Tema 4

Instrucciones condicionales

Guillermo Peris Ripollés

Objetivos

Cuando finalice este tema, el alumno deberá ser capaz de:

- Utilizar correctamente los operadores relacionales y lógicos.
- Entender el funcionamiento de las funciones lógicas `any`, `all` y `find`.
- Conocer y utilizar donde corresponda las instrucciones `if...end`, `if...else...end` y `if...elseif...else...end`.

Aplicación

Cuando finalice este tema, el alumno deberá ser capaz de resolver problemas como el siguiente, cuya resolución se indica a lo largo del propio tema.

Resolución de una ecuación de segundo grado

Escribe un programa OCTAVE que resuelva la ecuación de segundo grado

$$ax^2 + bx + c = 0$$

obteniendo las soluciones correctas en función de a, b y c.

Contenidos

4.1. Introducción

Como todo lenguaje de programación imperativo, OCTAVE dispone de instrucciones de bifurcación que permiten el control del flujo de órdenes de un programa, basándose en decisiones lógicas. Para ello, dispone de estructuras clásicas como `if`, `if...else`, `if...elseif...else` y `switch-case`[1], así como de funciones lógicas específicas de este lenguaje. Además, la aplicación directa de operadores lógicos a vectores y matrices resulta mucho más potente que en otros lenguajes, en los que esta posibilidad ni siquiera existe.

4.2. Instrucción `if` simple

La instrucción `if` simple nos permite comprobar si se cumple una cierta condición antes de ejecutar una serie de órdenes. Su forma general es la siguiente:

```
if condición
    instrucciones
end
```

Es decir, el programa evalúa si la *condición* es cierta, y en ese caso ejecuta las *instrucciones*. Si la *condición* es falsa, las *instrucciones* se ignoran. El flujo del programa se muestra en la Figura 4.1.

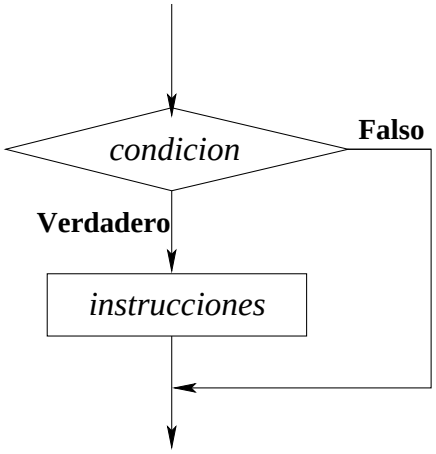


Figura 4.1: Flujo de la instrucción condicional `if`.

Veamos un ejemplo sencillo de la utilización de esta instrucción con las siguientes líneas de código OCTAVE:

```
if g < 50
   count = count + 1;
   disp(g)
end
```

Supongamos que `g` es un escalar. Si `g` es menor que 50, `count` se incrementa en 1 y la variable `g` se muestra por pantalla. En caso contrario, las dos instrucciones del interior de la estructura `if` no se tienen en cuenta. Fijémonos en que las instrucciones se encuentran desplazadas hacia la derecha para así entender mejor el código. Este *sangrado de líneas* es más que aconsejable para un buen mantenimiento del programa.

[1] Esta estructura no se va a considerar en el presente curso.

Ejercicios

✍ ▶ 1 ¿Cuál será el valor de `c` tras ejecutarse el siguiente programa OCTAVE?

```
≫a = 1; b = 2; c = 3;
≫if a < b
     c = 1 ;
  end
≫if b < c
     c = 2 ;
  end
```

4.3. Operadores relacionales y lógicos

En la sección anterior hemos utilizado el operador *menor que* (<) para evaluar si la condición de la instrucción `if` es verdadera o falsa. A este tipo de operadores se les denomina *operadores de relación* o *relacionales*, y su función es comparar dos expresiones y proporcionar una respuesta de tipo *VERDADERO* o *FALSO*. En la siguiente tabla se muestran los operadores de relación de OCTAVE [2]:

Operador de relación	Interpretación
<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que
==	Igual que
∼=	Distinto que

Tabla 4.1: Operadores de relación de OCTAVE.

