Práctica voluntaria (1er. parcial) 0.3 Puntos — Técnicas Numéricas
Francisco R. Villatoro y Carmen M. García — 12 de Noviembre de 2002

TEMAS 2 y 3: RESOLUCIÓN DE SISTEMAS LINEALES

Edite un fichero llamado `cheb.m` que contenga la siguiente función (tenga cuidado en no equivocarse).

```
function [D,x] = cheb(N)
% [D,x] = cheb(N)
% N número entero, D matriz de derivadas, x malla de Chebyshev

if N==0, D=0; x=1; return, end
x = cos(pi*(0:N)/N)';
c = [2; ones(N-1,1); 2].*(-1).^(0:N)';
X = repmat(x,1,N+1);
DX= X-X';
D = (c*(1./c)')./(DX+(eye(N+1)));
D = D-diag(sum(D'));
```

Un método numérico de Galerkin para resolver el problema de contorno de la ecuación no homogénea siguiente,

$$\frac{\partial^2 u(x)}{\partial x^2} = e^{4\,x}, \quad -1 < x < 1, \qquad u(-1) = 0, \quad u(1) = 1,$$

conduce al siguiente programa (que será estudiado en el tema 12 de la asignatura), que deberá escribir en el fichero `solvebvp.m`.

```
function [u,x] = solvebvp (N)
% [u,x] = solvebvp (N)
% N el número de nodos de la malla
% x el vector de abcisas, y u la solución (vector de ordenadas)

[D,x]=cheb(N); D2=D^2; D2=D2(2:N,2:N);

f=exp(4*x(2:N));

u=D2\f; %%% RESOLUCION DEL SISTEMA LINEAL DE MATLAB

u=[0;u;0] + (x+1)/2;
```

Para dibujar esta solución y comprobar que las dos funciones anteriores funcionan puedes utilizar el siguiente código.

```
N=10;
[u,x]=solvebvp(N);
clf
subplot(2,2,1); plot(x,u,x,u,'.','markersize',16);
xlabel('x'); ylabel('u'); title('u(x) Numerico');
xx=-1:0.01:1;
uu=polyval(polyfit(x,u,N),xx);
exact = (exp(4*xx)-sinh(4)*xx-cosh(4))/16+(xx+1)/2;
subplot(2,2,2); plot ( xx, uu-exact);
xlabel('x'); ylabel('u-uexact');
title('          Error cometido con este método','fontsize',8);
```

Problema práctico 1. El procedimiento anterior (`solvebvp.m`) requiere la solución de un sistema lineal (`u=D2\f`). Matlab utiliza el método de factorización LU con pivotaje parcial para realizar dicha operación. Para calcular el tiempo que un comando requiere se utiliza la función `t=cputime; operacion; tiempo=cput`

(a) Escriba una gráfica que presente el tiempo necesario en ejecutar dicho comando (`solvebvp.m`) en función del tamaño `N` de la matriz (use `N=5:5:100`).

(b) Puede aproximar dicha función por un polinomio de grado `M` si utiliza la función `polyfit(N,T,M)`, donde `N` es el vector de órdenes de las matrices,`T` es el vector de tiempos de cpu, y finalmente, `M` es el grado del polinomio que aproxima los datos. Pruebe con polinomios de orden $M = 1 : 5$. ¿Cuál es el que conduce a la mejor aproximación? ¿Cómo mide el error de dicha aproximación?

(c) Concluya indicando cuál es el orden computacional de la operación de resolución de sistemas lineales interna de Matlab. En teoría hemos indicado que puede ser $O(2\,n^3/3)$. ¿Lo es realmente? ¿Por qué?

Problema práctico 2. Implemente el algoritmo de eliminación de Gauss con pivotaje completo mediante una función de Matlab llamada `GEPC` con cabecera de la siguiente forma:

```
function x=GEPC(A,b)
% x=GEPC(A,b)
```

```
% A es un matriz de NxN
% b es un vector Nx1
% x es el vector Nx1 tal que x=A\b
% x = NaN si A es una matriz singular
```

(a) Sustituya en la función `solvebvp.m`) la operación `u=D2\f` por `u=GEPC(D2,f)`. ¿Cómo puede comparar el error entre estos dos procedimientos? Para `N=5:5:100` compare el error entre estos dos procedimientos. ¿Cuál es aparentemente más preciso?

(b) Represente gráficamente el tiempo necesario en ejecutar `solvebvp` en función del tamaño `N` de la matriz (use `N=5:5:100`). Aproxime dicha función por un polinomio de grado $M = 1 : 5$. ¿Cuál es el que conduce a la mejor aproximación?

