

Pràctica 0

Introducció a Scilab

Índex

1	Introducció	2
2	Introducció de dades en Scilab	2
3	Ajuda	5
4	Com guardar i recuperar sessions	5
5	Operacions amb escalars	6
6	Operacions amb matrius	7
6.1	Operacions bàsiques	7
6.2	Altres operacions amb matrius	8
6.3	Operacions element a element	8
6.4	Tipus especials de matrius	9
7	Manipulació interna de matrius	9
8	Polinomis	11
9	Estructures de decisió i bucle amb Scilab	12
9.1	El condicional if	12
9.2	El bucle for	13
9.3	El bucle while	14
10	Funcions amb Scilab	14
11	Fi de sessió	16

1 Introducció

El programa Scilab treballa fonamentalment amb matrius de coeficients reals, complexos o booleans. A aquesta pràctica es pretén descriure el funcionament bàsic del programa i tots els aspectes bàsics sobre Scilab que els alumnes han de conèixer per poder fer les pràctiques de Matemàtica Discreta i d'Àlgebra.

Scilab és un programa lliure. En particular, es pot utilitzar i distribuir lliurement i consultar-ne i modificar-ne el codi font en les condicions de la seua llicència. El programa es pot descarregar de <http://www.scilab.org>. Aquestes pràctiques, amb lleugeres modificacions, poden fer-se amb altres programes de sintaxi semblant, per exemple, sistemes lliures, com ara Octave, o sistemes comercials com Matlab.

Començarem indicant el funcionament bàsic del programa, indicant les diverses maneres d'introduir matrius en Scilab i com manipular-les. A continuació descriurem com fer amb Scilab les principals operacions matemàtiques entre matrius. Conclourem la pràctica mostrant algunes de les estructures de decisió i de bucles més habituals en Scilab.

El programa Scilab està disponible per a diverses plataformes, entre les quals podem destacar GNU/Linux, Microsoft Windows i MacOSX.

Al llarg de les diferents pràctiques anirem introduint més operadors, comandes i funcions bàsiques segons els anem necessitant i descriurem amb més detall el seu funcionament. Únicament pretenem donar ací els principals operadors, comandes i funcions bàsics per poder-los buscar després amb més facilitat.

Els exemples que es mostren estan desenvolupats amb la versió 4.1.2 de Scilab sota el sistema operatiu GNU/Linux amb un processador Intel® Core™2 Duo E8400 a 3 GHz. L'execució de Scilab en altres plataformes o sistemes operatius pot donar resultats lleugerament diferents, per la qual cosa caldrà para atenció.

A aquests butlletins farem servir els següents convenis tipogràfics:

- Les instruccions que s'introdueixen des del teclat apareixeran en un tipus de lletra mecanogràfica en negreta (per exemple, **entrada**).
- Els resultats o l'eixida de Scilab apareixeran en un tipus de lletra mecanogràfica de gruix mitjà (per exemple, *eixida*).
- Les expressions genèriques en la descripció de les comandes de Scilab apareixeran amb lletra mecanogràfica cursiva (per exemple, si apareix *nom=expressió*, entendrem que *nom* i *expressió* s'han de substituir per les cadenes adequades en Scilab).

2 Introducció de dades en Scilab

El programa Scilab es pot executar escrivint a una línia de comandes

scilab

o, en un entorn gràfic de finestres, fent doble clic (o clic senzill) sobre la icona corresponent o buscant-lo al menú adequat. Hi apareix una finestra amb el següent text:

scilab-4.1.2

Copyright (c) 1989-2007
Consortium Scilab (INRIA, ENPC)

Startup execution:
loading initial environment

-->


Si s'invoca el programa Scilab amb l'opció -nw, el text apareix en el terminal, sense finestra nova. Aquesta opció és adequada si no es necessiten capacitats gràfiques o només disposem d'un terminal de text.

La marca --> ens indica que podem introduir comandes.

Els *vectors* s'introdueixen escrivint ordenadament les seues components separades per una coma (,) o per un espai en blanc i tancant tota l'expressió entre claudàtors ([]). Els *escalars* s'introdueixen directament, sense necessitat de claudàtors, i, en el cas de nombres decimals, separant la part entera de la decimal amb un punt (.).