Al igual que ocurre en otros lenguajes de programación, OCTAVE asocia el valor 0 a las respuestas falsas y 1 (o distinto de 0) a las verdaderas:

Comparación		Valor
Verdadero	SI	1
Falso	NO	0

Podemos comprobar fácilmente esto desde la línea de órdenes de OCTAVE:

```
≫5 < 6
  ans = 1
≫5 >= 6
  ans = 0
```

Estas comparaciones son relativamente sencillas en el caso de escalares pero, ¿qué ocurre si comparamos dos vectores (o matrices) de la misma dimensión? En ese caso, OCTAVE compara los elementos *uno a uno* y crea un vector (o matriz) en el que cada elemento es el resultado de la comparación correspondiente. Veámoslo con un ejemplo:

[2] Fijémonos en que los operadores `=` y `==` significan cosas distintas: `==` compara dos expresiones, mientras que `=` asigna el resultado de la expresión de su derecha a la variable de la izquierda.

```
≫A =1:9
A =
 1  2  3  4  5  6  7  8  9
≫A > 4
ans =
 0  0  0  0  1  1  1  1  1
```

Es decir, el vector resultante presenta un 1 (verdadero) en aquellas posiciones en las que el elemento del vector `A` es mayor que 4, y 0 (falso) en el resto.

Además de los operadores relacionales, OCTAVE también dispone de operadores lógicos, que se utilizan para combinar o negar expresiones lógicas o de relación. En la Tabla 4.2 aparece una lista de estos operadores, mientras que en la Tabla 4.3 se muestran los resultados de su aplicación (esta tabla se conoce como *tabla de verdad* de los operadores).

Operador lógico	Símbolo
no	$\sim$
y	&
o	|

Tabla 4.2: Operadores lógicos de OCTAVE.

A	**B**	**$\sim$A**	**A|B**	**A&B**
falso	falso	verdadero	falso	falso
falso	verdadero	verdadero	verdadero	falso
verdadero	falso	falso	verdadero	falso
verdadero	verdadero	falso	verdadero	verdadero

Tabla 4.3: Tabla de verdad de los operadores lógicos.

La utilización de los operadores lógicos nos permite combinar relaciones de comparación. Veamos un ejemplo: supongamos que queremos escribir un programa en OCTAVE que calcule las soluciones de una ecuación de segundo grado del tipo general $ax^2 + bx + c = 0$, que son las siguientes:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \; .$$

Si buscamos sólo soluciones reales, deben cumplirse las dos condiciones siguientes, *simultáneamente*:

$$b^2 - 4ac >= 0 \qquad \text{y} \qquad a \text{ distinto de } 0 \; .$$

Si no se cumple la primera condición, las dos soluciones serían números complejos, mientras que si no se cumple la segunda, la ecuación no sería de segundo grado y su solución sería $-c/b$ (si $b \neq 0$). Estas dos condiciones se han de cumplir al mismo tiempo, por lo que en OCTAVE escribiríamos:

```
≫if (b^2 - 4*a*c >= 0 ) & (a ~= 0)
```

La condición en el interior del `if` únicamente se considerará cierta si las expresiones que une el operador `&` son ciertas. En caso de que una de estas dos expresiones sea falsa, la condición general se evaluará como falsa. Esta idea es la que se recoge en la última columna de la Tabla 4.3.

Por el contrario, el operador *o* (`|`) evaluará una combinación de condiciones como verdadera si al menos una de ellas (o las dos) es cierta. Esta información se presenta en la cuarta columna de la Tabla 4.3. En el caso de vectores o matrices, este operador se aplica elemento a elemento, como en el caso de los operadores relacionales. Veamos un ejemplo para entenderlo mejor:

```
≫a = [1 0 3 0 0 2], b = [0 0 1 1 0 2]
a =
   1   0   3   0   0   2
b =
   0   0   1   1   0   2
≫a|b
ans =
   1   0   1   1   0   1
```

Aquí se observa que únicamente se evaluan a falso (0) aquellos elementos del vector en los que los elementos relacionados son 0 simultáneamente (fijémonos en que OCTAVE considera como verdadero cualquier valor distinto de cero, y no sólamente el uno).

Por último, el operador $\sim$ niega la expresión que le sigue y supone intercambiar verdadero y falso [3]. Siguiendo con el último ejemplo :

```
≫∼b
ans =
   1   1   0   0   1   0
≫∼(a<b)
ans =
   1   1   1   0   1   1
```

Observamos en la última expresión que podemos combinar operadores lógicos y de relación. Por supuesto, en estos casos también existe un orden de precedencia, teniendo los operadores de relación prioridad sobre los operadores lógicos. Sin embargo, resulta conveniente utilizar paréntesis para no cometer errores.

```
≫x = 3; y = 5;
≫∼( (x == y) | (x ==5.5) )
ans =
   1
```

[3] Mucho ojo con confundir el operador lógico $\sim$ con el operador relacional $\sim=$. ¿Podrías decir que diferencia hay entre ellos?

