Apéndice: MATLAB

Este apéndice es una introducción a las técnicas de programación con el paquete de programas MATLAB. Suponemos que quien lee este libro tiene ya alguna experiencia de programación con lenguajes de alto nivel y, en particular, conoce algunas técnicas esenciales como los bucles, la ramificación mediante relaciones lógicas, el empleo de subprogramas y la edición de archivos con un procesador de textos. Estas técnicas son directamente aplicables cuando se usa el paquete MATLAB.

El paquete MATLAB es un conjunto de programas matemáticos que se basa en el empleo de matrices. El paquete consta de una amplia colección de programas numéricos y programas gráficos para dibujos bi- y tridimensionales e incluye la posibilidad de realizar programas adicionales usando un lenguaje de alto nivel. También es posible desarrollar y modificar los programas de manera muy sencilla. Todo esto hace que el paquete MATLAB sea un banco de trabajo ideal para explorar y trabajar con los algoritmos que se explican en este libro.

Lo que sigue es una introducción guiada a la programación con el paquete MATLAB y nuestra sugerencia es que se trabaje a lo largo de las líneas que aquí se explican (las instrucciones del paquete MATLAB se escriben con el tipo de letra `máquina de escribir`). Los ejemplos ilustran lo que nos muestra la ventana de trabajo (en inglés, *command window*) del paquete MATLAB en una sesión típica. Lo que aparece a continuación de `>>` es lo que se introduce como dato o instrucción. Una vez que se escribe lo que se desea, hay que pulsar la tecla de retorno; entonces el computador realiza la operación y muestra la respuesta `ans =`. En las guías de uso y de referencia que acompañan el programa, así

como en el programa de ayuda que puede consultarse directamente en la ventana de trabajo, se puede hallar información adicional sobre las instrucciones y sus opciones y, también, varios ejemplos.

Operaciones aritméticas

`+`	Sumar
`-`	Restar
`*`	Multiplicar
`/`	Dividir
`^`	Elevar a una potencia
`pi, e, i`	Constantes

Ejemplo.
```
>>(2+3*pi)/2
ans =
     5.7124
```

Funciones incorporadas al programa

Damos a continuación una breve lista de algunas de las funciones disponibles en el paquete MATLAB; el programa de ayuda proporciona información sobre qué otras funciones hay disponibles. El ejemplo ilustra cómo se usan y se combinan las operaciones aritméticas y las funciones.

```
abs(#)    cos(#)    exp(#)    log(#)    log10(#)    cosh(#)
sin(#)    tan(#)    sqrt(#)   floor(#)  acos(#)     tanh(#)
```

Ejemplo.
```
>>3*cos(sqrt(4.7))
ans =
    -1.6869
```

En la respuesta se muestran, habitualmente, cinco cifras decimales significativas; la instrucción `format long` nos permite obtener hasta 15 cifras decimales significativas.

Ejemplo.
```
>>format long
3*cos(sqrt(4.7))
ans =
    -1.68686892236893
```

Instrucciones de asignación

Mediante el signo de igualdad podemos asignar un nombre al resultado de la evaluación de una expresión.

Ejemplo.
```
>>a=3-floor(exp(2.9))
a=
     -15
```

Cuando se escribe un punto y coma al final de una expresión, el computador realiza las operaciones correspondientes y almacena su resultado bajo el nombre que le hayamos asignado (para su uso en cálculos posteriores) pero no muestra el resultado en la pantalla.

Ejemplo.
```
>>b=sin(a);
>>2*b^2
ans=
     0.8457
```
Nota: `b` no se muestra.

Definición de nuevas funciones

Es posible definir nuevas funciones que se pueden usar con el paquete MATLAB escribiendo con un editor un archivo de texto cuya extensión sea `m`. Una vez definido, puede utilizarse como cualquier otra función.

Ejemplo. Vamos a definir la función $f(x) = 1 + x - x^2/4$ en un archivo `fun.m`. Para ello, con el editor de textos, escribimos:
```
function y=fun(x)
y=1+x-x.^2/4;
```
Explicaremos el uso de ".^" un poco más adelante. Podemos usar letras distintas para las variables y podemos darle un nombre distinto a la función, pero el formato debe ser el mismo. Una vez que almacenamos esta función en un archivo llamado `fun.m`, podemos usarla como cualquier otra función del paquete MATLAB.
```
>>cos(fun(3))
ans=
    -0.1782
```
La instrucción `feval` nos permite evaluar funciones de una manera útil y eficiente. Cuando se usa, la función se invoca como una cadena de caracteres.

