SCILAB
BREVÍSIMA INTRODUCCIÓN

Héctor Manuel Mora Escobar

1

INTRODUCCIÓN

Este documento es una pequeña introducción a Scilab. Está lejos de ser exhaustivo con respecto a los temas, es decir, muchos de los temas y posibilidades de Scilab no son tratados aquí. Además, tampoco es exhaustivo con respecto al contenido de cada tema, sólo están algunos aspectos de cada tema.

El autor estará muy agradecido por los comentarios, sugerencias y correcciones enviados a:

`hmmorae@unal.edu.co` o `hectormora@yahoo.com`

Scilab fue desarrollado en el INRIA, Institut National de Recherche en Informatique et Automatique, un excelente instituto francés de investigación. Posteriormente colaboró la escuela de ingenieros ENPC, Ecole Nationale de Ponts et Chaussées.

Actualmente hay un gran consorcio con empresas como Renault, Peugeot-Citroen, CEA Commissariat à l'Energie Atomique, CNES Centre National d'Etudes spatiales, Dassault Aviation, EDF Electricité de France, Thales...

Sus principales características son:

- software para cálculo científico
- interactivo
- programable

- **de libre uso**, con la condición de siempre hacer referencia a sus autores
- disponible para diferentes plataformas: Windows, Linux, Mac.

El sitio oficial de Scilab es

`www.scilab.org`

Allí se encuentra información general, manuales, FAQs (frequent asked questions), referencias sobre reportes, diferencias con Matlab, lista de errores, ... Allí se puede descargar la versión binaria o las fuentes para las diferentes plataformas.

Un libro bastante completo sobre el tema es:

Gomez C. ed.,
Engineering and Scientific Computing with Scilab,
Birkhauser, Boston, 1999.

Otros libros son:

- Allaire G y Kaber S.M., *Introduction à Scilab, Exercises pratiques d'algèbre linéaire*, Ellipses, Paris, 2002.
- Guerre-Delabrière S. y Postel M., *Méthodes d'approximation : Équations différentielles - Applications Scilab, niveau L3*, Ellipses, Paris, 2004.
- Yger A., *Théorie et analyse du signal : Cours et initiation pratique via MATLAB et SCILAB* (1 décembre 1999) Ellipses, Paris, 1999.
- Urroz G., *Numerical and Statistical Methods With Scilab for Science and Engineering*, vol. 1, 2, Booksurge, 2001.
- Gomez C., Bunks C. et al., *Integrated Scientific Computing with SciLab*, Birkhauser, Boston, 1998
- Chancelier J.P. et al., *Introduction à SCILAB (Collection IRIS)* Springer, Paris, 2002

La utilización es igual para las diferentes plataformas, pero obviamente hay algunas diferencias intrínsecas al usar una u otra plataforma. Este pequeño manual se refiere principalmente a la versión 6.1 para Linux y para Windows (la última, a la fecha de hoy: 4 de julio del 2020).

Deseo agradecer a todos los que tuvieron que ver con la creación y actualización de Scilab. Realmente es una herramienta de uso libre muy poderosa. Me ha sido muy útil para mi trabajo personal y para mis clases en la Universidad.

También deseo manifestar mi agradecimiento a todas las personas que de una u otra forma me han ayudado a corregir, modificar, actualizar y mejorar este documento, en particular, al profesor Manuel Mejía.

La sintaxis de Scilab para las órdenes básicas es, salvo contadas exepciones, igual a la de Matlab y Octave (este también de libre uso).

2

GENERALIDADES

Al activar Scilab aparece en la pantalla algo semejante a:

```
Startup execution:
  loading initial environment

-->
```

A partir de ese momento se puede escribir al frente de `-->` las órdenes de Scilab. Éstas deben ser acabadas oprimiendo la tecla `Enter`, denominada también `Intro` o simplemente ⟵┘.