Ejercicios

▶ 2 Determina si las siguientes expresiones son verdaderas o falsas, comprobándolo posteriormente con OCTAVE. Asume para ello los siguientes valores de variables:

```
a = 5.5        b = 1.5        k = -3
```

a) `a < 10.0`
b) `a+b >= 6.5`
c) `k ~= 0`
d) `b -k > a`
e) `~(a == 3*b)`
f) `-k <= k + 6`
g) `a < 10 & a > 5`
h) `a < 10 | a > 5`
i) `~(abs(k) > 3) | k < b-a`

▶ 3 A partir de los vectores `x = [1 5 2 8 9 0 1]` e `y = [5 2 2 6 0 0 2]`, determina los resultados de las siguientes expresiones, comprobándolo posteriormente con OCTAVE.

a) `x > y`
b) `x < y`
c) `x == y`
d) `x <= y`
e) `y >x`
f) `x | y`
g) `x & (~y)`
h) `(x > y) | (y > x)`

▶ 4 A partir de las matrices A, B y C siguientes, evalúa el resultado de las siguientes órdenes OCTAVE.

$$A = \begin{bmatrix} 1 & 7 & 13 \\ -2 & -7 & 4 \\ 0 & 1 & 3 \end{bmatrix} \qquad B = \begin{bmatrix} 12 & -6 & 0 \\ 2 & 12 & 22 \\ -3 & 0 & 3 \end{bmatrix} \qquad C = \begin{bmatrix} 14 & 2 & -3 \\ -1 & 0 & 5 \\ 10 & 3 & -4 \end{bmatrix}$$

a) `A >= 0`
b) `A > B`
c) `A >= C`
d) `A == C`
e) `B - (A > 2)`
f) `C + 0.001*(C==0)`

4.4. Funciones lógicas

Además de los operadores lógicos y relacionales propios de cualquier lenguaje de programación, OCTAVE presenta una serie de funciones lógicas aplicables a vectores y matrices [4] que van a resultar extremadamente útiles en el uso de estructuras de selección. Vamos a estudiar ahora sólo algunas de ellas (para más información, consulta la ayuda de OCTAVE o alguno de los libros de la bibliografía).

La función `any` (en castellano, *alguno*) al ser aplicada a un vector devuelve un escalar que indica si *alguno* de sus elementos es distinto de cero. De forma similar, la función `all` (en castellano, *todos*) proporciona un valor de verdadero si *todos* los elementos de la matriz son distintos de cero:

```
≫a = [ 1 0 2 3 4 0 ];
≫any(a)
ans = 1
≫all(a)
ans = 0
```

[4] En este curso no aplicaremos estas funciones a matrices.

Estas funciones resultan muy útiles en combinación con operadores lógicos y relacionales. Por ejemplo, si queremos saber si el vector `a` tiene algún valor negativo, podemos hacerlo con la orden `any(a<0)` (que se leería *¿hay algún número menor que 0 en `a`?*), mientras que si nos interesara comprobar que *todos* los componentes de `a` son negativos escribiríamos `all(a<0)` (*¿todos los componentes de `a` son menores que 0?*). Podemos comprobarlo con el siguiente ejemplo:

```
≫a = [ 1 2 3 4 -2 -3 6 7] ; b = [-1 -2 -5 -7];
≫any(a < 0)
ans = 1
≫any(b < 0)
ans = 1
≫all(a < 0)
ans = 0
≫all(b < 0)
ans = 1
```

Otra función lógica que puede resultarnos útil es `find`, para extraer elementos de vectores que cumplan ciertas propiedades[5]. Por ejemplo, supongamos que, además de querer saber si existen elementos negativos en el vector `a`, deseamos conocer la posición de dichos elementos en el vector. La expresión `find(a<0)` nos proporcionaría dicha lista, por lo que podemos extraer el subvector de elementos negativos como `a(find(a<0))`. Incluso podemos sustituir globalmente estos elementos negativos por uno nuevo (en el ejemplo, se sustituyen por 0):

```
≫find(a < 0)
ans =
   5    6
≫a(find(a < 0))
ans =
  -2   -3
≫a(find(a < 0)) = 0 ;
≫a
a =
1    2    3    4    0    0    6    7
```

Ejercicios

▶ **5** Dado el vector `a = [ -0.2 1.3 -2.1 0.5 3.4 2.7 -0.8]`, escribe una orden OCTAVE que:

a) Muestre todos los números negativos de `a`.

b) Sustituya por 0 todos los números negativos de `a`.

c) Muestre todos los valores de `a` cuyo valor absoluto sea menor que 1.

d) Sume 1 a todos los valores de `a` cuyo valor absoluto sea menor que 1.

