Ejercicios de programación

JUAN MIGUEL VILLEGAS

1. Considere la ecuación de advección con condiciones de contorno periódicas

$$\begin{cases} u_t + a\,u_x = 0,\ x \in [0,10],\ t \geq 0, \\ u(t,0) = u(t,10),\ t \geq 0, \\ u(0,x) = u^0(x)\ , \end{cases} \tag{1}$$

donde $a < 0$. Implemente un método de tipo beam-warming

$$U_j^{n+1} = U_j^n - \frac{a\Delta t}{2\Delta x}\Big(-3U_j^n + 4U_{j+1}^n - U_{j+2}^n\Big) + \frac{(a\Delta t)^2}{2(\Delta x)^2}\Big(U_j^n - 2U_{j+1}^n + U_{j+2}^n\Big)$$

(si $a > 0$ se simetrizan los índices) a partir del código visto para el algorítmo de Lax–Wendroff para la ecuación de advección. Utilícelo para resolver el problema anterior con $a = -1$, $T = 10$ y $u^0(x) = e^{-(x-5)^2}$. Haga pruebas con distintos valores de Δt y Δx y determine a partir de ellas si es necesario imponer algún tipo de condición de estabilidad. Indique los distintos valores utilizados.

2. Diseñe una rutina para resolver el problema genérico

$$\begin{cases} u_t + a\,u_x = 0,\ x \in [b_0,b_1],\ t \in [0,T], \\ u(t,b_0) = u(t,b_1),\ t \geq 0, \\ u(0,x) = u^0(x) \end{cases}$$

mediante el método de beam-warming. Diseñe la entrada de datos de forma que el usuario pueda especificar el dominio espacial de resolución $[b_0,b_1]$, el coeficiente de transporte a, el tiempo final T, la condición inicial u^0 y los pasos en espacio y tiempo deseados, en la forma que estime conveniente.

3. Considere condiciones iniciales de la forma $u^0(x) = \sin(2n\pi x/10)$ con $n \in \mathbb{N}$ moderado. Estudie la evolución generada por el método programado en los puntos anteriores para el problema (1) y compare con las soluciones exactas. ¿Qué conclusiones extrae?

4. Considere una condición inicial de la forma[1] $u^0(x) = (4-(x-5)^2)\chi_{[3,7]}(x)$. Resuelva la ecuación (1) -tiempo final $T = 10$- con el método beam-warming y calcule a cada paso en tiempo el valor de la cantidad

$$M(t) = \int_0^{10} u(t,x)\,dx$$

[1] Recuerde que $\chi_{[a,b]}(x) = 1$ si $x \in [a,b]$, cero en otro caso.

a partir de las aproximaciones generadas con su método. Compare con la solución exacta. ¿Qué conclusiones extrae?

1 Apartado I

En este apartado nos planteamos simular la ecuación de advección sobre el cuadrado $[0,10]\times[0,10]$ con $a=-1$ y condición inicial $u^0(x)=e^{-(x-5)^2}$ implementando el métode de Beam Warming.

El método a tomar la forma

$$U_j^{n+1}=U_j^n-\frac{a\Delta t}{2\Delta x}\left(-3U_j^n+4U_{j+1}^n-U_{j+2}^n\right)+\frac{(a\Delta t)^2}{2(\Delta x)^2}\left(U_j^n-2U_{j+1}^n+U_{j+2}^n\right)$$

El siguiente código se encuntra en el archivo adjunto

```
 function [x,uint] = apartado1
%
% Resuelve u_t + a u_x = 0 en [ax,bx] con condiciones de
% contorno periodicas empleando el metodo de Bean Warming
% con m nodos espaciales interiores.
%
% Representa la solucion en varios instantes de tiempo.
% Devuelve mallado y aproximacion en la ultima iteracion.
% Modificado del codigo propuesto en clase para la simulacion de
% la ecuacion de adveccion con el metodo de Lax-Wendroff


% Comenzamos introduciendo los datos del problema
global a
m= input('Introduzca el numero de nodos deseado: ');
a = -1;          % parametro de la ecuacion
ax = 0;          % inicio del intervalo espacial
bx = 10;         % fin del intervalo espacial
tfinal = 10;              % Tiempo maximo
eta= @(x) exp(-1*(x - 5).^2); % Cond inicial

% Datos de las discretizaciones
h = (bx-ax)/(m+1);       % h paso espacial
k = 0.4*h;                % k paso temporal
nu = a*k/h;              % numero de Courant
x = linspace(ax,bx,m+2)'; % Fijamos el mallado en el intervalo espacial
nsteps = round(tfinal / k); % numero de pasos temp
nplot = 20;      % representamos cada nplot pasos

% Comprobamos si el ultimo paso llega a tfinal
if abs(k*nsteps - tfinal) > 1e-5
  disp(' ')
  disp(sprintf('OJO *** k no divide a tfinal, k = %9.5e',k))
  disp(' ')
end

% En el caso con condiciones periodicas se tienen:
```

