

DEPARTAMENTO DE MATEMATICA APLICADA
UNIVERSIDAD COMPLUTENSE DE MADRID
Métodos Numéricos en Ingeniería Química, Curso 07–08
PRACTICAS. Hoja 1
Primeras prácticas con MATLAB

Ejecutar los siguientes comandos, trabajando de forma interactiva con Matlab. Anota a la derecha lo que hace cada una de las instrucciones que siguen.

Práctica 1 *Tablas y gráficos.*

```
i)
t=1:10
t=t+5
2*t
1./t
t=[1,t]
t=[t,100]
s=[t; 1./t]
s=[s,[0;0]]
t.^2
s.^2
ii)
t=0:0.1:4*pi;
s=linspace(0,4*pi,1000)
x=sin(t);
y=cos(t)
z=exp(-t)
plot(t,x)
plot(t,y)
plot(t,x,'r',t,y,'g-*')
plot(x,y)
hold on
plot(9*x,4*y)
hold off
plot3(x,y,z)
u=x.*z
plot(t,u)
iii)
A=[t;x;y]
size(A)
size(A,1)
size(A,2)
plot3(A(1,:),A(2,:),A(3,:))
j=size(A,2)-8:size(A,2)
C=A(:,j)
```

Práctica 2 *Matrices*

```
i)
clear A
A=[0,-1,4;4,2,1/3;-1,1/2,-2]
```

```

det(A)
eig(A)
[V,D]=eig(A)
B=V(:)
D=B-C'
p=poly(A)
roots(p)
ii)
B=[A(3,:);A(1,:)+A(2,:)]
v=[1,-1,0]
y=B*v
v=v(:)
y=B*v
z=A\v
A*z-v

```

Práctica 3 *Manipulación de funciones*

```

clear all
f=2*exp(-t).*sin(t)
f='2*exp(-t).*sin(t)''
ezplot(f)
fplot(f,[0,8])
fplot('sin(x)./x',[-20,20])
t=1
eval(f)
t=0:0.1:2
y=eval(f)
plot(t,y)

```

Práctica 4 *Representaciones graficas*

```

x = -7.5: .5:7.5;
y = x
[X, Y] = meshgrid(x, y);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R)./R;
mesh(x, y, Z)
surf(x, y, Z)

```

Práctica 5 *Vamos a escribir en un fichero instrucciones de MATLAB de forma que, utilizando el Algoritmo de Newton, calculemos aproximaciones a un cero de la función $f(x) = x - x^3$.*

```

function x=minewton(f,df,x0,TOL)
% metodo de Newton para ecuaciones, dadas f y su derivada df

% variables de entrada: f,df ficheros .m
% x0 dato inicial vector columna
% TOL tolerancia error

% variables de salida: x aproximacion al punto fijo en 'i=contador' iteracione

```

```

ERR=TOL+1;
x=x0;
contador=0;
while ERR>TOL
    contador=contador+1;
    DF=feval(df,x);
    F=feval(f,x);
    w= DF\F;
    ERR=norm(w,inf);
    x=x-w;
    if contador>1000
        breaks
        disp('excede numero maximo de iteraciones')
    end
end
end

```

Llamad a este fichero **minewton.m**. Ahora podemos ejecutarlo. Para ello necesitaremos en primer lugar definir un fichero .m con la correspondiente función f y otro con su derivada, df , como una matriz si f fuera de varias variables. Para ello escribimos en dos ficheros:

```

function F=fun1(x)
%
F=x-x.^3;

function DF=dfun1(x)
%
DF=1-3*x.^2;

```

y los guardamos con el nombre `fun1.m` y `dfun1.m` respectivamente. Esto nos permitirá evaluar la función y su derivada para cada x . A continuación escribimos las variables de entrada en la línea de comandos

```
>>TOL=1e-10;
```

```
>>x0=-2;
```

y ahora llamamos al programa **minewton.m** desde la línea de comandos

```
>> x=minewton (@fun1 , @dfun1 , x0 , TOL );
```

Práctica 6 Calcule las raíces de las siguientes funciones utilizando el método de Newton

i) $f(x) = x - x^2$,

ii) $f(x) = xe^{-x} + 1$,

iii) $f(x_1, x_2) = (x_1 - x_2^2, \sin(x_1) - 0,7x_2)$.