▶ **6** Representa la función $\sin(x)$ entre 0 y 3π, sustituyendo por 0.5 todos los valores negativos de la función.

[5] La aplicación de esta función es ligeramente distinta en el caso de matrices. Si te interesa conocer su funcionamiento en ese caso, consulta la ayuda de OCTAVE.

▶ 7 Hemos realizado una serie de medidas que hemos guardado en un vector x. Supongamos que consideramos erróneas todas las medidas en el intervalo $-0.1 < x < 0.1$. Queremos sustituir estos valores por ceros y almacenarlos al final del vector (ver el ejemplo en la tabla siguiente). Escribe las órdenes OCTAVE necesarias para ello.

	Antes	Después
$x(1)$	1.92	1.92
$x(2)$	0.05	-2.43
$x(3)$	-2.43	0.85
$x(4)$	-0.02	0.00
$x(5)$	0.09	0.00
$x(6)$	0.85	0.00
$x(7)$	-0.06	0.00

4.5. Instrucción `if...elseif...else`

La orden `else` permite ejecutar un bloque de líneas de código si no es cierta la condición en una instrucción `if`. El flujo asociado a una estructura condicional de este tipo es el siguiente:

```
if condición
    instrucciones(1) % si condición VERDADERA
else
    instrucciones(2) % si condición FALSA
end
```

Es decir, el programa evalúa si la *condición* es cierta, y en ese caso ejecuta las *instrucciones(1)*. Si la *condición* es falsa, se ejecutan las *instrucciones(2)*. El flujo del programa se muestra en la Figura 4.2.

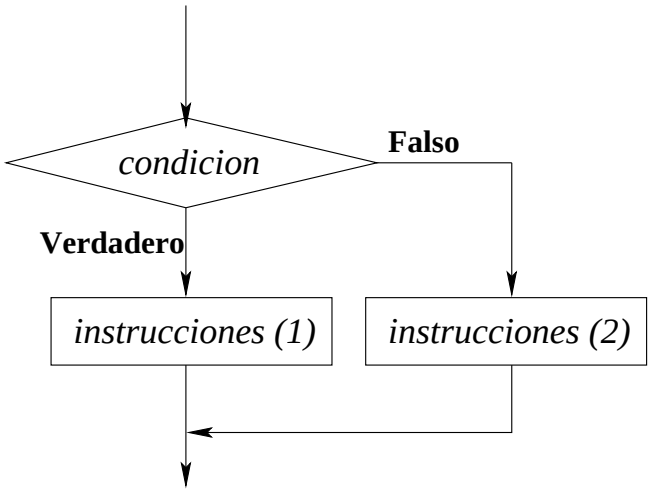


Figura 4.2: Flujo de la instrucción condicional `if...else` .

Como ejemplo, consideremos que tenemos una variable denominada `intervalo`. Si el valor de `intervalo` es menor que 1, asignamos el valor `intervalo/10` a la variable `incremento_x`. En caso contrario (*else*), le asignamos el valor 0.1. El código OCTAVE sería el siguiente:

```
if intervalo < 1
   incremento_x = intervalo/10;
else
   incremento_x = 0.1;
end
```

Existen ocasiones en que el número de condiciones que debemos comprobar es mayor que uno, lo cual nos obligaría a utilizar condiciones `if...else` anidadas. En estos casos, resulta más conveniente utilizar la condición `elseif` (que se traduciría como *si no se cumple, entonces si...*). Esta orden puede repetirse tantas veces como se quiera dentro de una estructura `if`. El flujo de ejecución en este caso sería el siguiente (ver Figura 4.3):

```
if condición 1
    instrucciones (1) % si condición 1 VERDADERA
elseif condición 2
    instrucciones (2) % si condición 2 VERDADERA
   .........
else
    instrucciones (3) % si condiciones anteriores FALSAS
end
```