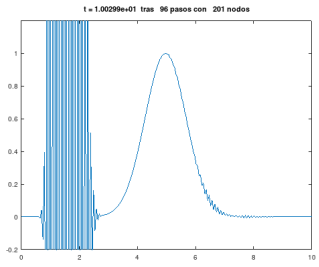
```
% m+1 incognitas u(2:m+2) y u(1) = u(m+2)
I = 2:(m+2);  % indices de las incognitas

% Definicion de condiciones iniciales:
tn = 0;
u0 = eta(x);
u = u0;

% Condiciones de contorno periodicas:
u(1) = u(m+2);
% OJO: nodo fantasma para cond periodicas necesitamos dos nodos
    "fantasma"
u(m+3) = u(2);
u(m+4)=u(3);
% representamos condicion inicial:
clf
plot(x,u0)
axis([0 10 -0.2 1.2]) %axis([0 1 -.2 1.2])
title('Condicion inicial a tiempo 0')
input('Pulse <return> para continuar ');

% Bucle temporal:
for n = 1:nsteps
  tnp = tn + k;  % = t_{n+1}

  % Implementamos el metodo de Beam-Warming:
  u(I) = u(I) - 0.5*nu*(-3*u(I)+4*u(I+1)-u(I+2)) + 0.5*nu^2 * (u(I) -
      2*u(I+1) + u(I+2));

  % Condiciones periodicas:
  u(1) = u(m+2); % redundante para dato inicial bien preparado
  % OJO: los nodos fantasma para cond periodicas
  u(m+3) = u(2);
  u(m+4)=u(3);
  % representamos resultados cada nplot pasos:
  if mod(n,nplot)==0 || n==nsteps
    uint = u(1:m+2); % representamos nodos en el intervalo
    plot(x,uint)
    axis([0 10 -0.2 1.2]) % axis([0 1 -.2 1.2])
    title(sprintf('t = %9.5e tras %4i pasos con %5i nodos',...
                  tnp,n,m+1))
    if n<nsteps, pause(.5); end;
  end

  tn = tnp;  % para el siguiente paso temporal
end % del for
uint = u(1:m+2);
endfunction
```

El principal problema que hemos encontrado en el diseño del método ha sido la necesidad de utilizar dos nodos fantasma por las condiciones periodicas en la frontera.

Estudiemos ahora si es necesario imponer algún tipo de condición de estabilidad, para ello haremos pruebas con 200 nodos en las que utizaremos distintas relaciones entre Δx y Δt, para ello haremos modificaciones en la línea 24 del código.(Debido a que la salida son simulaciones en movimiento no he encontrado la manera de ponerla en este archivo, de todas formas se pueden ejecutar haciendo la modificación anterior en el archivo apartado1.m)

1. Para $\frac{\Delta t}{\Delta x} \leq 2$ he realizado pruebas distintas relaciones $\Delta t = 0.2\Delta x$, $\Delta t = 1\Delta x$, $\Delta t = 1.2\Delta x$, $\Delta t = 1.7\Delta x$ y no he encontrado ningún problema en la simulación.

2. Para $\frac{\Delta t}{\Delta x} > 2$ al probar con $\Delta t = 2.1\Delta x$ he encontrado que la simulación comienza bien pero va habiendo problemas y muchas oscilaciones que no deberían de ocurrir, la siguiente imagen corresponde a la última iteración del método.



Después de ver este fenómeno y probar con $\Delta t = 2.5\Delta x$, $\Delta t = 2.7\Delta x$, $\Delta t = 3\Delta x$ en todas estas simulaciones ocurre lo mismo, luego esto nos lleva a pensar que en este caso el método no es estable.

2 Apartado II

En este apartado se me pedía resolver el problema genérico y el diseño de un método para introducir los datos del problema, todos ellos excepto la condición inicial se introducen cuando el programa se esta ejecutando, la condicion inicial ha de modificarse desde el código del programa, con mis conocimientos sobre Octave no he sido capaz de implementarlo de otra forma.

