Práctica 10. Resolución de EDO con el MDF.

Métodos Numéricos para la Computación

Grado en Ingeniería Informática. Mención Computación
Escuela de Ingeniería Informática
Universidad de Las Palmas de Gran Canaria

Contenidos

- Ensamblado de problemas de contornos en EDO con el MDF
- Utilización de LAPACK para resolver EDO con el MDF.
- Comparativa entre ODE45() y MDF en problemas de inicio.

Tarea 1. Ensamblado en C/C++ y tri-diagonal en LAPACK

1. Se suministrará una clase C/C++ denominada MDF, incluyendo ficheros .h y .cpp.

2. Se explicará en el laboratorio lo esencia de esta clase que usa LAPACK para resolver el sistema tri-diagonal.

3. El alumno construirá un programa Visual Studio. Se incluirá el código fuente y cabecera de la clase y se activará MKL, con las opciones Release y x64. Utilizando la clase MDF, se resolverá el problema del Ejemplo2 de la práctica anterior.

4. Recolectar los valores del error relativo promedio en función del número de puntos, construyendo una tabla y gráfica correspondiente.

Uso de la clase

La función única de la clase es solve(), donde los argumentos son:

```
int MDF::solve(const int N, const double h,
    double *y, const double *params, const int type,
    const double ca, const double cb){
```

Entrada:

N: el número de puntos
h: el intervalo de la variable x.
params: un array de 4*N con los valores de los coeficientes.
type: con valores predefinidos para los 4 tipos de problemas de contorno.
ca y cb: los valores de las condiciones de contorno en a y b.

Salidas:

y: El array con la solución de la función.
retorno de la función: 0 si correcto. El mismo valor info de la función de LAPACK

Condiciones de contorno:
CONTOUR_TYPE1: y(a), y(b)
CONTOUR_TYPE2: y(a), y'(b)
CONTOUR_TYPE3: y'(a), y(b)
CONTOUR_TYPE4: y'(a), y'(b)

Código del ejemplo que utiliza la clase MDF para el Ejemplo 2 del Tema 3-2.

```cpp
#include <cstdio>
#include <cstdlib>
#include <cmath>

#include <mkl.h>
#include "MDF.h"

void getparams(const int N, const double *x, double *params){

    double *A = params;
    double *B = &(params[N]);
    double *C = &(params[2*N]);
    double *D = &(params[3 * N]);

    for (int i = 0; i < N; i++){
        A[i] = x[i] * x[i];
        B[i] = x[i];
        C[i] = -1.0;
        D[i] = 3.0*x[i] * x[i];
    }
}

double exactSolution(double x){

    return x + 1.0 / x + x*x;
}
```

```cpp
int main(int argc, char *argv[]){

    MDF *mdf = new MDF();

    double a = 0.5;
    double b = 2.0;
    int N = 150;
    double h = (b - a) / (double)(N - 1);

    double *x = (double *)mkl_malloc(N*sizeof(double),64);
    double *y = (double *)mkl_malloc(N*sizeof(double), 64);
    double *params = (double *)mkl_malloc(4*N*sizeof(double), 64);

    for (int i = 0; i < N; i++) x[i] = a + (double)i*h;
    getparams(N, x, params);

    int ret = mdf->solve(N, h, y, params, CONTOUR_TYPE1, exactSolution(a), exactSolution(b));

    double *yexact = (double*)mkl_malloc(N*sizeof(double), 64);
    for (int i = 0; i < N; i++) yexact[i] = exactSolution(x[i]);

    double error=0.0;
    for (int i = 0; i < N; i++) error += fabs((yexact[i] - y[i]) / yexact[i]);
    error /= N;
    printf("\nError relativo promedio(%%): %g\n",100.0*error);

    mkl_free(x);
    mkl_free(y);
    mkl_free(yexact);

    delete mdf;

    std::getchar();
    return 0;

}
```

Tarea 2. Comparativa en problemas con condiciones de inicio.

