

Tema 1

Introducción a Octave

Guillermo Peris Ripollés

Objetivos

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

- Realizar cálculos simples con `OCTAVE`, utilizando los operadores aritméticos básicos, sus reglas de precedencia y algunas funciones matemáticas.
- Asignar valores a variables y utilizarlas correctamente.
- Gestionar el entorno de trabajo `OCTAVE` para conocer las variables definidas, así como gestionar ficheros y directorios.
- Crear ficheros-M con programas que pidan información al usuario y proporcionan información por la salida estándar.
- Utilizar la ayuda de `OCTAVE` para obtener información sobre su uso.

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.

Cálculo de pesos moleculares

Escribe un programa en `OCTAVE` que pida al usuario el número de átomos de carbono, oxígeno e hidrógeno de una molécula orgánica (sin otro tipo de átomos) y calcule su peso molecular y el porcentaje en peso de oxígeno. Como ejemplo, aplica el programa a las siguientes moléculas:

- Aspirina ($C_9H_8O_4$).
- Benceno (C_6H_6).
- Alcohol Etilico (CH_3CH_2OH).
- Ácido Acético (CH_3COOH).
- Acetona (CH_3COCH_3).

Pesos atómicos: Carbono, 12 g/mol; Hidrógeno, 1 g/mol; y Oxígeno, 16 g/mol.

Contenidos

1.1. Introducción	1-3
1.2. Conceptos básicos	1-3
1.2.1. Cálculos simples	1-3
1.2.2. Variables	1-5
1.2.3. Comentarios y signos de puntuación	1-6
1.2.4. Formato numérico	1-7
1.3. Funciones matemáticas	1-8
1.3.1. Funciones matemáticas elementales	1-9
1.3.2. Funciones trigonométricas	1-9
1.4. El entorno de trabajo Octave	1-10
1.4.1. Gestión de variables	1-10
1.4.2. Historia de la sesión	1-11
1.5. Ficheros-M	1-11
1.6. Instrucciones de entrada/salida	1-12
1.6.1. Salidas por pantalla	1-12
1.6.2. Entrada de datos	1-14
1.7. Ayuda de Octave	1-15
1.8. Ejercicios prácticos	1-17

1.1. Introducción

OCTAVE es un potente programa de cálculo matemático basado en software libre, originalmente desarrollado para trabajar con vectores y matrices por ingenieros químicos, aunque sus últimas versiones exceden con mucho este propósito inicial. Incorpora distintas funcionalidades útiles para el cálculo técnico y científico, como algebra matricial, manipulación de polinomios, cálculo numérico, cálculo simbólico, y creación y manipulación de gráficos ¹. Además, incluye un lenguaje de programación propio, que aún no siendo tan eficiente como los lenguajes compilados (como C/C++, Fortran, etc), es más fácil de utilizar y nos permitirá introducir en este curso conceptos básicos de programación comunes con estos lenguajes. Este lenguaje es muy similar al que proporciona el software propietario **Matlab**, por lo que puede consultarse la bibliografía de este último programa (mucho más extensa) para aprender a utilizar **OCTAVE**.

En este curso se asume que se va a ejecutar **OCTAVE** en un sistema operativo basado en GNU/Linux.

1.2. Conceptos básicos

Para ejecutar **OCTAVE** bajo Linux abriremos un terminal bajo *Linux* y ejecutaremos en él la orden `octave`. Al hacerlo, aparecerá una ventana en la que, tras información sobre el programa, aparecerá el indicador de órdenes `octave:1>`². En esta ventana únicamente podremos escribir después del último indicador aparecido.

1.2.1. Cálculos simples

Para empezar a trabajar con **OCTAVE**, vamos a resolver poco a poco la aplicación práctica expuesta al principio de la unidad. Para ello, calcularemos inicialmente el peso molecular y el porcentaje en peso de oxígeno del ácido acetilsalicílico (aspirina), de fórmula $C_9H_6O_4$, siendo los pesos atómicos aproximados: C, 12 g/mol; H, 1 g/mol; y O, 16 g/mol.

Para resolver este problema debemos calcular primero el peso molecular del compuesto. Podemos ayudarnos con **OCTAVE** utilizándolo como una calculadora³:

```
>> 9*12 + 6*1 + 4* 16
ans = 178
```

Fijémonos en que **OCTAVE** almacena el resultado con el nombre `ans`, abreviatura de *answer* (solución). Si ahora calculamos los gramos de oxígeno por mol de producto,

```
>> 4* 16
ans = 64
```

podemos obtener el porcentaje en peso de oxígeno como

```
>> 64 / 178 * 100
ans = 35.955
```

¹ Realmente, para la representación gráfica **OCTAVE** se apoya en otro programa libre: **Gnuplot**.

² Para simplificar esta notación, en los apuntes utilizaremos el *prompt* (`>>`).

