilizan para generar dichas gráficas.

A manera de ejemplo considérese la función $f(x, y) = \sin(\sqrt{x^2 + y^2}) / \sqrt{x^2 + y^2}$ y suponga que se quiere graficar para $x \in [-6\pi, 6\pi]$ y $y \in [-6\pi, 6\pi]$. Lo primero que se tiene que hacer para graficar la función anterior es definir una malla en el dominio de las dos variables, es decir, definir puntos en el interior del rectángulo delimitado por los extremos de los intervalos anteriores. Para esto primeramente se generan dos vectores que definirán los puntos sobre los cuales se va a generar la malla, para el ejemplo anterior serán

```
x = linspace(-6*pi,6*pi,100);
y = linspace(-6*pi,6*pi,100);
```

Una vez generados los vectores que definen los puntos de la malla, se procede a generar los puntos interiores de la malla utilizando el comando **meshgrid** como sigue

```
[X,Y] =meshgrid(x,y);
```

Los parámetros de salida X e Y en la instrucción anterior son dos matrices cuyas columnas son los vectores x y y. Ahora se define la función que se quiere graficar en términos de las dos matrices anteriores, es decir,

```
Z = sin(sqrt(X.*X +Y.*Y))./ sqrt(X.*X +Y.*Y);
```

Por último haciendo uso del comando **mesh** se genera la gráfica de la función anterior como sigue

`mesh(Z)`

dando como resultado la gráfica de la figura 8.

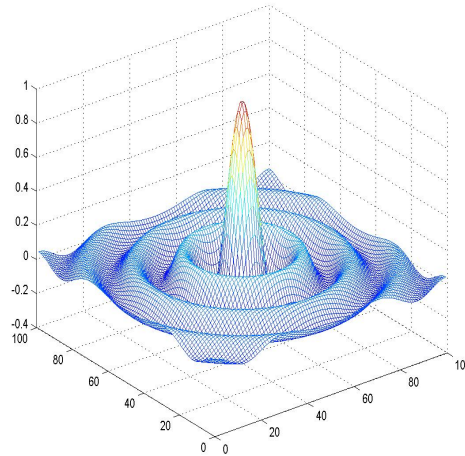


Figure 8: Gráfica de la función $f(x, y) = \frac{\text{sen}\left(\sqrt{x^2 + y^2}\right)}{\sqrt{x^2 + y^2}}$ utilizando la instrucción mesh

Existen otros comandos que trabajan en forma muy similar al mesh, como el **meshz** y el **waterfall**. Para ver las gráficas que dichos comandos producen utilice la matriz Z del ejemplo anterior y haciendo uso de las instrucciones meshz(Z) y waterfall(Z) genere en una misma figura las dos gráficas que producen los comando anteriores.

Ejercicios Para cada una de las siguientes funciones en dos variables, utilice los comandos **mesh**, **meshz** y **waterfall** para generar las gráficas correspondientes en los dominios especificados, poniéndolas en la misma figura.

1. $f(x, y) = x^4 - y^4 \quad x \in [-20, 20], \quad y \in [-20, 20]$
2. $f(x, y) = 4x^2 - 2.1x^4 + \frac{1}{3}x^6 + xy - 4y^2 + 4y^4 \quad x \in [-3, 3], \quad y \in [-1.5, 1.5]$
3. $f(x, y) = \text{sen}(x + y) + 1 - \frac{x^2}{40} + e^{y^2} \quad x \in [-6, 10], \quad y \in [-4, 20]$
4. $f(x, y) = 4\cos(x) + 2\text{sen}(y) - 5y \quad x \in [-3, 3], \quad y \in [-3, 3]$

En las gráficas anteriores el interior de la retícula de la superficie es incoloro; podría resultar más atractivo graficar las superficies con color. Para esto *Matlab* le ofrece al usuario comandos como el

surf que trabaja de la misma manera que el comando **mesh**, excepto que las gráficas de la funciones resultan coloreadas, por ejemplo si ya se tiene la matriz de la función que se quiere graficar, en este caso Z, al teclear

```
surf(Z)
```

se producirá la gráfica de la figura 9.

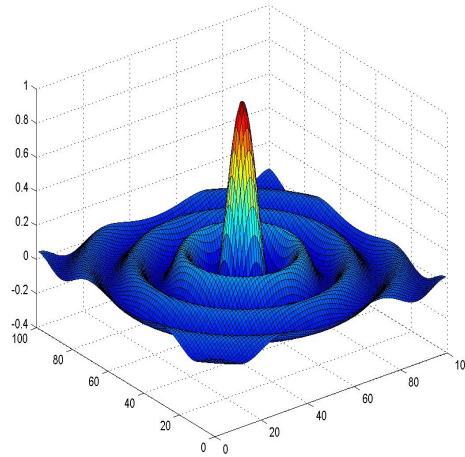


Figure 9: Gráfica de la función $f(x, y) = \frac{\sin(\sqrt{x^2 + y^2})}{\sqrt{x^2 + y^2}}$ utilizando la instrucción surf

El color interior de las retículas puede ser modificado utilizando la instrucción **colormap** y a continuación el nombre de los mapas de colores predefinidos por *Matlab* como **hsv**, **pink**, **gray**, **jet**, **colorcube**, **winter** y **hot**, entre otros. Para ver el efecto de cada colormap sobre la gráfica anterior, teclee en la pantalla de comandos de *Matlab*

```
colormap pink
colormap hot
⋮
```

Una manera más atractiva de generar superficies es utilizando el comando **surf1**, que funciona igual que los comandos ya mencionados. Para aplicar este comando a nuestra función anterior, es decir, a la matriz Z, se debe teclear

```
surf1(Z)
```