Práctica 7 i) Escribe el fichero función **mitridiagonal.m** que tomando como datos cuatro vectores columna A, B, C, R de tamaños respectivos $N, N - 1, N - 1, N$ resuelva el sistema lineal tridiagonal $Mx = R$, donde la Matriz M tiene dimensión $N \times N$, la diagonal esta dada por el vector A , la diagonal inferior por el vector B , la diagonal superior por el vector C y el resto de los elementos de la matriz son ceros.

```

function [x]=mitridiagonal(A,B,C,R)
% Esta funcion resuelve un sistema lineal Mx=R
% aplicando el metodo de eliminacion de Gauss
% La matriz M es tridiagonal y de dimension N.
% A: elementos de la diagonal. Dimension=N. %Vectores columna

```


DEPARTAMENTO DE MATEMÁTICA APLICADA
UNIVERSIDAD COMPLUTENSE DE MADRID
Métodos Numéricos en Ingeniería Química, Curso 07–08
PRACTICAS. Hoja 2
Resolución Numérica de Problemas de contorno

Práctica 8 *El programa Matlab tiene integrado el programa **bvp4c.m** que permite resolver numéricamente Problemas de contorno*

$$u''(x) = f(x, u(x), u'(x)), \quad x \in (a, b)$$
$$bc(u(a), u'(a), u(b), u'(b)) = 0$$

utilizando métodos de colocación. La sintaxis básica de esta función es:

```
sol=bvp4c (@odefun , @bcfun , solinit )
```

donde

```
@odefun
```

es el nombre del fichero función de la EDO que se quiere resolver, como un sistema de dos EDO

```
dv=func ( x , v )
```

x es escalar, $v=[v_1; v_2] = [u(x); u'(x)]$, $dv=[f_1; f_2] = [u'(x); f(x, u(x), u'(x))]$

donde

```
@bcfun
```

es el nombre del fichero función de la condición de contorno

```
bc= bcfun (ua , ub )
```

debe devolver un vector columna $bc=[bc_1; bc_2]$ con datos de frontera tipo general, por ejemplo $ua(1) + \alpha ua(2) - \beta$ representa una condición de frontera mixta no homogénea

$$u(a) + \alpha \frac{\partial u}{\partial v}(a) = \beta$$

`solinit` es el dato inicial de la EDO en un conjunto de nodos ordenados, i.e. $a = x_1 < x_2 < \dots < x_N = b$, $u_1, u_2, \dots, u_N = b$ donde u_j es una aproximación inicial de la solución u en el nodo x_j . Se puede usar la función `bvpinit` que forma una aproximación inicial

```
solinit= bvpinit (xmesh , uinit )
```

para un mallado inicial `xmesh`, o bien usar un fichero de función

```
solinit= bvpinit (0:0.1:1 , @initfun )
```

La solución se evalúa en los puntos `xint` usando la salida `sol` de `bvp4c` y la función `deval`

```
uint= deval (sol , xint )
```

La salida `sol` es una estructura con

`sol.x` – mallado seleccionado por `bvp4c`

`sol.y` – aproximación a $u(x)$ en los puntos del mallado `sol.x`

`sol.yp` – aproximación a $u'(x)$

`sol.solver` – 'bvp4c'

Se pueden considerar problemas singulares del tipo

$$v' = D * v/x + f(x, v), \quad x \in (0, b)$$
$$bc(v(0), v(b)) = 0$$

con $v = [v_1; v_2]$. La matriz constante D se especifica como el valor del término 'SingularTerm' en las opciones de `bvpset` y la función `@odefun` evalúa solamente $f(x, v)$. Las condiciones de contorno deben ser consistentes con la condición necesaria $D * v(0) = 0$ y el dato inicial debe verificar esta condición.

Práctica 9 Crea un fichero tipo script con nombre **testmibvp4c.m** que

- 1.- guarde los datos del problema,
- 2.- ejecute el fichero **bvp4c.m** y
- 3.- pinte la gráfica de la solución aproximada.

