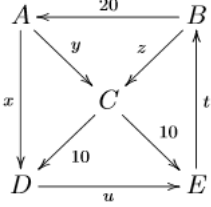


Comandos Scilab

a/b	a dividido por b
a\b	b dividido por a
inv(A), rref([A,eye(A)])	Inversa de la matriz A
“.” “/” “.^”	Hace la operación elemento por elemento
1.- zeros (m,n) 2.- zeros (A)	1.- Devuelve matriz nula de dimensión m x n 2.- Devuelve matriz nula = dimensiones que A
1.- ones (m,n) 2.- ones(A)	1.- Devuelve matriz formada por “1” de dimensión m x n 2.- Devuelve matriz formada por “1” = dimensiones que A
1.- eye(m,n) 2.- eye(A)	1.- Devuelve matriz formada por 1 en diagonal y 0 el resto de dimensiones m x n Si m=n 2.- Devuelve matriz formada por 1 en diagonal y 0 el resto = dimensiones que A Identidad
A(i,j)=expresión	Substituye el elemento i,j de la matriz por la expresión
C=[A,B] o C=[A B]	C = (A B) construir nueva matriz, columnas de B se ponen a la derecha de las de A
C=[A;B]	$C = \begin{pmatrix} A \\ B \end{pmatrix}$ construir nueva matriz, filas de B se ponen bajo de A
Tabajo con matrices: A(i,j) A(:,j) A(i,:) A(r:s,:) A(:,r:s) A([r s],:) A(:,[r s])	Para el elemento que ocupa la posición fila i y columna j de la matriz A Para la columna j de la matriz A Para la fila i de la matriz A Submatriz formada por las filas entre la r y la s de la matriz A Submatriz formada por las columnas entre la r y la s de la matriz A Submatriz formada por las filas r y s de la matriz A Submatriz formada por las columnas r y s de la matriz A
A(i,:)=nueva fila A(:,j)=nueva columna	Modificar la fila i de la matriz A Modificar la columna j de la matriz A
s=poly(0,"s")	Definir una variable polinómica en la indeterminada s
roots(p)	Raíces del polinomio
horner(p,n)	Sustituye la variable polinómica por “n” en el polinomio p
If condición then <i>instrucción</i> elseif cond. then <i>instrucción</i> else instrucciones end	Evaluar expresión lógica y ejecutar instrucciones, elseif y else son opcionales, puede haber más de un elseif Ejemplo: x=-4; if x<0 then r=-x,elseif x==0 then r=0,else r=x,end Sol: r=4
For variable=expresión do ... instrucción, ..., instrucción, end ejemplo: for x=1:2:9 do [x,x^2], end	La expresión suele ser una matriz y las ordenes se ejecutan para todas las columnas de la matriz, los formatos más habituales son: “comienzo:paso:fin” o “comienzo:fin” Escribe los cuadrados de los números impares entre 1 y 9
while condición do instrucciones, ... [else instrucciones], end	do puede sustituirse por then , la cláusula else se ejecuta cuando condición es falsa
función(arg1, arg2...)	Sintaxis de una función en scilab
Exec(‘fichero.sci’)	Ejecuta un fichero de texto de scilab
A([i,j],:)=A([j,i],:)	Intercambio de filas aplicado a las filas i y j
A(i,:)=p*A(i,:)	La fila i-ésima se multiplica por p
A(i,:)=A(i,:)+p*A(j,:)	La fila i se cambia por la suma de la fila i y p veces la fila j
Rref(A)	Proporciona la forma escalonada reducida de A