³ Fíjate en que los espacios no son importantes en **OCTAVE**.

En el ejemplo anterior hemos utilizado los operadores aritméticos suma, multiplicación y división. Los operadores aritméticos básicos que proporciona `OCTAVE` son los siguientes⁴:

Operador	Símbolo	Ejemplo
suma, $a + b$	+	$3 + 22$
resta, $a - b$	-	$90 - 54$
multiplicación, $a * b$	*	$3.14 * 0.85$
división, $a \div b$	/ o \	$56/8$
exponenciación, a^b	^	2^8
cambio de signo, $-a$	-	$-(4*8)$

Tabla 1.1: Operadores aritméticos básicos de `OCTAVE`.

La ejecución de estas operaciones en una expresión matemática se realiza siguiendo las *reglas de precedencia*, y de izquierda a derecha en el caso de operadores con igual precedencia. La mayor precedencia la posee el operador de exponenciación, seguido por los operadores de multiplicación y división (igual precedencia), siendo la suma y la resta los operadores con menor precedencia (igual entre ellos).

Por ejemplo, la expresión

$$2 + 3 * 4^2$$

se evalúa considerando primero la exponenciación ($4^2 = 16$), por ser el operador de mayor precedencia, a continuación la multiplicación ($3 * 16 = 48$), y por último la suma ($48 + 2 = 50$).

Este orden de precedencia puede ser alterado mediante el uso de paréntesis. Así, $(2 + 3) * 4^2 = 80$. En caso de duda sobre el orden de precedencia, es recomendable hacer uso de paréntesis.

Ejercicios

 **► 1** Evalúa las siguientes expresiones según las reglas de precedencia de `OCTAVE`.

- a) $2 + 3 + 4 + 2$
- b) $2 + 3 * 4 + 2$
- c) $(2 + 3) * 4 + 2$
- d) $(2 + 3) * (4 + 2)$
- e) $1/2 * (2 + 3 * 4^2)$

 **► 2** Traduce las siguientes expresiones matemáticas a `OCTAVE` utilizando el menor número de paréntesis posible.

- a) $2 + (3 \cdot (6/2))$
- b) $\frac{4 + 6}{2 + 3}$
- c) $(4/2)^5$
- d) $(4/2)^{5+1}$
- e) $(-3)^2$

⁴ El operador `\` realiza operaciones de matrices a izquierdas, aunque no lo utilizaremos en este curso.

1.2.2. Variables

Una *variable* es un identificador que se utiliza para representar cierto tipo de información dentro de una determinada parte del programa. En su forma más sencilla, una variable es un identificador que se utiliza para representar un dato individual, como por ejemplo una cantidad numérica. Este valor se puede recuperar después en el programa simplemente haciendo referencia al nombre de la variable, e incluso cambiar su contenido por otro del mismo tipo.

Podemos resolver el problema anterior utilizando variables y asignándoles valores con el operador de asignación (=). De esta forma, podemos reutilizar los resultados intermedios para operaciones posteriores. En las siguientes líneas se resuelve el problema definiendo 5 variables (`peso_C`, `peso_O`, `peso_H`, `peso_molecular` y `porcentaje_O`).

```

>> peso_C = 9*12;
>> peso_O = 4*16;
>> peso_H = 6*1 ;
>> peso_molecular = peso_C + peso_O + peso_H
peso_molecular = 178
>> porcentaje_O = peso_O/peso_molecular*100
porcentaje_O = 35.955

```

En las tres primeras líneas de código se observa que la utilización de un punto y coma (;) al final de la expresión matemática evita que se muestre el resultado en pantalla. Esta opción de `OCTAVE` se utiliza con frecuencia para evitar un exceso de líneas de salida de programa.

En las órdenes de asignación, en primer lugar se calcula la expresión a la derecha del signo =, y el valor obtenido se asigna a la variable que aparece a la izquierda del operador. Así, mientras la expresión

```

>> peso_C = 9*12;

```

es válida en `OCTAVE`, la siguiente no lo es:

```

>> 9*12 = peso_C ;

```

Además, teniendo en cuenta el orden de evaluación de las asignaciones, la siguiente secuencia de órdenes es correcta:

```

>> a = 10;
>> a = 2*a
a = 20

```

ya que en la segunda línea se calcula en primer lugar `2*a` y el resultado (20) se asigna a la variable `a`, sustituyendo el valor anterior.

La utilización de nombres de variables en `OCTAVE` sigue una serie de reglas que se indican a continuación:

- Variables que sólo se diferencian en el uso de mayúsculas/minúsculas son distintas.
Ejemplo: `peso_o`, `peso_O` y `Peso_O` se pueden utilizar como tres variables distintas.

- Los nombres deben empezar por letras, y pueden contener tanto letras como números y guiones bajos (`_`). Los signos de puntuación no están permitidos.
Ejemplo: Este_es_1_nombre_valido y ~~Este,no lo es~~.

