

Herramientas en GNU/Linux para estudiantes universitarios

Gnu/Octave: cálculo numérico por ordenador

Juan José García Rojo

Herramientas en GNU/Linux para estudiantes universitarios: Gnu/Octave: cálculo numérico por ordenador

por Juan José García Rojo

Copyright (c) 2.003 Juan José García Rojo..

<jugaro@users.sf.net>

Se permite la copia, distribución y/o modificación de este documento bajo los términos de la GNU Free documentation License, Versión 1.2 o cualquier otra versión posterior publicada por la Free Software Foundation, sin partes no modificables y sin añadidos en la portada o contraportada. Una copia de esta licencia se incluye en la sección titulada "GNU Free Documentation License".

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Tabla de contenidos

1. Introducción a Octave	1
1.1. ¿Qué es Octave?	1
1.2. Usar Octave	1
1.2.1. Ejemplos sencillos	1
1.2.2. Comentarios en Octave	5
2. Sintaxis de Octave	6
2.1. Tipos de datos	6
2.1.1. Objetos numéricos	6
2.1.2. Estructuras	9
2.2. Variables	10
2.3. Operadores	11
2.3.1. Operadores aritméticos	11
2.3.2. Operadores de comparación	12
2.3.3. Operadores booleanos	13
2.3.4. Operadores booleanos "short-circuit"	14
2.3.5. Operador de asignación	14
2.4. Expresiones de control de flujo	15
2.4.1. Sentencia if	15
2.4.2. Sentencia while	16
2.4.3. Sentencia for	16
2.4.4. Sentencias break y continue	16
2.4.5. Manejo de excepciones	17
2.5. Funciones y scripts de usuario	17
3. Funciones de entrada/salida	20
3.1. Funciones básicas de escritura en disco	20
3.2. Gráficas	20
4. Problemas de álgebra lineal	21
5. Estadística	22
6. Procesado de señal	23
6.1. Procesado de audio	23
6.2. Procesado de imagen	23
7. Referencias	24
A. GNU Free Documentation License	25
A.1. PREAMBLE	25
A.2. APPLICABILITY AND DEFINITIONS	25
A.3. VERBATIM COPYING	26
A.4. COPYING IN QUANTITY	26
A.5. MODIFICATIONS	27
A.6. COMBINING DOCUMENTS	28
A.7. COLLECTIONS OF DOCUMENTS	28
A.8. AGGREGATION WITH INDEPENDENT WORKS	29
A.9. TRANSLATION	29
A.10. TERMINATION	29

A.11. FUTURE REVISIONS OF THIS LICENSE29
A.12. ADDENDUM: How to use this License for your documents30

Lista de tablas

2-1. Secuencias de escape en strings.....	8
2-2. Operadores aritméticos.....	11
2-3. Operadores de comparación.....	13
2-4. Operadores booleanos.....	13
2-5. Operadores binarios de corto circuito.....	14

Tabla de ejemplos

1-1. Definir números.....	2
1-2. Crear un vector.....	2
1-3. Crear una matriz.....	3
1-4. Aritmética de matrices.....	3
1-5. Solución de ecuaciones lineales.....	4
1-6. Obteniendo ayuda.....	4

Capítulo 1. Introducción a Octave

1.1. ¿Qué es Octave?

Octave es un lenguaje de alto nivel para realizar cálculos numéricos en el ordenador, y también es un programa capaz de interpretar este lenguaje y realizar los cálculos. Octave ofrece una interfaz de usuario interactiva, orientada a línea de comandos, pero también puede ser utilizado en modo no interactivo, leyendo sus órdenes de fichero.

Octave originalmente fue desarrollado para facilitar la tarea a los estudiantes de Ingeniería Química de la universidad de Texas, sin que estos tuvieran que enfrentarse a las dificultades de la programación. Su flexibilidad en seguida lo hizo popular y su uso se expandió a otros problemas relacionados con el álgebra lineal y las ecuaciones diferenciales y favoreció su desarrollo, agregando las aportaciones de la comunidad de usuarios.

Otros programas de características similares, y hasta cierto punto compatibles, son el lenguaje R de la FSF, Matlab y Scilab. Estos dos últimos propietarios. Octave es software libre (bajo licencia GNU), lo que significa que se puede usar y redistribuir libremente, y que cualquiera puede ayudar para mejorarlo. Octave está disponible en Internet en <http://www.octave.org>.

1.2. Usar Octave

En la mayoría de los casos, la forma de arrancar Octave es desde la línea de comandos tecleando **octave**. Octave muestra un mensaje inicial y un prompt indicando que está esperando órdenes del usuario.

```
$ octave
GNU Octave, version 2.0.16.92 (i386-pc-linux-gnu).
Copyright (C) 1996, 1997, 1998, 1999, 2000 John W. Eaton.
This is free software with ABSOLUTELY NO WARRANTY.
For details, type 'warranty'.
```

```
octave:1>
```

Para terminar octave, basta con teclear **quit** o **exit** en el prompt de Octave.

```
octave:1> exit
```

1.2.1. Ejemplos sencillos

Los capítulos siguientes describen las características de octave en mayor detalle, pero antes de continuar es recomendable mostrar (e intentar) algunos ejemplos. Cada vez que se complete una línea con una pulsación de retorno de carro, Octave responderá con un segundo prompt para seguir recibiendo entrada del usuario, o con la respuesta (puede tardar más o menos según el número de operaciones necesaria para computarla) si el usuario introdujo una orden completa.