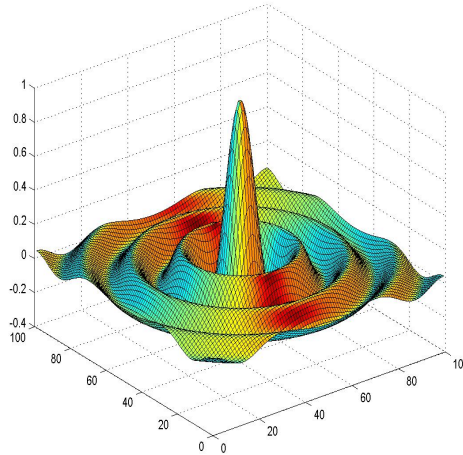


Figure 10: Gráfica de la función $f(x, y) = \frac{\sin(\sqrt{x^2 + y^2})}{\sqrt{x^2 + y^2}}$ utilizando la instrucción `surf`

y se obtiene la figura 10.

La diferencia que tiene esta gráfica con las anteriores es que se agrega luz al ambiente de la gráfica, obteniéndose una mejor presentación. De igual manera se puede cambiar el mapa de colores para lograr mejores resultados.

Una vez que se genera la gráfica de la función en dos variables es posible mejorar la presentación de la malla. Para esto existe el comando **shading** que controla la forma en que los bordes de la malla aparecen en la gráfica, siendo tres las opciones disponibles que son **faceted** (default), **flat** e **interp**. Para aplicar una de las opciones anteriores a la gráfica con la que se está trabajando, es necesario, después de haber generado la superficie mediante los comandos `surf(Z)`, `mesh(Z)` o `surfl(Z)`, aplicar la instrucción

`shading interp`

dando como resultado la figura 11.

Ejercicios Grafique las siguientes funciones en dos variables. Para cada función genere en una misma figura las gráficas utilizando las tres opciones del comando **shading**.

1. $f(x, y) = x^4 - y^4$, para $x \in [-20, 20]$, $y \in [-3, 3]$
2. $f(x, y) = 4x^2 - 2.1x^4 + \frac{1}{3}x^6 + xy - 4y^2 + 4y^4$ $x \in [-3, 3]$, $y \in [-3, 3]$

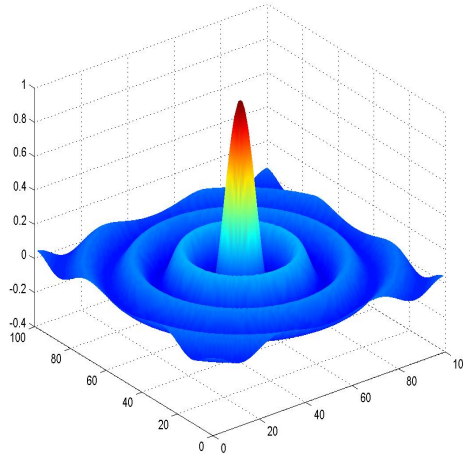


Figure 11: Graficación utilizando la instrucción shading interp

3.1 Curvas de nivel

Para funciones en dos variables muchas de las veces es conveniente graficar lo que se conoce como curvas o contornos de nivel ya sea en 2 o en 3 dimensiones. Para esto *Matlab* dispone de algunos comandos que permiten fácilmente generar dichas gráficas. Uno de estos comandos es **contour**, que grafica las curvas de nivel de la función en el plano. Para usarlo es necesario generar previamente la matriz de evaluación, a la que hemos estado llamando Z . Para ejemplificar consideramos la función $f(x, y) = (y^3 - 3y)/(1 + x^2)$ definida en $x, y \in [-1.5, 1.5]$ cuya gráfica se genera siguiendo las instrucciones descritas en la sección anterior. Empleando

```
surf(X,Y,Z)
```

se obtiene la figura 12. ¿Qué diferencia se tiene respecto a las gráficas anteriores en los ejes?

Para graficar las curvas de nivel de la figura anterior simplemente teclee

```
contour(Z)
```

lo que producirá la gráfica de la figura 13.

En la instrucción anterior el valor empleado para generar cada una de las curvas de nivel es seleccionado automáticamente por *Matlab*, sin embargo se puede especificar para qué valor de la función se requiere la gráfica de la curva de nivel. Por ejemplo si se desea graficar la curva de nivel para $Z = 0.5$ entonces se introduce la instrucción

```
contour(Z,[0.5 0.5])
```

dando como resultado la figura 14.

Si se requieren las curvas de nivel para diferentes valores de Z , estos se especifican en un vector V , utilizando la sintaxis **contour(Z,V)** o **contour(X,Y,Z,V)**.

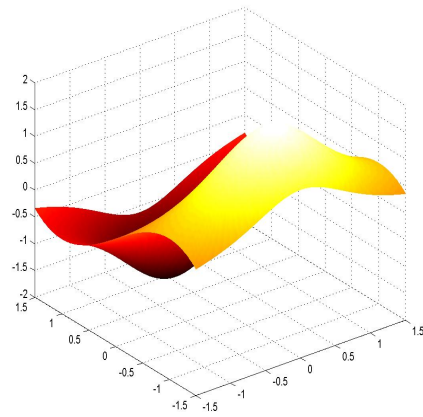


Figure 12: Gráfica de la función $f(x, y) = \frac{(y^3 - 3y)}{(1 + x^2)}$

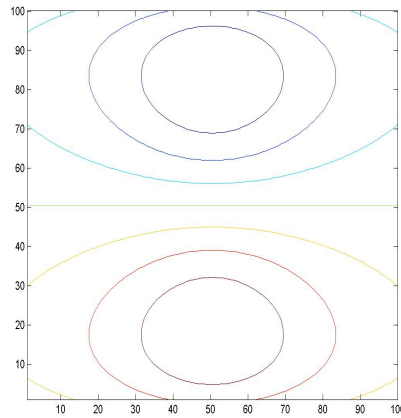


Figure 13: Gráfica de las curvas de nivel de la función $f(x, y) = \frac{(y^3 - 3y)}{(1 + x^2)}$