Resulta conveniente escoger nombres que nos indiquen claramente para qué se utiliza la variable. Además, hay que tener en cuenta que `OCTAVE` utiliza una serie de variables predefinidas cuyo valor no conviene cambiar, aunque es posible hacerlo. Algunas de estas variables son:

<code>ans</code>	Variable que almacena el último resultado.
<code>pi</code>	Valor $\pi = 3.1415\dots$
<code>eps</code>	Menor número que, sumado a 1, da un resultado mayor que 1.
<code>inf</code>	Infinito, p.e., $1/0$.
<code>NaN</code> o <code>nan</code>	Resultado no numérico (<i>Not a number</i>), p.e., $0/0$.
<code>i</code> y <code>j</code>	$\sqrt{-1}$.
<code>e</code>	Base de los logaritmos naturales (neperianos).

Tabla 1.2: Variables predefinidas por `OCTAVE`.

Ejercicios

🚩 ▶ **3** Indica cuáles de las siguientes expresiones son incorrectas y por qué.

- `numero_bajas = 6 + 2 ;`
- `8colores = 6*8;`
- `numero bajas = 6 + 2;`
- `i = 1 ;`
- `A1234_5678 = i ;`
- `A1234_5678 = A1234_5678*2`
- `B-52 = 0 ;`

🚩 📄 ▶ **4** ¿Qué resultado esperarías encontrar para el valor de x e y tras la ejecución del código `OCTAVE` siguiente?

```

>> x = 5;
>> x = 2*x;
>> y = x^2;
>> x = y/x;

```

1.2.3. Comentarios y signos de puntuación

`OCTAVE` permite introducir varias órdenes en una misma línea separándolas por signos de puntuación. Así, haciendo uso de comas podemos escribir,

```

>> peso_C = 9*12, peso_0 = 4*16, peso_H = 6*1
peso_C = 108
peso_0 = 64
peso_H = 6

```

Ya hemos visto que la utilización de un punto y coma evita que se muestre el resultado en pantalla. Podemos introducir varias órdenes en una misma línea, terminando con coma o fin de línea aquellas cuyo resultado queremos visualizar, y con punto y coma aquellas que no nos interesa observar.

```

>> peso_C = 9*12, peso_O = 4*16; peso_H = 6*1
peso_C = 108
peso_H = 6

```

En **OCTAVE**, al igual que en otros lenguajes de programación, se pueden añadir comentarios a los programas, lo cual permite que otros programadores entiendan más fácilmente la estructura del código, y que incluso los propios autores de los programas comprendan sus propios trabajos al cabo del tiempo. Para ello, en **OCTAVE** se utiliza el carácter `%` para iniciar comentarios, de forma que el *intérprete de OCTAVE* ignora todo lo que sigue a este signo hasta el final de la línea.

```

>> % Calculo de los g/mol de cada elemento.
>> peso_C = 9*12; % Carbono
>> peso_O = 4*16; % Oxigeno
>> peso_H = 6*1 ; % Hidrogeno
>> % Calculo del peso molecular
>> peso_molecular = peso_C + peso_O + peso_H
peso_molecular = 178
>> % Calculo del porcentaje de oxigeno
>> porcentaje_O = peso_O/peso_molecular*100
porcentaje_O = 35.9551

```

Ejercicios

-  ▶ **5** Ejecuta el programa anterior con **OCTAVE**, comprobando que el resultado es correcto.

Por último, podemos utilizar puntos suspensivos (...) para continuar una orden larga en otra línea. En este caso, no se permite la división de nombres y continuación de comentarios:

```

>> peso_molecular = peso_C + ... % Continua en siguiente linea
peso_O + peso_H
peso_molecular = 178

```

1.2.4. Formato numérico

Al hablar del lenguaje de programación **OCTAVE**, al contrario que en otros lenguajes, no tiene demasiado sentido hablar de tipos de variables (entero, real, doble precisión, etc), ya que **OCTAVE** almacena todas las variables con doble precisión (con 16 decimales), aún cuando sean enteros simples⁵. Sin embargo, sí que podemos cambiar el formato con el que se muestran las variables. Para ello, podemos utilizar las órdenes que se muestran en la Tabla 1.3, en la que se muestra la representación de `pi` tras ejecutar las órdenes respectivas.

⁵ Esto sorprenderá a aquellos que conozcan otros lenguajes de programación, como C, en los que hay que declarar el tipo de variable. En **OCTAVE**, todas las variables se presuponen que son de doble precisión, por lo que no es necesaria la declaración de tipos.

Orden	pi
<code>format short</code>	3.1416
<code>format long</code>	3.14159265358979
<code>format short e</code>	3.1416e+00
<code>format long e</code>	3.14159265358979e+00

Tabla 1.3: Distintos formatos numéricos de OCTAVE.

