PRACTICAS-MATLAB.pdf

PRACTICAS MATLAB

3º Métodos Numéricos

Doble Grado en Economía y Matemáticas y Estadística

Facultad de Ciencias Económicas y Empresariales
Universidad Complutense de Madrid

KEEP CALM AND ESTUDIA UN POQUITO

WUOLAH + #QuédateEnCasa

GAUSS

```
A = input('Introducir matriz ');
n = size(A,1);
w = zeros(1,n);
u = zeros(1,n);
punt = 1:n;
mas = 1;

for j = 1:n-1
    [p,q] = max(abs(A(punt(j:n),j))); %p=valor pivote, q=lugar del
pivote
    if p == 0
        error('Matriz no válida')
    end

    punt([j, q+j-1]) = punt([q+j-1, j]); %permutacion de filas lo
guardo en puntero, intercambio de lugar elemtnos de puntero
    A(punt(j+1:n),j) = A(punt(j+1:n),j) / A(punt(j),j); %en la columna
j guardo multiplicadores

    for k=punt(j+1:n)
        A(k,j+1:n)=A(k,j+1:n)-A(k,j)*A(punt(j),j+1:n);  %al resto de
elementos de cafa fila les resto el primero
        %(que calcule arriba, pues es el multipicador)
        %multiplicado por el elemento que corresponda de la fila j
    end
end

if A(punt(n), n) == 0
    error('Matriz no válida')
    %me quedaria todo 0s a ultima linea, asi que no puedo aplicar
remonte,
    %infinitas soluciones
end
while mas == 1
    b = input('Introducir el vector ');
    w(1) = b(punt(1));

    for k=2:n
        w(k) = b(punt(k))-w(1:k-1)* A(punt(k), 1:k-1)';
        %formula del libro para remonte en gauss
        %w(i)=b(punt(i)) -suma de j=1 a i-1 de A(punt(i),j)*w(j)

        %creo que transforma vector b en otro w que tiene en cuenta
        %las permutaciones de filas que hemos hecho, y operaciones
entre
        %ellas
       %en particular coje el elemtno que corresponderia a ese lugar
por la
       %permutacion y luego resta: : suma de cada elemento de w ya
calculado
       %multiplicado por el elemtno que esta en la matriz en la fila
de
       %ahora y columna que va corresponiendo al elemento
    end
    u(n) = w(n) / A(punt(n), n);
    for i = n-1:-1:1
        u(i) = (w(i)-A(punt(i),i+1:n) * u(i+1:n)')/A(punt(i), i);
```

```
        %u(i)=(w(i)- suma de j=i+1 a n de A(punt(i),j)*u(j), todo
partido
        %de A(punt(i),i))
    end
    disp('Solución del sistema')
    disp('u = ')
    disp(u)
    mas = input('Introducir otro sistema con la misma matriz(Si=1,
No=0): ');
end
```

LU

```
A = input('Introducir matriz ');
n = size(A,1);
l = zeros(1,n);
u = zeros(1,n);
mas = 1;

for i=1:n %formula es desde i =1 hasta n
    % para cada i, misma formula se aplica en U desde j=i,...n y en L
desde
    % j= i+1...n
    A(i, i) = A(i,i) - A(i, 1:i-1)*A(1:i-1, i);
    %U(i,i)= A(i,i)-suma de k=1 a i-1 de L(i,k)*U(k,j)
    if A(i, i) == 0
            error('La matriz introducida no admite factorización LU')
    end
    for j=i+1:n
            A(i, j) = A(i, j) - A(i, 1:i-1)*A(1:i-1, j);
            %U(i,j)= A(i,j)-suma de k=1 a i-1 de L(i,k)*U(k,j)
            A(j, i) = (A(j, i) - A(j, 1:i-1) * A(1:i-1, i)) / A(i, i);
            %L(j,i) = A(j,i)- suma de k=1 a i-1 de L(j,k)*U(k,i)
partido de
            %U(i,i)
    end
end

while mas == 1; %L*w=b U*u=w aqui w es l
    b = input('Introducir el vector ');
    %remonte para triangular inferior, no pongo partido por L(i,i)
porque
    %son unos
    l(1) = b(1);
    for k = 2:n
       l(k) = b(k) - A(k, 1:k-1)*l(1:k-1)';
    end
    %remonte para triangular superior
    u(n) = l(n) / A(n, n);
    for k = n-1:-1:1
       u(k) = (l(k) - (A(k, k+1:n)*u(k+1:n)'))/A(k,k);
    end

    disp('Solución del sistema')
    disp('u = ')
```

```
    disp(u)
    mas = input('Introducir otro sistema con la misma matriz(Si=1,
No=0): ');
end
```