Según se van introduciendo los comandos Octave los almacena en un "histórico de comandos" y permite su recuperación y edición. Simplemente pulsando las flechas del teclado numérico, o si esto no funcionara con las teclas C-b o C-f para mover el cursor a izquierdas o derechas en el comando que estamos tecleando, y con C-p y C-n para recuperar órdenes anteriores (retroceder en la historia) o volver a las últimas órdenes.

Sugerencia: Existen otras muchas combinaciones de teclas útiles, la mayor parte de las cuales coinciden con las de emacs.

Ejemplo 1-1. Definir números

El siguiente ejemplo define dos números, el primero el número real 2, y el segundo el número complejo $2+2j$.

```
octave:1> a=2.1
a = 2.1
octave:2> b=a+2j
b = 2.1 + 2i
```

Inicialmente, los símbolos i y j son representaciones de la unidad imaginaria ($i*i=-1$, $j*j=-1$). Ambos símbolos son equivalentes.

Atención

Esto es cierto siempre y cuando no se asigne otro valor a las variables i y j o a sus correspondientes mayúsculas. Por eso motivo se desaconseja su uso como variable. Lo mismo sucede con las constantes π (relación entre la longitud de la circunferencia y su diámetro) y e (base del logaritmo neperiano)

En la definición de la variable b se ha usado la variable a . Así, si una variable está a la izquierda del signo $=$ se estará asignando valor a esa variable, y si está a la derecha será sustituida por su valor a la hora de hacer los cálculos.

También se observa que no hay ninguna diferencia a la hora de definir un número real, complejo o imaginario, o al usarlo, lo que supone una gran comodidad a la hora de programar.

Atención

Octave usa el $.$ (punto) para definir números decimales (exactamente igual que en las calculadoras).

Para obtener el valor de una variable basta con teclear el nombre de la variable a continuación del prompt.

```
octave:3> b
b = 2.1 + 2i
```

Ejemplo 1-2. Crear un vector

Si queremos crear un vector fila con los elementos 1, 2, 1:

```
octave:4> v= [1 2, 1]
v =
  1 2 1
```

Para crear un vector columna

```
octave:5> w= [
> 1; -1
> 2
> ]
w =
    1
   -1
    2
```

Los símbolos [y] se usan para definir vectores y matrices. Si se trata de un vector fila, se introducen los elementos separados por espacios (o tabuladores) o por , (coma). Si es un vector columna se introducen los elementos separados por retornos de carro o por ; (puntos y coma). En el caso de las matrices, los elementos se introducen por filas. Está permitido usar cualquier cantidad de espacios para separar elementos.

Ejemplo 1-3. Crear una matriz

Para almacenar la matriz identidad de rango 3 en una variable escribiríamos lo siguiente:

```
octave:6> I= [
> 1 0 0
> 0 1 0; 0,0,1 ]
I =
    1 0 0
    0 1 0
    0 0 1
```

Octave responde imprimiendo por pantalla la matriz correctamente formateada. A partir de este momento la variable I contendrá la matriz identidad de orden 3. El mismo resultado hubiéramos podido conseguir usando la función **eye(3)**. Otros ejemplos de funciones que crean matrices son ones(), zeros() y rand().

Sugerencia: Si desea saber como utilizar estas funciones teclee en el prompt de Octave help seguido por el nombre de la función. Por ejemplo: **help ones**. Por desgracia la ayuda estará en inglés.

Ejemplo 1-4. Aritmética de matrices

La notación de octave para aritmética de matrices, vectores y números es clara y sencilla. Por ejemplo:

```
octave:7> b * I
ans =
    2.10000 + 2.00000i    0.00000 + 0.00000i    0.00000 + 0.00000i
    0.00000 + 0.00000i    2.10000 + 2.00000i    0.00000 + 0.00000i
    0.00000 + 0.00000i    0.00000 + 0.00000i    2.10000 + 2.00000i
```


Y para multiplicar vectores:

```
octave:8> v * w
ans = 1
octave:9> w * v
ans =
    1    2    1
   -1   -2   -1
    2    4    2
octave:10> v .* w'
ans =
    1  -2  2
```

Los vectores v y w son vectores de 1×3 y 3×1 respectivamente, esto explica que en el primer caso el resultado sea escalar y en el segundo una matriz 3×3 . En el tercero se multiplican las componentes una a una (operador `.*`), y para que esta operación no de error es necesario que la dimensión de los dos operadores sea la misma. En este caso se transpone w mediante el operador `'` (apóstrofe):

```
octave:11> w'
ans =
    1  -1  2
```

Ejemplo 1-5. Solución de ecuaciones lineales

Para solver un conjunto de ecuaciones lineales del tipo $\mathbf{Ax}=\mathbf{b}$ disponemos del operador "división por la izquierda", `\`:

```
octave:12> A=[1 2 3; 3 1 2; 2 3 1];
octave:13> A\b
ans =
  -0.55556
   1.11111
  -0.22222
```

Sugerencia: Un `;` (punto y coma) al final de una orden hace que octave después de ejecutar la orden suprima su respuesta y nos presente el siguiente prompt.