Per exemple, l'escalar $a = 2$ s'introdueix teclejant

a=2

i quan es prem la tecla  apareix

-->**a=2**

a =

2.

-->

El vector $\vec{v} = (1, -1, 3)$ s'introdueix amb

-->**v=[1 -1 3]**

v =

1. - 1. 3.

o amb


-->**v=[1,-1,3]**

v =

1. - 1. 3.

Nota: En algunes versions del programa apareixen signes d'exclamació al voltant de les matrius.

Una matriu de tipus $m \times n$ és una col·lecció de $m \times n$ nombres organitzats en m files i n columnes. Si A és una matriu, $a_{i,j}$ denota el terme que ocupa la fila i i columna j de la matriu i podem escriure que $A = (a_{i,j})$.

En Scilab, els elements d'una matriu s'introdueixen entre claudàtors [], escrivint ordenadament els elements de cada fila separats per comes o per espais en blanc i separant les files per punt i coma (;) o el caràcter  (retorn de carro). No cal indicar prèviament la grandària de les matrius.

Per exemple, la matriu

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

es pot introduir en Scilab com a


A=[1 2 3;4 5 6;7 8 9]

o com a

**A=[1 2 3
4 5 6
7 8 9]**

o fins i tot com a

**A=[1 2,3;4 5 6
7 8 9]**

En introduir alguna d'aquestes expressions i prémer  s'obté

A =

1.	2.	3.
4.	5.	6.
7.	8.	9.

Exemple 1. Introduïu la matriu A de l'exemple anterior. La pantalla de Scilab ha de quedar-vos aproximadament com es mostra a figura 1.


Figura 1: Introducció d'una matriu a Scilab

Les instruccions de Scilab tenen la forma

variable=expressió

o, simplement,

expressió

Els noms de les variables han de començar per una lletra, seguida de més lletres o dígit, fins a un màxim de 25 caràcters. Scilab no executa les sentències fins que no es prem . Si una sentència acaba amb punt i coma (;), el resultat no es mostra a la pantalla, encara que el resultat s'emmagatzema a la memòria. Scilab distingeix entre majúscules i minúscules. Naturalment, no es poden fer servir altres comandes de Scilab com a noms de variables.

És possible incloure diverses instruccions a una mateixa línia separant-les per una coma (,), per mostrar-ne els resultats, o per punt i coma (;), per no mostrar-los. De la mateixa manera, si una sentència és molt llarga i no cap a una línia, es pot separar en diverses línies posant al final de cada línia tres o més punts (...).

És possible recordar ordres anteriors amb les tecles de les fletxes.

Si una expressió és assignada a una variable, aquesta roman emmagatzemada com a variable de treball durant tota la sessió (o fins que se li done un altre valor o s'esborri amb la instrucció **clear**, que esborra totes les variables, o **clear** seguit del nom d'una o més variables per esborrar-les). D'aquesta manera, per recuperar aquesta expressió en qualsevol moment, n'hi haurà prou amb escriure el nom de la variable. Si utilitzem la segona forma de les instruccions, el resultat s'emmagatzema a la variable `ans`. Açò ens permet recuperar el resultat de la operació immediatament anterior si no l'hem assignat a una altra variable.

Les comandes **who** i **whos** ens indiquen quines són les variables de l'entorn de treball.

Els comentaris s'introdueixen precedits dels caràcters `//` (dues barres) fora d'una cadena. Tot el que hi haja darrere no s'executa.

3 Ajuda

Podem trobar ajuda sobre qualsevol de les comandes o funcions de Scilab escrivint **help** o, si es fa servir Scilab en un entorn gràfic, punxant amb el ratolí sobre el botó **Help**. També podem trobar ajuda sobre una comanda específica teclejant

help *comanda*

4 Com guardar i recuperar sessions

És possible guardar totes les variables de l'entorn de treball amb la comanda **save**. Per exemple,

```
-->save('fitxer.dat')
```

guarda totes les variables al fitxer `fitxer.dat`, mentre que

```
-->save('fitxer.dat',A,b,c)
```

guarda les variables A, b i c al fitxer. Per recuperar el contingut dels fitxers, es fa servir la comanda **load** amb la sintaxi

```
-->load('fitxer.dat','A','b','c')
```