CHOLESKY

```
n = size(A,1);
l = zeros(1, n);
u = zeros(1, n);
mas = 1;
for i = 1:n
    raiz = A(i,i) - A(i,1:i-1)*A(i,1:i-1)'; %A(i,i)-suma de k=1 a i-1
de B(i,k)^2
    if raiz <=0
        error ('La matriz introducida no admite factorización de
Cholesky')
    end
    A(i, i) = sqrt(raiz);
    %B(i,i)= sqrt(raiz)
    %ambas tienen la misma diagonal asi que no hay ningun problema
    for j = i+1:n
        A(j, i) =(A(i, j) - A(i,1:i-1)*A(j, 1:i-1)')/A(i,i);
        %B(j,i) = A(i,j) - suma de k= 1 a i-1 de B(i,k)*B(j,k), todo
        %partido de B(i,i)
    end
end
while mas == 1 %B*w=b Bt*u=w
    b = input('Introducir el vector ');
    %remonte para triangular inferior
    l(1) = b(1)/A(1,1);
    for k = 2:n
       l(k) = (b(k) - A(k, 1:k-1)*l(1:k-1)')/A(k,k);
    end
    %remonte para triangular superior
    u(n) = l(n) / A(n, n);
    for k = n-1:-1:1
       u(k) = (l(k) - (u(k+1:n)*A(k+1:n,k)))/A(k,k);
    end

    disp('La solución del sistema es ')
    disp('u = ')
    disp(u)
    mas = input('Introducir otro sistema con la misma matriz(Si=1,
No=0): ');
end
```

TRIDIAGONAL

```
unction x = tridiagonal2(a,b,c,d)
%( b1 c1 0 0 ...
% (a2 b2 c2 0 ...
%.............000 a(n-1) b(n-1) c(n-1) )
.............00000000000 a(n)   b(n)   )
n = length(b)
x = zeros(1,n);
m = zeros(1, n);
```

```
g = zeros(1,n);

m(1) = b(1); %esto es tal cual
g(1) = d(1)/m(1);

for k = 2:n
    m(k) = b(k) - c(k-1)/m(k-1)*a(k-1);
    %m(k) = b(k) - c(k-1)/m(k-1)*a(k), tomo k-1 por que no pongo el
primer elemento de a como 0, y en realidad empieza en a(2);
    g(k) = (d(k)-g(k-1)*a(k-1))/m(k); %igual que anterior, todo tal
cual menos que en la formula es a(k) pero aqui k-1
end
%a partir de aqui la formula es tal cual
x(n)= g(n);
for k = n-1:-1:1
    x(k) = g(k) - c(k)*x(k+1)/m(k);
end
```

JACOBI

```
A = input('Introducir matriz ');
b = input('Introducir el vector ');
n = size(A,1);
eps = input('Introducir precisión del test de parada ');
k = input('Introducir número máximo de iteraciones ');
u = zeros(n,1);
d = zeros(1,n);
r = b'-A*u;
j = 1;
e = norm(b)*eps;
if ~all(diag(A))%ver si ALGUN elemento de la diagonal de A es cero)
    error('No se puede aplicar el método de relajación')
end
while j<=k && norm(r)>= e
    r = b' - A*u; % rk(i)= b(i) - suma de j=1 a n de A(i,j)*uk(j)
    d = r./diag(A); %dk(i)=rk(i)/A(i,i)
    u = u + d; %uk+1(i) = uk(i)+ dk(i)

    j = j+1;
end

if j > k
    disp('El cálculo se detiene porque se alcanzó el número máximo de
iteraciones y en ese momento la solución valía')
else
    disp('El cálculo se detiene porque se ha acotado correctamente la
solución y se han hecho ')
    disp(j-1)
    disp('iteraciones y la solución es')
end
disp(u)
```