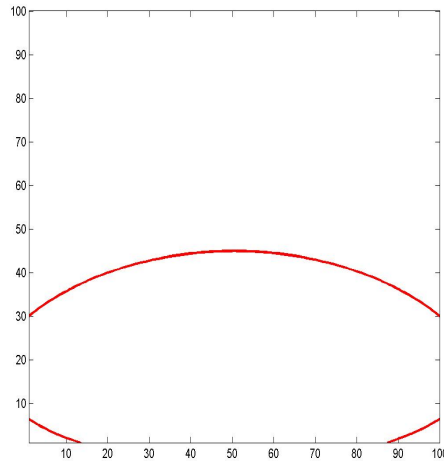


Figure 14: Gráfica de la curva de nivel de valor 0.5 de la función $f(x, y) = (y^3 - 3y)/(1 + x^2)$

Ejercicios Utilizando **contour(X,Y,Z)**, **contour(X,Y,Z,N)** (donde N es un natural) y **contour(X,Y,Z,[v₁, v₂, ...])** grafique las curvas de nivel para los valores especificados en cada inciso.

1. $f(x, y) = \text{sen}(y^2 + x) - \cos(y - x^2)$
 $x, y \in [0, \pi]$
2. $f(x, y) = \text{sen}(3y - x^2 + 1) + \cos(2y^2 - 2x)$
 $x, [-2, 2] \quad y \in [-1, 1], N = 20$
3. $f(x, y) = \text{sen}(y^2 + x) - \cos(y - x^2) \quad x, y \in [0, \pi],$
 nivel de curva = -1.2, -0.5, 0, 1.2

Cuando se grafican curvas de nivel en el plano algunas veces es difícil identificar que curva corresponde a un valor específico de Z, por eso es conveniente etiquetar cada curva que se grafica. Esto se puede hacer en *Matlab* con la instrucción **clabel**.

Para emplear el comando **clabel** es necesario que al momento de generar las gráficas de las curvas de nivel con el comando **contour** se asignen dos parámetros de salida a dicho comando. El primero de ellos será una matriz y el segundo, un vector columna que se empleará por el comando **clabel** como argumentos de entrada.

Por ejemplo, si denotamos como C y H a los parámetros de salida del comando **contour** las instrucciones que deben teclearse son

```
[C,H]=contour(X,Y,Z);
clabel(C,H)
```

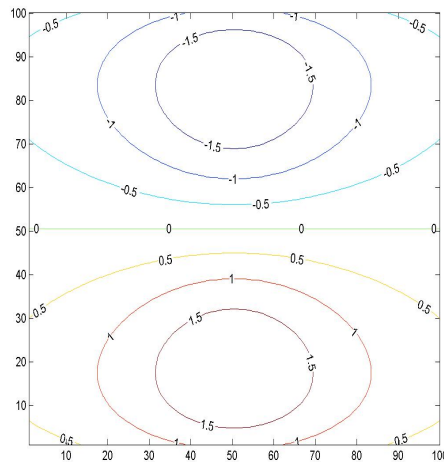


Figure 15: Gráfica de las curvas de nivel mostrando los valores

produciendo así la gráfica de la figura 15.

Esto permite al usuario identificar el nivel correspondiente a cada gráfica.

Si el usuario quiere visualizar en la misma figura las curvas de nivel y la gráfica de la función, puede hacerlo utilizando los comandos **surf** y **meshc**, cuya sintaxis es la misma que emplea el comando **surf**.

Ejercicio:

Para las funciones del ejercicio anterior genere la gráfica de la función y sus curvas de nivel, etiquetando cada una de ellas.

3.2 Campo direccional

Si se tienen las gráficas de las curvas de nivel es posible darse una idea de cómo es el comportamiento de la función en dos variables a través del campo direccional. En *Matlab* el campo direccional se genera con el comando **quiver**, pero este a su vez requiere los datos de las derivadas direccionales en cada punto de la matriz Z o los valores de la función. A continuación veremos como generar el campo direccional del ejemplo con el que se ha estado trabajando.

Primero se debe generar la matriz Z

```
x = -2:.2:2;
y = x;
[xx, yy] = meshgrid(x,y);
Z = (yy.^3 - 3*yy)./(1+xx.^2);
```

Después de haber generado la matriz Z es necesario construir dos matrices, F_x y F_y , cuyos elementos sean los valores de las derivadas direccionales de la función en los puntos en que fue evaluada; esto se logra mediante el comando

```
[Fx, Fy]=gradient(Z);
```

finalmente, con la instrucción

```
quiver(Fx,Fy)
```

se obtiene la gráfica de la figura 16.