Per guardar el contingut d'una sessió de treball, es fa servir la comanda **diary**:

```
-->diary('fitxer.txt')
```

comença la gravació de tota la sessió de treball en el fitxer de text `fitxer.txt`. Per concloure la gravació utilitzarem

```
-->diary(0)
```

Si el fitxer ja existeix prèviament s'esborrarà. Per això cal parar molta atenció en escriure el nom del fitxer. Una bona estratègia és incloure la data (i, si s'escau, l'hora) en el nom del fitxer. Per exemple,

```
-->diary('md221010.txt');
```

5 Operacions amb escalars

Scilab és capaç de treballar amb escalars reals i complexos. Per treballar amb nombres decimals, la part entera i la decimal se separen mitjançant un *punt* (.). Les operacions suma, resta i multiplicació es fan amb els operadors **+**, **-** i *****, respectivament. Les potències d'un escalar a a un escalar p les escriurem a^p . Les arrels quadrades es poden calcular també amb l'operador **sqrt**. Les arrels d'altres índexs es poden representar com potències d'exponents fraccionaris, com per exemple $2^{(1/3)}$ per a $\sqrt[3]{2}$. Per a la divisió hi ha dos operadors, **/** i ****. Per exemple, a/b vol dir a dividit entre b , mentre que $a \backslash b$ vol dir b dividit entre a .

La jerarquia dels operadors esmentats és l'habitual: de major (el darrer en execució) a menor (el primer en execució) és la següent:

+ - * / \ ^

Per variar l'ordre d'execució farem servir els parèntesis.

Exemple 2. Calculem amb Scilab el valor de l'expressió $2x^3y + \sqrt{y^5z}$, on $x = 1,5$, $y = 3,7$, $z = 9$.

Primer introduïm les variables

```
-->x=1.5; y=3.7; z=9;
```

Tot seguit, introduïm l'expressió:

```
-->2*x^3*y+sqrt(y^5*z)
ans =
```

103.97472

El resultat és 103,97472.

6 Operacions amb matrius

Seguidament mostrarem com s'executen amb Scilab les operacions matricials habituals i unes altres que ens seran útils per a algunes pràctiques. Com que els vectors no són més que matrius amb una única fila, les operacions que es descriuran seran també vàlides per a vectors.

6.1 Operacions bàsiques

Suma i resta de matrius: Per sumar dues matrius utilitzarem l'operador $+$ i per restar-les l'operador $-$. Les dues matrius han de tenir la mateixa grandària (és a dir, el mateix nombre de files i de columnes). Si es pretén sumar o restar matrius que no tenen la mateixa grandària, Scilab ens indicarà que hem comés un error.

Producte d'un escalar per una matriu: Es fa amb l'operador $*$.

Producte de matrius: Es fa també amb l'operador $*$. El nombre de columnes de la primera matriu ha de coincidir amb el nombre de files de la segona matriu. Si es pretén sumar dues matrius que no satisfacen aquesta condició, s'obté un missatge d'error.

Potència d'una matriu: Per calcular el producte d'una matriu A per ella mateixa n vegades, farem servir l'operador $^$. Per exemple, A^n . Si la matriu no és quadrada, Scilab ens donarà un missatge d'error.

Exemple 3. A continuació mostrem càlculs amb matrius.

```
-->X=[1 2;3 -1]
X =

    1.    2.
    3.   -1.

-->Y=[0 1]
Y =

    0.    1.

-->X*Y
!--error 10
inconsistent multiplication

-->X+Y
!--error 8
inconsistent addition
```

```
-->X-Y
      !--error 9
inconsistent subtraction
```

```
-->Y*X
ans =
```

```
3. - 1.
```

6.2 Altres operacions amb matrius

Inversa d'una matriu: Scilab calcula la inversa d'una matriu quadrada (si existeix) amb la funció `inv()`. Si la matriu introduïda no té inversa, apareixerà un missatge d'error. A les pràctiques d'Àlgebra Lineal estudiarem el funcionament detallat d'aquesta funció.

Transposada d'una matriu: La matriu transposada d'una matriu donada és la matriu obtinguda intercanviant files per columnes. Scilab la calcula amb l'operador apòstrof (`'`), per exemple, `A'`.