Esta parte del ejercicio se puede ejecutar deslde el archivo adjunto apartado2.m

El código es el siguiente:

```
function [x,uint] = apartado2
%
% Resuelve u_t + a u_x = 0 en [ax,bx] con condiciones de
% contorno periodicas empleando el metodo de Bean Warming
% con m nodos espaciales interiores.
%
% Representa la solucion en varios instantes de tiempo.
% Devuelve mallado y aproximacion en la ultima iteracion.
% Modificado del codigo propuesto en clase para la simulacion de
% la ecuacion de adveccion con el metodo de Lax-Wendroff


printf(" Comenzamos introduciendo los datos del problema, por motivos
    tecnicos la
condicion inicial debe introducirse desde el codigo\n");
ax= input('Introduzca el inicio del intervalo espacial: ');
bx= input('Introduzca el final del intervalo espacial: ');
a= input('Introduzca el coeficiente de transporte: ');
tfinal= input('Introduzca el tiempo final: ');
eta= @(x) x; %Intoducir aqui la condicion inicual deseada
h= input('Introduzca el paso en espacio: ');
k= input('Introduzca el paso en tiempo: ');



% Datos de las discretizaciones
m=round((bx-ax)/h)-1;   % nodos en espacio
nu = a*k/h;             % numero de Courant
x = linspace(ax,bx,m+2)'; % Fijamos el mallado en el intervalo espacial
nsteps = round(tfinal / k); % numero de pasos temp
nplot = 20;     % representamos cada nplot pasos

% Comprobamos si el ultimo paso llega a tfinal
if abs(k*nsteps - tfinal) > 1e-5
  disp(' ')
  disp(sprintf('OJO *** k no divide a tfinal, k = %9.5e',k))
  disp(' ')
```

```
end

% En el caso con condiciones periodicas se tienen:
% m+1 incognitas u(2:m+2) y u(1) = u(m+2)
I = 2:(m+2);  % indices de las incognitas

% Definicion de condiciones iniciales:
tn = 0;
u0 = eta(x);
u = u0;

% Condiciones de contorno periodicas:
u(1) = u(m+2);
% OJO: nodo fantasma para cond periodicas necesitamos dos nodos
    "fantasma"
u(m+3) = u(2);
u(m+4)=u(3);
% representamos condicion inicial:
clf
plot(x,u0)
axis([ax bx -5 10]) %axis([0 1 -.2 1.2])
title('Condicion inicial a tiempo 0')
input('Pulse <return> para continuar ');

% Bucle temporal:
for n = 1:nsteps
  tnp = tn + k;  % = t_{n+1}

  % Implementamos el metodo de Beam-Warming:
  if a<0
    % Condiciones de contorno periodicas:
  u(1) = u(m+2);
% OJO: nodo fantasma para cond periodicas necesitamos dos nodos
    "fantasma"
  u(m+3) = u(2);
  u(m+4)=u(3);
  u(I) = u(I) - 0.5*nu*(-3*u(I)+4*u(I+1)-u(I+2)) + 0.5*nu^2 * (u(I) -
      2*u(I+1) + u(I+2));
endif
if a>0
  % Condiciones de contorno periodicas:
  u(m+2) = u(1);
% OJO: nodo fantasma para cond periodicas necesitamos dos nodos
    "fantasma"
  u(m+3) = u(2);
  u(m+4)=u(3);
  u(I) = u(I) - 0.5*nu*(3*u(I)-4*u(I+1)+u(I+2)) + 0.5*nu^2 * (u(I) -
      2*u(I+1) + u(I+2));
endif
```

```
  % Condiciones periodicas:
  u(1) = u(m+2); % redundante para dato inicial bien preparado
  % OJO: los nodos fantasma para cond periodicas
  u(m+3) = u(2);
  u(m+4)=u(3);
  % representamos resultados cada nplot pasos:
  if mod(n,nplot)==0 || n==nsteps
    uint = u(1:m+2); % representamos nodos en el intervalo
    plot(x,uint)
    axis([ax bx -5 10]) % axis([0 1 -.2 1.2])
    title(sprintf('t = %9.5e tras %4i pasos con %5i nodos',...
                  tnp,n,m+1))
    if n<nsteps, pause(.5); end;
  end

  tn = tnp;  % para el siguiente paso temporal
end % del for
uint = u(1:m+2);
endfunction
```

