Actividad grupal: Problemas de contorno unidimensionales, métodos de diferencias finitas

**Objetivos**

Con esta actividad vas a conseguir poner en práctica los conceptos relacionados con los problemas de frontera unidimensionales estudiados en la asignatura. Concretamente, se aplicará el método de diferencias finitas a un problema de frontera cuyas condiciones no son de tipo Dirichlet.

**Descripción**

Consideremos el siguiente problema de frontera:

.

Plantea el sistema lineal que resulta al discretizar el problema anterior mediante diferencias finitas de segundo orden con 100 subintervalos.

Si tenemos en cuenta las diferencias divididas centrales para y para, el sistema lineal que resulta de discretizar el problema que queremos resolver mediante diferencias finitas es:

con *k=0, …, 100*

Explica con detalle el papel de las condiciones de contorno en este sistema. Indica la estructura de la matriz de coeficientes del sistema lineal resultante.

Queremos resolver un sistema de 101 ecuaciones con 101 incógnitas, basta con darnos cuenta que los puntos conflictivos son la frontera, que es donde tenemos las condiciones de contorno. Así, para todos aquellos *k* que no estén en la frontera tenemos que resolver (1).

Para ***k*=0**, tenemos que resolver:

Como no tenemos información de , vamos a utilizar las condiciones de contornoen en diferencias finitascentrales para sacar este valor:

Así, introduciendo en eliminamos la dependencia de , y obtenemos:

Para ***k*=100**, tenemos que resolver:

Como no tenemos información de , vamos a utilizar las condiciones de contorno en en diferencias finitas centrales para sacar este valor:

Así, introduciendo en eliminamos la dependencia de , y obtenemos:

Juntando así y yatenemos todo el problema que queremos resolver. Si lo pasamos a notación matricial, el problema que queremos resolver nos queda:

Aproxima la solución del problema de contorno tomando 100 subintervalos. Proporciona la solución numérica en los diez últimos nodos utilizando el algoritmo de Crout y representa gráficamente la solución completa.

La implementación de las matrices que hemos visto, adaptándolo a Matlab, es la siguiente:  
function[x,y] = DiFiLineal\_Entr2(a,b,alfa,beta,N)

h=(b-a)/(N-1);

x=a:h:b;

X=x(:);

dp=(h^2-2)\*ones(1,length(x)); % diagonal principal

dp(1)=-h^2-2\*h-2; dp(end)=-1/3\*h^2+4/3\*h-2;

ds=(1-h)\*ones(1,length(x)-1); % diagonal superior

ds(1)=2;

di=(1+h)\*ones(1,length(x)-1); % diagonal inferior

di(end)=2;

d=-2\*h^2.\*cos(X); % terminos independientes

d(1)=d(1)-2\*h\*(1+h)\*alfa;

d(end)=d(end)+4/3\*h\*(1-h)\*beta;

y=Crout(dp,ds,di,d); %Resolvemos el sistema de ecuaciones por Crout

x=x(:);

end

Una vez lo tenemos implementado, lo corremos mediante el comando  
[x,y] = DiFiLineal\_Entr2(0,pi/2,-1,1,101)  
y obtenemos los valores de la tabla 1 donde se tiene la solución aproximada en los diez últimos nodos mediante el algoritmo de Crout.

|  |  |
| --- | --- |
|  |  |
| 1.429425 | 0.990242 |
| 1.445133 | 0.992336 |
| 1.460841 | 0.994185 |
| 1.476549 | 0.995788 |
| 1.492257 | 0.997146 |
| 1.507940 | 0.998258 |
| 1.523672 | 0.999123 |
| 1.539380 | 0.999742 |
| 1.555088 | 1.000115 |
| 1.570796 | 1.000241 |

Tabla 1: solución aproximada en los diez últimos nodos mediante el algoritmo de Crout.

Además, si nos damos cuenta que la solución exacta que cumple el problema que nos plantean es , podemos graficar la solución aproximada enfrentada a la exacta, obteniendo la Figura 1.