Cabe insistir en que estas órdenes únicamente afectan a la impresión de un número en pantalla, y no a la representación interna, que siempre es de 16 dígitos.

Ejercicios

-  ▶ **6** Comprueba el efecto de los formatos de la Tabla 1.3 sobre la variable `pi`. Para ello, ejecuta primero la orden `format` correspondiente antes de mostrar el valor de `pi`.

1.3. Funciones matemáticas

Hasta ahora hemos estudiado las operaciones aritméticas básicas (suma, resta, multiplicación, división y exponenciación), que son las más utilizadas en cálculos básicos. Sin embargo, en ocasiones son necesarias operaciones más complejas, como cálculo de raíces cuadradas, operaciones trigonométricas, valores absolutos, etc. Para ello, OCTAVE dispone de una serie de *funciones predefinidas* que nos permiten realizar estas operaciones. Así, para calcular la raíz cuadrada de un número utilizaremos la función `sqrt`, introduciendo el número entre paréntesis después del nombre de la función:

```
>> a = 18;
>> b = sqrt(a)
b = 4.2426
>> sqrt(4)
ans = 2
```

A los parámetros (constantes, variables o expresiones) que se le pasan a una función entre paréntesis se les denomina *argumentos*. Una función puede no tener ningún argumento, uno (`sqrt`), o varios argumentos separados por comas. Por ejemplo, la función `rem` calcula el resto de la división entera de dos números enteros:

```
>> rem(25,4)
ans = 1
```

ya que si dividimos $25/4$, el resultado es 6 con un resto de 1. Por supuesto, los argumentos de una función también pueden contener otras llamadas a otras funciones,

```
>> rem(sqrt(16),4)
ans = 0
```

1.3.1. Funciones matemáticas elementales

En la Tabla 1.4 se proporciona una lista de algunas funciones matemáticas simples, como raíces cuadradas, exponenciales y logaritmos, así como otras utilizadas para redondear números.

Función	Comentario	Ejemplo	
		Orden	Resultado
<code>sqrt(x)</code>	Raíz cuadrada de x	<code>sqrt(25)</code>	5
<code>exp(x)</code>	e^x (e = base de logaritmos neperianos)	<code>exp(1)</code>	2.7183
<code>log(x)</code>	$\ln(x)$, logaritmo neperiano de x	<code>log(2.7183)</code>	1.0000
<code>log10(x)</code>	$\log_{10}(x)$, logaritmo de x en base 10	<code>log10(100)</code>	2
<code>rem(x,y)</code>	Resto de la división x/y	<code>rem(17,7)</code>	3
<code>abs(x)</code>	Valor absoluto de x	<code>abs(-3.4)</code>	3.4000
<code>round(x)</code>	Redondea x al entero más cercano	<code>round(4.7)</code>	5
<code>floor(x)</code>	Redondea al entero inferior	<code>floor(4.7)</code>	4
<code>ceil(x)</code>	Redondea al entero superior	<code>ceil(4.3)</code>	5

Tabla 1.4: Funciones matemáticas elementales.

1.3.2. Funciones trigonométricas

En OCTAVE, se asume que los argumentos de funciones trigonométricas están en radianes⁶. En la Tabla 1.5 se ejemplifica el uso de algunas funciones trigonométricas.

Función	Comentario	Ejemplo	
		Orden	Resultado
<code>sin(x)</code>	Seno de x	<code>sin(30*pi/180)</code>	0.50000
<code>cos(x)</code>	Coseno de x	<code>cos(30*pi/180)</code>	0.86603
<code>tan(x)</code>	Tangente de x	<code>tan(30*pi/180)</code>	0.57735
<code>asin(x)</code>	Arcoseno de x	<code>asin(0.5)*180/pi</code>	30.000
<code>acos(x)</code>	Arcocoseno de x	<code>acos(0.87)*180/pi</code>	29.541
<code>atan(x)</code>	Arcotangente de x	<code>atan(0.58)*180/pi</code>	30.114

Tabla 1.5: Funciones trigonométricas.

Fíjate en que el resultado de las tres últimas funciones se encuentra en radianes, por lo que es necesario transformarlo para obtener grados sexagesimales.

Ejercicios

► 7 Evalúa las siguientes expresiones sin utilizar OCTAVE, y después comprueba tu respuesta haciendo uso de él. Si has realizado el ejercicio anterior, ejecuta antes la orden `format`.

- | | |
|-----------------------------|----------------------------|
| a) <code>round(-2.6)</code> | e) <code>round(2.6)</code> |
| b) <code>floor(-2.6)</code> | f) <code>floor(2.6)</code> |
| c) <code>abs(-2.6)</code> | g) <code>abs(2.6)</code> |
| d) <code>ceil(-2.6)</code> | h) <code>ceil(2.6)</code> |

⁶ Recordemos las expresiones para transformar radianes a grados sexagesimales y viceversa:

$$\text{radianes} = \text{grados} * \pi / 180$$

$$\text{grados} = \text{radianes} * 180 / \pi .$$

En este ejemplo, todas las variables definidas son escalares de dimensión 1×1 .