2.1 Primeros pasos

La orden

`--> t = 3.5` ⟵┘

crea la variable `t` y le asigna el valor . Además Scilab muestra el resultado de la orden desplegando en pantalla lo siguiente:

```
 t  =

   3.5
```

```
-->
```

De ahora en adelante, en este manual no se indicará la tecla ↵ ni tampoco el "prompt" `-->` . Por lo tanto deben sobreentenderse. Generalmente tampoco se mostrará el resultado desplegado en la pantalla por Scilab ante una orden recibida.

Al dar la orden

```
T = 4.5;
```

se crea otra variable diferente con nombre `T` y se le asigna el valor `4.5`. Scilab diferencia las letras minúsculas de las mayúsculas. La presencia de punto y coma al final de la orden hace que Scilab no muestre el resultado en la pantalla. Sin embargo, la orden tuvo efecto.

Scilab no hace diferencia entre números enteros y números reales. Los números se pueden escribir usando la notación usual o la notación científica. Por ejemplo, son válidos los siguientes números.

```
3.5

-4.1234

3.14e-10

3.14E-10

0.0023e20

-12.345e+12
```

En las órdenes de Scilab los espacios en blanco antes y después del signo igual no son indispensables. Se podría simplemente escribir `t=3.5` obteniéndose el mismo resultado. Los espacios en blanco sirven simplemente para facilitar la lectura.

Los nombres en Scilab constan hasta de 24 caracteres. El primero debe ser una letra o `$` . Los otros pueden ser letras, números, `#`, `_`, `$`, `!` . Por ejemplo:

```
a12, pesoEsp, valor_ini
```

Cuando en la orden no hay ninguna asignación, sino simplemente una operación válida, Scilab crea o actualiza una variable llamada `ans` . Por

ejemplo,

```
t+T
```

produce el resultado

```
ans  =

   8.
```

Las asignaciones pueden incluir operaciones con variables ya definidas, por ejemplo

```
x = t+T
```

Si se da la orden

```
t = 2*t
```

se utiliza el antiguo valor de `t` para la operación, pero, después de la orden, `t` queda valiendo 7.

Si se quiere conocer el valor de una variable ya definida, basta con digitar el nombre de la variable y oprimir Enter.

Es posible volver a repetir fácilmente una orden dada anteriormente a Scilab. Utilizando las teclas correspondientes a las flechas hacia arriba y hacia abajo, se obtiene una orden anterior y se activa oprimiendo $\boxed{\longleftarrow\!\!\lrcorner}$. Por ejemplo al repetir varias veces la orden

```
x = (x+3/x)/2
```

se obtiene una aproximación muy buena de $\sqrt{3}$.

También es posible, por medio de las flechas (hacia arriba y hacia abajo), buscar una orden anterior para editarla y enseguida activarla.

En una misma línea de Scilab puede haber varias órdenes. Éstas deben estar separadas por coma o por punto y coma. Por ejemplo,

```
t1 = 2, t2 = 3; dt = t2-t1
```

2.2 Operaciones y funciones

Los símbolos

```
+   -   *   /
```

sirven para las 4 operaciones aritméticas. El signo `-` también sirve para indicar el inverso aditivo. Por ejemplo

```
u = -t
```

Para elevar a una potencia se utiliza el signo `^` o también `**` . Por ejemplo

```
u = 2^8,  v = 2**8
```

Scilab utiliza para agrupar los paréntesis redondos: `(  )`, como en la orden

```
x = (x+3/x)/2.
```

Puede haber parejas de paréntesis metidas (anidadas) dentro de otras.

En una expresión puede haber varios operadores. Las reglas de precedencia son semejantes a las de la escritura matemática usual. Los paréntesis tienen prioridad sobre todos los operadores. Entre los operadores vistos hay tres grupos de prioridad. De mayor a menor, estos son los grupos de prioridad:

```
^  **
*  /
+  -
```

Entre operadores de igual prioridad, se utiliza el orden de izquierda a derecha. Por ejemplo, `2*3+4^5-6/7` es equivalente a `((2*3)+(4^5))-(6/7)`.