Ejemplo 1-6. Obteniendo ayuda

Para poder obtener información de Octave es necesario conocer el nombre de la orden que se quiere usar. Este nombre no tiene por qué ser obvio. Un buen sitio para empezar es simplemente teclear `help`. Esto nos mostrará todos los operadores, palabras reservadas, funciones, variables predefinidas (built-in) y ficheros de funciones. Si ya se conoce el nombre el comando, simplemente hay que pasarlo como parámetro:

```
octave:14> help rand
zeros is a builtin function
```

`zeros (N)`, `zeros (N, M)`, `zeros (X)`: create a matrix of all zeros

Additional help for builtin functions, operators, and variables is available in the on-line version of the manual. Use the command `'help -i <topic>'` to search the manual index.

Help and information about Octave is also available on the WWW at <http://www.che.wisc.edu/octave/octave.html> and via the `help-octave@bevo.che.wisc.edu` mailing list.

La tercera forma de obtener ayuda es por medio de `help -i concepto`, donde *concepto* es la palabra clave que queremos buscar.

Atención

Para que este comando funcione es necesario tener instalada la utilidad externa `info` (también software libre) y la documentación de octave en el formato de esta herramienta.

1.2.2. Comentarios en Octave

Siempre es recomendable documentar el código, para hacerlo más inteligible a otras personas o al propio programador, por si en alguna ocasión tuviera que modificarlo. En Octave los comentarios empiezan con un signo `#` o `%`. Todo el texto que siga a continuación hasta fin de línea es considerado como un comentario y no es evaluado por Octave.

```
function y = determinante(X)
#
# Uso: y = determinante(X)
#
# Esta función calcula el determinante de la matriz X
# X debe ser una matriz cuadrada.
...
end
```

En el ejemplo se muestra parte de la definición de una función. El comando de Octave `help` es capaz de buscar el primer bloque de comentarios (aquel que sucede justo después de la palabra clave `function`) e imprimirlo en pantalla. Por ejemplo, después de definir la función anterior, cuando se teclee `help determinante` se obtendrá el siguiente resultado:

```
function y = determinante(X)

Uso: y = determinante(X)

Esta función calcula el determinante de la matriz X
X debe ser una matriz cuadrada.
```

Capítulo 2. Sintaxis de Octave

Una de las características principales de Octave y que hacen de él una herramienta de fácil aprendizaje es la simplicidad e intuitividad de su lenguaje de programación. En los siguientes apartados se describen las reglas básicas para escribir órdenes y programas usando este lenguaje.

2.1. Tipos de datos

Octave ofrece soporte para datos predefinidos, que incluyen escalares (reales y complejos), vectores, matrices, cadenas de caracteres y estructuras. También es posible la definición de nuevos datos por parte el usuario, programados en algún lenguaje que produzca código máquina (FORTRAN, C++, ...), aunque esta posibilidad aún no ha sido suficientemente documentada. Por este motivo en esta sección nos centraremos únicamente en los datos predefinidos (Built-in).

2.1.1. Objetos numéricos

Los objetos numéricos predefinidos de octave son los escalares (reales y complejos), y las matrices (los vectores se consideran matrices especiales, en que una de sus dimensiones es 1). Todos los datos numéricos son almacenados como números de doble precisión, lo que significa que en sistemas que usan el formato en punto flotante de la IEEE se pueden representar números en el rango $2.2e-308$ a $1.7e+308$ y la precisión aproximada de $2.2e-16$.

Sugerencia: Los valores exactos se encuentran en las variables `realmin`, `realmax` y `eps`, respectivamente.

2.1.1.1. Escalares

Los números escalares se pueden especificar en formato decimal, hexadecimal (precedidos por el prefijo `0x`). Los números en formato hexadecimal sólo pueden ser enteros. Los números decimales, además permiten usar notación científica, pudiéndose utilizar los símbolos `e`, `E`, `d` o `D` seguidos por un número `n` para significar "multiplicado por 10 elevado a la potencia `n`". Algunos ejemplos:

```
105
1.05e2
1050E-1
0x69
```

Todos ellos representan el número 105. Para representar el número complejo $10+11i$:

```
10 + 11i
1.0d1 + 0xBi
10 + 110D-1j
0xA + 0xbj
```

Recordar que `i` y `j` y sus correspondientes mayúsculas representan la unidad imaginaria, y que debe ser el último dígito de la cantidad imaginaria y sin espacios con el resto del número. Tanto `100je-1`, `j100` o `100 j` son incorrectos.

2.1.1.2. Rangos numéricos

Un rango es una forma más cómoda de escribir un vector con elementos equiespaciados. Un rango se define como una base o primer valor del rango, un valor opcional de incremento entre elementos y un valor máximo que el rango no superará. Estos tres elementos se separan por el símbolo : (dos puntos). El incremento puede ser positivo o negativo, y en caso de omitirse se asumirá valor 1.

```
octave> 1:10
ans =
    1    2    3    4    5    6    7    8    9   10
octave> 1:2:10
ans =
    1    3    5    7    9
```

Sugerencia: El máximo del rango puede que no sea alcanzado en la expansión del rango. Si es necesario que forme parte del conjunto de elementos y se conoce el número de elementos que debe contener, se recomienda usar la función `linspace()` en su lugar.