<p>Operador \</p> $A\vec{x} = \vec{b} \quad A \backslash b$	<p>Para resolver $A\vec{x} = \vec{b}$, A=matriz de coeficientes, b=vector términos independientes y escribimos A\b</p> <p>*Si es SCD, proporciona la solución única</p> <p>*Si es SCI, A\b proporciona una de las soluciones</p> <p>*Si es SI, calcula un vector aproximado por mínimos cuadrados de la solución, es decir, un vector \vec{x} tal que el valor de la norma $\ A\vec{x} - \vec{b}\$ es el mínimo posible entre todos los posibles vectores de \vec{x}</p>
<p>Solución general de un sistema utilizando \ y kernel</p> <p>(cantidad infinita de soluciones) → SCI</p>	<p>calcular el núcleo: $A\vec{x} = \vec{0}$ instrucción: kernel(A)</p> <p>ejemplo: $A=[1 \ 0 \ 2 \ 3; \ 7 \ 1 \ 1 \ 1; \ 8 \ 1 \ 3 \ 4; \ 9 \ 1 \ 5 \ 7]; b=[6; \ 10; \ 16; \ 22];$</p> <p>$x=A \backslash b \rightarrow$ sol: 1.2; 0; 0; 1.6 (solución particular) (\vec{u})</p> <p>clean(A*x-b) sol: 0; 0; 0; 0 (si da esto, SCI, si da algo que no sea 0, SI)</p> <p>kernel(A) sol: (-0.149;0.843;0.451;-0.250) (\vec{v}_1) (-0.041;0.514;-0.706;0.484) (\vec{v}_2) (si da [] SCD y la solución es $x=A \backslash b$)</p> <p>solución Sistema: $\vec{u} + \lambda_1[\vec{v}_1] + \lambda_2[\vec{v}_2]$</p>
<p>Redes de flujo:</p> <p>Red consta de puntos (nodos, vértices...) y de arcos que conectan todos o parte de los nodos)</p>	<p>*flujo total que entra a un nodo = flujo total que sale del nodo</p> <p>*flujo total que entra dentro de la red = flujo total que sale de la red</p> <p>Ejemplo:</p>  <p> $\left. \begin{array}{l} \text{Nodo A: } x + y = 20 \\ \text{Nodo B: } z + 20 = t \\ \text{Nodo C: } y + z = 20 \\ \text{Nodo D: } x + 10 = u \\ \text{Nodo E: } u + 10 = t \end{array} \right\}, \quad M = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 20 \\ 0 & 0 & 1 & -1 & 0 & -20 \\ 0 & 1 & 1 & 0 & 0 & 20 \\ 1 & 0 & 0 & 0 & -1 & -10 \\ 0 & 0 & 0 & -1 & 1 & -10 \end{bmatrix}$ </p> <p>y calculamos, con Scilab, la forma escalonada reducida de M:</p> <pre>-->M=[1 1 0 0 0 20;0 0 1 -1 0 -20;0 1 1 0 0 20; --> 1 0 0 0 -1 -10;0 0 0 -1 1 -10]; rref(M) ans =</pre> <pre>1. 0. 0. 0. -1. -10. 0. 1. 0. 0. 1. 30. 0. 0. 1. 0. -1. -10. 0. 0. 0. 1. -1. 10. 0. 0. 0. 0. 0. 0.</pre> <p>SCI → $x=-10+\lambda, y=30-\lambda, z=-10+\lambda, t=10+\lambda, u=\lambda$, como son líquidos (en este ejemplo) han de cumplir que $x,y,z,t,u \geq 0$, por tanto el intervalo será $[10,30]$</p>
<p>Método jacobi:</p> $A\vec{x} = \vec{b} \quad A = L + D + U$ <p>\underline{L} (triangular inferior)</p> <p>(tril(A)-D)</p> <p>\underline{D} (<<parte diagonal>>)</p> <p>(diag(A))</p> <p>\underline{U} (triangular superior)</p> <p>(triu(A)-D)</p>	<p>Ejemplo: $A = \begin{bmatrix} 10 & 3 & 1 \\ 2 & -10 & 3 \\ 1 & 3 & 10 \end{bmatrix} A = L + D + U \rightarrow L = \begin{bmatrix} 0 & 0 & 0 \\ 2 & 0 & 0 \\ 1 & 3 & 0 \end{bmatrix}, D = \begin{bmatrix} 10 & 0 & 0 \\ 0 & -10 & 0 \\ 0 & 0 & 10 \end{bmatrix}$</p> <p>$U = \begin{bmatrix} 0 & 3 & 1 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \end{bmatrix}$ el vector \vec{x} es solución del sistema $A\vec{x} = \vec{b} \Leftrightarrow (L+D+U)\vec{x} = \vec{b} \Leftrightarrow$</p> <p>$\Leftrightarrow D\vec{x} = \vec{b} - (L+U)\vec{x} \Leftrightarrow \{\vec{x} = D^{-1}[\vec{b} - (L+U)\vec{x}]\} (1)$, D es invertible (elementos NO nulo de su diagonal).</p> <p>El método jacobi es una técnica iterativa basa en la igualdad, siguientes pasos:</p> <p>Tomamos una aproximación inicial de la solución \vec{x}_0, la sustituimos en el primer miembro de (1) y calculamos otra aproximación \vec{x}_1, la sustituimos en el primer miembro de (1) y calculamos otra aproximación \vec{x}_2 y así sucesivamente, obtenemos una sucesión de vectores $\vec{x}_0, \vec{x}_1, \vec{x}_2 \dots$ tal que:</p> $\vec{x}_{k+1} = D^{-1}[\vec{b} - (L+U)\vec{x}_k], \quad k = 0, 1, 2, 3 \dots$ <p>Proposición 1: Si esta sucesión de vectores es convergente, el vector límite es una solución del sistema de ecuaciones lineales.</p> <p>Ejemplo:</p> <p>TEORÍA PRÁCTICA 2 PÁGINA 2</p>