Scilab tiene predefinidas muchas funciones matemáticas. A continuación está la lista de las funciones elementales más comunes.

`abs` : valor absoluto

`acos` : arcocoseno

`acosh` : arcocoseno hiperbólico

`asin` : arcoseno

`asinh` : arcoseno hiperbólico

`atan` : arcotangente

`atanh` : arcotangente hiperbólica

`ceil` : parte entera superior

`cos` : coseno

`cosh` : coseno hiperbólico

`cotg` : cotangente

`coth` : cotangente hiperbólica

`exp` : función exponencial: e^x

`fix` : redondeo hacia cero (igual a `int`)

`floor` : parte entera inferior

`int` : redondeo hacia cero (igual a `fix`)

`log` : logaritmo natural

`log10` : logaritmo decimal

`log2` : logaritmo en base dos

`max` : máximo

`min` : mínimo

`modulo` : residuo entero

`rand` : número aleatorio

`round` : redondeo

`sin` : seno

`sinh` : seno hiperbólico

`sqrt` : raiz cuadrada

`tan` : tangente

`tanh` : tangente hiperbólica

El significado de la mayoría de estas funciones es absolutamente claro. La siguiente tabla muestra varios ejemplos utilizando las funciones de parte entera y redondeo.

x	ceil(x)	floor(x)	int(x)	round(x)
2.0	2.	2.	2.	2.
1.8	2.	1.	1.	2.
1.5	2.	1.	1.	2.
1.2	2.	1.	1.	1.
- 3.1	- 3.	- 4.	- 3.	- 3.
- 3.5	- 3.	- 4.	- 3.	- 4.
- 3.8	- 3.	- 4.	- 3.	- 4.

Otra función matemática, ésta ya con dos parámetros de entrada, es `modulo`. Sus dos parámetros deben ser enteros. El resultado es el residuo de la división entera.

```
modulo(17,5)
```

da como resultado 2.

Para tener información más detallada sobre alguna función basta con digitar `help` y a continuación el nombre de la función o de la orden. Por ejemplo

```
help floor
```

Obviamente se requiere que la función `floor` exista. Si no se conoce el nombre de la función, pero se desea buscar sobre un tema, se debe utilizar `apropos`. Por ejemplo:

```
apropos polynomial
```

da información sobre las funciones que tienen que ver con polinomios. En cambio,

```
help polynomial
```

informa que no hay manual para `polynomial`.

Además de estas funciones elementales, Scilab tiene muchas más.

3

VECTORES Y MATRICES

En Scilab no hay vectores como tales, los vectores se deben asimilar a matrices de una sola fila o vectores fila (tamaño $1 \times n$) o a matrices de una sola columna o vectores columna (tamaño $n \times 1$).

3.1 Creación

La matriz

$$\begin{bmatrix} 11 & 12 & 13 & 14 & 15 \\ 21 & 22 & 23 & 24 & 25 \\ 31 & 32 & 33 & 34 & 35 \end{bmatrix}$$

se puede definir por medio de

```
a = [ 11 12 13 14 15; 21 22 23 24 25; 31 32 33 34 35]
```

o también por medio de

```
a = [ 11,12,13,14,15; 21,22,23,24,25; 31,32,33,34,35]
```

o por

```
a = [ 11 12 13 14 15
21 22 23 24 25
31 32 33 34 35]
```

Scilab mostrará el siguiente resultado en la pantalla:

```
 a  =

!   11.    12.    13.    14    15. !
!   21.    22.    23.    24    25. !
!   31.    32.    33.    34    35. !
```

La definición de la matriz se hace por filas. Los elementos de una misma fila se separan por medio de espacios en blanco o por medio de comas. Una fila se separa de la siguiente por medio de punto y coma o por medio de cambio de línea.