2.1.1.3. Matrices

Es muy simple definir matrices en Octave. La definición se hace introduciendo los elementos por filas, o bien concatenando otras matrices. Las dimensiones de la matriz son determinadas automáticamente a partir de los datos. Se produce un error si alguna de las filas o columnas de la matriz final tiene distinto número de elementos que el resto. Para separar filas podemos usar ; (punto y coma) o presionar un retorno de carro.

```
octave> a = [ 1 2; 3, 4 ]
a =
    1 2
    3 4
octave> b = [ a a ]
b =
    1 2 1 2
    3 4 3 4
```

Para recuperar un elemento de una matriz, simplemente tenemos que indicar el nombre de la variable seguido por unos paréntesis, y dentro el índice de fila y el de columna --en este orden-- del elemento que queremos recuperar.

```
octave> b(2,3)
ans = 3
```

En el caso de un vector basta con indicar un sólo índice (el de la posición del elemento), o se puede usar la notación de dos índices, pero en este caso si el vector es un vector fila el primer índice valdrá uno y si es un vector columna será el segundo índice el que valga uno.

Para recuperar varios elementos consecutivos de una misma fila de una matriz (esto es un vector fila), usaremos un rango como segundo índice. Si en cambio queremos recuperar un vector columna, usaremos un rango como primer índice. Y si lo que queremos es recuperar una submatriz usaremos rangos tanto para el primer como para el segundo índice.

```
octave> b(1,2:3)
ans =
    2    1
octave> b(1:2,3)
ans =
    1
    3
octave> b(:,2:3)
ans =
    2    1
    3    4
```

Sugerencia: Si se desean todos los elementos de una fila o de una columna, basta con indicar el rango sólo con los dos puntos: `b(1, :)` nos devolvería la primera fila de `b`.

Sugerencia: Para más información, desde el prompt de Octave teclear `help -i Matrices`

2.1.1.4. Cadenas de caracteres o strings

Las cadenas de caracteres o strings, como se les denomina en Octave, son secuencias de caracteres encerrados entre comillas simples (') o dobles (").

Sugerencia: Como la comilla simple (') es también el operador de transposición, se recomienda utilizar comillas dobles (").

Algunos caracteres no pueden ser incluidos de forma literal en un string, y es necesario introducir secuencias de caracteres equivalentes (secuencias de escape). La siguiente tabla muestra las secuencias de escape:

Tabla 2-1. Secuencias de escape en strings

Secuencia	Descripción
\\	Contra-barra (backslash): \.
\"	Comillas dobles ". Sólo es necesario si se utilizan las comillas dobles para limitar el string.

Secuencia	Descripción
\'	Comillas simples '. Sólo es necesario si se utilizan las comillas simples para limitar el string.
\a	Representa el carácter "alert" (normalmente suena el pitido de terminal).
\b	Representa el carácter de borrado (backspace o control-h).
\f	Representa el carácter "formfeed" (nueva página).
\n	Representa el carácter "nueva línea".
\r	Representa el carácter "retorno de carro".
\t	Representa el carácter tabulador. Normalmente no es necesario escaparlos, pero desde la línea de comandos de Octave el carácter tabulador invoca la función de edición completar palabra.
\v	Representa un tabulador vertical.

Las cadenas de caracteres pueden concatenarse utilizando la notación para definir matrices. Por ejemplo:

```
octave> [ "Bien" , "venidos", " al ", "curso" ]
ans = Bienvenidos al curso
```

Sugerencia: Para más información, desde el prompt de Octave teclear `help -i strings`

2.1.2. Estructuras

Octave posee soporte para organizar los datos en estructuras. La sintaxis es muy similar a la del lenguaje C. Los datos de una estructura pueden ser de cualquier tipo. Por ejemplo:

```
octave> x.numero = 1;
octave> x.matriz = [1 2; 2 1];
octave> x.cadena = "Hola";
octave> x
x =
{
  cadena = hola
  numero = 1
  matriz =
    1 2
    2 1
}
```

Las estructuras son ellas mismas objetos que pueden formar parte de otras estructuras. Las estructuras se pueden copiar (operador =) pero no tienen definida aritmética ni pueden ser elementos de una matriz. En casos en que hay anidamiento de estructuras, al mostrar el valor de un objeto de tipo estructura, octave puede elegir truncar la representación de algunos miembros para evitar confusión. Por ejemplo:

```
octave> a.b.c.d=1;
octave> a
a =
{
  b =
  {
    c = <structure>
  }
}
```

Sugerencia: El número de niveles de anidamiento de estructuras que se muestra se puede cambiar con la variable predefinida `struct_levels_to_print`. Su valor por omisión es de 2.