Ejemplo 1 para la **Práctica 10** con $d = 0$.

Fichero odefuncreac1.m

```
function f=odefuncreac1(x,u)
global Thiele k m n
f=zeros(2,1);
f(1)=u(2);
f(2)=(Thiele.^2)*(u(1).^m)./(1+k*u(1).^n);
```

Fichero bcdirichlet.m

```
function res=bcdirichlet(ua,ub)
res=zeros(2,1);
res(1)=ua(2);
res(2)=ub(1)-1; %CC Dirichlet
```

Fichero testmibvp4c.m

```
clear all
close all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global Thiele k m n Bim
Thiele=1; k=0; m=2; n=0;

    odefun=@odefuncreac1;
    bcfun=@bcdirichlet;
    solinit=bvpinit([0 .25 .5 .75 1],[1 0]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

sol = bvp4c(odefun,bcfun,solinit);
xint = linspace(0,1);
uint = deval(sol,xint);
plot(xint,uint(1,:));
title(['la grafica de la solucion aproximada con f= ', func2str(odefun)])
xlabel('x');
ylabel('solucion u');
```

Ejemplo 2 para la **Práctica 11** de datos iniciales generados por un fichero función.

Fichero initfuncreac1.m

```
function yinit = initfuncreac1(x)
global Thiele
yinit = [ cosh(Thiele*x)/cosh(Thiele)
          Thiele*sinh(Thiele*x)/cosh(Thiele) ]
```

Modificaciones al Fichero testmibvp4c.m

```
initfun=@initfuncreac1;
solinit = bvpinit(0:.1:1,initfun);
```

Ejemplo 3 para la Práctica 10 con $d = 2$.

Modificaciones al Fichero **testmibvp4c.m**

```
d=2;
options = bvpset('singularterm',[0 0;0 -d]);
sol = bvp4c(odefun,bcfun,solinit,options);
```

Ejemplo 4 para la Práctica 12 con $d = 2$.

Modificaciones al Fichero **testmibvp4c.m**

```
d=2;
options = bvpset('singularterm',[0 0 0 0; 0 -d 0 0; 0 0 0 0; 0 0 0 -d]);
sol = bvp4c(odefun,bcfun,solinit,options);
```

Práctica 10 Resuelve numéricamente el problema de contorno

$$\frac{1}{x^d}(x^d u')' = \tau^2 \frac{u^m}{1 + k u^n}, \quad u'(0) = 0, \quad u(1) = 1. \quad (1)$$

para $d = 0, 1, 2$. Otras condiciones de contorno habituales son

$$u'(1) + Bi_m u(1) = Bi_m, \quad (2)$$

donde Bi_m es el número de Biot de transferencia de masa. Este problema corresponde a la reacción-difusión dentro de la partícula del catalizador poroso, en una ecuación adimensionalizada; $u = u(x)$ representa la concentración de un reactante que desaparece a una velocidad $-f(u)$ por unidad de volumen por unidad de tiempo de la partícula catalítica. El parámetro τ es el módulo de Thiele.

Escribe los ficheros tipo función **odefuncreac1.m**, **bcdirichlet.m** con la ecuación y las condiciones de contorno de (1). Define τ , k , m , n y Bi_m como variables globales. Toma $\tau = 0, 1; 1; 10$, $k = 0$, $m = 1$, $n = 0$ y como aproximación inicial $c_0(r) = 1$ y $c_0(r) = \frac{\cosh(\tau r)}{\cosh(\tau)}$, y compara con las graficas en Finlayson, p. 85. Toma ahora $\tau = 10$, $k = 0$, $m = 2$, $n = 0$, compara con las graficas en Davis, p. 121. Selecciona distintos valores de $\tau = 7, 10, 12$, (cf. Davis, p. 121 y ss. para $\tau^2 = 50 \approx 49$, $\tau^2 = 100$, $\tau^2 = 150 \approx 144$).

Escribe el ficheros tipo función **bcrobin.m** con las condiciones de contorno (2). Toma $\tau = 5$ y selecciona distintos valores de $k \in (1, 10)$, $m = 1, 2$, $n = 1, 2$. Toma tambien valores de $\tau < 1$ y $\tau > 2$. Toma $Bi_m = 10$, (cf. Froment).