6.3 Operacions element a element

En algunes ocasions pot ser útil realitzar operacions amb matrius element a element que no corresponen a cap operació matemàtica entre matrius. En mostrarem tot seguit algunes d'aquestes operacions.

Suma d'un escalar a tots els elements d'una matriu: S'executa posant l'operador `+` entre la matriu i l'escalar. Per exemple, `A+1` és la matriu que s'obté quan sumem 1 a cada element de la matriu A.

Producte i quocient element a element: Per multiplicar element a element es fa servir l'operador `«.*`». Per exemple, `A.*B`. Cal que les matrius siguin de la mateixa grandària per poder fer aquesta operació. Anàlogament podem calcular quocients de matrius element a element amb l'operador `«.*/`».¹

Potència element a element: Per elevar a un nombre tots els elements d'una matriu es fa servir l'operador `«. ^ »`. La matriu pot ser de qualsevol grandària i l'exponent, qualsevol nombre.

¹Aquest darrer operador es pot fer servir, per exemple, per dividir un nombre entre tots els elements d'una matriu. En aquest cas, si el primer nombre és un enter, pot ser necessari separar-lo de l'operador amb un espai (per exemple, `1 ./A`) o posar-li el punt decimal (`1. ./A`) perquè `1./A` s'interpretaria com el quocient entre el nombre real 1. i la matriu A.

6.4 Tipus especials de matrius

A Scilab hi ha implementades unes quantes matrius especials. Per exemple, **zeros**(m, n) és la matriu nul·la de m files i n columnes, **ones**(m, n) és la matriu de m files i n columnes formada tota per uns i **eye**(m, n) és la matriu de grandària m files i n columnes que té uns a la diagonal i zeros a la resta de les posicions (quan $m=n$, és la matriu identitat d'ordre n , I_n). Anàlogament, si *mat* és una matriu, **zeros**(*mat*), **ones**(*mat*) i **eye**(*mat*) tornen les matrius de les mateixes dimensions que *mat* formades, respectivament, per zeros, uns, i uns a la diagonal i zeros a la resta de les posicions.

7 Manipulació interna de matrius

Suposem que tenim introduïda una matriu *A* a la nostra sessió de Scilab i en volem canviar l'element que ocupa la fila *i* i la columna *j* pel valor *expressió*. Aleshores farem servir l'ordre

$$A(i, j) = \text{expressió}$$

Aquesta acció modifica la matriu *A*. Per tant, si no volem perdre la matriu *A* original podem copiar-la a una altra matriu, fent per exemple

```
-->B=A
```

i efectuant després la modificació a la matriu **B**, per exemple, amb

```
-->B(2,3)=10
```

Si els índexs són més grans que la grandària de la matriu, Scilab afegeix files o columnes suficients per acomodar els nous elements. També convé tenir en compte que Scilab identifica les matrius 1×1 amb els escalars.

Scilab permet introduir matrius definides *per blocs*. Suposem que *A* i *B* són matrius amb el mateix nombre de files, podem construir la matriu

$$C = \begin{pmatrix} A & B \end{pmatrix}$$

obtinguda posant les columnes de *B* a la dreta de les de *A*. Aquesta matriu s'introdueix a Scilab escrivint

```
-->[A,B]
```

o

```
-->[A B]
```

Anàlogament, si *A* i *B* són dues matrius amb el mateix nombre de columnes, podem construir la matriu

$$D = \begin{pmatrix} A \\ B \end{pmatrix}$$

posant les files de *B* davall de les files de *A*. Aquesta matriu s'introduiria en la forma

```
-->[A;B]
```

o com a

```
-->[A  
--> B]
```

De la mateixa manera, podem afegir a una matriu A $m \times n$ una nova fila escrivint

```
-->[A; [a1 a2 ... an]]
```

i una nova columna escrivint

```
-->[A,[a1;a2; ...; am]]
```

Aquesta manera de treballar per blocs és vàlida per a qualsevol nombre de matrius sempre que els nombres de files i de columnes siguin compatibles.