La orden `clear` borra variables del espacio de trabajo `OCTAVE`. En el siguiente ejemplo, se elimina la variable `peso_C`, de forma que un intento de utilización posterior de dicha variable genera un mensaje de error.

```
>> clear peso_C
>> who
Your variables are:
__nargin__      peso_0          peso_H
peso_molecular  porcentaje_0
>> peso_C
??? Undefined function or variable 'peso_C'.
```

Pero hay que ir con cuidado con esta orden, ya que si se ejecuta sin ningún nombre de variable borra todas las variables y funciones de la memoria.

1.4.2. Historia de la sesión

`OCTAVE` también memoriza las expresiones introducidas en una sesión de trabajo en el mismo orden en que fueron ejecutadas, de forma que podemos *navegar* por la historia de la sesión repitiendo órdenes ya ejecutadas. Para ello se hace uso de los cursores: con las teclas \uparrow y \downarrow recordamos las expresiones introducidas hacia el principio o el final, respectivamente, y con las teclas \leftarrow y \rightarrow podemos editar la orden para modificarla convenientemente.

1.5. Ficheros-M

En el caso de problemas sencillos, como los tratados hasta ahora, es útil introducir directamente las órdenes en la ventana de `OCTAVE`, pero en el caso de problemas más complejos o en la repetición de las mismas órdenes con distintos valores de variables, esta técnica resulta muy incómoda. Para automatizar la ejecución de secuencias de órdenes, `OCTAVE` permite utilizar *ficheros-M*, cuyo nombre proviene de su extensión “.m”. La secuencia de órdenes contenida en un fichero-M constituye un *programa*, que podremos ejecutar fácilmente cuando lo deseemos.

Para crear un fichero-M utilizaremos un editor de texto cualquiera guardándolo como `nombre.m`, siendo `nombre` el nombre del programa. En el ejemplo siguiente se ha guardado el fichero-M con el nombre `pmol.m`.

```
%*****
% Programa : pmol.m
% Descripcion: Este programa calcula el peso molecular de una
% molecula organica.
%*****
% Calculo de los g/mol de cada elemento.
peso_C = 12*9 ;
peso_H = 1*6 ;
peso_O = 16*4;
% Calculo del peso molecular
peso_molecular = peso_C + peso_H + peso_O
% Calculo del porcentaje de oxigeno
porcentaje_0 = peso_O/peso_molecular*100
```

Para ejecutar este programa basta con escribir en la ventana de `OCTAVE` su nombre sin la extensión “.m”. Así, ejecutaríamos el ejemplo anterior como se muestra a continuación⁷:

```

>> pmol
peso_molecular = 178
porcentaje_0 = 35.9551

```

Al ejecutar el programa, se ejecutan una tras otra las instrucciones contenidas en el fichero `pmol.m`, en el mismo orden en que aparecen en éste. Como era de esperar, sólo se muestran los resultados de las instrucciones no finalizadas con punto y coma.

Las variables que se usan en el programa se quedan en el entorno de trabajo de `OCTAVE` cuando finaliza la ejecución del programa. Por ello, debemos evitar utilizar en ficheros-M nombres de variables que coincidan con el nombre del propio programa, ya que en ese caso al realizar posteriores llamadas al programa se mostraría simplemente el valor de la variable. Es decir, si nuestro programa lo hemos guardado con el nombre `pmol.m`, debemos evitar el uso de una variable `pmol`.

Ahora resulta evidente la utilidad de los ficheros-M a la hora de repetir conjuntos de órdenes. Así, si quisiéramos calcular el peso molecular de distintas moléculas, bastaría con que cambiáramos los números de cada tipo de átomos. Además, es aconsejable documentar los ficheros-M con comentarios, para que no olvidemos en el futuro qué pretendíamos al escribir el programa.

Ejercicios

► **9** Escribe con editor de texto el programa `pmol.m`, y prueba a ejecutarlo.

► **10** Escribe y ejecuta un programa (llámalo `nitrogeno.m`) que calcule la presión P que ejerce 1 mol de N_2 en un volumen de 0.419 l. a 227C, según la ecuación de Van der Waals:

$$P = \frac{nRT}{V - nb} - \frac{n^2a}{V^2} .$$

En esta ecuación, n es el número de moles de un gas, T la temperatura y V el volumen a las que se encuentra. R es la constante de los gases ideales ($R = 0.082 \frac{\text{litró-atm}}{\text{mol-grad}}$). Las constantes del N_2 son $a = 1.390 \text{ l}^2\text{atm/mol}^2$ y $b = 3.913\text{e-}02 \text{ l/mol}$.