Práctica 11 Resuelve

$$\begin{cases} \frac{1}{r^d}(r^d c_r)_r = \tau^2 c^n e^{\gamma - \gamma/T}, & r \in (0, 1) \\ c_r(0) = 0 & c_r(1) + Bi_m c(1) = Bi_m \end{cases}$$

donde $T(r) = 1 + \beta[c(1) - c(r)] + \beta\delta[1 - c(1)]$. Cf. B. Finlayson "Nonlinear Analysis in Chemical Engineering" pp 82 y ss.

Escribe el fichero tipo función **odefuncreacarrhenius.m**. Define $\tau, \gamma, n, \beta, \delta$ y Bi_m como variables globales. Utilizar los valores de $d = 0$, $\tau = 1$, $\gamma = 30$, $n = 1, 2$, $\beta = 0,02$, $Bi_m \rightarrow \infty$ es decir $c(1) = 1$, $T(r) = 1 + \beta[1 - c(r)]$ y como aproximación inicial $c_0(r) = \frac{\cosh(\tau r)}{\cosh(\tau)}$.

Práctica 12 Resuelve el sistema

$$\begin{cases} \frac{1}{r^d}(r^d c_r)_r = \tau^2 R(c, T), \\ \frac{1}{r^d}(r^d T_r)_r = -\beta\tau^2 R(c, T), & r \in (0, 1) \\ c_r(0) = T_r(0) = 0 \\ c_r(1) + Bi_m c(1) = Bi_m, & T_r(1) + Bi T(1) = Bi, \end{cases}$$

donde $R(c, T) = c^n e^{\gamma - \gamma/T}$. Cf. B. Finlayson "Nonlinear Analysis in Chemical Engineering" pp 80 y ss.

Escribe los ficheros tipo función **odefuncreacarrhenius2.m**, **bcdirichlet2.m** con la ecuación y las condiciones de contorno. Utilizar los valores de $d = 2$, $\tau = 1$, $\gamma = 30$, $n = 1$, $\beta = 0,02$, $Bi, Bi_m \rightarrow \infty$ es decir $c(1) = T(1) = 1$, y como aproximación inicial $c_0(r) = 1$ y $T_0(r) = 1$. Utilizar ahora $\beta = 0,4$. y como aproximación inicial $c_0(r) = 0$, $T_0(r) = 1 + \beta$. Utilizar ahora $\tau = 0,5$. Observa que hay múltiples estados estacionarios seleccionando distintas aproximaciones iniciales $c_0(r) = 1$, $c_0(r) = 0$ y $T_0(r) = 1 + \beta[1 - c_0(r)]$. Compara con los resultados de la p. 102.

DEPARTAMENTO DE MATEMATICA APLICADA
UNIVERSIDAD COMPLUTENSE DE MADRID
Métodos Numéricos en Ingeniería Química, Curso 07–08
PRACTICAS. Hoja 3
Diferencias Finitas

Práctica 13 Crea el fichero tipo función **midiffin.m** que tomando como datos las funciones $p(x)$, $q(x)$, $r(x)$ de la ecuación diferencial implemente el método de diferencias finitas para el problema lineal

$$\begin{cases} -u'' + p(x)u' + q(x)u = r(x) \\ u'(0) = a, u'(L) + \beta u(L) = b, (CC = 1) \quad (\text{o bien } u(L) = b, CC = 2) \end{cases}$$

Los datos de entrada de este fichero función son: $p, q, r, L, a, b, \beta, N, CC$ donde N es el tamaño de la discretización y $CC = 1, 2$ especifica las diferentes condiciones de contorno. Los datos de salida deben ser $[x, u]$, donde x es la discretización de $[0, L]$ y u es el vector solución en los nodos x_i .

Indicación: No almacenar la matriz cuadrada que aparece con este método.

Práctica 14 i) Crea el fichero tipo script **testmidiffin.m** que ejecute **midiffin.m** y pinte la solución del problema de contorno para la **Práctica 10** con $m = 1, n = 0$.

ii) Aplicarlo a la **Práctica 10** con $m = 1, n = 0$ y las condiciones de contorno modificadas.