Exemple 4. Considerem les matrius

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, \quad B = \begin{pmatrix} 7 & 8 & 2 \\ 4 & 1 & 6 \\ 0 & 0 & 1 \end{pmatrix}.$$

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

1.	2.	3.
4.	5.	6.
7.	8.	9.

```
-->B=[7 8 2  
--> 4 1 6  
--> 0 0 1]  
B =
```

7.	8.	2.
4.	1.	6.
0.	0.	1.

```
-->C=[A,B]  
C =
```

1.	2.	3.	7.	8.	2.
----	----	----	----	----	----

4.	5.	6.	4.	1.	6.
7.	8.	9.	0.	0.	1.

-->**D=[A;B]**

D =

1.	2.	3.
4.	5.	6.
7.	8.	9.
7.	8.	2.
4.	1.	6.
0.	0.	1.

Una vegada introduïda una matriu *matriu*, és possible treballar amb submatrius extrems d'ella, fent servir expressions com les següents:

- *matriu(i,j)* per a l'element que ocupa la posició fila *i* i columna *j* de *matriu*,
- *matriu(:,j)* per a la columna *j* de *matriu*,
- *matriu(i,:)* per a la fila *i* de *matriu*,
- *matriu(r:s,:)* per a la submatriu formada per les files entre la *r* i la *s* de *matriu*,
- *matriu(:,r:s)* per a la submatriu formada per les columnes entre la *r* i la *s* de *matriu*,
- *matriu([r s],:)* per a la submatriu formada per les files *r* i *s* de *matriu*, i
- *matriu(:, [r s])* per a la submatriu formada per les columnes *r* i *s* de *matriu*.

Observem que aquestes expressions també ens permeten fer modificacions en una matriu que ja ha sigut introduïda. Així, per exemple, si volem modificar la fila *i* de *matriu*, podem escriure *matriu(i,:)=nova fila* i el programa ens tornarà la matriu *matriu* modificada. Cal tenir en compte que les dimensions de la submatriu i la nova matriu han de ser compatibles. D'altra banda, podem assignar a tots els elements d'una submatriu un mateix element amb la mateixa sintaxi.

Com abans, cal prendre precaucions si no volem perdre la matriu inicial.

8 Polinomis

Per definir una variable polinòmica en la indeterminada **s**, podem definir la variable **s** en la forma

```
-->s=poly(0,"s")
```

```
s =
```

```
s
```

```
-->p=3+4*s+s^2
```

```
p =
```

```
          2  
3 + 4s + s
```

Les arrels d'un polinomi es calculen amb la funció **roots**. Per al polinomi **p** que hem introduït tindríem

```
-->roots(p)
```

```
ans =
```

```
- 1.
```

```
- 3.
```

Per a avaluar un polinomi en un valor concret utilitzarem la funció **horner**. Per exemple

```
-->horner(p,2)
```

```
ans =
```

```
15.
```

A les pràctiques d'Àlgebra Lineal treballarem amb polinomis.

9 Estructures de decisió i bucle amb Scilab

Scilab és un llenguatge de programació en el qual és possible utilitzar instruccions condicionals i executar diversos bucles. Per referència futura en descrivim algunes d'aquestes instruccions. Totes aquestes instruccions acaben amb la paraula reservada **end**.

9.1 El condicional **if**

La instrucció **if** s'utilitza per avaluar una expressió lògica i executar una sèrie d'instruccions si l'expressió és vertadera.

La sintaxi és:

```
if condició then instruccions  
elseif condició then instruccions
```

```
...
else instruccions
end
```

on les parts **elseif** i **else** són opcionals, i pot haver-hi més d'una clàusula **elseif**. La paraula **then** pot ser substituïda per un retorn de carro o una coma i *ha de trobar-se sempre a la mateixa línia que la corresponent comanda if o elseif*.

Si la condició de la línia amb **if** és vertadera, s'executen les instruccions següents. En un altre cas, s'executa la primera de les sèries d'instruccions **elseif** per a les quals la condició siga vertadera, i, si cap d'elles no és vertadera, s'executen les instruccions de la clàusula **else**.

A una pràctica futura estudiarem amb detall els operadors de comparació i els operadors lògics. En presentem un exemple senzill.

```
-->x=-4; if x<0 then r=-x,elseif x==0 then r=0,else r=x,end
r =
```

4.

```
-->x=5; if x<0 then r=-x,elseif x==0 then r=0,else r=x,end
r =
```

5.