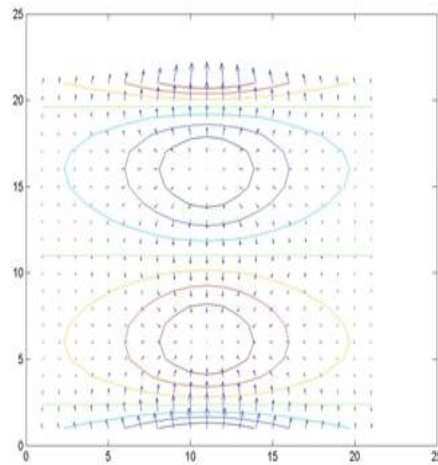


Figure 16: Gráfica del campo direccional utilizando la instrucción quiver

Las curvas de nivel que se generan con el comando **contour** aparecen como gráficas en el plano, sin embargo podría resultar más atractivo que dichas gráficas aparecieran en 3D. Esto se puede lograr con el comando **contour3**. Para ver qué resultados produce teclee

```
contour3(Z)
```

donde Z es la matriz de la función del ejemplo con el que se ha estado trabajando en esta sección . El comando anterior produce la figura 17.

Ejercicio:

Grafique las curvas de nivel en 3D para las siguientes funciones.

1. $Z=\text{peaks}$
2. $f(x, y) = x^3 + 4xy - 2y^2, \quad x \in [-1, 1], y \in [-1, 1]$
3. $f(x, y) = (1 - y^2)\cos(x), \quad x \in [-1, 1], y \in [-1, 1]$

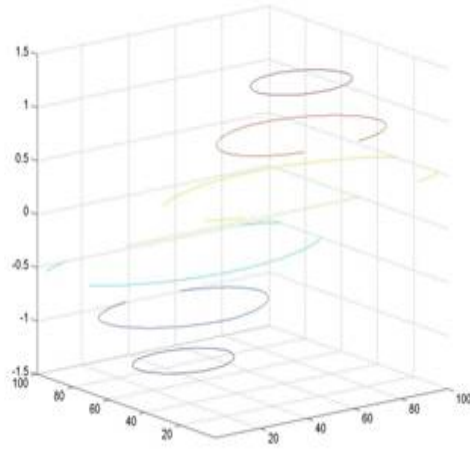


Figure 17: Graficación utilizando contour3

4 Manipulación de matrices

Veremos ahora una de las principales fortalezas de *Matlab*: el manejo de matrices. Iniciemos viendo cómo introducir una variable que contenga una matriz. Considérese la matriz

$$a = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

en *Matlab* dicha matriz se introduce de la siguiente forma:

```
a=[1 2;3 4]
```

o bien

$$a = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Algo que resulta particularmente útil cuando no es posible escribir una instrucción de *Matlab* en una sola línea, es teclear tres puntos al final del renglón, para indicar que se espera en la línea siguiente la continuación del comando. Por ejemplo, para introducir la matriz a se usaría:

$$a = \begin{bmatrix} 1 & 2; \dots \\ 3 & 4 \end{bmatrix}$$

Para hacer referencia al elemento del renglón i y la columna j de la matriz a , se utiliza la siguiente notación:

$$a(i, j)$$

Por ejemplo, teclear `a(2,1)` y se obtendrá por resultado 3.

El i -ésimo renglón de a se obtiene por:

$$a(i, :)$$

y para la j -ésima columna de a se utiliza:

$$a(:,j)$$

Por ejemplo teclear

$$a(1,:)$$

para obtener el vector que contiene el primer renglón de la matriz a y utilice la instrucción

$$a(:,2)$$

para obtener la segunda columna. Podemos alterar un sólo elemento de la matriz sin necesidad de modificarla toda. Por ejemplo

$$a(1,2)=5$$

nos produce

$$a = \begin{pmatrix} 1 & 5 \\ 3 & 4 \end{pmatrix}$$

y con

$$a(2,1)=a(2,2)$$

obtenemos

$$a = \begin{pmatrix} 1 & 5 \\ 4 & 4 \end{pmatrix}$$

Para introducir vectores se procede como sigue: El vector $x = (1 \ 2 \ 3)$, se introduce como

$$x=[1 \ 2 \ 3]$$

para vectores columna

$$x=[1 \ 2 \ 3] ',$$

o bien

$$x=[1;2;3].$$

4.1 Variables e instrucciones

Las variables pueden tener una longitud de hasta 19 caracteres comenzando con una letra.

Matlab distingue las variables mayúsculas de las minúsculas, es decir A y a pueden tener valores distintos.

La forma de introducir instrucciones en *Matlab* es la siguiente

$$variable = expresión$$

o simplemente

$$expresión$$

Por ejemplo al introducir $x = \exp(5 - \sin(\pi/2))$ produce

$$x = 54.5982$$

mientras que si no se asigna una variable a la expresión, el resultado obtenido se guarda en la variable **ans**, por ejemplo, si se introduce

$$\exp(5 - \sin(\pi/2))$$

se obtendrá

$$ans = 54.5982$$

Si el último carácter de una instrucción es un punto y coma, no se despliega en pantalla el resultado, pero si éste se asigna a una variable, guardará el nombre de la variable y su valor.

Es posible conocer las dimensiones de una matriz (es decir, la cantidad de renglones y columnas que la componen) mediante la función **size**:

$$[m,n] = \text{size}(A)$$

Por otro lado, si v es un vector renglón o un vector columna, la instrucción

$$n = \text{length}(v)$$

asigna a n la dimensión de v .

Como ejercicio teclear

$$\begin{aligned} [m,n] &= \text{size}(a) \\ n &= \text{length}(x) \end{aligned}$$

También existen algunas instrucciones ya definidas en *Matlab*, por ejemplo se puede calcular la inversa de la matriz A utilizando el comando

$$\text{inv}(A)$$

De igual forma $\det(A)$ calcula el determinante de la matriz A , $\text{rank}(A)$ proporciona su rango, por mencionar algunas.