Práctica 15 Crea el fichero tipo función **midiffinnol.m** que tomando como datos los coeficiente $p(x), q(x)$ y las funciones $f(x, u)$ y $\frac{\partial f}{\partial u}, g(x, u), \frac{\partial g}{\partial u}$ implemente el método de diferencias finitas para el problema no lineal

$$-u'' + p(x)u' + q(x)u = f(x, u)$$

con condiciones de contorno:

$$\begin{aligned} i1) \quad & u'(0) = 0, \quad \frac{\partial u}{\partial x}(L) + \beta u(L) = g(L, u(L)), \quad (CC = 1) \\ i2) \quad & u'(0) = 0, \quad u(L) = b, \quad (CC = 2) \end{aligned}$$

Los datos de entrada de este fichero función son: $f, \frac{\partial f}{\partial u}, g, \frac{\partial g}{\partial u}, p, q, L, \beta, N, u_0, TOL, CC$ donde N es el tamaño de la discretización, u_0 es la condición inicial para iterar el método de Newton, TOL es la tolerancia del método de Newton y CC es una variable para especificar el tipo de condición de contorno. Los datos de salida deben ser $[x, u]$, donde x es la discretización de $[0, L]$ y u es el vector solución en los nodos x_i .

Práctica 16 Crea el fichero tipo script **testmidiffinnol.m** que ejecute **midiffinnol.m** y pinte la solución (o soluciones) del problema de contorno para las ecuaciones de las **Prácticas 10, 11, 12**

DEPARTAMENTO DE MATEMÁTICA APLICADA
UNIVERSIDAD COMPLUTENSE DE MADRID
Métodos Numéricos en Ingeniería Química, Curso 07–08
PRACTICAS. Hoja 4
Métodos de Colocación

Práctica 17 Escribe el fichero función **legendregaussradau.m** que dado el número de puntos de colocación ($N = 1, \dots, 6$) y la geometría de la ecuación (plana, cilíndrica o esférica) calcule los puntos de colocación de Gauss-Radau del método de colocación ortogonal. **Indicación** utiliza los comandos `poly`, `polyval`, `roots`, `sort`, sabiendo que $\text{poly}(A) = \det(xI - A) = [c_1, \dots, c_{N+1}]$ donde $p(x) = c_1x^N + \dots + c_Nx + c_{N+1}$, que `polyval(p, 1)` evalúa el polinomio p en el punto 1, y que `roots(p)` calcula los ceros de p

Práctica 18 Escribe el fichero función **legendregausslobato.m** que dado el número de puntos de colocación ($N = 1, \dots, 6$), calcule los puntos de colocación de Gauss-Lobato del método de colocación ortogonal. **Indicación** utiliza el comando `polyder`. Utiliza el Help de MATLAB

Práctica 19 Escribe el fichero función **mismatricesradau.m** que dado el número de puntos de colocación de Gauss-Radau ($N = 1, \dots, 6, \dots$) y los puntos de colocación x calcule las matrices Q , $P = Q^{-1}$, A y B del método de colocación ortogonal. **Indicación** efectúa operaciones por columnas.

Práctica 20 Escribe el fichero función **mismatriceslobato.m** que dado el número de puntos de colocación de Gauss-Lobato ($N = 1, \dots, 6, \dots$) y los puntos de colocación x calcule las matrices Q , $P = Q^{-1}$, A y B del método de colocación ortogonal.

Práctica 21 Escribe el fichero función **micolradau.m** que implemente el método de colocación ortogonal para los problemas

$$\begin{cases} -\frac{1}{r^d}(r^d u_r)_r = f(r), & r \in (0, 1), \\ u_r(0) = 0 \\ u'(1) + \beta u(1) = b, & (CC = 1) \quad \text{o bien } u(1) = b, & (CC = 2). \end{cases}$$

con $d = 0, 1, 2$.

Los datos de entrada del fichero función son: d (la dimensión del problema), N (el número de puntos del método de colocación), la función f , los datos de contorno b, β, CC . Las variables de salida deben incluir los puntos de colocación x , la matriz P y los valores u .