Scilab també disposa de la comanda **select/case** que pot ser una alternativa útil quan una expressió només pot prendre un petit nombre de valors interessants. Per més informació, teclegeu **help select** a Scilab.

9.2 El bucle **for**

El bucle **for** s'utilitza per a executar unes instruccions per a totes les columnes d'una matriu. La seua sintaxi és

```
for variable=expressió do instrucció, ... instrucció,end
```

Ací *expressió* sol ser una matriu i les ordres s'executen per a totes les columnes de la matriu. Els formats habituals per a aquesta expressió són

```
començament:pas:fi
```

o

```
començament:fi
```

Exemple 5. Amb el següent exemple podem escriure els quadrats dels nombres senars entre 1 i 9:

```
-->for x=1:2:9 do [x,x^2],end
ans =

    1.    1.
ans =

    3.    9.
ans =

    5.   25.
ans =

    7.   49.
ans =

    9.   81.
```

Amb la següent instrucció calculem la suma dels quadrats dels 10 primers nombres naturals. L'ús del punt i coma en comptes de la coma evita que s'escriguen tots els valors intermedis.

```
-->suma=0;for x=1:10 do suma=suma+x^2;end;suma
suma =

    385.
```

9.3 El bucle **while**

Aquest bucle s'utilitza per a executar unes instruccions mentre una certa expressió booleana siga vertadera. La seua sintaxi és

```
while condició do instructions,...[,else instructions], end
```

La paraula **do** pot substituir-se per **then**, per una coma o per un salt de línia. La paraula **do** o **then** han de trobar-se a la mateixa línia que **while**. La clàusula **else** s'executa quan *condició* deixa de ser vertadera.

10 Funcions amb Scilab

Ja hem vist algunes funcions com **zeros** o **ones**, que permeten construir objectes de Scilab a partir dels seus paràmetres o arguments. La sintaxi d'una funció a Scilab és

funció(*arg1*, *arg2*...)

per a tornar un resultat, o

$[var1, var2 \dots] = funció(arg1, arg2 \dots)$

per a funcions que tornen diversos resultats, assignats a les variables *var1*, *var2*...

Per definir una funció a Scilab podem fer servir les paraules clau **function** i **endfunction**. La seua sintaxi és

```
function  argseixida=nomfunció(argsentrada)
           instruccions
endfunction
```

on *argsentrada* és una llista d'identificadors de variables que s'assignaran ordenadament als paràmetres i *argseixida* és un identificador o una llista d'identificadors de variables separats per comes i tancats entre claudàtors. La llista d'instruccions ha de contenir assignacions d'aquestes variables per calcular-ne el resultat.

Per exemple, la següent funció calcula el quadrat d'un nombre donat:

```
-->function y=quadrat(x)
-->y=x*x
-->endfunction
```

```
-->z=quadrat(3)
z  =
```

9.

Com veiem, l'argument d'eixida és **y** i el d'entrada és **x**. El valor de la funció és el valor de **y** quan acaba la definició de la funció després de substituir el paràmetre **x** per l'argument utilitzat.

La següent funció admet dos paràmetres, **x** i **y**, i torna dos resultats, que corresponen a la suma i a la diferència dels seus arguments:

```
-->function [suma, dif]=sumadif(x,y)
-->  suma=x+y
-->  dif=x-y
-->endfunction
```

```
-->[m,n]=sumadif(5,3)
n  =
```

2.

m =

8.

Observem que la variable **m** queda assignada a la suma dels dos nombres i la variable **n** queda assignada a la diferència entre els dos nombres.

Sol ser interessant emmagatzemar una o diverses funcions a un fitxer de text, per tal de fer-les servir a diverses sessions. Per executar totes les instruccions (no sols definicions de funcions) d'un fitxer de text *fitxer.sci*, podem fer servir la comanda **exec**:

```
-->exec('fitxer.sci')
```

La versió gràfica del programa Scilab inclou un editor de fitxers, al qual es pot accedir des del menú, que permet l'execució del codi que se'n seleccione.

11 Fi de sessió

Per acabar la sessió de Scilab cal introduir la instrucció **exit**. Si es fa servir un entorn de finestres, també se'n pot eixir tancant la finestra de Scilab.