1.6. Instrucciones de entrada/salida

1.6.1. Salidas por pantalla

En ocasiones, estaremos interesados en mostrar texto por pantalla, por ejemplo, para informar al usuario del uso del programa. Esto se puede conseguir con la orden `disp('texto')`. Si queremos mostrar el valor de una variable previamente calculada, podemos utilizar la misma orden incluyendo el nombre de la variable sin comillas:

```

>> disp('Este programa calcula...')
Este programa calcula...
>> t = 5 ;
>> disp(t)
5

```

⁷ También podemos ejecutar el programa sin entrar en `OCTAVE`: basta con ejecutar en un terminal Linux la orden `octave -q pmol.m`. ¿Por qué no pruebas a hacerlo?

La orden `fprintf`⁸ permite tener un mayor control sobre las salidas por pantalla que el que se tiene con `disp`, ya que permite especificar el formato con el que se van a mostrar los valores. La sintaxis de esta orden es la siguiente:

```
fprintf(formato, expresiones)
```

El formato contiene el texto y las especificaciones de formato para las salidas, y va seguido de los nombres de las expresiones cuyo valor se desea visualizar separadas por comas. Dentro del formato se pueden usar los especificadores `%e`, `%f` y `%g`. Si se usa `%e` los valores se exhiben en una notación exponencial; si se usa `%f` se muestran en notación decimal; y si se usa `%g`, se usará el formato que sea más corto de los dos anteriores. La cadena `\n` fuerza un cambio de línea en la salida. Veamos un ejemplo:

```
>> presion = 5.64;
>> temp = 245 ;
>> fprintf('A %g atm. \n la temperatura es %f K', presion, temp);
   A 5.64 atm.
     la temperatura es 245.000000 K
```

Fíjate en que se ha producido un cambio de línea a la mitad de la frase debido al especificador `\n`. Además hay una variable por cada uno de los especificadores de formato: `presion` se corresponde con `%g` y `temp` con `%f`.

Los especificadores de formato también pueden contener información para especificar el número de posiciones decimales que se muestran y las posiciones totales que se van a ocupar en la salida. Veamos un ejemplo para entender mejor el uso de esta funcionalidad: si ejecutamos la orden

```
>> fprintf('La temperatura es %6.2f K\n', temp);
```

lo que indicamos es que se reserven **6** posiciones para mostrar la variable `temp`, con **2** decimales. Las posiciones reservadas se ocupan (de derecha a izquierda) con los decimales, el punto, la parte no decimal y, en su caso, el signo. Si se reservan más posiciones de las necesarias, éstas se muestran como espacios en blanco a la izquierda. Veamos la salida de la orden anterior:

```
La temperatura es 245.00 K
```

Fíjate en qué ocurre si aumentamos el número de posiciones reservadas:

```
>> fprintf('La temperatura es %8.2f K\n', temp);
La temperatura es   245.00 K
```

En general, si sólo estamos interesados en indicar el número de decimales, podemos obviar la reserva de posiciones.

⁸ La orden original de `OCTAVE` es `printf`, pero aquí utilizaremos la orden `fprintf` porque, además de ser también correcta, es compatible con `Matlab`.

```
>> fprintf('La temperatura es%.4f K\n', temp);
La temperatura es 245.0000 K
```

Ejercicios

-  **11** Escribe las instrucciones OCTAVE adecuadas para que se impriman las siguientes frases con el formato que se indica. Ten en cuenta que previamente se han definido las siguientes variables:

$$a = 5 \quad b = 48.56 \quad c = -4.7864 \quad d = 1111111111$$

- | | |
|------------------------------|--------------------------------|
| 1. El valor de a es 5 | 6. El valor de c es -4.7864 |
| 2. El valor de a es 5.00 | 7. El valor de c es -4.8 |
| 3. El valor de b es 49 | 8. El valor de d es 1.111e+010 |
| 4. El valor de b es 48.56 | 9. El valor de d es 1.e+010 |
| 5. El valor de b es 48.56000 | |

1.6.2. Entrada de datos

Con lo visto hasta ahora, hemos podido realizar cálculos con distintas moléculas sin más que modificar el programa `pmol.m`. Sin embargo, sería más interesante que ni siquiera hubiera que modificar dicho fichero, sino que el propio programa nos “preguntara” los cambios que queremos hacer. Por ello, cualquier lenguaje de programación posee instrucciones que permiten al usuario introducir datos durante la ejecución del programa. En el caso de OCTAVE disponemos de la orden `input`. Consideremos el siguiente fichero-M, modificación del archivo `pmol.m` y documentado con comentarios:

```
%*****
% Programa : pmol.m
% Descripción: Este programa calcula el peso molecular de una
% molecula organica.
%*****
% Calculo de los g/mol de cada elemento.
disp('Programa para el calculo de pesos moleculares');
numero_C = input('Introduce el numero de atomos de carbono: ');
numero_H = input('Introduce el numero de atomos de hidrogeno: ');
numero_O = input('Introduce el numero de atomos de oxigeno: ');

% Calculo de los pesos de los elementos
peso_C = 12*numero_C ;
peso_H = 1*numero_H ;
peso_O = 16*numero_O;

% Calculo del peso molecular
peso_molecular = peso_C + peso_H + peso_O ;
fprintf('El peso molecular de la molecula es %g g/mol. \n', ...
    peso_molecular);
```

```
% Calculo del porcentaje de oxigeno
porcentaje_0 = peso_0/peso_molecular*100;
fprintf('El porcentaje en oxigeno de la molecula es%8.4f.\n', ...
    porcentaje_0);
```