Sugerencia: Para más información, desde el prompt de Octave teclear `help -i struct`

2.2. Variables

Las variables nos permiten dar nombre a los valores que nos interesa guardar y poder referirnos a ellos más tarde. Ya se han visto muchos ejemplos de variables en los ejemplos anteriores. El nombre de una variable debe ser una secuencia de letras, dígitos, y subrayados (guión bajo), pero no puede empezar por dígito. Octave no impone ninguna restricción a la longitud de los nombres de variables.

Sin embargo nombres que empiecen por dos subrayados se reservan para el uso interno de Octave y sólo se deberán en el código para acceder a las variables predefinidas (y documentadas) de Octave.

El nombre de una variable es una expresión válida. Representa el valor almacenado en la variable. Una variable se define cuando se le asigna valor por primera vez, y no hay ningún problema en asignarle posteriormente distintos valores, aunque sean de otros tipos (por ejemplo entero y string). Se da valor a una variable utilizando el operador igual (=) y si se trata de variables numéricas, con los operadores de incremento.

Sugerencia: Este tema se tratará con más profundidad en la sección Sección 2.3.

Existen algunas variables predefinidas que tienen un significado especial, por ejemplo algunas permiten configurar la forma en que se comporta Octave, y otras almacenan constantes matemáticas útiles. Algunas de ellas son constantes y no pueden cambiarse, otras son pueden ser modificadas exactamente igual que una variable normal.

Las variables en Octave no tienen tipo fijo, así que es posible almacenar distintos tipos de dato en la misma variable. Únicamente hemos de ser cuidadosos para no utilizar una variable antes de haberla definido.

Un tipo especial de variable son las variables globales. Una variable global es aquella que se declara utilizando la palabra reservada `global`. Una variable declarada de esta manera puede ser accedida desde cualquier ámbito, independientemente de dónde se haya definido.

Sugerencia: Para más información, consultar `help -i Variables`

2.3. Operadores

Los operadores permiten construir sentencias más complejas. Es posible concatenar operaciones. Su precedencia es semejante a la de otros lenguajes, pero es posible modificarla agrupando las expresiones entre paréntesis.

Sugerencia: Para obtener la lista completa de la precedencia de los operadores teclear `help -i precedence`

2.3.1. Operadores aritméticos

Operan sobre escalares y matrices.

Tabla 2-2. Operadores aritméticos

Operador	Descripción
$\mathbf{x} + \mathbf{y}$	Suma. Si los dos operadores son matrices las dimensiones deben coincidir. Si no el escalar se suma a cada elemento de la matriz.
$\mathbf{x} .+ \mathbf{y}$	Suma elemento a elemento. Esta operación es equivalente a $+$.
$\mathbf{x} - \mathbf{y}$	Resta. El resultado es equivalente a $\mathbf{x} + (-\mathbf{y})$, donde $-\mathbf{y}$ representa el opuesto de \mathbf{y} .
$\mathbf{x} .- \mathbf{y}$	Resta elemento a elemento. Esta operación equivale a $-$.
$\mathbf{x} * \mathbf{y}$	Multiplicación de matrices. El número de columnas de \mathbf{x} debe coincidir con el número de filas de \mathbf{y} .
$\mathbf{x} .* \mathbf{y}$	Multiplicación de matrices elemento a elemento.

Operador	Descripción
<code>x / y</code>	División a derechas. Conceptualmente es equivalente a $(\text{inverso}(y') * x')$, pero sin necesidad de calcular la inversa de <code>y</code> ni calcular las transpuestas de las matrices.
<code>x ./ y</code>	División elemento a elemento, de los elementos de <code>x</code> divididos por los elementos de <code>y</code> .
<code>x \ y</code>	División por la izquierda. Conceptualmente es equivalente a $\text{inverso}(x) * y$.
<code>x .\ y</code>	División elemento a elemento, de los elementos de <code>y</code> entre los elementos de <code>x</code> .
<code>x ^ y</code> o <code>x ** y</code>	Operación de exponenciación. Ambos operadores no pueden ser matrices, y si alguno es matriz, deberá ser cuadrada.
<code>x .^ y</code> o <code>x .** y</code>	Exponenciación, elemento a elemento. Si ambos operadores son matrices, deberán tener igual dimensión.
<code>-x</code>	Negación. Se obtiene el escalar opuesto o la matriz de igual dimensión y cuyos elementos son los opuestos de la matriz original.
<code>+x</code>	Operador suma unitario. No tiene ninguna consecuencia sobre el operando.
<code>x'</code>	Conjugado complejo y transpuesta. Para números reales es equivalente a la transposición.
<code>x.'</code>	Transposición de los elementos de <code>x</code> . No se conjugan los elementos.
<code>++x</code>	Operador de preincremento. Equivalente a ejecutar la expresión <code>x=x+1</code> antes de acceder al valor de <code>x</code> .
<code>--x</code>	Operador de predecremento. Equivalente a ejecutar la expresión <code>x=x-1</code> antes de acceder al valor de <code>x</code> .
<code>x++</code>	Operador de postincremento. Equivalente a ejecutar la expresión <code>x=x+1</code> después de acceder al valor de <code>x</code> .
<code>x--</code>	Operador de postdecremento. Equivalente a ejecutar la expresión <code>x=x-1</code> después de acceder al valor de <code>x</code> .