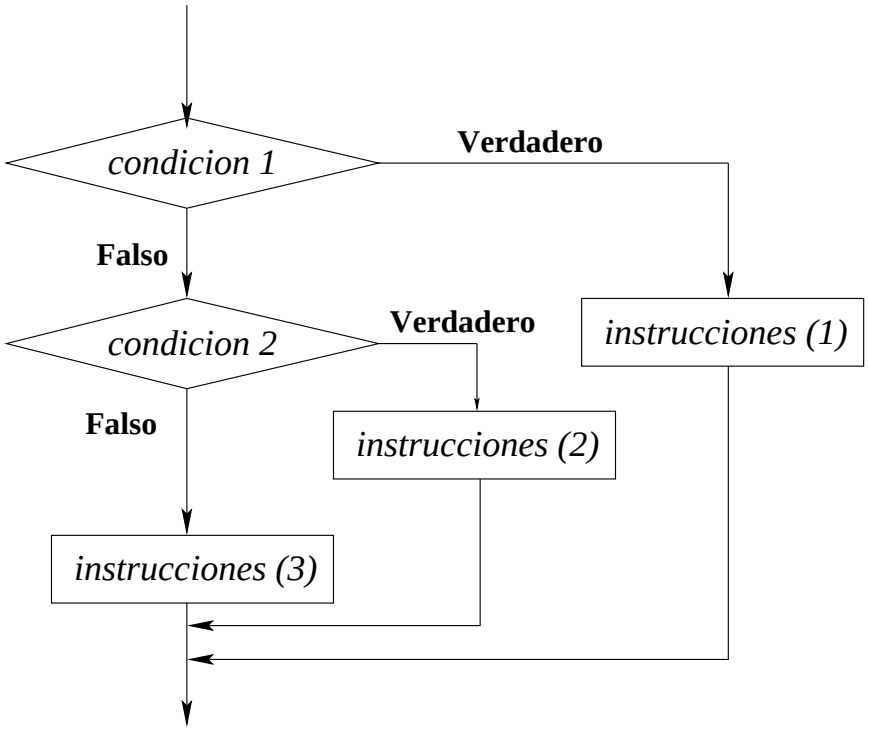
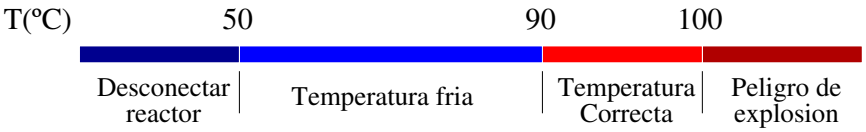


Figura 4.3: Flujo de la instrucción condicional `if...elseif` .

Consideremos el siguiente ejemplo: un reactor químico debe funcionar entre 90 y 100 grados centígrados. Por encima de 100 grados, el reactor entra en situación crítica, peligrando la seguridad de la planta; entre 50 y 90 grados, el reactor funciona aunque su rendimiento no es óptimo, mientras que por debajo de 50 grados el funcionamiento no es correcto, debiéndose desconectar el reactor. Este comportamiento se resumen en la siguiente figura.



El siguiente código OCTAVE utiliza la temperatura del reactor para generar un mensaje adecuado:

```
if temperatura >100
  disp('Demasiado caliente - Peligro de explosion !!!');
elseif temperatura >90
  disp('Temperatura dentro de los limites adecuados');
elseif temperatura >50
  disp('Temperatura por debajo de los limites adecuados');
else
  disp('Demasiado frio - Desconectar el reactor') ;
end
```

Analicemos este código: el primer mensaje sólo se mostrará si la temperatura está por encima de 100 grados; el segundo mensaje aparecerá cuando la temperatura sobrepase los 90 grados *pero sea inferior a los 100 grados* (si fuera superior a 100 grados, se cumpliría la primera condición y no se examinarían el resto de posibilidades); el tercer mensaje sólo se mostrará para temperaturas entre 50 y 90 grados, mientras que el último mensaje sólo aparecerá cuando no se cumplan ninguna de las condiciones anteriores, por lo que la temperatura será menor que 50 grados. Fijémonos, en que el orden en que aparecen las condiciones es importante.

Ejercicios

✍ ▶ **8** Escribe las órdenes necesarias para que se desarrollen los siguientes pasos:

a) Si la diferencia entre los voltajes `volt1` y `volt2` es mayor que 10, imprime dichos valores.

b) Si el logaritmo neperiano de `x` es mayor o igual a 3, pon a cero la variable `tiempo` y aumenta en uno el contador `cont`.

c) Si la distancia `dist` es menor que 50.0 y el tiempo `t` mayor que 10.0, aumenta `t` en 2.0; en caso contrario, aumenta `t` en 2.5.

d) Si `dist` es mayor o igual a 100.0, aumenta `t` en 2.0. Si `dist` se encuentra entre 50 y 100, aumenta `t` en 1.0. En cualquier otro caso, aumenta `t` en 0.5.