Al ejecutar este programa, cada una de las instrucciones `input` mostraría en pantalla el mensaje que incluye, y esperaría a que el usuario introdujera un dato y presionara la tecla *Intro*, almacenando dicho dato en la variable correspondiente:

```
>> pmol
Programa para el calculo de pesos moleculares
Introduce el numero de atomos de carbono: 9
Introduce el numero de atomos de hidrogeno: 6
Introduce el numero de atomos de oxigeno: 4
El peso molecular de la molecula es 178 g/mol.
El porcentaje en oxigeno de la molecula es 35.955.
```

Ejercicios

 **12** Modifica el programa `pmol.m` para que incluya los cambios de entrada de datos e impresión de resultados. Ejecuta dicho programa y comprueba que da los resultados esperados, aplicándolo sobre las siguientes moléculas orgánicas:

- Aspirina ($C_9H_8O_6$).
- Benceno (C_6H_6).
- Alcohol Etilico (CH_3CH_2OH).
- Ácido Acético (CH_3COOH).
- Acetona (CH_3COCH_3).

1.7. Ayuda de Octave

OCTAVE es capaz de proporcionar ayuda al usuario mediante distintos medios. Por ejemplo, si queremos recordar la forma de uso de la orden `rem` (vista anteriormente), podemos obtener una respuesta rápida con la orden `help rem`:

```
>> help rem
rem is the user-defined function from the file
/usr/share/octave/2.1.40/m/general/rem.m

- Mapping Function: rem (X, Y)
  Return the remainder of 'X / Y', computed using the expression

      x - y .* fix (x ./ y)

An error message is printed if the dimensions of the arguments
do not agree, or if either of the arguments is complex.

See also: mod, round.
```

Ejercicios

-  **▶ 13** Una función interesante que no hemos estudiado es la función `diary`. Busca ayuda sobre el funcionamiento de esta función.
-

1.8. Ejercicios prácticos

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

Ejercicios

► **14** Escribe un programa llamado `OperBas.m` que pida al usuario dos números y calcule con ellos las operaciones básicas suma, resta, multiplicación y división, mostrando los resultados en pantalla.

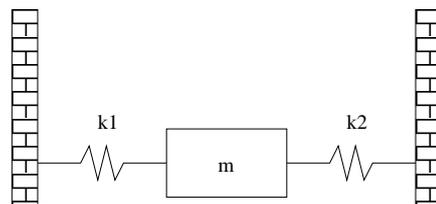
► **15** Escribe un programa llamado `fahrenheit.m` que, dada una temperatura en grados Celsius, la convierta en grados Fahrenheit. La expresión matemática para realizar dicho cambio es:

$$F = \frac{9}{5}C + 32$$

► **16** La siguiente figura muestra una masa m en reposo sobre una superficie sin rozamiento. La masa está conectada a dos muros por muelles con constantes elásticas k_1 y k_2 . El periodo de este sistema viene dado por la expresión

$$T = 2\pi\sqrt{\frac{m}{k_1 + k_2}}$$

Escribe un programa `OCTAVE` llamado `muelles.m` que pida al usuario los valores de m , k_1 y k_2 y que calcule y muestre el periodo T .

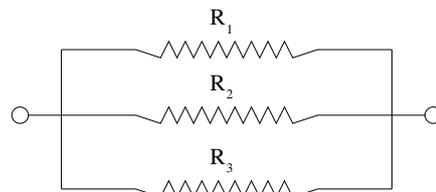


► **17** Escribe un programa llamado `distancia.m` que calcule la distancia entre dos puntos de un plano (x_1, y_1) y (x_2, y_2) .

$$\text{distancia} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Deberás pedir las coordenadas al usuario del programa.

► **18** Escribe un programa de nombre `resistencia.m` que calcule la resistencia equivalente de un circuito de tres resistencias en paralelo como el que se muestra en la siguiente figura.