RELAJACION

```
A = input('Introducir matriz ');
```

```
b = input('Introducir el vector ');
n = size(A,1);
eps = input('Introducir precisión del test de parada ');
k = input('Introducir número máximo de iteraciones ');
u = zeros(n,1);
w = input('Introducir parámetro de relajación ');
d = zeros(1,n);
r = b'-A*u;
j = 1;
e = norm(b)*eps;
if ~all(diag(A))
    error('No se puede aplicar el método de relajación')
end
while j<=k && norm(r)>= e %norma de rk menor que norma de b por
epsilon
    for i = 1:n
        r(i) = b(i)-A(i,:)*u;
        %rk(i)= b(i) - de j=1 a i-1 de A(i,j)*uk+1(j) - de j=i a n de
        %A(i,j)*uk(j)
        d(i) = w*r(i)/A(i, i); %dk(i)= w * rk(i) /A(i,i)
        u(i) = u(i)+d(i); %uk+1(i) = uk(i) +dk(i)
    end
    j = j+1;
end
if j > k
    disp('El cálculo se detiene porque se alcanzó el número máximo de
iteraciones y en ese momento la solución valía')
else
    disp('El cálculo se detiene porque se ha acotado correctamente la
solución y se han hecho ')
    disp(j-1)
    disp('iteraciones y la solución es')
end
disp(u)
```

INTERPOLACION

```
x = input('Introducir los puntos ');
n = length(x) ;
version = input('Introducir tabla(1) o funcion(0) ');

if version
    y = input('Introducir valores de la función ');
else
    f = input('Introduce la función a interpolar con sintaxis
@(x)función: ');
    y = f(x);
end

d = y; %en d ire guardando las diferencias divididas
%no hago matriz porque como par acalcualr el siguiente me hacne falta
solo
%dos anteriores, al calcularlo me puedo cargar los anteriores
pol = d(1); %f(x0) primer termino de pol interpolacion
prod1 = 1; %prod1 va a ser producto de (x-x0)*(x-x1)*(x-x2)*...
for i = 2: n
    for j = 1 : n-i+1 %en cada paso de las diferencias dividads tengo
que dar un  paso menos
        %el +1 es porque i empieza en 2
```

```
        d(j) = (d(j) - d(j+1)) / (x(j) - x(j+i-1)); %en cada cambio de
i amplio 1 la distancia de los puntos
        %el -1 es porque i empieza en 2

        %EN CADA PASO ME QUEDA GUARDADA EN d(1) la diferencia que me
        %interesa
    end
    prod1 = [prod1,0] - x(i-1) * [0,prod1];
    %primer termino es multiplicar por x, y segundo termino es
multiplicar
    %por x(i-1) y pongo un 0 a la izquierda de prod 1 para poder
restarlo
    %con lo otro
    %ademas multiplico por x(i-1) porque siempre va un paso por detras
esta
    %multiplicacion de la de las diferencias divididas, y porque
empiezo en
    %i= 2
    pol = [0,pol] + prod1 * d(1);
    %polinomio le añado un 0 a la izquierda solo para poder sumarlo
con lo
    %otro, no estoy haciendo ninguna operacion
    %y ademas le sumo el productorio por la diferencia dividida de ese
paso

end

disp('El polinomio de interpolación de Newton es ')
disp(pol)
hold on
plot(x,y,'*')%dibujo lo que vale la funcion de verdad o los valores
que tenia
inter = min(x):(max(x)-min(x))/100:max(x);
%doy valores entre el minimo y el maximo de los puntos
hold on
z = polyval(pol,inter);
%veo lo qu evale el polinomio en cada punto
plot(inter,z)%dibujo

disp('¿Quieres añadir otro punto de interpolación?')
r = input('Si quieres, pulsa 1 y si no, pulsa 0: ');
while r

    nuevo = input('Introduce el punto de interpolación que quieres
añadir: ');
    x = [x, nuevo]; %añado el nuevo punto
    n = length(x);

    if version %tabla
        a = input('Introduce el valor de la función en dicho punto de
interpolación: ');
        d(n) = a;
        y = [y,a];

    else
        d(n) = f(nuevo);
        y = [y,f(nuevo)];
    end

    for i = n-1:-1:1 %tabla de diferencias divididas
```

```
        d(i) = (d(i+1) - d(i)) / (x(n) - x(i));
        %(x(n) - x(i)) en cada paso se hace un punto mas grande
        %empieza x(n)-x(n-1) y acaba en x(n)-x(1)
        %he ido guardando la diagonal de la tabla en d
    end

    prod1 = [prod1 0] - x(n-1)*[0 prod1];

    pol = [0 pol] + prod1 * d(1);

    disp('Los coeficientes del polinomio de interpolacion de Newton
son:  ')
    disp(pol)

    hold on
    plot(x,y,'*')
    inter = min(x):(max(x)-min(x))/100:max(x);
    hold on
    z = polyval(pol,inter);
    plot(inter,z)

    disp('¿Quieres añadir otro punto de interpolación?')
    r = input('Si quieres, pulsa 1 y si no pulsa 0 ') ;

end
```