✍ ▶ **9** Evalúa los siguientes fragmentos de código OCTAVE teniendo en cuenta los distintos valores de variables iniciales indicados.

```
if n > 1
   m = n + 2
else
   m = n - 2
end
```

- $n = 7 \quad m = ?$
- $n = 0 \quad m = ?$
- $n = -7 \quad m = ?$

```
z = 1;
if s <= 1
   t = 2z
elseif s < 10
   t = 9 - z
elseif s < 100
   t = sqrt(s)
else
   t = s
end
```

- $s = 1 \quad t = ?$
- $s = 7 \quad t = ?$
- $s = 57 \quad t = ?$
- $s = 300 \quad t = ?$

```
if t >= 24
   z = 3t + 1
elseif t < 9
   z = t^2/3 - 2t
else
   z = -t
end
```

- $t = 50 \quad z = ?$
- $t = 19 \quad z = ?$
- $t = -6 \quad z = ?$
- $t = 0 \quad z = ?$

4.6. Aplicación

La solución de la ecuación de segundo grado

$$ax^2 + bx + c = 0$$

presenta distintas soluciones en función de a y de su discriminante D, que se define como

$$D = b^2 - 4ac.$$

Así, las distintas soluciones que se presentan son las siguientes:

a) $a = 0$: Si $a = 0$ la ecuación no es de segundo grado, sino lineal, y su solución es[6]:

$$x = \frac{-c}{b}$$

b) $D < 0$ y $a \neq 0$: En este caso, la ecuación presenta dos soluciones complejas, por lo que el programa mostrará un mensaje indicándolo, sin calcular las soluciones[7].

c) $D = 0$ y $a \neq 0$: La ecuación presenta una solución única,

$$x = \frac{-b}{2a}$$

El programa deberá calcularla y mostrarla en pantalla.

d) $D > 0$ y $a \neq 0$: La ecuación tiene dos soluciones:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

A continuación se muestra el programa OCTAVE que resuelve ecuaciones cuadráticas.

[6] Por supuesto, esta solución es válida si $b \neq 0$. Si $b = 0$, no hay ningún término en x, por lo que no se trata de ninguna ecuación. Esta posibilidad no se tiene en cuenta en el programa que se muestra a continuación. ¿Se te ocurre como podrías incluirla?

[7] En este curso no vamos a trabajar con números complejos.

```
%*****************************************************************
% Programa : ecuacuad.m
% Descripcion: Este programa calcula las raices de una
%           ecuacion del tipo ax^2+bx+c=0.
%*****************************************************************

   % Pedimos al usuario los valores de a, b y c .
   disp('Este programa calcula las soluciones de una ecuacion');
   disp('de segundo grado del tipo ax2 + bx + c.');

   % Introducimos los valores de las constantes a,b,c.
   a = input('Introduce el valor de a: ');
   b = input('Introduce el valor de b: ');
   c = input('Introduce el valor de c: ');

   % Calculamos el valor del discriminante.
   D = b^2 - 4*a*c ;

   %Caso a = 0
   if a == 0
      x = -c/b;
      fprintf('La ecuacion es lineal y su solucion es: ');
      fprintf('x =%6.2f.\n', x);

   %Caso discriminante negativo
   elseif D < 0
      disp('Esta ecuacion tiene dos soluciones complejas.');

   % Caso discriminante cero
   elseif D == 0
      x = -b/(2*a);
      fprintf('La ecuacion presenta una solución unica:');
      fprintf('x =%6.2f.\n',x);

   % Dos soluciones reales
   else
      x1 = (-b + sqrt(D))/(2*a);
      x2 = (-b - sqrt(D))/(2*a);
      fprintf('Esta ecuacion tiene dos soluciones:');
      fprintf('x1 =%6.2f y x2 =%6.2f.\n', x1, x2) ;
   end
   disp('Fin del programa') ;
```

Ejercicios

▶ **10** Escribe este programa en un fichero de nombre `ecuacuad.m` y obtén con él la solución a las siguientes ecuaciones:

a) $5x + 4 = 0$

b) $4x^2 + 2x + 1 = 0$

c) $x^2 - 12x + 6 = 0$

d) $x^2 - 2x + 1 = 0$

4.7. Ejercicios prácticos

Es conveniente que pienses y realices los ejercicios que han aparecido a lo largo de la unidad marcados con el símbolo ✍ antes de acudir a la sesión de prácticas correspondiente. Deberás iniciar la sesión realizando los ejercicios marcados con el símbolo 💾. A continuación, deberás hacer el mayor número de los ejercicios siguientes.