Ejercicio Introducir la siguiente matriz, calcular su rango, inversa y determinante.

$$\begin{aligned} A &= \text{hilb}(4) \\ &\text{inv}(A) \\ &\det(A) \\ &\text{rank}(A) \end{aligned}$$

Matlab cuenta con distintos formatos de salida para las variables numéricas, por ejemplo si desplegamos el valor de π (con la instrucción **pi**) en diversos formatos obtenemos:

FORMATO	VALOR DE PI
format short	3.1416
format short e	3.1416e+00
format long	3.14159265358979
format long e	3.141592653589793e+00

Para verificarlo, teclear

```
format short
pi
format short e
pi
```

Existen otras variantes que pueden leerse tecleando `help format`.

4.2 Dimensiones variables

Se puede cambiar la dimensión de una matriz redefiniéndola con una nueva dimensión, por ejemplo si $b = [1 \ 2 \ 3; 4 \ 5 \ 6]$ es una matriz de 3×2 , si la redefinimos por $b = [3 \ 8; 9 \ 5]$ entonces b será una matriz de 2×2 .

La instrucción $c(3,2) = 2$ nos produce una matriz de 3×2 cuyo elemento $(3,2)$ es 2 y en las otras entradas tendrá ceros. Esto es,

```
c(3,2)=2
```

produce

$$c = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 2 \end{pmatrix}$$

```
d(1,3)=1
```

nos produce el vector

$$(0, 0, 1)$$

y la instrucción `c=3` asigna a c el valor constante de 3.

4.3 Operaciones con Matrices

Si a y b son dos matrices de igual dimensión, podemos efectuar la operación suma, resta, multiplicación, por ejemplo defina las matrices a y b como

$$a = \begin{pmatrix} 1 & 3 \\ 7 & 9 \end{pmatrix} \quad b = \begin{pmatrix} 2 & 6 \\ 4 & 5 \end{pmatrix}$$

entonces podemos calcular su suma con la siguiente instrucción:

```
c=a+b
```

lo que nos producirá

$$c = \begin{pmatrix} 3 & 9 \\ 11 & 14 \end{pmatrix}$$

En forma análoga para la resta, la instrucción

```
c=a-b
```

resulta

$$c = \begin{pmatrix} -1 & -3 \\ 3 & 4 \end{pmatrix}$$

La multiplicación viene dada por

$$c=a*b$$

obtenemos

$$c = \begin{pmatrix} 14 & 21 \\ 50 & 87 \end{pmatrix}$$

cuidando las dimensiones para poderla efectuar.

Una vez definida una matriz a , la transpuesta se puede obtener con la instrucción

$$c=a'$$

En nuestro ejemplo se obtiene

$$c = \begin{pmatrix} 1 & 7 \\ 3 & 9 \end{pmatrix}$$

también podemos redefinir a como su transpuesta, esto es,

$$a=a'$$

Otra de las operaciones que podemos efectuar con matrices es elevarlas a una potencia dada, por ejemplo para elevar al cuadrado la matriz a , tenemos

$$c = a^2$$

produce

$$c = \begin{pmatrix} 22 & 30 \\ 70 & 102 \end{pmatrix}$$

de esta forma $c = a * a$. Y $c = a^3$ nos produce $c = a * a * a$.

Ejercicios.

1. Defina las siguientes matrices

$$A = \begin{pmatrix} 1 & -2 \\ 4 & 6 \end{pmatrix}$$

y

$$B = \begin{pmatrix} 2 & 3 \\ -5 & 9 \end{pmatrix}$$

Calcule $A * B$ y $B * A$, compárelas.

2. Considere ahora

$$A = \begin{pmatrix} 3 & -7 \\ 7 & 3 \end{pmatrix}$$

y

$$B = \begin{pmatrix} 5 & 2 \\ -2 & 5 \end{pmatrix}$$

Calcule $A * B$ y $B * A$, compárelas. ¿Hay una contradicción entre los resultados?

3. La aritmética de matrices no es la misma que la aritmética de los números reales, los siguientes ejemplos dan muestra de ello.

(a) Sea

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

verifique que $A^2 = 0$.

(b) Sea

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

verifique que $A^2 = A$.

(c) Sea

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

verifique que $A^2 = I$.

(d) Sean

$$A = \begin{pmatrix} 1 & -1 \\ -2 & 2 \end{pmatrix}$$

y

$$X = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}$$

verifique que $A * X = 0$.

(e) Sean

$$A = \begin{pmatrix} 1 & -1 \\ -2 & 2 \end{pmatrix}$$

$$B = \begin{pmatrix} 3 & 5 \\ 3 & 5 \end{pmatrix}$$

y

$$C = \begin{pmatrix} 2 & 3 \\ 2 & 3 \end{pmatrix}$$

verifique que $A * B = A * C$ para $B \neq C$, así que en general, no se satisface la cancelación.

4. Análogo a los números reales, podemos definir funciones polinomiales con matrices cuadradas, por ejemplo si consideramos en polinomio, $p(x) = 3x^2 + 5x - 1$, se puede definir la función

$$3A^2 + 5A - I$$

donde I representa la matriz identidad del mismo orden de A .

(a) Definir una matriz A de dimensión 2×2 . Evaluar el polinomio anterior en esta matriz.

(b) Definir

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$$

calcular

$$A^2 - 2 * A - 3 * I$$

Otras de las operaciones que se pueden realizar son las siguientes, si queremos calcular $a^{-1} * b$, tecleamos la instrucción:

$$c = a \backslash b$$

lo que resulta

$$c = \begin{pmatrix} -0.5000 & -3.2500 \\ 0.8333 & 3.0833 \end{pmatrix}$$

En particular para resolver un sistema de ecuaciones lineales de la forma $Ax = b$, cuya solución es $x = A^{-1}b$, en *Matlab* utilizamos la instrucción

$$x = A \backslash b$$

Ejercicio. Defina una matriz A de 3×3 y un vector b de 3×1 , calcule

$$x = A \backslash b$$

verifique que x es solución del sistema lineal, calcule

$$A * x$$

y compara con b .