Scilab permite crear rápidamente algunos tipos especiales de matrices:

`ones(4,5)` es una matriz de unos de tamaño 4×5

`zeros(4,5)` es una matriz de ceros de tamaño 4×5

`rand(20,30)` es una matriz aleatoria de tamaño 20×30

`eye(4,4)` es la matriz identidad de orden 4

Algunas veces es útil, especialmente para gráficas de funciones (tema que se vera en un capítulo posterior), crear vectores con elementos igualmente espaciados, por ejemplo

```
x = [1 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6]
```

Esto también se puede hacer más fácilmente así:

```
x = 1:0.2:2.6
```

Los valores 1.0, 0.2 y 2.6 corresponden respectivamente al límite inferior, al incremento y al límite superior. Si no se especifica incremento se supone que es uno. Escribir `y = 2:9` es equivalente a `y = 2:1:9`. Cuando se desea una columna basta con utilizar el operador de trasposición, por ejemplo, `z = (1:0.2:2.6)'`

3.2 Notación y operaciones

`a(2,5)` denota el elemento o entrada de `a` en la fila 2 y en la columna 5.

`a(3,:)` denota la tercera fila de la matriz `a`.

`a(:,4)` denota la cuarta columna de la matriz `a`.

`a(1:2,2:5)` denota la submatriz de tamaño 2×4, formado por los elementos que están en las filas 1, 2 y en las columnas 2, 3, 4, 5, o sea, la matriz

$$\begin{bmatrix} 12 & 13 & 14 & 15 \\ 22 & 23 & 24 & 25 \end{bmatrix}.$$

La anterior definición de submatrices se puede generalizar utilizando vectores(fila o columna). Por medio de las siguientes órdenes

```
u = [ 1 2 1],  v =[2 4], aa = a(u,v)
```

se asigna a la variable `aa` la matriz

$$\begin{bmatrix} 12 & 14 \\ 22 & 24 \\ 12 & 14 \end{bmatrix}.$$

Si `a` es una matriz $m \times n$, entonces `a(:)` es un vector columna de tamaño $mn \times 1$, obtenido colocando primero la primera columna de `a`, enseguida la segunda columna, luego la tercera y así sucesivamente.

Si `x` es un vector columna, entonces `x(:)` es el mismo vector columna. Si `x` es un vector fila, entonces `x(:)` es lo mismo que `x'`. Dicho de otra forma, si `x` es un vector (fila o columna), entonces `x(:)` es un vector columna con las mismas entradas de `x`. Así, `x = x(:)` convierte el vector `x` en un vector columna.

Si `x` es un vector fila, se puede escribir `x(3)` en lugar de `x(1,3)`. Análogamente, si `y` es un vector columna, se puede escribir `y(2)` en lugar de `y(2,1)`.

Por medio de `*` se puede hacer el producto entre un número y una matriz. Los signos `+` y `-` permiten hacer sumas y restas entre matrices del mismo tamaño. Cuando el producto de matrices es posible (número de columnas de la primera matriz igual al número de filas de la segunda), éste se indica por medio de `*` . La transposición de una matriz se indica por medio de comilla, por ejemplo,

```
c = a'*a
```

crea la matriz `c`, producto de la traspuesta de `a` y de `a`. Por construcción `c` es una matriz cuadrada.

La mayoría de las funciones de Scilab utilizadas para números reales se pueden aplicar a matrices. En este caso se obtiene una matriz del mismo

tamaño, en la que se ha aplicado la función a cada uno de los elementos de la matriz. Por ejemplo, las dos órdenes siguientes son equivalentes.

```
D = sin([1 2; 3 4]), D = [sin(1) sin(2); sin(3) sin(4)]
```

Para matrices (y vectores) del mismo tamaño se puede hacer la multiplicación elemento a elemento utilizando `.*` . De manera análoga se puede hacer la división elemento a elemento. Por ejemplo:

```
P = rand(3,5),  Q = rand(3,5),  r = P.*Q
```

También es posible elevar a una potencia los elementos de una matriz, por ejemplo,

```
H = a.^3
```

Si `a` es una matriz rectangular

```
a.^(1/2)
```

es lo mismo que

```
sqrt(a)
```

es decir, una matriz del mismo tamaño, cuyas entradas son las raices cuadradas de las entradas de `a`.