SPLINE

```
p = input("Introducir los puntos de una partición: ");
n = length(p) - 1; %cuento que como trabajo con n+1 puntos, tomo n
elegir = input("Elegir tabla o función (Tabla = 1, Función = 0): ");

if elegir == 0
    funct = input("Función para interpolar con @(x) con analisis
vectorial: ");
    y = zeros(n+1,1);
    y = funct(p);
    fplot(funct,[p(1) p(n+1)] ) %en azul sale la funcion de verdad
    hold on

else
    y = input("Tabla de interpolación: ");
end

a = zeros (n,1); %en tridiagonal a y c tienen un elemnto menos
b = 2* ones (n + 1,1); %vector columna con n+1 doses
c = zeros (n,1);
d = zeros (n + 1,1) ; %este tiene n+1 elementos, actua como el vector

%h(j+1)=x(j+1)-x(j)

for i = 2:n %formulas van de 1 a n-1, y el primer c que aparece seria
el c(0),
    %y el ultimo a que aparece seria el a(n)
```

```
    %por tanto esta bien que empiece en MI c2 y a1 y acabe en cn y an-
1
    %nota:notacion de numeros de arriba es la del libro, y la segunda
es la
    %mia. seria que c(1)=0 y a(n)=0
    c(i) = (p(i+1) - p(i)) / (p(i+1) - p(i-1)); %c(j)=
h(j+1)/[h(j)+h(j+1)]=[x(j+1)-x(j)]/[x(j)-x(j-1)+x(j+1)-x(j)]
    a(i- 1) = 1 - c(i); %a(j)=1-c(j) pero como el primer a es 0, a va
un paso por detras
    d(i) = (6 / p(i+1) - p(i-1)) * (((y(i+1)- y(i)) / (p(i+1) -p(i)))
- ((y(i) - y(i-1)) / (p(i) - p(i-1))));
    %d(j)= 6/[h(j)+h(j+1)]  * [ [y(j+1)-y(j)]/h(j+1) - [y(j)-y(j-
1)]/h(j) ]
end

M = tridiagonal2(a,b,c,d);
%M es un vector,momentos
%tridiagonal*M=d
A = zeros (n,4); %n filas y 4 columnas
%cada fila de A será la spline para un intervalo
%S=alfaj + betaj*(x-x(j) + fij*(x-x(j)^2 + deltaj*(x-x(j))^3
for i = 1:n
    %lo guardo asi para luego funcione bien polyval
    A(i,4) = y(i); %alfaj
    %alfaj = y(j)
    A(i,3) = ((y(i+1) - y(i)) / (p(i+1) - p(i))) - ((2* M(i) +
M(i+1))/ 6) * (p(i+1) - p(i));%betaj
    %betaj = [y(j+1)-y(j)]/h(j+1) - [[2M(j) + M(j+1)]/6 ]*h(j+1)
    A(i,2) = M(i) / 2; %fij
    %fij =M(j)/2
    A(i,1) = (M(i+1) -M(i)) / (6 * (p(i+1)-p(i)));%deltaj
    %deltaj = [M(j+1)-M(j)]/(6*h(j+1))
end

disp ("La matriz de coeficientes de la spline cubica")
disp (A)

for i = 1:n
    %tomo valores entre dos puntos, pues cada funcion se aplica a un
    %intevralo
    x = (p(i):(p(i)+ p(i+1))/100:p(i+1));
    g = polyval(A(i,1:4),x-p(i));
    %porque la spline en cada intervalo se multplica por los alfas
beta etc
    %por x-extremo izquierdo del intevalo

    %polyval evalua el primero qu es el de la primera columna con
exponente
    %3 en x-p(i), luego en 2, luego 1, luego 0
    hold on
    %(x,g,"K") %negro

end
plot(p, y ,"Rx")%cruces rojas, en cada putno de la particion
```