$$\left. \begin{array}{l} 10x + 3y + z = 14 \\ 2x - 10y + 3z = -5 \\ x + 3y + 10z = 14 \end{array} \right\},$$

Con scilab:

```
-->A=[10 3 1; 2 -10 3; 1 3 10];
-->D=diag([diag(A)])    -->L=tril(A)-D    -->U=triu(A)-D
D =                      L =                      U =
  10.    0.    0.        0.    0.    0.        0.    3.    1.
  0.   -10.    0.        2.    0.    0.        0.    0.    3.
  0.    0.   10.        1.    3.    0.        0.    0.    0.

-->F=inv(D)              -->R=L+U              -->x0=[0; 0; 0];x1=F*(b-R*x0)
F =                      R =                      x1 =
  0.1    0.    0.        0.    3.    1.        1.4
  0.   -0.1    0.        2.    0.    3.        0.5
  0.    0.    0.1        1.    3.    0.        1.4

-->x2=F*(b-R*x1) -->x3=F*(b-R*x2) -->x4=F*(b-R*x3)
x2 =                      x3 =                      x4 =
  1.11                    0.929                    0.9906
  1.2                     1.055                    0.9645
  1.11                    0.929                    0.9906

-->x5=F*(b-R*x4) -->x6=F*(b-R*x5)
x5 =                      x6 =
  1.01159                 1.000251
  0.9953                  1.005795
  1.01159                 1.000251
.....
```

Vemos que converge a (1,1,1), para hacerlo con bucle:

```
-->for i=1:6
-->x=F*(b-R*x);
-->end;

-->x
x =
  1.000251
  1.005795
  1.000251
```

Para comprobar si necesitamos hacer más operaciones, hacemos el mismo bucle pero hasta 50:

	<pre> -->for i=1:50 -->x=F*(b-R*x); -->end; -->x x = 1. 1. 1. </pre> <p>Efectivamente, converge a (1,1,1) que es la solución.</p>
<p>Método Gauss-Seidel:</p> $A\vec{x} = \vec{b}$ $A = L + D + U$ \underline{L} (triangular inferior) (tril(A)-D) \underline{D} (<<parte diagonal>>) (diag(A)) \underline{U} (triangular superior) (triu(A)-D)	<p>Es una ligera modificación de jacobi (generalmente nº más pequeño de iteraciones)</p> <p>La igualdad ahora será la siguiente: $(L + D)\vec{x} = \vec{b} - U\vec{x}$ Comenzamos con un vector inicial cualquiera \vec{x}_0 y calculamos \vec{x}_1 de manera que: $(L + D)\vec{x}_1 = \vec{b} - U\vec{x}_0$, ahora calculamos \vec{x}_2 de manera que: $(L + D)\vec{x}_2 = \vec{b} - U\vec{x}_1$ y así sucesivamente.</p> <p>Ejemplo:</p> $\left. \begin{array}{l} 10x + 3y + z = 14 \\ 2x - 10y + 3z = -5 \\ x + 3y + 10z = 14 \end{array} \right\},$ <p>Scilab:</p> <p>Primero hemos de ejecutar el fichero "SustitucionProgresiva.sci"</p> <pre> -->M=L+D; x0=[0; 0; 0]; -->x1=SustitucionProgresiva(M,b-U*x0) -->x2=SustitucionProgresiva(M,b-U*x1) x1 = x2 = 1.4 1.0634 0.78 1.02048 1.026 0.987516 -->x3=SustitucionProgresiva(M,b-U*x2) -->x4=SustitucionProgresiva(M,b-U*x3) x3 = x4 = 0.9951044 1.0012266 0.9952757 1.0008174 1.0019069 0.9996321 -->x5=SustitucionProgresiva(M,b-U*x4) -->x6=SustitucionProgresiva(M,b-U*x5) x5 = x6 = 0.9997916 1.000039 0.9998480 1.0000277 1.0000665 0.9999878 </pre> <p>Con la función for podemos efectuar más iteraciones, si queremos:</p> <pre> -->x0; -->for(i=1:50) x=SustitucionProgresiva(M,b-U*x); end; </pre> <p style="text-align: right;"> <pre> -->x x = 1. 1. 1. </pre> </p>
Criterio de convergencia	<p>Es posible que al aplicar jacobi o Gauss-Seidel, la sucesión sea divergente, pero si la matriz de coeficientes es de tipo especial, podemos garantizar la convergencia de ambos métodos.</p> <p>Def. 1: Una matriz cuadrada es <u>estrictamente diagonalmente dominante</u> si en todas las filas el valor absoluto del elemento de la diagonal es más grande que la suma de los valores del resto de los elementos de esta fila. (también se puede hacer por columnas)</p>