La expresión de la resistencia equivalente viene dada por

$$R_{eq} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}}$$

- **19** Escribe un programa llamado `GasIdeal.m` que utilice la ecuación de los gases ideales

$$P = \frac{nRT}{V}$$

de modo que pida al usuario el número de moles n de un gas, la temperatura T y volumen V a las que se encuentra, y calcule la presión P según esta ecuación ($R = 0.082 \frac{\text{litró}\cdot\text{atm}}{\text{mol}\cdot\text{grad}}$).

▷ *Ejemplo: 1 mol de N_2 que ocupa 0.419 l. a 227C, ejerce una presión de 97.85 atm.*

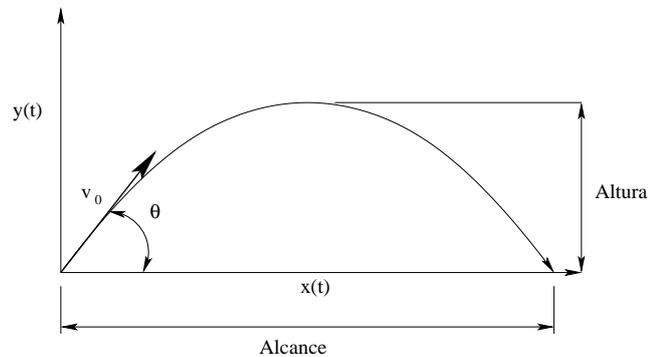
- **20** Repite el ejercicio anterior utilizando la ecuación de Van der Waals para gases imperfectos,

$$P = \frac{nRT}{V - nb} - \frac{n^2a}{V^2} .$$

Aquí deberás pedir al usuario, además los valores de a y b . El programa deberá llamarse `VanDerWaals.m`.

▷ *Ejemplo: 1 mol de N_2 que ocupa 0.419 l. a 227C, ejerce una presión de 100 atm. Las constantes del N_2 son $a = 1.390 \text{ l}^2\text{atm/mol}^2$ y $b = 3.913\text{e-}02 \text{ l/mol}$*

- **21** La siguiente figura muestra la trayectoria de un proyectil lanzado a una velocidad v_0 y un ángulo θ sobre un plano horizontal.



Suponiendo que pueden despreciarse todos los efectos de resistencia del aire, las ecuaciones del movimiento en los dos ejes vienen dadas por las ecuaciones siguientes:

$$x(t) = v_0 \cos(\theta)t$$

$$y(t) = v_0 \sin(\theta)t - \frac{1}{2}gt^2$$

donde $g = 9.81\text{m/s}^2$. Teniendo en cuenta que la altura máxima del proyectil se corresponde con el punto donde $dy/dt = 0$, y que el movimiento es parabólico y, por lo tanto, simétrico, es relativamente fácil obtener la duración del vuelo del proyectil y su alcance.

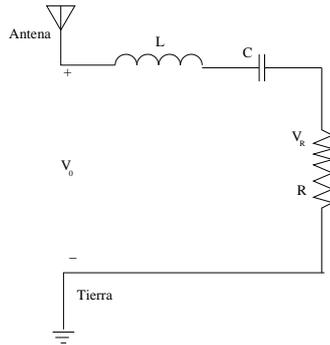
$$\text{Tiempo de vuelo} = \frac{2v_0 \sin(\theta)}{g}$$

$$\text{Alcance} = \frac{v_0^2 \sin(2\theta)}{g}$$

Escribe un programa en `OCTAVE` de nombre `proyectil.m` que calcule e imprima el tiempo de vuelo y alcance de un proyectil, pidiendo al usuario los valores de v_0 y θ . Ejecútalo con distintos valores de ángulos entre 0 y 90 grados (de 5 en 5 grados, por ejemplo), comprobando que se obtiene el alcance máximo para $\theta = 45$ y el tiempo de vuelo máximo para $\theta = 90$.

▷ *Ejemplo: Para $v_0 = 15\text{m/s}$ y $\theta = 45^\circ$, el tiempo de vuelo es de 2.16 s. y el alcance 22.94 m.*

► 22 En la siguiente figura se muestra una versión simplificada de un radio receptor AM.



El voltaje que atraviesa la resistencia V_R varía en función de la frecuencia según la ecuación

$$V_R = \frac{R}{\sqrt{R^2 + \left(\omega L - \frac{1}{\omega C}\right)^2}} V_0$$

donde $\omega = 2\pi f$, y f es la frecuencia en hercios. Escribe un programa que pida al usuario los valores de R , V_0 , f , L y C , y calcule el voltaje V_R . La salida del programa deberá ser lo más parecida a la que se muestra a continuación.

```
% Ejemplo de uso
>> Voltaje_antena
Introduce el valor de la resistencia R (en ohms): 50
Introduce el valor del voltaje V0 (en voltios): 0.01
Introduce el valor de la frecuencia f (en Hertzios): 1E6
Introduce el valor de la inductancia L (en henries): 0.0001
Introduce el valor de la capacitancia C (en faradays): 0.25E-9
El valor de VR es 0.00986496 voltios.
```