Práctica 22 Escribe el fichero script **testmicolradau.m** que ejecute **micolradau.m** en los ejemplos de la **Práctica 10** con $m = 1$ y pinte en una gráfica la solución obtenida. Comparar los resultados con los obtenidos por el método de diferencias finitas. **Indicación** utiliza el comando `flipud(c)` que cambia el orden de las componentes de un vector columna, de $[c_1, \dots, c_{N+1}]$ a $[c_{N+1}, \dots, c_1]$, el comando `c=[zeros(1, N+1); c']` y el comando `c=c(:)'` que cambia una matriz en un vector.

Práctica 23 Escribe el fichero función **micollobato.m** que implemente el método de colocación ortogonal para el problema

$$\begin{cases} -u_{xx} + p(x)u_x + q(x)u = r(x), & r \in (-1, 1), \\ u(-1) = a \\ u'(1) + \beta u(1) = b, & (CC = 1) \quad \text{o bien } u(1) = b, & (CC = 2). \end{cases}$$

Los datos de entrada del fichero función son: p, q, r, N (el número de puntos del método de colocación), los datos de contorno a, b, β, CC .

Práctica 24 Escribe el fichero script **testmicollobato.m** que ejecute **micollobato.m** y pinte en una gráfica la solución obtenida. Comparar los resultados con los obtenidos por el método de diferencias finitas.

Práctica 25 Escribe el fichero función **micolradaunol.m** que implemente el método de colocación ortogonal para los problemas

$$\begin{cases} -\frac{1}{r^d}(r^d u_r)_r = f(u), & r \in (0, 1), \\ u_r(0) = 0 \\ u_r(1) = g(u(1)). \end{cases}$$

con $d = 0, 1, 2$.

Los datos de entrada del fichero función son: d (la dimensión del problema), N (el número de puntos del método de colocación), las funciones f , g y sus derivadas (que debe estar en un fichero función), la tolerancia del método de Newton TOL , la condición inicial del método de Newton v y el dato de contorno b . Los datos de salida deben ser los puntos de colocación x y los valores de la solución en los puntos de colocación u .

Práctica 26 Escribe el fichero script **testmicolradaunol.m** que ejecute **micolradaunl.m** y pinte en una gráfica la solución (o soluciones) del problema de contorno para las ecuaciones de las **Prácticas 10, 11, 12**.

DEPARTAMENTO DE MATEMÁTICA APLICADA
UNIVERSIDAD COMPLUTENSE DE MADRID
Métodos Numéricos en Ingeniería Química, Curso 07–08
PRACTICAS. Hoja 5
Resolución Numérica de Problemas de evolución

Práctica 27 El programa Matlab tiene integrado el programa **pdepe.m** que permite resolver numéricamente **Problemas de evolución** del tipo

$$c \left(x, t, u, \frac{\partial u}{\partial x} \right) \frac{\partial u}{\partial t} = x^{-d} \frac{\partial}{\partial x} \left[x^d f \left(x, t, u, \frac{\partial u}{\partial x} \right) \right] + s \left(x, t, u, \frac{\partial u}{\partial x} \right), \quad x \in (x_l, x_r) \quad t \in (t_0, T)$$

$$u(x, t_0) = u_0(x), \quad x \in (x_l, x_r)$$

$$p(x, t, u) + q(x, t) f \left(x, t, u, \frac{\partial u}{\partial x} \right) = 0, \quad x = x_l, x = x_r \quad t \in (t_0, T)$$

También se pueden resolver sistemas de ecuaciones. La sintaxis básica de esta función es:

`sol=pdepe(d, @pdefun, @icfun, @bcfun, xmesh, tspan)`

donde

`@pdefun`

es el nombre del fichero función de la EDP que se quiere resolver,

$$[c, f, s] = \text{pdefun} \left(x, t, u, \frac{\partial u}{\partial x} \right)$$

donde

`@icfun`

es el nombre del fichero función dato inicial

$$u_0 = \text{icfun}(x)$$

donde

`@bcfun`

es el nombre del fichero función dato de frontera

$$[p_l, q_l, p_r, q_r] = \text{bcfun}(x_l, u_l, x_r, u_r, t)$$

`xmesh` es un vector $[x_0, x_1, \dots, x_N]$ especificando el mallado espacial de puntos $x_l = x_0 < x_1 < \dots < x_N = x_r$