Sugerencia: Si el valor de la variable predefinida `warm_divide_by_zero` es distinto de cero, y en algún caso se produce una división por cero, Octave imprime un aviso y continúa con las operaciones.

2.3.2. Operadores de comparación.

Todos los operadores de comparación devuelven un valor 1 si la comparación es cierta, y 0 si es falsa. Para matrices

las operaciones se realizan operando a operando. Por ejemplo:

```
octave> [1 2; 3 4] == [1 3; 2 4]
ans =
    1 0
    0 1
```

Si un operador es escalar y el otro una matriz, el escalar se compara con cada elemento de la matriz y el resultado tiene las mismas dimensiones que la matriz.

Tabla 2-3. Operadores de comparación

Operador	Descripción
$x < y$	Cierto si x es menor que y .
$x <= y$	Cierto si x es menor o igual que y .
$x == y$	Cierto si x es igual que y .
$x > y$	Cierto si x es mayor que y .
$x >= y$	Cierto si x es mayor o igual que y .
$x \neq y$ o $x \sim= y$ o $x <> y$	Cierto si x no es igual a y .

Sugerencia: Las comparaciones con cadenas de caracteres pueden hacerse con la función `strcmp`.

2.3.3. Operadores booleanos

El resultado de un operador booleano es una matriz de dimensión equivalente a los operandos, donde cada elemento es el resultado de aplicar el operador booleano a los elementos correspondientes. Se considera como cierto un valor distinto de cero, y falso un valor igual a cero. Los operadores booleanos se pueden emplear en las mismas situaciones que los operadores de comparación. Si además se utilizan en estructuras de control de flujo (if o while) sólo serán cierto si todos los elementos son distintos de cero.

```
octave> [1 0] & [1 1]
ans = 1 0
```

Únicamente hay tres operadores booleanos:

Tabla 2-4. Operadores booleanos

Operador	Descripción
<code>boolean1 & boolean2</code>	Operador 'and' lógico. Cada elementos del resultado es cierto si los elementos correspondientes de los operandos lo son.

Operador	Descripción
<code>boolean1 boolean2</code>	Operador 'or' lógico. Cada elementos del resultado es cierto si alguno de los elementos correspondientes de los operandos lo es.
<code>! boolean o ~ boolean</code>	Operador 'not' lógico. Cada elemento del resultado toma el valor booleano opuesto al del operando.

2.3.4. Operadores booleanos "short-circuit"

Los operadores booleanos de "corto circuito" son semejantes a los operadores booleanos, con la diferencia de que si después de evaluar el primer operando, ya es suficiente para obtener el resultado, no se comprueba el segundo operando.

Tabla 2-5. Operadores binarios de corto circuito

Operador	Descripción
<code>boolean1 && boolean2</code>	El operando <code>boolean1</code> es evaluado y convertido a un escalar (el resultado es análogo a <code>all(boolean1)</code>). Si el resultado es falso, la operación termina con resultado falso. Si es cierto se realiza la misma operación con el segundo operando y este será el resultado de la operación.
<code>boolean1 boolean2</code>	El operando <code>boolean1</code> es evaluado y convertido a un escalar (el resultado es análogo a <code>any(boolean1)</code>). Si el resultado es verdadero, la operación termina con resultado verdadero. En cambio si fue falso se evalúa de la misma manera el segundo operando el resultado obtenido será el resultado de la operación.

La diferencia entre los operadores binarios normales y los de cortocircuito se verá mejor con un ejemplo:

```
octave> a=0; b=0; a & b++, b
ans = 0
b = 1
octave> a=0; b=0; a && b++, b
ans = 0
b = 0
```

En el segundo caso, `b` será incrementado sólo si `a` es verdadero. En el primer caso `b` siempre será incrementado.

2.3.5. Operador de asignación

El signo `=` es el operador de asignación. Después de una asignación una variable cambia de valor y de tipo para acomodarse al del nuevo valor. El operador asignación es la única manera de poder almacenar valores. En el lado derecho puede aparecer cualquier expresión de las descritas anteriormente en este capítulo, o funciones que devuelvan

un valor. En el lado izquierdo podemos tener variables, elementos de una matriz o vector, o listas de valores de retorno (este concepto se aclarará en el apartado dedicado a funciones). Ejemplos:

```
octave> a=1;
octave> b=ones(2,3);
octave> b(:,1)= 2
b =
  2 1 1
  2 1 1
octave> [d c]= size(b);
octave> e=["esta " "es " "una " "cadena"];
```

2.4. Expresiones de control de flujo

Las expresiones de control de flujo influyen en la forma en que el código es ejecutado, y dividen el código en bloques. Los bloques empiezan con una palabra reservada, por ejemplo **while** y terminan con la palabra reservada **end-while**, formada por el prefijo end y la misma palabra que empezó el bucle. También es posible acabar un bloque con la palabra **end**. El código que se entre dichas palabras se denomina el cuerpo de la expresión de control.