Resolveremos en MATLAB un problema de EDO de segundo orden con condiciones de inicio, utilizando ode45(). Posteriormente, e igualmente en MATLAB, resolveremos el mismo problema con MDF (lo cual no es lo más recomendable) pero con un número de puntos equivalente de puntos al número de evaluaciones intermedias que utiliza ode45(). De esa forma la comparativa tiende a ser justa.

"El método Dormand–Prince tiene siete etapas, pero solo usa seis evaluaciones de función por paso …" https://es.wikipedia.org/wiki/M%C3%A9todo_de_Dormand-Prince

La función de ensamblado de condiciones de inicio se proporciona al alumno.

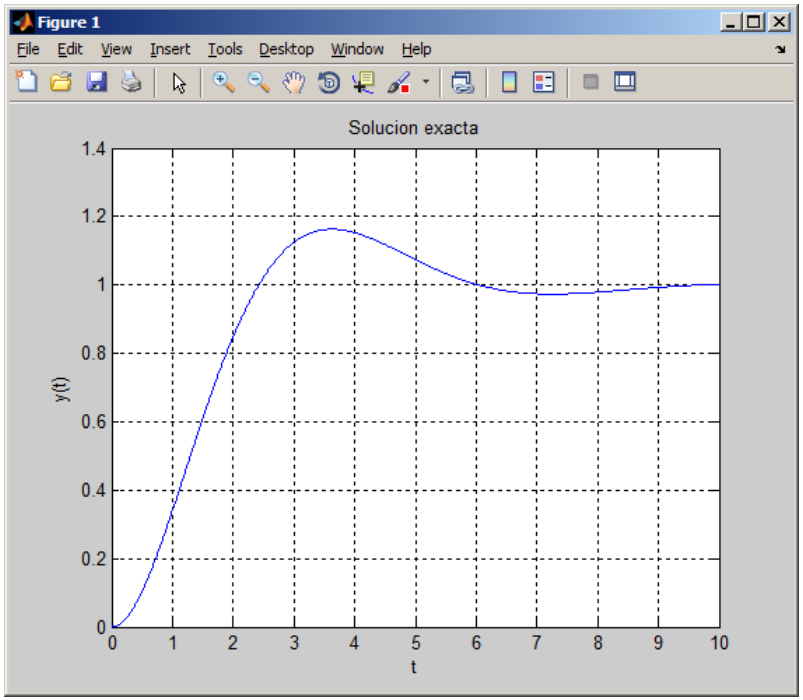
Problema que deseamos resolver

Segundo orden con condiciones de inicio.

$$\frac{d^2y}{dt^2}+\frac{dy}{dt}+y=1 \quad t\in[0,T]$$
$$y(0)=0$$
$$y'(0)=0$$

La solución analítica exacta:

$$y(t)=1-e^{-\frac{t}{2}}\left[\cos\left(\frac{\sqrt{3}}{2}t\right)+\frac{1}{\sqrt{3}}\sin\left(\frac{\sqrt{3}}{2}t\right)\right]$$



Formulado como sistema de EDO

$$y_1=y \qquad \frac{dy_1}{dt}=y_2 \qquad y_1(0)=y(0)$$
$$y_2=y' \qquad \frac{dy_2}{dt}=1-y_1-y_2 \qquad y_2(0)=y'(0)$$

```
function  comparaEDO_MDF

    T =5;
    % muchos puntos para dibujo preciso de la solución exacta
    tplot = linspace(0,T,200);
    yplot=SolucionExacta(tplot);

    figure;
    hold on;
    plot(tplot,yplot,'r-');

    % Solucion con ODE45()
    [tode,Yode]=ode45(@sistemaODE,[0,T],[0,0]);
    yode = Yode(:,1);
    plot(tode,yode,'b-');
    y1 = SolucionExacta(tode); % solucuión ecata en los puntos de ODE45
    % se evitan los primeros terminos dado que son casi nulos
    % al ser error relativo implica dividir por casi cero.
    errorODE = 100*mean(abs(y1(5:end)-yode(5:end))./y1(5:end));

    % Solucion con MDF con 6 veces el número de puntos
    N = length(tode);
    [tMDF,yMDF] = inicioMDF(0,T,6*N,0,0,@terminosMDF,[]);
    plot(tMDF,yMDF,'k-');
    % solución ezacta para los puntos de MDF
    y2 = SolucionExacta(tMDF);
    % se evitan 6 veces los primeros terminos
    errorMDF = 100*mean(abs(y2(5*6:end)-yMDF(5*6:end))./y2(5*6:end));


    grid on;
    xlabel('t');
    ylabel('y(t)');
    h=legend('Exacta','ode45','MDF');
    set(h,'Location','SouthEast');
    title(sprintf('Error ODE(%%): %g, Error MDF(%%): %g',errorODE,errorMDF));
    hold off;

end
```