`tspan` es un vector $[t_0, t_1, \dots, T]$ especificando el mallado temporal $t_0 < t_1 < \dots < T$

Práctica 28 Crea un fichero tipo script con nombre **testmipdepe.m** que

1.- guarde los datos del problema,

2.- ejecute el fichero **pdepe.m** y

3.- pinte

a) Si la EDP es una ecuación: la gráfica en 3D de la solución aproximada.

b) Si la EDP es un sistema de dos ecuaciones: la gráfica en 3D de cada componente de la solución aproximada.

Ejemplo 1 para la Práctica 30

Crea los ficheros **pdefuncreac1.m**, **pdefuncreac1c.m** y **pdefuncbdirichlet.m**

Fichero **testmipdepe.m**

```

close all
clear all
global gamma beta Thiele
gamma=20; beta=.6; Thiele=0.5;

    pdefun=@pdefuncreac1;
    icfun=@pdefuncreaclic;
    bcfun=@pdefuncreac1bc

d=0;%d=1;d=2;
xmesh=0:.1:1;
tspan=0:1:100;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

sol = pdepe(d,pdefun,icfun,bcfun,xmesh,tspan);

x=xmesh;
t=tspan;

if size(sol,3)==1
% Extrae la primera componente de la solucion u.
u = sol(:,:,1);

% dibuja en 3D
surf(x,t,u)
title(['Solucion numerica calculada con ', num2str(size(xmesh,2)),' puntos de
xlabel('Distancia x')
ylabel('Tiempo t')
colorbar

% El perfil de la solucion en tiempo final
figure
plot(x,u(end,:))
title(['Solucion en t = ',num2str(t(end))])
xlabel('Distancia x')
ylabel(['u(x, ',num2str(t(end)),')'])
[UOUTEND,DUOUTDXEND] = pdeval(d,xmesh,sol(t(end),:,1),xmesh);

% -----

elseif size(sol,3)==2
u1 = sol(:,:,1);
u2 = sol(:,:,2);

figure
surf(x,t,u1)
title('u_1(x,t)')
xlabel('Distancia x')
ylabel('Tiempo t')
colorbar

```

```

figure
surf(x,t,u2)
title('u_2(x,t)')
xlabel('Distancia x')
ylabel('Tiempo t')
colorbar

end

```

Práctica 29 Resuelve el problema concreto

$$\begin{cases} c_t - c_{rr} = -\tau^2 R(c, T), & r \in (0, 1), t \in [0, T] \\ c_r(0, t) = 0 \\ c(1, t) = 1 \\ c(r, 0) = 1,01 \end{cases}$$

donde $R(c, T) = (1 - c)e^{\gamma - \gamma/T}$ y toma $T(r, t) = 1 + \beta(1 - c(r, t))$. Utilizar los valores de $\gamma = 20$, $\tau = 0,5$ y $\beta = 0,6$.

Práctica 30 Resuelve

$$\begin{cases} T_t - T_{rr} = \beta' R(c, T), & r \in (0, 1), t \in [0, T] \\ c_t - c_{rr} = \beta R(c, T), & r \in (0, 1), t \in [0, T] \\ T_r(0, t) = c_r(0, t) = 0 \\ T_r(1, t) + Bi_m T(1, t) = Bi_m T_w(t), & c_r(1, t) = 0 \\ T(r, 0) = T_0 \quad c(r, 0) = c_0 \end{cases}$$

donde $R(c, T) = (1 - c)e^{\gamma - \gamma/T}$. Cf. el libro de B. Finlayson "Nonlinear Analysis in Chemical Engineering" pp 192 y ss. Utilizad los valores de $\beta = 0,3$, $\beta' = 0,2$, $\gamma = 20$, $Bi_m = 1$, $T_w(t) = 0,92$, y comparad con la solución obtenida en la p. 195. ¿Podrías modificar tu programa para cubrir esas condiciones de contorno?. Utilizad ahora $Bi_m = 20$, $T_w(t) = 1$, y comparad con la solución obtenida en las p. 201-202.