4.4 Algunas matrices especiales

4.4.1 Matriz identidad

La matriz identidad de orden n se obtiene por medio de la instrucción

$$a = \text{eye}(n)$$

donde n es el orden de la matriz, así $a = \text{eye}(4)$ nos produce la matriz identidad de orden 4.

También podemos obtener matrices no cuadradas, así

$$a = \text{eye}(m,n)$$

nos produce una matriz de $m \times n$, con unos en la diagonal principal y ceros en las demás entradas.

4.4.2 Matriz de ceros

También podemos producir una matriz cuyas entradas sean todas iguales a cero. La matriz de orden n con estas características se obtiene con la siguiente instrucción:

$$a = \text{zeros}(n)$$

y

$$a = \text{zeros}(m,n)$$

nos produce una matriz de orden $m \times n$ con entradas cero.

4.4.3 Matriz de unos

Otra de las matrices muy utilizadas es la matriz con todas sus entradas iguales a uno. La matriz de orden n de unos se obtiene con:

$$a = \text{ones}(n)$$

para matrices no cuadradas se obtiene con $a = \text{ones}(m, n)$. ¿Cómo obtenemos una matriz con todas sus entradas iguales a dos?

4.4.4 Matriz aleatoria

Podemos obtener matrices de orden n generadas aleatoriamente con distribución uniforme en $[0, 1]$ en la siguiente forma:

$$a = \text{rand}(n)$$

Análogamente

$$a = \text{rand}(m, n)$$

genera una matriz aleatoria de $m \times n$.

Ejercicio.

1. Introducir las siguientes instrucciones y describir brevemente lo que sucede en cada caso.

```
a=eye(5)
a=magic(5)
b=eye(a)
a=zeros(3,4)
a=zeros(3,3)
a=zeros(3)
a=magic(5)
b=zeros(a)
b=triu(a)
c=tril(a)
d=tril(a,-1)
b=[a zeros(3,2); zeros(2,3) eye(2)]
```

2. Repetir lo anterior con ones en lugar de zeros.

4.5 Operaciones elemento a elemento

Podemos efectuar operaciones con matrices que operen elemento a elemento, si A y B son dos matrices dadas y

$c = a.*b$	entonces	$c_{ij} = a_{ij} * b_{ij}$
$c = a./b$	entonces	$c_{ij} = a_{ij} / b_{ij}$
$c = a.\backslash b$	entonces	$c_{ij} = b_{ij} / a_{ij}$
$c = a.^b$	entonces	$c_{ij} = a_{ij}^{b_{ij}}$

Para estas operaciones se requiere que las dimensiones de las matrices sean iguales.

Ejercicios.

1. Generar las matrices

$$A = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}, \quad B = \begin{pmatrix} -2 & 1 \\ 3 & 6 \end{pmatrix}$$

Calcular

`A.*B`
`A./B`
`A.\ B`

2. Introducir las siguientes instrucciones.

```
x = 1 : 6  
a = diag(x)  
b = diag(a)  
a = diag(x, 1)  
a = diag(x, 2)  
a = diag(x, -3)  
a = 2*diag(x)
```

3. Generar la matriz

$$A = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix}$$

usando el comando `diag`.

4. Generar

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 \end{pmatrix}$$

4.6 Subíndices

Para referirnos a los elementos de una matriz A , utilizamos los subíndices entre paréntesis, por ejemplo si la matriz A está definida como

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

La instrucción

$$c=A(3,2)$$

produce $c = 8$. Un subíndice puede ser un vector, si x y v son vectores, entonces $x(v)$ es $[x(v(1)), x(v(2)), \dots, x(v(n))]$.

Por ejemplo, genere una matriz A de (10×10) , entonces

$$A(1:5,3)$$

especifica una submatriz de (5×1) , o un vector columna que consiste en los primeros 5 elementos de la tercer columna de A . Similarmente,

$$A(1:5,7:10)$$

es la submatriz de (5×4) con los elementos del primer al quinto renglón de las últimas cuatro columnas de A .

Si se usan los dos puntos en lugar de un subíndice, esto corresponde a considerar todos los renglones o todas las columnas de la matriz, por ejemplo,

$$A(:,3)$$

es la tercera columna de A , y

$$A(1:5,:)$$

son los primeros cinco renglones de la matriz. Verifíquelo para la matriz A .

El uso de vectores como subíndices es de gran ayuda, y se pueden efectuar operaciones mas complejas, por ejemplo genere una matriz B de 10×10 , la instrucción

$$A(:, [3 \quad 5 \quad 10])=B(:, 1:3)$$

reemplaza las columnas 1,5 y 10 de la matriz A por las primeras 3 columnas de la matriz B . En general, si v y w son vectores con componenetes enteras, entonces

$$A(v,w)$$

es la matriz obtenida tomando los elementos de A con los subíndices de renglones de v y subíndices de columnas de w . Entonces

$$A(:,n:-1:1)$$

reordena las columnas de A de derecha a izquierda.