Las siguientes construcciones son posibles

2.4.1. Sentencia if

```
if (condicion)
    cuerpo-entonces
end

if (condicion)
    cuerpo-entonces
else
    cuerpo-alternativa
end

if (condicion)
    cuerpo-entonces
elseif (condicion)
    cuerpo-entonces
else
    cuerpo-alternativa
end
```

Si la condición del **if** es cierta, entonces se ejecuta el cuerpo-entonces que está a continuación. Si es falso se pasa a evaluar la condición del siguiente **elseif**, ejecutando se el código correspondiente si su condición es cierta, o pasando al siguiente en caso contrario. Finalmente, si ninguna condición evaluó cierta se ejecuta el cuerpo-alternativa.

Tanto los **elseif**, como el **else** son opcionales, sin embargo puede haber tantos **elseif** como sean necesarios.

2.4.2. Sentencia while

La sintaxis es la siguiente

```
while (condicion)
    cuerpo
endwhile
```

Las sentencias del cuerpo se ejecutaran mientras condición se siga evaluando a cierto. Si en el cuerpo no se modifica alguna de las variables que se evalúan en la condición entraremos en un bucle infinito.

2.4.3. Sentencia for

La sintaxis de la sentencia **for** es muy semejante a la del **while**.

```
for var = expresion
    body
endfor
```

En el bucle **for** se evalúa la expresión una vez al principio y se asigna a la variable **var**. Sin embargo la expresión de asignación del **for** funciona un poco distinto que en el resto de los casos. En lugar de asignar la expresión completa, se asigna sólo una columna y el bucle va iterando en todas las columnas. Por ejemplo

```
octave> for a=ones(2,1)*[1:3]
>     a
> endfor
a =
  1
  1
a =
  2
  2
a =
  3
  3
```

Normalmente el bucle **for** se usa cuando se quieren hacer las cosas cierto número de veces (tantas como columnas tenga el resultado de evaluar la expresión).

2.4.4. Sentencias break y continue

Estas sentencias sólo pueden usarse dentro de bucles **for** y **while**.

break salta fuera del bucle más interno que lo encierra. En el siguiente ejemplo, el bucle terminará cuando la variable **var** valga 3:

```
for var=1:10
    a
    if(a==3)
```

```

        break
    endif
endfor

```

La sentencia **continue** es semejante, pero en lugar de terminar el bucle termina sólo la iteración actual. Ejemplo:

```

for var=1:10
    if(a==3)
        continue
    endif
    a
endfor

```

En este caso el bucle llegará hasta el final (**a=10**), pero no se imprimirá el valor de **a** cuando esta valga 3.

Normalmente estas dos sentencias se usan desde dentro de condiciones **if**.

2.4.5. Manejo de excepciones

Octave soporta un manejo limitado de excepciones con la sentencia **try**. La sintaxis es la siguiente:

```

try
    cuerpo
catch
    código_de_recuperacion
end_try_catch

```

El código_de_recuperación únicamente es ejecutado si sucede algún error en el cuerpo. Durante la ejecución del cuerpo no se imprime ningún mensaje de error. Si alguno ocurre se almacenan en la variable `__error_text__`, accesible desde el código_de_recuperación.

2.5. Funciones y scripts de usuario

Las funciones y scripts son el mecanismo que ofrece Octave para simplificar la escritura de programas o la carga de datos iniciales. Se pueden escribir directamente en la línea de comandos de Octave, o en ficheros externos. En ese caso los scripts y las funciones se invocan exactamente igual que cualquier otra función predefinida de Octave. La única condición es que esos ficheros se encuentren en un directorio dentro del path de Octave

Sugerencia: Para más información sobre el path de búsqueda de ficheros de Octave consultar `help -i LOAD-PATH`.

Sugerencia: Los ficheros de scripts y funciones de Octave tienen extensión ".m" por compatibilidad con Matlab.

Las funciones tienen el siguiente formato:

```
function variable_retornada = nombre_funcion (lista_argumentos)
    cuerpo_de_función
endfunction
```

El formato tiene los siguientes campos:

- Empiezan con la palabra clave `function` que indica el inicio de la declaración de una función de octave.
- Opcionalmente sigue la declaración de las variables retornadas. Puede ser una única variable (de cualquier tipo permitida en octave) o una lista de variables entre corchetes (`[]`). Las variables pueden ser de distintos tipos.

Si no existe la lista de variables, tampoco deberá aparecer el signo igual y la función no devolverá ningún valor.

- El nombre de la función
- Opcionalmente sigue la lista de argumentos entre paréntesis (`()`) y separados por comas. Si no aparece la lista de argumentos la función no tendrá parámetros de entrada.
- A continuación sigue el código o cuerpo de la función.
- La función termina con la palabra `end` o `endfunction`.

```
function [x y] = polar2cartesiano(radio, angulo)
    x= radio*cos(angulo);
    y= radio*sin(angulo);
endfunction
```

Para que sea posible la devolución de algún valor, dentro del cuerpo de la función deben existir variables de igual nombre que las que aparecen en el campo `variable_retornada`. No es necesario devolver ningún valor explícitamente.