	<p>Si la matriz no es diagonal dominante (ni por filas o columnas) a veces es posible manipular el sistema (multiplicarlo por matrices elementales) para obtener un sistema equivalente cuya matriz si sea estrictamente diagonal dominante.</p> <p>Teorema: si una matriz es diagonalmente dominante, los métodos de jacobi y gauss-seidel son convergentes.</p>
Matrices estocásticas:	Una matriz es estocástica si NO hay ningún elemento negativo y si la suma de los elementos de las columnas da 1.
Cadena de Markov:	<p>Para estudiar la convergencia escribiremos las siguientes instrucciones:</p> <p>Ejemplo: $P = \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1 & 0 \\ 1/2 & 0 & 1 \end{bmatrix}$ y $\vec{x}_0 = (\frac{1}{3}, \frac{2}{3}, 0)$</p> <pre>-->P=[1/2 0 0; 0 1 0; 1/2 0 1];x0=[1/3; 2/3; 0];P^10*x0 ans = 0.0003255 0.6666667 0.3330078 -->P^30*x0 ans = 3.104D-10 0.6666667 0.3333333 -->P^50*x0 ans = 2.961D-16 0.6666667 0.3333333 -->clean(ans) ans = 0. 0.6666667 0.3333333</pre> <p>Vemos que el proceso es convergente al vector de probabilidad (0, 2/3, 1/3), Puede haber más de 1 vector estacionario, si $\vec{x}_0 = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, el vector es (0, 1/3, 2/3)</p>
Vector Estacionario cuando hay más de 1:	<p>kernel(A-eye(A)) ejemplo:</p> <pre>-->kernel(A-eye(A)) ans = 0. 0. 0. 0. 0. 1. 1. 0.</pre> <p>Si scilab nos devuelve esto, el conjunto de vectores estacionarios de A es: $\{\lambda(0,0,0,1)+\mu(0,0,1,0)=(0,0,\mu,\lambda) : \lambda \neq 0 \text{ ó } \mu \neq 0\}$</p>
Vector estacionario de probabilidad:	<p>Dividiendo todos los vectores de este conjunto entre la suma de sus componentes obtendremos los vectores de probabilidad estacionarios.</p> <p>Ejemplo: vector estacionario = $(0, \lambda_1, \lambda_2)$, los vectores de probabilidad serán: $\frac{1}{\lambda_1 + \lambda_2} (0, \lambda_1, \lambda_2)$</p> <p>Ejemplo: vector estacionario = $\lambda(0.5449493, 0.7784989, 0.3113996)$, los vectores de probabilidad serán:</p> $\frac{1}{\lambda(0.5449493, 0.7784989, 0.3113996)} \lambda \begin{bmatrix} 0.5449493 \\ 0.7784989 \\ 0.3113996 \end{bmatrix} = \begin{bmatrix} 0.3333333 \\ 0.4761905 \\ 0.1904762 \end{bmatrix}$
Descomposición LU	<p>[L,U]=lu(A): nos devuelve la descomposición LU sin permutar (L no será triangular inferior)</p> <p>[L,U,P]=lu(A): nos devuelve la descomposición LU de la matriz A permutada, L será triangular inferior. (L es la triangular inferior, U la triangular superior y P no es nada)</p> <p>Resolución de sistemas mediante LU:</p> <p>Ejemplo:</p>

	<pre> -->C=[0 -2 1;3 0 2;0 2 4];[L,U]=lu(C) U = 3. 0. 2. 0. 1. 0. 0. -2. 1. 1. 0. 0. 0. 0. 5. 0. -1. 1. -->y=L\[1;2;3] -->x=U\y y = 2. 1. 4. x = 0.1333333 - 0.1 0.8 </pre> <p>La solución será $x=(0.1333333, -0.1, 0.8)=(2/15, -1/10, 4/5)$</p>
Calculo de determinantes	<p>Scilab lo hace mediante LU, teniendo en cuenta que $A=LU$, $\det(A)=\det(L)\det(U)$</p> <p>*$\det(L)=\pm 1$ (+ cuando número de operaciones elementales de intercambio sea par y – cuando sea impar)</p> <p>*$\det(U)=$producto de la diagonal</p>
Calculo de inversas	<p>Scilab hace lo siguiente: 1.- obtiene LU de A, 2.- calcula $\det(U)$, 3.- si $\det(U)=0$, no tiene inversa, 4.- si $\det(U)\neq 0$, calcula U^{-1}, finalmente, $A^{-1}=U^{-1}L^{-1}$</p>
Ortogonalidad	<p>Comprobamos si 2 vectores son ortogonales con su producto escalar: $uv=u'v$, si el resultado da 0, son ortogonales</p>
Norma de un vector	<p>$\text{norm}(w)$</p>
Vector unitario asociado a w	<p>$t=w/\text{norm}(w)$, el resultado de t es el vector unitario asociado a w, lo podemos comprobar calculando la norma de t (esta debe de dar 1)</p>
Base subespacio fila	<p>$NF=\text{kernel}(A)$</p>
Base subespacio columna	<p>$NC=\text{kernel}(A')$</p>
Proyección ortogonal (RECTA)	<p>Ejemplo: recta $W=\langle(1,-2,5)\rangle$ vector $x=(0,1,1)$ sobre W: Scilab: $u=[1;-2;5]$ $x=[0;1;1]$ $q=u/\text{norm}(u)$ $(q'x)*q$ (el resultado de esto último es la proyección ortogonal)</p>
Proyección ortogonal (Subespacio vectorial)	<p>Ejemplo: $W=\text{subespacio vectorial de } R^3 \text{ con base } S=\{(1,2,3),(-3,5,1)\}$ y vector $x=(2,3,4)$ Scilab: $u1=[1;2;3]; u2=[-3;-5;1];$ $MS=[u1 \ u2]$ matriz de proyección Pw: $x=[2; 3; 4];$ $PW=MS*\text{inv}(MS'*MS)*MS'$ $PW*x$ (el resultado de esto último es la proyección ortogonal)</p>
Solución por mínimos cuadrados	<p>$x=A\b b$, proporciona una solución por mínimos cuadrados en caso de que A no sea una matriz cuadrada</p>