Práctica 31 Resuelve el problema concreto

$$\begin{cases} M_1 T_t - \frac{1}{r^d} (r^d T_r)_r = \beta \tau^2 R(c, T), & r \in (0, 1), t \in [0, T] \\ M_2 c_t - \frac{1}{r^d} (r^d c_r)_r = -\tau^2 R(c, T), & r \in (0, 1), t \in [0, T] \\ T_r(0, t) = c_r(0, t) = 0 \\ T(1, t) = c(1, t) = 1 \\ T(r, 0) = 1,05 \quad c(r, 0) = 1,0 \end{cases}$$

donde $R(c, T) = ce^{\gamma - \gamma/T}$. Utilizar los valores de $d = 0$, $M_1 = 176$, $M_2 = 199$, $\gamma = 20$, $\beta = 0,6$, $\tau = 0,5$. Comparad la solución obtenida con las del libro de B. Finlayson "Nonlinear Analysis in Chemical Engineering" pp 206 y ss.

Modifica ahora las condiciones de contorno a

$$T_r(1, t) + Bi_m T(1, t) = Bi_m, \quad c_r(1, t) + Bi c(1, t) = Bi$$

toma $Bi = 27, 65$, $Bi_m = 33, 25$, $T(r, 0) = 1,1$. y comparad la solución obtenida con las de la p 208.

DEPARTAMENTO DE MATEMÁTICA APLICADA
UNIVERSIDAD COMPLUTENSE DE MADRID
Métodos Numéricos en Ingeniería Química, Curso 07–08
PRACTICAS. Hoja 6

Métodos de Colocación para problemas de evolución no lineales

Práctica 32

i) Crea el fichero tipo función **micolevolnl.m** que tomando como datos la función f , N (número de puntos de colocación interiores), T , N_t (tamaño de la discretización temporal) y la condición inicial v implemente el método de colocación ortogonal de Gauss-Lobatto para el problema de evolución

$$\begin{cases} u_t = u_{xx} + f(x, u), & x \in (-1, 1), t \in (0, T) \\ u(-1, t) = u(1, t) = 0 \\ u(x, 0) = v \end{cases}$$

Las variables de salida deben ser la discretización temporal t , los puntos de colocación x y los valores de la solución en los puntos del mallado (x, t) dados por la matriz u y la matriz P .

Para resolver la ecuación diferencial ordinaria que aparece al implementar el método de colocación utilizar el método de Euler mejorado, que viene dado por la fórmula:

$$U^{j+1} = U^j + \frac{k}{2}[F(t_j, U^j) + F(t_{j+1}, U^j + kF(t_j, U^j))]$$

donde k es el paso de la discretización temporal.

Práctica 33 Crea el fichero tipo script **testmicolevolnl.m** que ejecute **micolevolnl.m** para el problema anterior con las funciones $f(x, u) = \alpha(u - u^3)$ para distintos valores de $\alpha > 0$. Este fichero tiene que dibujar dinámicamente la solución obtenida, realizando la “película” de N_t fotografías de forma que en la fotografía k se pinte la función $u(t_k, x)$. De esta forma se ve dinámicamente la evolución del sistema desde la condición inicial dada por v hasta el estado final dado por $u(T, x)$. Para esto se puede utilizar los siguientes comandos

```
M=moviein(Nt);  
figure(1)  
for j=1:Nt  
    plot(h,w(:,j)), axis([fijar los ejes]);  
    M(:,j)=getframe;  
end
```

```
movie(M)
```

Opcionalmente se puede dibujar la solución en 3D y la solución en tiempo final

```
% dibuja en 3D  
surf(x,t,u)  
title(['Solucion numerica calculada con ', num2str(size(xmesh,2)), ' puntos de  
xlabel('Distancia x')  
ylabel('Tiempo t')  
colorbar
```

```
% El perfil de la solución en tiempo final  
figure  
plot(x,u(end,:))  
title(['Solucion en t = ', num2str(t(end))])
```