Problema práctico 3. Escriba un algoritmo para determinar la inversa de una matriz que utilice el comando `x=A\b` de Matlab con la siguiente cabecera:

```
function Ainv=Inversa(A)
% Ainv=GEPC(A)
% A es un matriz de NxN
% Ainv es la matriz inversa de A
% Ainv = NaN si A es singular
```

(a) Sustituya en la función `solvebvp.m`) la operación `u=D2\f` por `u=Inversa(D2)*f`. Para `N=5:5:100` compare el error entre estos dos procedimientos. ¿Cuál es aparentemente más preciso?

(b) Represente gráficamente el tiempo necesario en ejecutar `solvebvp` en función del tamaño `N` de la matriz (use `N=5:5:100`) para la nueva implementación. Aproxime dicha función por un polinomio de grado $M = 1 : 5$. ¿Cuál es el que conduce a la mejor aproximación?

Problema práctico 4. Para calcular la inversa de un matriz se puede utilizar el procedimiento de Jubete y Castillo, basado en ortogonalización, y cuyo código es:

```
function B=JubeteCastilloInversa(A);
% function B=JubeteCastilloInversa(A);
%
```

```
% Calcula la inversa de la matriz A en la matriz B
% mediante el algoritmo de Jubete-Castillo

[p1,p2]=size(A); if (p1==p2), n=p1; end
B=eye(n); C=eye(n); det=1;
for k=1:n,
   C(k,:)=A(k,:)*B(:,:);
   B(:,k)=B(:,k)*inv(C(k,k));
   B(:,rowind)=B(:,rowind)-B(:,k)*C(k,rowind);
end
```

(a) Sustituya en la función `solvebvp.m` la operación `u=D2\f` por `u=JubeteCastilloInver` Para `N=5:5:100` compare el error entre estos todos los procedimientos que ha utilizado hasta el momento. ¿Cuál es aparentemente más preciso?

(b) Represente gráficamente el tiempo necesario en ejecutar `solvebvp` en función del tamaño `N` de la matriz (use `N=5:5:100`) para la nueva implementación. Aproxime dicha función por un polinomio de grado $M = 1 : 5$. ¿Cuál es el que conduce a la mejor aproximación?

Problema práctico 5. Considere el algoritmo iterativo SOR basado en el algoritmo de Jacobi. Escriba una función que calcule el radio espectral de su matriz de iteración en función de w:

```
function rho=rhoSORJacobi(A,w);
% function rho=rhoSORJacobi(A,w)
%
% Calcula el radio espectral de la matriz de convergencia del método
% SOR basado en Jacobi con parámetro w para la matriz A (NxN)
```

Escriba otra función que aplique el algoritmo iterativo SOR basado en Jacobi para resolver un sistema lineal:

```
function x=SORJacobi(A,b,w);
% function x=SORJacobi(A,b,w)
%
% Calcula la solución de Ax=b utilizando SOR-Jacobi con parametro w
```

¿Cómo ha decidido cortar la iteración del método iterativo?

(a) Para la matriz `D2` de la función `solvebvp.m` con $N = 5$ dibuje el radio espectral de la matriz de convergencia del método SORJacobi en función de

w (por supuesto, dibuje sólo el intervalo $[0, 2]$). Determine el mínimo de dicha función. ¿Converge este método iterativo para esta matriz? En su caso, ¿Para qué valor se ha producido el valor mínimo? Este valor es el w óptimo.

(b) Para $N = 5 : 5 : 100$, dibuje el w óptimo para el algoritmo de SOR-Jacobi en función de N. ¿Converge el método para algún N? ¿Le parece razonable el resultado obtenido?

(c) Para los valores de `N=5:5:100` para los que SOR-Jacobi converge, si los hubiere, sustituya en la función `solvebvp.m` la operación `u=D2\f` por `u=SORJacobi(A,b,w)`, utilizando el w óptimo. Compare el error entre estos todos los procedimientos que ha utilizado hasta el momento. ¿Cuál es aparentemente más preciso?

(d) Represente gráficamente el tiempo necesario en ejecutar `solvebvp` en función del tamaño `N` de la matriz (use `N=5:5:100`) para la nueva implementación. Aproxime dicha función por un polinomio de grado $M = 1 : 5$. ¿Cuál es el que conduce a la mejor aproximación?

Problema práctico 6. Repita los apartados (a)-(d) del problema práctico 5 para el método SOR basado en Gauss-Seidel (`function x=SORJacobi(A,b,w)`).

Problema práctico 7. Repita los apartados (a) y (b) del problema práctico 4 para los métodos numéricos iterativos de solución de sistemas lineales `gmres`, `bicg`, `cgs` y `bicgstab`. Estudie la ayuda para aprender a utilizarlos. ¿Convergen todos estos métodos? ¿Cuál es el más eficiente?

Problema práctico 8. Compare todos los métodos que ha desarrollado entre sí. ¿Cuál es el más preciso de todos? ¿Cuál es el más rápido de todos? ¿Son más rápidos los métodos directos o los iterativos?