En cambio, si `a` es una matriz cuadrada,

```
a1 = a^(1/2)
```

es una matriz raiz cuadrada de `a`, o sea, `a1*a1` es lo mismo que `a`.

Como se verá más adelante, para graficar se requieren dos vectores, uno con los valores de la variable x y otro (del mismo tamaño) con los valores de la variable y. Las dos órdenes siguientes permiten crear los dos vectores para un polinomio:

```
x = (-2:0.01:3)';
```

```
y = 3*x.^4 + x.^3 - 5*x.*x + 3.14;
```

Para matrices cuadradas, es posible calcular directamente una potencia. Por ejemplo,

```
G = rand(6,6)^3
```

Si los tamaños son compatibles, dos o más matrices se pueden "pegar" para obtener una matriz de mayor tamaño, por ejemplo,

```
AA = [a rand[3,2)]
B = [rand(2,5); a; eye(5,5)]
```

Como `a` fue definida de tamaño 3×5, entonces `AA` es de tamaño 3×7 y `B` es de tamaño 10×5.

La orden `y = []` permite crear la matriz `y` de tamaño 0×0.

Si `x` es un vector, la orden `x(2) = []` suprime la segunda componente y desplaza el resto. Por ejemplo,

```
x = [2 3 5 7 11]; x(2) = []
```

produce el resultado

```
 x  =

!   2.    5.    7.    11. !
```

De manera análoga, si `a` es un matriz, la orden `a(:,2) = []` suprime la segunda columna.

3.3 Funciones elementales

Hay numerosas funciones de Scilab para el manejo de matrices.

Algunas de las más usadas son:

- `rank(a)` calcula el rango de `a`.
- `det(c)` determinante de una matriz cuadrada `c`. Scilab hace los cálculos por métodos numéricos muy buenos, pero todas las operaciones de Scilab tienen un error muy pequeño que en la mayoría de los casos no se ve. Entonces es posible que el determinante de una matriz sea exactamente cero, pero Scilab muestre algo semejante a `1.285D-13`

  Consideremos una matriz $x \times 4$:

  ```
  a = [ 1 2 3 4; 5 6 7 8; 9 10 11 12;13 14 15 16]
  ii = 1:3, det( a(ii,ii))
  ```

La línea anterior permite calcular el determinante de una submatriz 3×3 formada por las primeras tres filas y las primeras tres columnas.

```
ii = [1 3], det( a(ii,ii))
```

La línea anterior permite calcular el determinante de una submatriz 2×2 formada por la primera y tercera filas y la primera y tercera columnas.

- `inv(c)` inversa de una matriz cuadrada e invertible `c`.
- `rref(a)` matriz escalonada reducida por filas equivalente a `a`.
- `expm(C)` produce la matriz e^C, donde `C` es una matriz cuadrada.

$$e^C = I + C + \frac{1}{2}C^2 + \frac{1}{6}C^3 + \frac{1}{24}C^4 + \cdots$$

- `diag(c)` produce un vector columna con los elementos diagonales de la matriz cuadrada `c`.
- `diag(x)` produce una matriz diagonal con los elementos del vector (fila o columna) `x`.
- `y = gsort(x)` ordena el vector `x` de manera decreciente.
- `y = gsort(x, 'g', 'i')` ordena el vector `x` de manera creciente. También puede ser `-gsort(-x)`
- `[y, k] = gsort(x)`: `y` es el vector ordenado de manera decreciente, `k` es un vector que contiene los índices del ordenamiento, o sea, `y = x(k)`.
- `b = gsort(a)` ordena la matriz `a` de manera decreciente, considerando cada matriz como un vector formado por la primera columna, la segunda columna, ..., la última columna.