EL código que compara la resolución mediante ODE45 y mediante inicioMDF() una función MATLAB para este problema.

Código solución exacta y sistema

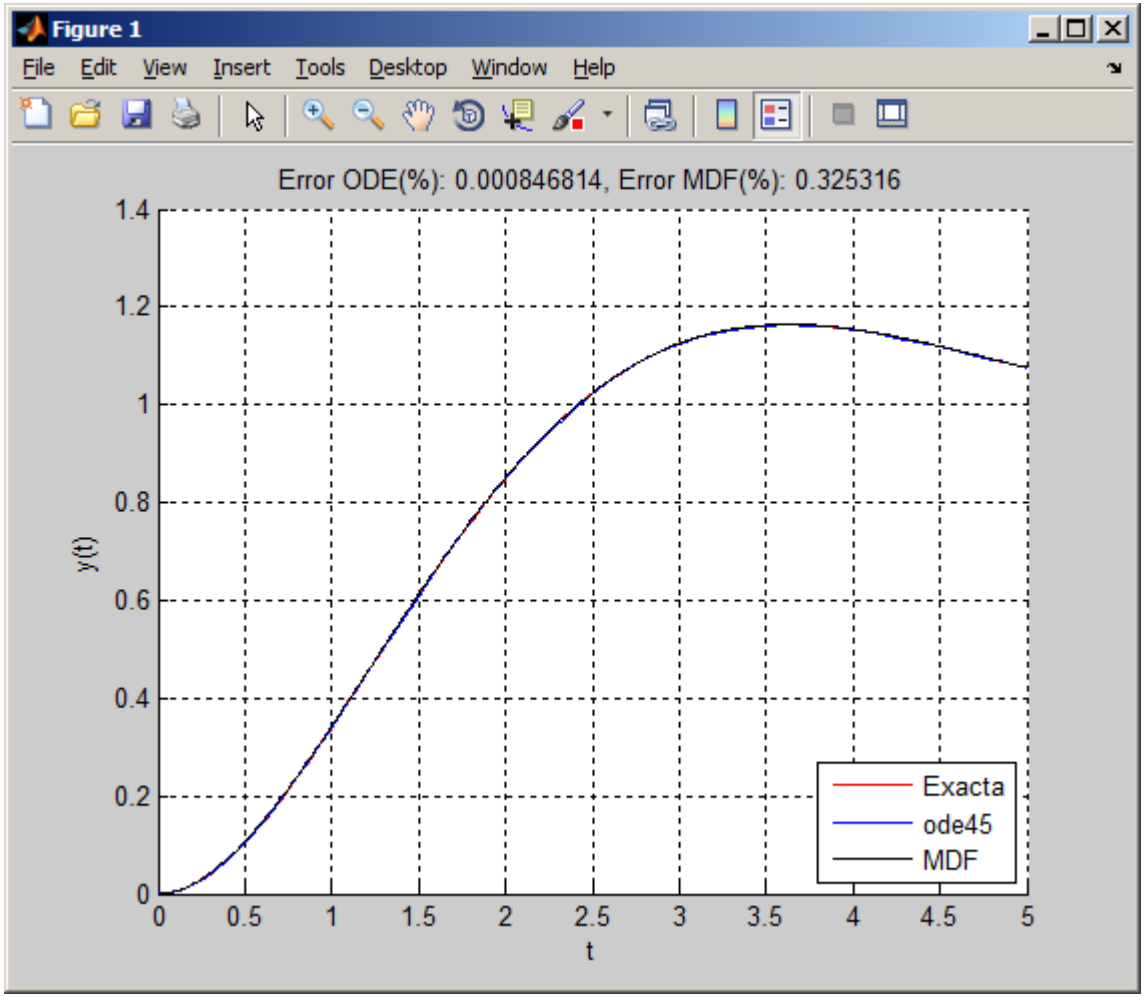
```
% solución analítica de la EDO
function y = SolucionExacta(t)
    y = 1 - exp(-t/2).*(cos((sqrt(3)/2)*t)+(1/sqrt(3))*sin((sqrt(3)/2)*t));
end

% codificación como sistema para ODE
function dy = sistemaODE(t,y)

    dy=zeros(2,1);
    dy(1,1) = y(2);
    dy(2,1) = 1-y(1)-y(2);
end

% cálculo de los coeficientes para MDF
function P = terminosMDF(x,params)

    P = ones(4,length(x));
end
```

```
function [x,y] = inicioMDF(a,b,N,ya,y1a,func,params)
% Metodo de Diferencias Finitas para problemas de contornos
% ULPGC, EII, MNC

    % espaciado constante
    x = linspace(a,b,N);
    h = (b-a)/(N-1);

    % función de cálculo de los coeficientes
    P = func(x,params);
    A = P(1,:);
    B = P(2,:);
    C = P(3,:);
    D = P(4,:);

    % factores E,F, G y H
    E = A;
    F = h*h*C - h*B - 2*A;
    G = h*B + A;
    H = h*h*D;

    % Ensamblado de RHS
    RHS = zeros(N,1);
    RHS(1)   = ya;
    RHS(2)   = h*y1a;
    RHS(2:N) = H(1:N-1)';

    % Ensamblado de LHS
    LHS = zeros(N,N);
    LHS(1,1) = 1;
    LHS(2,1) = -1;
    LHS(2,2) = 1;
    for a=3:N
        LHS(a,a-2)= E(a-2);
        LHS(a,a-1) = F(a-2);
        LHS(a,a) = G(a-2);
    end

    %disp(LHS);
    %disp(RHS);

    % si el determinante es nulo o muy pequeño,
    % el sistema es no compatible -> no hay solución
    deter = det(LHS);
    if abs(deter) < 1.0e-6
        error('El sistema no tiene solución');
    end
    X = LHS\RHS;
    y = X';
end
```