Otra de las instrucciones que se pueden utilizar es

$$b=A(:)$$

indica que b es un vector columna que contiene todos los elementos de A . Por ejemplo

$$A = [1 \quad 2; 3 \quad 4; 5 \quad 6]$$

$$b = A(:)$$

produce

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$
$$b = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix}$$

La instrucción $b = A(:)$ puede ser usada para reconstruir una matriz. Para hacer esto, A debe estar definida anteriormente. Entonces $A(:) = v$ denota una matriz con las mismas dimensiones de A , pero con las componentes del lado derecho, por ejemplo en el caso anterior en que A tiene 3 renglones y dos columnas, la instrucción

$$A(:) = 11:16$$

reconstruye los 6 elementos de un vector renglón en la matriz de (3×2) ,

$$A = \begin{pmatrix} 11 & 14 \\ 12 & 15 \\ 13 & 16 \end{pmatrix}$$

4.7 Matrices vacías

Es posible crear matrices vacías, la instrucción

$$x = [\]$$

asigna a x una matriz de dimensión cero por cero. Esta instrucción es diferente de

$$\text{clear } x$$

la cual elimina x de la lista de variables actuales. Las matrices vacías son de tamaño cero. La función `exist` puede ser usada para probar la existencia de una matriz, mientras que la instrucción `isempty` indica si una matriz es vacía.

Es posible generar vectores vacíos. Si n es un número menor que 1, entonces $1:n$ no contiene elementos y

$$x = 1:n$$

es una manera complicada de crear un vector vacío.

Una manera eficiente de eliminar renglones y columnas de una matriz es asignárselas a una matriz vacía. Entonces

$$A(:, [2 \ 4]) = [\]$$

elimina las columnas 2 y 4 de A .

4.8 Almacenar y recuperar variables

Antes de salir de *Matlab* las variables utilizadas pueden ser salvadas para ser usadas en otra sesión de *Matlab*, esto se hace con el comando

```
save
```

Esta instrucción guarda todas las variables en un archivo llamado `matlab.mat`. En la siguiente sesión de *Matlab* las variables pueden ser restauradas del archivo `matlab.mat` con el comando

```
load
```

`save` y `load` pueden ser usados con un nombre específico de un archivo o guardar sólo algunas de las variables. El comando

```
save temporal
```

almacena las variables actuales en el archivo `temporal.mat`. El comando

```
save temporal x
```

salva solamente la variable x , y

```
save temporal x y z
```

guarda las variables x , y y z .

```
load temporal
```

restaura todas las variables almacenadas en el archivo `temporal.mat`.

Ejercicio. Salvar la matriz A actual en un archivo, siga las siguientes instrucciones.

1. Teclear el comando

```
whos
```

con éste se describen las variables actuales y su dimensión, verificar qué variables están almacenadas en este momento en *Matlab*.

2. Salvar la matriz A en un archivo, llamado `abril22`, con la instrucción

```
save abril22 A
```

3. Eliminar la variable A de las variables actuales, esto se hace con la instrucción

```
clear A
```

si solo se da la instrucción `clear` sola, se eliminan todas las variables de memoria.

4. Verificar que se haya eliminado la variable A , teclear nuevamente `whos`.
5. Leer el archivo donde se guardó la matriz A con la instrucción

```
load abril22
```

y teclear nuevamente `whos` para ver que efectivamente la matriz `A` es una variable actual del sistema.

El formato utilizado por `save` solo puede ser leído desde *Matlab*, si se desea guardar datos para utilizarlos con otros programas se puede usar

```
save abril22a.dat A -ascii
```

así se guardará la matriz con formato `ascii`, éste archivo puede ser leído desde el block de notas o algún otro programa, conviene ponerle la extensión `.dat` para diferenciarlo de los archivos `.mat`

4.9 Funciones matemáticas elementales

Las funciones trigonométricas disponibles en *Matlab* son:

<code>sin</code>	seno
<code>cos</code>	coseno
<code>tan</code>	tangente
<code>asin</code>	arco seno
<code>acos</code>	arco coseno
<code>atan</code>	arco tangente
<code>atan2</code>	arco tangente para los cuatro cuadrantes
<code>sinh</code>	seno hiperbólico
<code>cosh</code>	coseno hiperbólico
<code>tanh</code>	tangente hiperbólica
<code>asinh</code>	arco seno hiperbólico
<code>acosh</code>	arco coseno hiperbólico
<code>atanh</code>	arco tangente hiperbólica

Las funciones elementales disponibles son:

<code>abs</code>	valor absoluto o magnitud de un número complejo
<code>sqrt</code>	raíz cuadrada
<code>real</code>	parte real
<code>imag</code>	parte imaginaria
<code>conj</code>	complejo conjugado
<code>round</code>	redondeo al entero más cercano
<code>fix</code>	redondeo hacia cero
<code>floor</code>	redondeo hacia $-\infty$
<code>ceil</code>	redondeo hacia ∞
<code>sign</code>	función signo
<code>rem</code>	residuo o módulo
<code>exp</code>	exponencial de base e
<code>log</code>	logaritmo natural
<code>log10</code>	logaritmo base 10

4.10 Formato racional

Matlab permite trabajar los vectores y matrices en formato racional, por ejemplo construir una matriz de Hilbert de 4×4 con la instrucción

```
A=hilb(4)
```

que produce

$$A = \begin{pmatrix} 1.0000 & 0.5000 & 0.3333 & 0.2500 & 0.2000 \\ 0.5000 & 0.3333 & 0.2500 & 0.2000 & 0.1667 \\ 0.3333 & 0.2500 & 0.2000 & 0.1667 & 0.1429 \\ 0.2500 & 0.2000 & 0.1667 & 0.1429 & 0.1250 \\ 0.2000 & 0.1667 & 0.1429 & 0.1250 & 0.1111 \end{pmatrix}$$

la podemos desplegar en formato racional, la instrucción es

```
A=rats(A)
```

lo que nos produce

$$A = \begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 & 1/5 \\ 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \end{pmatrix}$$

5 Ejercicios

En los siguientes ejercicios se necesitará leer un archivo de datos, en el primero el archivo se llama `proteina.dat`, para leer este archivo se utiliza la instrucción `load` y enseguida el nombre del archivo, en nuestro ejemplo,

```
load proteina.dat
```

el archivo se lee y se guardan los datos en una matriz que tiene el mismo nombre del archivo, por ejemplo la variable se llama `proteina` y se puede desplegar su contenido en pantalla y utilizar la función `size` para ver su tamaño, se manipula igual que un variable tipo matriz. Siempre que se lean archivos de datos, deben tener el formato de matriz y tener igual número de columnas por renglón.