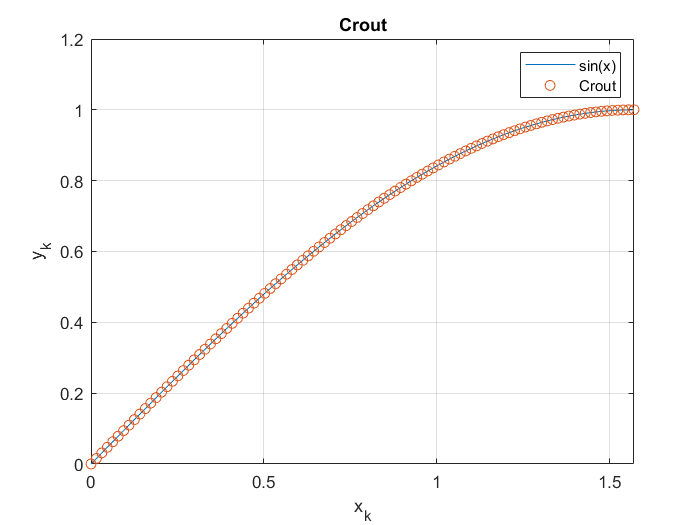


Figura 1: gráfica de la representación de los valores obtenido al aproximar mediante el algoritmo de Crout enfrentado a la solución exacta.

Repite el apartado anterior utilizando el método de Gauss (\ en Matlab) para resolver el sistema. Compara el tiempo de ejecución entre esta técnica y la del apartado (c).

Simplemente bastaría reutilizar el programa ejecutado en el apartado anterior, cambiando la condición donde se llama a Crout por:

A=diag(dp,0)+diag(ds,1)+diag(di,-1); %Creamos la matriz con las 3 diagonales

y=A\d; %Resolvemos el sistema de ecuaciones por Gauss

A partir de esta corrección y corriendo el programa modificado, obtenemos los valores de la tabla 2, donde mostramos la solución aproximada en los diez últimos nodos mediante el algoritmo de Gauss, y la diferencia entre la solución de Gauss y de Crout en valor absoluto.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1.429425 | 0.990242 | 2.194109 |
| 1.445133 | 0.992336 | 2.220446 |
| 1.460841 | 0.994185 | 2.247091 |
| 1.476549 | 0.995788 | 2.273737 |
| 1.492257 | 0.997146 | 2.300382 |
| 1.507940 | 0.998258 | 2.325917 |
| 1.523672 | 0.999123 | 2.351452 |
| 1.539380 | 0.999742 | 2.376987 |
| 1.555088 | 1.000115 | 2.402523 |
| 1.570796 | 1.000241 | 2.426948 |

Tabla 2: solución aproximada en los diez últimos nodos mediante el algoritmo de Gauss, y la diferencia entre la solución de Gauss y de Crout en valor absoluto.

A partir de esto vemos que las diferencias entre las soluciones mediante ambos métodos son del orden de , por lo que ambas aproximaciones son muy buenas.Además, si graficamos la solución de ambos métodos, obtenemos la Figura 2.

Para saber con qué tipo de matriz estamos trabajando, podemos ver cómo de bien condicionada está la matriz A que usamos para resolver el problema, mediante:  
cond(A) %Condición de la matriz A que hemos creado

De esto, obtenemos el valor , viendo que se aleja muchísimo de la unidad, donde diríamos que está bien condicionada. Por ello, pequeños cambios en los valores frontera pueden provocar un cambio brusco en la solución, pudiendo diferir mucho de la solución exacta. Además, al haber un gran número de operaciones por Gauss, el error de redondeo aumenta notablemente, pudiendo provocar estas diferencias con la solución exacta. Sin embargo, hemos comprobado que en este caso no se da este disparo de errores, debido a que en este caso el método \ llega a la solución por un método de descomposición, simplificando los cálculos y tiempos internos de Matlab, tal y como se ve en la Figura 3.

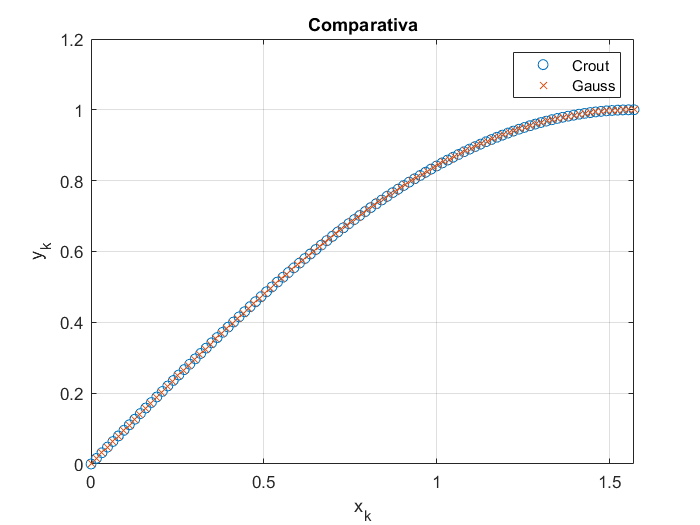


Figura 2: gráfica de la representación de los valores obtenido al aproximar mediante el algoritmo de Crout y los obtenidos al aproximar mediante Gauss.