$$\texttt{a} = \begin{bmatrix} 3.2 & 3.1 \\ 3.4 & 3.8 \end{bmatrix}, \qquad \texttt{b} = \begin{bmatrix} 3.8 & 3.2 \\ 3.4 & 3.1 \end{bmatrix}$$

- `b = gsort(a, 'r')` ordena la matriz `a` de manera decreciente por columnas. **Atención**, aunque `'r'` tiene un significado interno de filas, el resultado externo es un ordenamiento de las columnas.

$$\texttt{b} = \begin{bmatrix} 3.4 & 3.8 \\ 3.2 & 3.1 \end{bmatrix}$$

- `b = gsort(a, 'c')` ordena la matriz `a` de manera decreciente por filas. **Atención**, aunque `'c'` tiene un significado interno de columnas, el resultado externo es un ordenamiento de las filas.

$$\texttt{b} = \begin{bmatrix} 3.2 & 3.1 \\ 3.8 & 3.4 \end{bmatrix}$$

- `m = max(x)` calcula el máximo del vector (fila o columna) `x`.
- `[m, k] = max(x)`: `m` es el máximo del vector `x`, `k` indica la posición donde está el máximo.
- `m = max(a)` calcula el máximo de la matriz `a`.
- `[m, k] = max(a)`: `m` es el máximo de la matriz `a`, `k` es un vector 1×2 e indica la posición donde está el máximo.
- `m = max(a, 'r')` : `m` es un vector fila (row) que contiene los máximos de las columnas de `a`.
- `[m, k] = max(a, 'r')` : `m` es un vector fila que contiene los máximos de las columnas de `a`, `k` es un vector fila que contiene las posiciones de los máximos.
- `min` semejante a `max` pero para el mínimo.
- `m = mean(x)` calcula el promedio del vector (fila o columna) `x`.
- `m = mean(a)` calcula el promedio de la matriz `a`.
- `m = mean(a, 'r')` : `m` es un vector fila (row) que contiene los promedios las columnas de `a`.
- `m = mean(a, 'c')` : `m` es un vector columna que contiene los promedios las filas de `a`.
- `median` semejante a `mean` pero para la mediana.

- `st-deviation` semejante a `mean` pero para la desviación estándar.
- `sum` semejante a `mean` pero para la suma.
- `prod` semejante a `mean` pero para el producto.
- `norm(x)` calcula la norma euclidiana del vector `x` (fila o columna).
- `norm(x, p)` calcula la norma l_p del vector `x`:

$$\left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}.$$

- `norm(x, 'inf')` calcula la norma "del máximo" del vector `x`:

$$\max_{1\leq i\leq n} |x_i|.$$

- `norm(a) = norm(a, 2)` calcula, para la matriz `a`, la norma matricial generada por la norma euclidiana, o sea, el mayor valor singular de `a`.
- `norm(a, 1)` calcula, para la matriz `a`, la norma matricial generada por la norma l_1, o sea,

$$\max_{1\leq j\leq n} \sum_{i=1}^{m} |a_{ij}|.$$

- `norm(a, 'inf')` calcula, para la matriz `a`, la norma matricial generada por la norma l_∞, o sea,

$$\max_{1\leq i\leq m} \sum_{j=1}^{n} |a_{ij}|.$$

- `norm(a, 'fro')` calcula, para la matriz `a`, la norma de Frobenius, o sea,

$$\left(\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2\right)^{1/2}.$$

- La orden `fc = size(a)` proporciona un vector fila 1×2 con el número de filas y el número de columnas de `a`. La orden `size(a,1)` proporciona el número de filas de `a`. Análogamente, `size(a,2)` proporciona el número de columnas de `a`.