3 Apartado III

En el programa implementado comenzamos introduciendo los nodos en espacio deseados y el parámetro n del enunciado.
Simularemos las soluciones aproximádas por el método y las soluciones reales para ponerlas las dos sobre una misma gráfica y así poder compararlas.

```
function [x,uint] = apartado3
%
% Resuelve u_t + a u_x = 0 en [ax,bx] con condiciones de
% contorno periodicas empleando el metodo de Bean Warming
% con m nodos espaciales interiores.
%
% Representa la solucion en varios instantes de tiempo.
% Devuelve mallado y aproximacion en la ultima iteracion.
% Modificado del codigo propuesto en clase para la simulacion de
% la ecuacion de adveccion con el metodo de Lax-Wendroff

%Intoduzcamos los nodos deseados y el n de la condicion inicial
m=input('Introduzca el numero de nodos espacial desado: ');
n=input('Introduzca el n de la condicion inicial: ');
% Comenzamos introduciendo los datos del problema
global a
a = -1;          % parametro de la ecuacion
ax = 0;          % inicio del intervalo esspacial
bx = 10;         % fin del intervalo espacial
tfinal = 10;              % Tiempo maximo
eta= @(x) sin((2*n*pi*x)./(10)); % Cond inicial

% Datos de las discretizaciones
h = (bx-ax)/(m+1);       % h paso espacial
k = 0.8*h;                % k paso temporal
nu = a*k/h;              % numero de Courant
x = linspace(ax,bx,m+2)'; % Fijamos el mallado en el intervalo espacial
nsteps = round(tfinal / k); % numero de pasos temp
nplot = 20;      % representamos cada nplot pasos

% Comprobamos si el ultimo paso llega a tfinal
if abs(k*nsteps - tfinal) > 1e-5
  disp(' ')
  disp(sprintf('OJO *** k no divide a tfinal, k = %9.5e',k))
  disp(' ')
end

% En el caso con condiciones periodicas se tienen:
% m+1 incognitas u(2:m+2) y u(1) = u(m+2)
I = 2:(m+2);  % indices de las incognitas

% Definicion de condiciones iniciales:
```

```
tn = 0;
u0 = eta(x);
u = u0;

% Condiciones de contorno periodicas:
u(1) = u(m+2);
% OJO: nodo fantasma para cond periodicas necesitamos dos nodos
    "fantasma"
u(m+3) = u(2);
u(m+4)=u(3);
% representamos condicion inicial:
clf
plot(x,u0)
axis([0 10 -1.2 1.2]) %axis([0 1 -.2 1.2])
title('Condicion inicial a tiempo 0')
input('Pulse <return> para continuar ');
% Bucle temporal:
for n = 1:nsteps
  if n>2
    clear eta1,u1,u_sol;
  endif
  tnp = tn + k;
  eta1=@(x) sin((2*n*pi*(x+tnp))./(10)); %Introducimos la solucion del
      problema (la que no es aprimada)
  u1 = eta1(x);
  u_sol = u1;
  % Implementamos el metodo de Beam-Warming:
  u(I) = u(I) - 0.5*nu*(-3*u(I)+4*u(I+1)-u(I+2)) + 0.5*nu^2 * (u(I) -
      2*u(I+1) + u(I+2));

  % Condiciones periodicas:
  u(1) = u(m+2); % redundante para dato inicial bien preparado
  % OJO: los nodos fantasma para cond periodicas
  u(m+3) = u(2);
  u(m+4)=u(3);
  % representamos resultados cada nplot pasos:
  if mod(n,nplot)==0 || n==nsteps
    uint = u(1:m+2); % representamos nodos en el intervalo
    plot(x,uint,u_sol)
    %plot(x,uint)
    %plot(x,uint)
    axis([0 10 -1.2 1.2]) % axis([0 1 -.2 1.2])
    title(sprintf('t = %9.5e tras %4i pasos con %5i nodos',...
                  tnp,n,m+1))
    if n<nsteps, pause(.5); end;
  end
  tn = tnp;  % para el siguiente paso temporal
end % del for
uint = u(1:m+2);
clear x,eta,u0,u;
```

```
endfunction
```

La ejecución de este apartado se puede realizar desde el archivo apartado3.m
Después de ejecutarlo con 200 nodos en espacio y $n = 150, 200, 250$ apreciamos que hay mucha diferencia entre las soluciones aproximadas y las soluciones del problema de advección, esto puede ser debido al fenómeno que mencionamos en clase donde la condición inicial oscilaba muchísimo y por tanto los nodos no pueden captar el verdadero comportamiento de la condición inicial ni de sus aproximaciones.

4 Apartado IV

En este ejercicio aproximaremos el valor de la integral por la siguiente expresión

$$\int_0^{10} u(x,t)dx = h \cdot \sum u_{aprox}$$

donde u_{apox} son las aproximaciones obtenidas por el método.