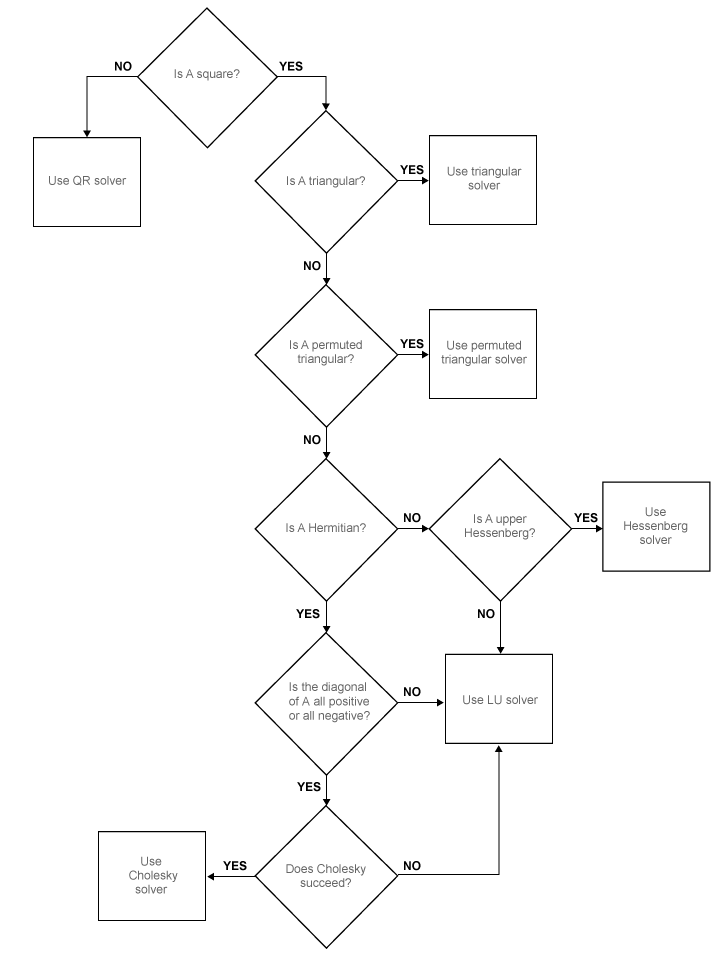


Figura 3: esquema del preanálisis de \ para resolver el sistema [1].

Si ahora contamos el tiempo promediado entre 50 ejecuciones en 5 máquinas distintas para eliminar la dependencia con el rendimiento de un único ordenador, obtenemos para Crout el valor , y para el método \ obtenemos el valor. Viendo que el valor de Crout es más eficiente que el de \ puesto que es un poco menor, y por ello es más recomendable usar Crout para matrices tridiagonales, ya que es más eficiente. Destacar que hemos hecho una prueba con rref (que es el método de Gauss-Jordan) y vemos que, en este caso, los tiempos son de 100 veces el obtenido para Gauss, por lo que claramente, el método elegido influye mucho en cuanto a eficiencia computacional.

Bibliografía

[1] <https://es.mathworks.com/help/matlab/ref/mldivide.html>

**Extensión máxima:** 10 páginas, fuente: Calibri12, interlineado 1.5.

**Rúbrica**

|  |  |  |  |
| --- | --- | --- | --- |
| Problemas de contorno unidimensionales, métodos de diferencias finitas | Descripción | Puntuación máxima  (puntos) | Peso  % |
| Criterio 1 | Calidad en la presentación: expresiones matemáticas escritas con editor de ecuaciones, tablas en formato tabla, legibilidad de gráficos… La no presentación en Word supone un 0 en este apartado. | 1.5 | 15 % |
| Criterio 2 | Problema 1. Apartado a. El planteamiento teórico es correcto. | 3 | 30 % |
| Criterio 3 | Problema 1. Apartado b. El planteamiento teórico es correcto. | 2.5 | 25 % |
| Criterio 4 | Problema 1. Apartado c. Tabla de resultados y gráfica correctas | 1 | 10 % |
| Criterio 5 | Problema 1. Apartado d. Tabla de resultados y gráfica correctas | 1 | 10 % |
| Criterio 6 | Problema 1. Apartado d. Comparación entre tiempos de ejecución | 1 | 10 % |
|  |  | **10** | **100 %** |