- Por medio de `tril` se puede obtener la parte triangular inferior (low) de una matriz cuadrada. Por ejemplo,

  ```
  a = [1 2 3; 4 5 6; 7 8 9]; L = tril(a)
  ```

  produce el resultado

  ```
   L  =

  !   1.    0.    0. !
  !   4.    5.    0. !
  !   7.    8.    9. !
  ```

- La función `triu` produce la parte triangular superior (upper).

- Si `a` es una matriz cuadrada, por medio de

  ```
  v = spec(a)
  ```

  se obtiene un vector columna con los valores propios (reales o complejos) de `a`.

  Con la orden

  ```
  [v, s] = spec(a)
  ```

  se obtiene la matriz `v` cuyas columnas son vectores propios y la matriz diagonal `s` cuyos elementos diagonales son los valores propios.

  En Matlab, una de las pocas diferencias la orden es `eig`

- Si A es una matriz simétrica y definida positiva, entonces se puede factorizar en la forma $A = U^{\mathrm{T}} U$, donde U es una matriz triangular superior de diagonal positiva. Esta es la llamada factorización de Cholesky. Esta matriz se puede obtener por

  ```
  U = chol(A)
  ```

- La factorización QR de una matriz A es la expresión

  $$A = QR$$

  donde Q es una matriz ortogonal y R es una matriz del tamaño de A, "*triangular*" (no necesariamente cuadrada). Por ejemplo,

  ```
  [q, r] = qr(a)
  ```

- La factorización LU de una matriz cuadrada invertible A es la expresión

$$PA = LU$$

donde P es una matriz de permutación, L es una matriz triangular inferior con unos en la diagonal y U es una matriz triangular superior. Por ejemplo,

```
[L, U, P] = lu(A)
```

- La descomposición SVD, descomposición en valores singulares, de una matriz A es la expresión

$$A = UDV^{\mathrm{T}}$$

donde U y V son matrices ortogonales y D es una matriz del tamaño de A, "*diagonal*" (no necesariamente cuadrada), cuyos elementos diagonales son los valores singulares de A. Por ejemplo,

```
[U, D, V] = svd(A)
```

- La inversa generalizada de Moore-Penrose o seudoinversa de una matriz se puede obtener por medio de

```
a1 = pinv(a)
```

3.4 Solución de sistemas de ecuaciones

En matemáticas (y todas las ciencias y técnicas relacionadas) uno de los problemas más frecuentes, o posiblemente el más frecuente, consiste en resolver un sistema de ecuaciones lineales $Ax = b$, donde se conocen la matriz A y el vector columna b.

Si A es una matriz cuadrada e invertible, el sistema tiene, teóricamente, una única solución y se puede resolver por una de las dos órdenes siguientes. La primera conlleva el cálculo de la inversa. La segunda usa un algoritmo eficiente de Scilab para hallar la solución.

A = [1 1 1; 2 3 4; 1 -1 1] b = [6 16 2]'

```
x1 = inv(A)*b
x2 = A\b
```

Teóricamente, el resultado debe ser el mismo. Desde el punto de vista de precisión numérica, los resultados son semejantes. Para matrices medianamente grandes hay diferencias en tiempo. La primera forma gasta, aproximadamente, tres veces más tiempo que la segunda.

Fuera del caso de solución única hay otros dos casos: el caso de sistema inconsistente (sin solución) y el caso de muchas soluciones (número infinito de soluciones).

Si el sistema es inconsistente, entonces se desea encontrar una seudosolución, la solución por mínimos cuadrados, o sea, se busca x que minimice $||Ax-b||_2^2$. En este caso la orden

```
x = A\b
```

encuentra una de estas"soluciones". Si las columnas de A son linealmente independientes, esta seudosolución es única. Para mayor información, use la ayuda:

```
help backslash
```