Ejercicios

▶ **11** Escribe un programa de nombre `raizcuad.m` que pida al usuario un número y:

- si el número es menor que cero, indique con un mensaje que el número es negativo.
- si el número es mayor que 100, indique con un mensaje que el número es demasiado grande.
- en cualquier otro caso, muestre el número y su raíz cuadrada.

Comprueba que funciona correctamente ejecutando el programa y dándole un número negativo, uno mayor que cien, y otro positivo menor que 100.

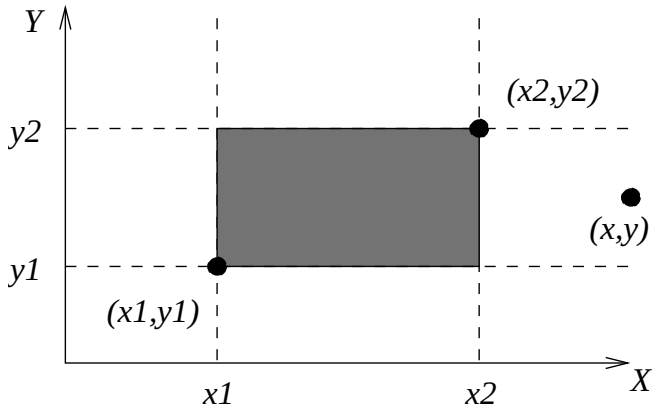
▶ **12** Escribe un programa de nombre `multiplo2_3.m` que pida un entero y compruebe si es divisible por 2 o 3. Considera todas las posibilidades, como que sea divisible por 2 y 3, sólo divisible por 2, sólo divisible por 3 o no divisible por estos números (utiliza la función `rem`).

▶ **13** Escribe un programa de nombre `reynolds.m` que calcule el coeficiente de arrastre de un fluído, que depende del número de Reynolds Re según la siguiente ecuación:

$$C = \begin{cases} 0, & Re \leq 0 \\ 24/Re, & Re \in]0, 0.1] \\ (24/Re) \cdot (1 + 0.14 Re^{0.7}), & Re \in]0.1, 1000] \\ 0.43, & Re \in]1000, 500000] \\ 0.19 + 80000/Re, & Re > 500000 \end{cases}$$

Calcula el coeficiente de arrastre para Re = -3000, 0.05, 56, 1000, 7000, 3000000.

▶ **14** Supongamos un plano real de dos dimensiones con un sistema de coordenadas. En el plano representamos un punto mediante sus coordenadas (x, y) y un rectángulo mediante las coordenadas de sus esquinas inferior izquierda y superior derecha, es decir, mediante los puntos $(x1, y1)$ y $(x2, y2)$. Esto puede observarse en el ejemplo de la siguiente figura, donde además puede apreciarse que $x1$, $y1$, $x2$ e $y2$ definen las rectas paralelas a los ejes X e Y a partir de las cuales se delimita el rectángulo. Realiza un programa de nombre `rectangulo.m`



que, a partir de las coordenadas de un rectángulo y de un punto, determine si el punto está contenido en el rectángulo. Compruébalo con el rectángulo de coordenadas $(x1, y1) = (-4, -4)$ y $(x2, y2) = (3, 0)$, y los puntos $(1, -1)$ (interior) y $(1, 3)$ (exterior).

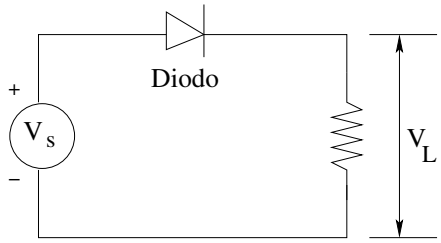


Figura 4.4: Diodo rectificador

▶ 15 Un *diodo* ideal bloquea el flujo de corriente en la dirección opuesta al símbolo de flecha que lo representa. Puede utilizarse para diseñar *rectificadores* como el que se muestra en la siguiente figura:

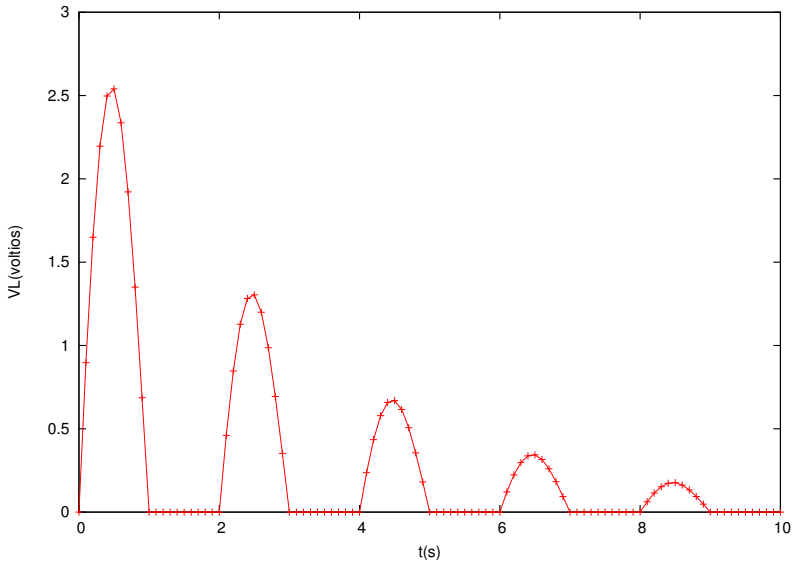
Para este diodo ideal, el voltaje V_L a través de la resistencia R_L viene dado por la ecuación,

$$v_L = \begin{cases} v_S & \text{si } v_S > 0 \\ 0 & \text{si } v_S \leq 0 \end{cases}$$

Supongamos que el voltaje de la pila viene dado por la expresión

$$v_S(t) = 3 \cdot e^{-t/3} \sin(\pi t) \qquad \text{voltios.}$$

donde t es el tiempo en segundos. Escribe un program en `Octave` que represente el voltaje v_L en función del tiempo para $0 \leq t \leq 10$. A continuación, se muestra una posible forma de la gráfica obtenida.



► **16** Una matriz `temp` contiene los valores de temperatura del agua en dos estanques en grados Celsius, medida al mediodía durante diez días, de forma que cada fila contiene los resultados de un estanque. Por ejemplo, la matriz podría presentar los siguientes valores:

```
≫temp = [18 23 25 17 20 21 15 18 22 19;   % Primer estanque
          19 22 25 18 19 22 17 19 21 19];   % Segundo estanque
```

Escribe en cada uno de los siguientes apartados **una sóla** instrucción Octave para:

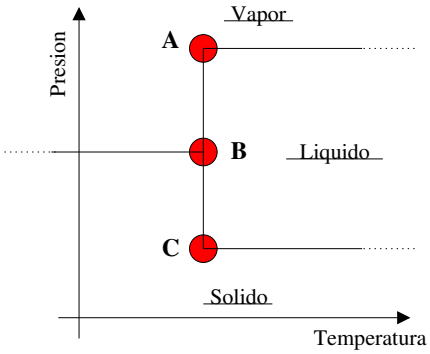
1. Determinar cuántos días la temperatura del estanque 1 estuvo por encima de 20 grados.
2. Determinar en qué días se cumplió la condición anterior.
3. Determinar cúantos días la temperatura del estanque 1 fue superior a la del estanque 2.
4. Determinar en qué días se cumplió la condición anterior.
5. Determinar si hubo algún día en que la temperatura de ambos estanques fuera la misma (verdadero o falso).

► **17** El pasado 1 de Julio de 2006 entró en vigor el nuevo permiso de conducir por puntos, según el cual se restan puntos al conductor en función de las infracciones cometidas. Entre las infracciones más comunes se encuentra el exceso de velocidad. La resta de puntos por este concepto sigue las siguientes condiciones:

- Superar el límite de velocidad entre 21 y 30 km/h : **2 puntos**.
- Sobrepasar el límite de velocidad entre 31 y 40 km/h : **3 puntos**.
- Conducir a una velocidad superior al límite establecido en más de 40 km/h : **4 puntos**.
- Conducir superando en más del 50 % el límite de velocidad máxima autorizada, siempre que ello suponga superar, al menos, en 30 km/h dicho límite : **6 puntos**.

Escribe un programa que pida al usuario la velocidad del vehículo y la velocidad máxima de la vía, y calcule e imprima el número de puntos que pueden restarse por la infracción. Ten en cuenta, además, que la velocidad puede ser correcta, e incluso ser incorrecta pero no conllevar sanción.

► **18** Recientemente se ha descubierto un nuevo compuesto químico, el **Guillermio**, del cual se ha hecho un estudio intensivo. De dicho estudio se ha obtenido el diagrama de fases del compuesto, que nos indica su estado físico en función de la temperatura y la presión. Dicho diagrama se muestra en la siguiente figura, y los valores de los puntos críticos en la tabla a su derecha.



Punto	Temp. (K)	Presión (bar)
A	1000	1575
B	1000	1250
C	1000	980

Escribe un PROGRAMA que pida al usuario los valores de temperatura y presión, y le imprima una frase diciendo si el compuesto es sólido, líquido o gaseoso.