Ejemplo.
```
>>feval('fun',4)
ans=
     1
```

Matrices

En el paquete MATLAB todas las variables son matrices. Las matrices se introducen de una manera directa:

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

Los puntos y comas separan las filas de la matriz, mientras que los elementos de la misma fila deben separarse mediante un espacio en blanco (o una coma). Alternativamente, podemos introducir las matrices fila a fila:

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

Podemos generar algunas matrices especiales usando funciones ya incorporadas:

Ejemplo.
```
>>Z=zeros(3,5);          crea una matriz de ceros de orden 3 × 5
>>X=ones(3,5);           crea una matriz de unos de orden 3 × 5
>>Y=0:0.5:2              crea la matriz de orden 1 × 5 siguiente
Y=
0 0.5000 1.0000 1.5000 2.0000

>>sin(Y)                 crea una matriz de orden 1 × 5 tomando
                         el seno de cada elemento de Y
ans=
    1.0000 0.8776 0.5403 0.0707 -0.4161
```

Podemos trabajar con los elementos de una matriz de diversas maneras.

Ejemplo.
```
>>A(2,3)                 selecciona una entrada concreta de A
ans=
     6
>>A(1:2,2:3)            selecciona una submatriz de A
ans=
     2 3
     5 6
>>A([1 3],[1 3])        otra forma de seleccionar
                         una submatriz de A
ans=
     1 3
     7 9
>>A(2,2)=tan(7.8);      asigna un nuevo valor a una entrada
                         concreta de A
```

El programa de ayuda y la documentación que acompañan el paquete proporcionan información sobre otras funciones matriciales disponibles.

Operaciones con matrices

+	Sumar
-	Restar
*	Multiplicar
^	Elevar a una potencia
'	Traspuesta conjugada

Ejemplo.
```
>>B=[1 2;3 4];
>>C=B'                    C es la traspuesta de B
C=
     1 3
     2 4
>>3*(B*C)^3              3(BC)^3
ans=
     13080 29568
     29568 66840
```

Operaciones que se realizan elemento a elemento

Una de las características más útiles del paquete MATLAB es que dispone de un gran número de funciones que operan sobre una matriz elemento a elemento; vimos antes un ejemplo de esto cuando tomamos el seno de cada elemento de una matriz de orden 1×5. Las operaciones matriciales de suma, resta y producto por un escalar se realizan elemento a elemento, lo que no ocurre con las operaciones matriciales de multiplicación, división y potenciación. Estas tres operaciones pueden realizarse elemento a elemento si anteponemos un punto al símbolo correspondiente: `.*`, `./` y `.^`. Es importante el entender cómo y cuándo deben usarse estas operaciones ya que las operaciones elemento a elemento son cruciales a la hora de diseñar e implementar eficientemente programas numéricos y gráficos con el paquete MATLAB.

Ejemplo.
```
>>A=[1 2;3 4];A^2
                          calcula el producto AA
ans=
     7 10
     15 22
>>A.^2                    eleva al cuadrado cada elemento de A
ans=
     1 4
     9 16
```

```
>>cos(A./2)          divide cada elemento de A entre 2 y,
                     después, calcula el coseno
ans=
     0.8776   0.5403
     0.0707  -0.4161
```

Gráficos

El paquete MATLAB puede producir dibujos bi- y tridimensionales de curvas y superficies. Las diversas opciones y aspectos adicionales de las instrucciones gráficas básicas pueden consultarse en el paquete de ayuda o en la documentación.

La instrucción `plot` permite generar gráficas de curvas planas. En el siguiente ejemplo se muestra cómo podemos obtener las gráficas de las funciones $y = \cos(x)$ e $y = \cos^2(x)$ en el intervalo $[0, \pi]$.

Ejemplo.
```
>>x=0:0.1:pi;
>>y=cos(x);
>>z=cos(x).^2;
>>plot(x,y,x,z,'o')
```

En la primera línea se especifican el dominio y el tamaño de paso 0.1. En las dos líneas siguientes se definen las funciones. Hagamos notar que las tres primeras líneas terminan con un punto y coma; este punto y coma se escribe para evitar que aparezcan en la pantalla los treinta y tantos elementos de cada una de las matrices `x`, `y` y `z`. La cuarta línea contiene la instrucción de dibujo que produce las gráficas. Los dos primeros términos, `x` e `y`, dibujan la función $y = \cos(x)$. Los términos tercero y cuarto, `x` y `z`, dibujan la función $y = \cos^2(x)$. El último término, `'o'`, hace que se dibuje una 'o' en cada punto (x_k, z_k) con $z_k = \cos^2(x_k)$.

El uso en la tercera fila del indicador de operación elemento a elemento "`.^`" es esencial: primero se calcula el coseno de cada elemento de la matriz `x` y, después, cada elemento de la matriz `cos(x)` se eleva al cuadrado usando la instrucción `.^`.

La instrucción de dibujo `fplot` es una alternativa útil a la instrucción `plot`. La sintaxis de esta instrucción es `fplot('nombre',[a,b],n)`, que produce la gráfica de la función `nombre.m` determinando su valor en n puntos del intervalo $[a, b]$. Si no se especifica otra cosa, el valor de n es 25.