Dentro del cuerpo de la función son accesibles las variables declaradas en la lista de argumentos, que tendrán el valor que se les asignó al hacer la llamada (la forma de asignar los valores a las variables es por la posición, es decir, el primer valor será asignado a la primera variable de la lista de argumentos). Si el último de los argumentos de la función es el símbolo `...`, significa que la función admite un número indeterminado de parámetros. Es responsabilidad del programador afrontar el tratamiento de los parámetros no declarados explícitamente.

Se puede terminar en cualquier momento la ejecución de la función usando la palabra `return`.

Además de lo anterior, dentro del cuerpo de la función está disponible la variable `nargin`. Su valor es asignado automáticamente al número de parámetros que se han pasado en la llamada. Esto puede permitirnos hacer un pequeño chequeo antes de continuar con la ejecución del resto del código y evitar fallos por falta de parámetros. Por ejemplo:

```
function [x y] = polar2cartesiano(radio, angulo)
    if(nargin!=2)
        fprintf(2,"Error, el número de parámetros debe ser 2\n");
        return
    endif
    x= radio*cos(angulo);
    y= radio*sin(angulo);
```

endfuncion

Excepto para pequeños "programillas de juguete", no es práctico tener que definir todas las funciones cada vez. En lugar de ello es posible guardarlos en ficheros de texto y así conservarlos para futuros usos.

Octave no requiere que se carguen las funciones antes de usarlas. Simplemente es necesario que se guarden en un fichero con extensión ".m", en un directorio de los indicados en la variable LOADPATH, y que el nombre del fichero coincida con el de la función.

Los scripts son simples ficheros de texto en el que se introducen las sentencias de la misma manera que se introducen en la línea de comandos. Su principal utilidad es definir un entorno de trabajo en cualquier momento, definiendo una serie de variables y/o funciones. Cada vez que queramos recuperar ese estado deberemos volver a invocar el script por su nombre (in la extensión). La única restricción es que la primera sentencia no debe ser la definición de una función. En caso contrario Octave pensará que se trata de un fichero de función y los resultados no serán los esperados.

Capítulo 3. Funciones de entrada/salida

3.1. Funciones básicas de escritura en disco.

Estas funciones son `save` y `load`. La primera guarda variables a un fichero, la segunda carga variables de un fichero al espacio de trabajo.

`help load`

`help save`

3.2. Gráficas

Octave normalmente está configurado para usar `gnuplot` para representar sus gráficos. El programa externo elegido puede configurarse con la variable `gnuplot_binary`.

Existen cuatro funciones para representar gráficas en Octave: `plot()` y `mesh()` para representaciones en 2 y 3 dimensiones respectivamente. Admiten como parámetros vectores y matrices para representar (cuidado con las dimensiones).

Las otras dos funciones son `gplot` y `gspot`. Son simples interfaces para las funciones `plot` y `splot` del programa `gnuplot`, y por lo tanto admiten la misma sintaxis.

Para obtener más información `help nombre_funcion`

Capítulo 4. Problemas de álgebra lineal

En esta sección se nombrarán algunas de las utilidades que posee Octave para resolver problemas de álgebra lineal y se plantearán algunos ejemplos.

Capítulo 5. Estadística

En esta sección se nombrarán algunas de las utilidades de Octave para realizar cálculos y representaciones estadísticas. Se plantearán algunos ejemplos.

Capítulo 6. Procesado de señal

Aquí se describen las capacidades de Octave para realizar procesado digital de la señal. Se describirán algunas de las funciones fundamentales, tales como `fft()` o transformada rápida de Fourier. Se resolverán algunos problemas típicos.

6.1. Procesado de audio

Se mostrará algunas de las facilidades que ofrece Octave para el procesado de audio: grabación, reproducción, codificación.

6.2. Procesado de imagen

Se describen algunas de las funciones de procesado de imagen y se muestra algún ejemplo de procesado.

Octave utiliza los programas `xv` o `xloadimage` para mostrar las imágenes.

Capítulo 7. Referencias

Octave es un programa muy extenso (y fácilmente extensible), de gran utilidad en el mundo de la ingeniería, las ciencias, la informática y muchos otros campos de la ciencia en que se necesite una herramienta suficientemente potente para realizar todo tipo de cálculo numérico, y a la vez sencilla que permita un fácil aprendizaje

Este documento ha tratado de ser una introducción a Octave, centrándose principalmente en las particularidades del lenguaje. Si se desea profundizar más en el aprendizaje de Octave, se recomiendan las siguientes referencias.

- La ayuda en línea: desde el prompt de Octave tecleando "help".
- La página oficial de Octave: <http://www.octave.org>

Esta página centraliza toda la información de octave. Es el principal punto de referencia.

- MatLinks/Chorus <http://sourceforge.net/projects/matlinks/>. Proyecto Open Source que pretende desarrollar, coordinar y organizar funciones y código de cálculo matemático. El objetivo final es desarrollar "ToolBoxes" similares a las de Matlab.

Octave forma parte de prácticamente todas las distribuciones de Linux, por lo que obtener el programa y la documentación es tan simple como instalar los paquetes correspondientes.

Apéndice A. GNU Free Documentation License

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

A.1. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

A.2. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a

Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

A.3. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

A.4. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that

carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

A.5. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

A.6. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

A.7. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

A.8. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

A.9. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

A.10. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

A.11. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

A.12. ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.