Ensamblado del MDF para un problema de condiciones de inicio.

NOTA: No es una buena idea resolver problemas de inicio con el MDF.

Comparación de la Solución



El error cometido por ODE es ínfimo si se compara con MDF.

Qué debe entregar el alumno

- Cada alumno entregará en el Campus Virtual una memoria en PDF o Word en la que estará contenida una descripción del trabajo realizado, incluyendo descripción, el listado MATLAB o C de la actividad realizada y la captura de pantalla de las gráficas o imágenes generadas. Para autentificar las imágenes cuando sea posible el alumno incluirá su nombre en cada imagen mediante la función title().
- En principio la tarea quedará abierta para su entrega hasta cierta fecha que se indicará.
- Se puede trabajar en grupo en el Laboratorio, pero la memoria elaborada y entregada será individual.

Bibliografía

1. Finite Difference Methods for Ordinary and Partial Differential Equations, R. J. LeVeque, SIAM, 2007
2. Introduction to Numerical Methods in Differential Equations, M.H. Holmes, Springer, 2000.
3. Tutorial de ODE: http://www.math.tamu.edu/reu/comp/matode.pdf

4. Ejemplos MATLAB

http://www.sc.ehu.es/sbweb/energias-renovables/MATLAB/numerico/diferencial/diferencial.html

http://www.sc.ehu.es/sbweb/energias-renovables/MATLAB/numerico/diferencial/diferencial_1.html

Scholar http://www.scholarpedia.org/article/Finite_difference_method