Ejemplo. `>>fplot('tanh',[-2,2])` dibuja $y = \tanh(x)$ en $[-2, 2]$

Las instrucciones `plot` y `plot3` se utilizan para dibujar curvas parametrizadas en el espacio bi- y tridimensional, respectivamente. Estas instrucciones son especialmente útiles para visualizar las soluciones de una ecuación diferencial en dimensión dos y tres.

Ejemplo. El dibujo de la elipse $c(t) = (2\cos(t), 3\sin(t))$, con $0 \leq t \leq 2\pi$, se obtiene con las siguientes instrucciones:

```
>>t=0:0.2:2*pi;
>>plot(2*cos(t),3*sin(t))
```

Ejemplo. El dibujo de la curva $c(t) = (2\cos(t), t^2, 1/t)$, con $0.1 \leq t \leq 4\pi$, se obtiene con las siguientes instrucciones:

```
>>t=0.1:0.1:4*pi;
>>plot3(2*cos(t),t.^2,1./t)
```

Para obtener dibujos tridimensionales de superficies hay que especificar un rectángulo del dominio de la función, mediante la instrucción `meshgrid`, y luego las instrucciones `mesh` o `surf` para obtener la gráfica. Estas instrucciones son útiles para visualizar la solución de una ecuación en derivadas parciales.

Ejemplo.

```
>>x=-pi:0.1:pi;
>>y=x;
>>[x,y]=meshgrid(x,y);
>>z=sin(cos(x+y));
>>mesh(z)
```

Bucles y ramificaciones

Operadores de relación

`==`	Igual que
`~=`	No igual que
`<`	Menor que
`>`	Mayor que
`<=`	Menor o igual que
`>=`	Mayor o igual que

Operadores lógicos

`~`	No	(Verdadero si, y sólo si, la proposición es falsa)
`&`	Y	(Verdadero si las dos proposiciones son verdaderas)
`|`	O	(Verdadero si alguna de las dos proposiciones es verdadera)

Valores booleanos

`1`	Verdadero
`0`	Falso

Las instrucciones `for`, `if` y `while` del paquete MATLAB operan de manera similar a sus homólogas en otros lenguajes de programación. Estas instrucciones adoptan la sintaxis básica siguiente:

```
for (variable del bucle = rango del bucle)
        instrucciones ejecutables
end

if (premisa)
        instrucciones ejecutables
else
        instrucciones ejecutables
end

while (premisa)
        instrucciones ejecutables
end
```

En el siguiente ejemplo mostramos cómo se pueden encajar varios bucles para generar una matriz. Guardando las líneas de texto en un archivo llamado `nido.m`, entonces cada vez que escribamos `nido` en la ventana de trabajo del paquete MATLAB obtendremos la matriz `A`. Hagamos notar que los elementos de la matriz `A` forman, empezando en la esquina superior izquierda, el triángulo de Pascal.

Ejemplo.
```
for i=1:5
  A(i,1)=1;A(1,i)=1;
end
for i=2:5
  for j=2:5
    A(i,j)=A(i,j-1)+A(i-1,j);
  end
end
A
```

Para salir de un bucle antes de que se complete, se usa la instrucción `break`.

Ejemplo.
```
for k=1:100
  x=sqrt(k);
  if ((k>10)&(x-floor(x)==0))
    break
  end
end
k
```

Para mostrar una línea de texto o una matriz se utiliza la instrucción `disp`.

Ejemplo.
```
n=10;
k=0;
while k<=n
  x=k/3; disp([x x^2 x^3]), k=k+1;
end
```

Programas

Una forma eficiente de construir programas es crear nuevas funciones que se almacenan como archivos cuya extensión es `m`. Estos programas nos permiten especificar los datos que deben introducirse y los resultados que deben mostrarse y pueden ser llamados como subprogramas desde otros programas. El siguiente ejemplo nos permite visualizar el efecto de calcular los elementos del triángulo de Pascal previa modulación con un número primo. Para ello, hay que escribir las siguientes líneas con el editor de textos del paquete y almacenarlas en un archivo llamado `pasc.m`.

Ejemplo.

```
function P=pasc(n,m)
% Datos     - n es la cantidad de filas
%           - m es el número primo
% Resultado - P es el triángulo de Pascal

for j=1:n
  P(j,1)=1;P(1,j)=1;
end
for k=2:n
  for j=2:n
    P(k,j)=rem(P(k,j-1),m)+rem(P(k-1,j),m);
  end
end
```

Ahora, escribiendo en la ventana de trabajo la instrucción `P=pasc(5,3)`, veremos las cinco primeras filas del triángulo de Pascal módulo 3. Podemos intentar también `P=pasc(175,3);` (use el punto y coma) y luego `spy(P)` (que genera una matriz dispersa para valores grandes de n).

Conclusión

Alcanzado este punto, usted debería ser capaz de crear y modificar programas basados en los algoritmos dados en este libro. Para obtener información adicional sobre las funciones del paquete MATLAB o sobre cómo se utiliza en su computador particular, debe usted utilizar el programa de ayuda o la documentación que acompaña el paquete.