En nuestro caso, las aproximaciones las tenemos almacenadas en el vector *uint*, luego para realizar dicho cálculo, multiplicamos h por el vector anterior y sumamos todos sus elementos, esto lo realizaremos en cada paso en tiempo.

```
function [x,uint] = apartado4
%
% Resuelve u_t + a u_x = 0 en [ax,bx] con condiciones de
% contorno periodicas empleando el metodo de Bean Warming
% con m nodos espaciales interiores.
%
% Representa la solucion en varios instantes de tiempo.
% Devuelve mallado y aproximacion en la ultima iteracion.
% Modificado del codigo propuesto en clase para la simulacion de
% la ecuacion de adveccion con el metodo de Lax-Wendroff

m=input('Introduzca el numero de nodos en espacio deseado: ')

% Comenzamos introduciendo los datos del problema
global a
a = -1;          % parametro de la ecuacion
ax = 0;          % inicio del intervalo esspacial
bx = 10;         % fin del intervalo espacial
tfinal = 10;              % Tiempo maximo
%eta= @(x) (0) .* (x < 3 ) +((4-(x-5)).^2 .* (x >= 3 & x <=7 ) + (0) .*
    (x >7 & x <=10 ); % Cond inicial
eta=@(x) (0).*(x <= 3)+(4-(x -5).^2).*(x >= 3 & x <= 7 ) +(0).*(x>7)
% Datos de las discretizaciones
h = (bx-ax)/(m+1);       % h paso espacial
k = 0.2*h;                % k paso temporal
nu = a*k/h;              % numero de Courant
x = linspace(ax,bx,m+2)'; % Fijamos el mallado en el intervalo espacial
nsteps = round(tfinal / k); % numero de pasos temp
nplot = 20;      % representamos cada nplot pasos

% Comprobamos si el ultimo paso llega a tfinal
if abs(k*nsteps - tfinal) > 1e-5
  disp(' ')
  disp(sprintf('OJO *** k no divide a tfinal, k = %9.5e',k))
  disp(' ')
end
```

```
% En el caso con condiciones periodicas se tienen:
% m+1 incognitas u(2:m+2) y u(1) = u(m+2)
I = 2:(m+2);  % indices de las incognitas

% Definicion de condiciones iniciales:
tn = 0;
u0 = eta(x);
u = u0;


% Condiciones de contorno periodicas:
u(1) = u(m+2);
% OJO: nodo fantasma para cond periodicas necesitamos dos nodos
    "fantasma"
u(m+3) = u(2);
u(m+4)=u(3);
% representamos condicion inicial:
clf
plot(x,u0)
axis([0 10 -1 5]) %axis([0 1 -.2 1.2])
title('Condicion inicial a tiempo 0')
 %%Procedemos a calcular la integral%%
   vector_integral=h.*u0;
   valor_integral=sum(vector_integral);
   disp("El valor de la integral a tiempo cero
       es="),disp(valor_integral);
   clear vector_integral,valor_integral;
input('Pulse <return> para continuar ');


% Bucle temporal:
for n = 1:nsteps
  tnp = tn + k;  % = t_{n+1}

  % Implementamos el metodo de Beam-Warming:
  u(I) = u(I) - 0.5*nu*(-3*u(I)+4*u(I+1)-u(I+2)) + 0.5*nu^2 * (u(I) -
      2*u(I+1) + u(I+2));

  % Condiciones periodicas:
  u(1) = u(m+2); % redundante para dato inicial bien preparado
  % OJO: los nodos fantasma para cond periodicas
  u(m+3) = u(2);
  u(m+4)=u(3);
  % representamos resultados cada nplot pasos:
  if mod(n,nplot)==0 || n==nsteps
    uint = u(1:m+2); % representamos nodos en el intervalo
    plot(x,uint)
    axis([0 10 -1 5]) % axis([0 1 -.2 1.2])
    title(sprintf('t = %9.5e tras %4i pasos con %5i nodos',...
                    tnp,n,m+1))
```

```
    if n<nsteps, pause(.5); end;
    %%Procedemos a calcular la integral%%
   vector_integral=h.*uint;
   valor_integral=sum(vector_integral);
   disp("El valor de la integral a tiempo(primera salida) es (segunda
       salida)"),disp(tn),disp(valor_integral);
   clear vector_integral,valor_integral;
  end

  tn = tnp;  % para el siguiente paso temporal
end % del for
uint = u(1:m+2);
endfunction
```

Después de ejecutar el programa con el archivo apartado4.m vemos que el valor de la integral no cambia mucho (con una eleccion de m que no sea muy pequeña), esto tiene sentido ya que en todo momento estamos integrando sobre $[0, 10]$ y la gráfica de nuesta función solo se esta transportando a lo largo del intervalo.

Una de las conclusiones que podríamos extraer es que la cantidad $M(t)$ es constante ya que las variaciones de esta cantidad podrían ser debidas a que estamos tomando aproximación de la función u