Ajuste de rectas (mínimos cuadrados)

$y = \beta_0 + \beta_1 x$ En forma matricial: (matriz diseño(X)*vector parámetro($\vec{\beta}$))=vector observación(\vec{y})

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \dots & \dots \\ 1 & x_n \end{bmatrix} * \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}$$

Vector residual= $\vec{\epsilon} = \vec{y} - X\vec{\beta}$

Ejemplo: Puntos: (0,1), (1,1), (2,2) y (3,2) recta = $y = \beta_0 + \beta_1 x$

Scilab:

X=[1 0;1 1;1 2;1 3], y=[1;1;2;2]

X1=X'*X, y1=X'*y

b=X1\y1 (solución)

norm(y-X*b) (error)

Ajuste de curvas (mínimos cuadrados)

$y = \beta_0 f_0(x) + \beta_1 f_1(x) + \dots + \beta_k f_k(x)$ (f0,f1... son funciones conocidas y β_0, β_1, \dots Son los parámetros que debemos determinar) en forma matricial:

$$\begin{bmatrix} f_0(x_1) & f_1(x_1) & \dots & f_k(x_1) \\ f_0(x_2) & f_1(x_2) & \dots & f_k(x_2) \\ \dots & \dots & \dots & \dots \\ f_0(x_n) & f_1(x_n) & \dots & f_k(x_n) \end{bmatrix} * \begin{bmatrix} \beta_0 \\ \beta_1 \\ \dots \\ \beta_k \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}$$

Ejemplo:

1	2	3	4	5
1.8	2.7	3.4	3.8	3.9

Determinar vector parámetro, vector residual y la curva de mínimos cuadrados asociada a la función:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

1.- resolver por mínimos cuadrados el sistema de ecuaciones lineales $\vec{y} = X\vec{\beta}$:

$$\begin{bmatrix} 1,8 \\ 2,7 \\ 3,4 \\ 3,8 \\ 3,9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

2.- Calculamos:

$$X^t X = \begin{bmatrix} 5 & 15 & 55 \\ 15 & 55 & 225 \\ 55 & 225 & 979 \end{bmatrix}, \quad X^t \vec{y} = \begin{bmatrix} 15,6 \\ 52,1 \\ 201,5 \end{bmatrix}$$

3.- Resolviendo el sistema $X^t X \vec{\beta} = X^t \vec{y}$ obtendremos el vector que minimiza el error.

$$\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} 0,58 \\ 1,34 \\ -0,136 \end{bmatrix}.$$

La parábola será: $y = 0.58 + 1.34x - 0.136x^2$

Si calculamos $\vec{\epsilon} = \vec{y} - X\vec{\beta}$, obtendremos el vector residual:

$$\begin{bmatrix} 0,114 \\ -0,026 \\ 0,008 \\ 0,014 \\ 0,008 \end{bmatrix}$$

Con **Scilab**:

Introducimos las matrices X , \vec{y} :

--> **X=[1 1 1;1 2 4;1 3 9;1 4 16;1 5 25]**, **y=[1.8;2.7;3.4;3.8;3.9];**

Y con la función \ obtenemos la solución por mínimos cuadrados:

-->**b=X\y**

b =

0.58

1.3442857

- 0.1357143

Luego la parábola que mejor se aproxima a los datos tiene los parámetros de b y su error residual será

-->**ER=norm(y-X*b)**

ER =

0.0338062