Aminoácidos Un secuenciador de proteínas es un equipo avanzado que desempeña un papel clave en la ingeniería genética. El secuenciador puede determinar el orden de los aminoácidos que constituyen una molécula de proteína, similar a una cadena. El orden de los aminoácidos ayuda a los ingenieros genéticos a identificar el gen que produjo la proteína. Se usan enzimas para disolver los enlaces con los genes vecinos, separando así el valioso gen ADN. A continuación, el gen se inserta en otro organismo, como una bacteria, que se multiplicará, multiplicando también el gen ajeno.

Aunque sólo existen 20 aminoácidos distintos en las proteínas, las moléculas de proteína tienen cientos de aminoácidos unidos en un orden específico. En este ejercicio suponemos que se ha identificado la secuencia de aminoácidos de una molécula de proteína y que deseamos calcular el peso molecular de la molécula de proteína. La siguiente tabla muestra un listado alfabético de los aminoácidos, su referencia de tres letras y pesos moleculares.

	Aminoácido	Referencia	Peso Molecular
1.	Alanina	Ala	89
2.	Arginina	Arg	175
3.	Asparagina	Asn	132
4.	Aspártico	Asp	132
5.	Cisteína	Cys	121
6.	Glutámico	Glu	146
7.	Glutamina	Gln	146
8.	Glicina	Gly	75
9.	Histidina	His	156
10.	Isoleucina	Ile	131
11.	Leucina	Leu	131
12.	Lisina	Lys	147
13.	Metionina	met	149
14.	Fenilalanina	Phe	165
15.	Prolina	Pro	116
16.	Serina	Ser	105
17.	Treonina	Thr	119
18.	Triptofano	Trp	203
19.	Tirosina	Tyr	181
20.	Valina	Val	117

Suponemos que se tiene un secuenciador de proteínas que genera un archivo de datos que contiene el número y tipo de moléculas de aminoácidos en cada molécula de proteína. Cada línea del archivo de datos corresponde a una proteína y cada línea contiene 20 enteros que corresponden a los 20 aminoácidos en el orden alfabético de la tabla. Cada entero indica cuántos de ese aminoácido en particular hay en la proteína. Por ejemplo, una línea que contiene los siguientes valores representa la proteína LysGluMetAspSerGlu:

0 0 0 1 0 2 0 0 0 0 0 1 1 0 0 1 0 0 0 0

Si queremos calcular el peso molecular de esta proteína, tenemos que identificar los pesos moleculares correspondientes de los aminoácidos, que en este caso son

147, 146, 149, 132, 105, 146

por lo que el peso molecular es de 825. Esto se puede calcular con un producto punto entre el vector de proteínas y el de pesos moleculares. Esto se puede hacer utilizando el producto de matrices, como se muestra en el siguiente ejemplo para dos proteínas dadas,

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 89 \\ 175 \\ 132 \\ 132 \\ 121 \\ 146 \\ 146 \\ 75 \\ 156 \\ 131 \\ 131 \\ 147 \\ 149 \\ 165 \\ 116 \\ 105 \\ 119 \\ 203 \\ 181 \\ 117 \end{pmatrix} = \begin{pmatrix} 825 \\ 1063 \end{pmatrix}$$

el vector de la derecha contiene los pesos moleculares de cada proteína.

Elaborar un programa que lea el archivo de datos llamado `proteina.dat` y calcule e imprima los pesos moleculares de cada una de las proteínas contenidas en el archivo.

Precipitaciones Pluviales El archivo `precipitacionp.html` contiene información sobre las precipitaciones pluviales promedio en los Estados de la República Mexicana, de 1941 a 2004 por meses del año. Esta información está también contenida en el archivo `precipita.txt`. Cargar el archivo en Matlab y guardar los datos en la matriz A . La primera columna de A contiene los totales de los promedios de precipitaciones, ¿Cómo lo puedes verificar? Si sumas las componentes de la matriz del rengón 1 de las columnas 2 a la 12 debes obtener el elemento $A(1, 1)$. Y lo mismo pasa para todos los renglones de la matriz, verificarlo.

Se puede hacer una gráfica con las precipitaciones para cada uno de los estados para ver en que meses del año llueve mas, por ejemplo para Aguascalientes esta gráfica se obtiene tecleando

`plot(A(2,2:13))`

Con la gráfica se puede deducir fácilmente que meses llueve mas en el año. Realizar la gráfica de las precipitaciones por meses de varios estados y comparar.

Utilizando el comando `[i,j]=max(v)` donde v es un vector, calcular en qué estado de la República llueve más en enero, en cuál en febrero y así para todos los meses de año.

Se puede obtener resultados por regiones, por ejemplo para la región Noreste del país que comprende los estados de Coahuila, Nuevo León y Tamaulipas, ¿Cuál es la precipitación media por mes? ¿Cuál es el promedio anual?

El trabajar con submatrices es muy útil en algunas aplicaciones.