VI CONGRESSO DE INICIAÇÃO CIENTÍFICA DA UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

PIBIC/CNPq/UFCG-2009

METODOS NUMÉRICOS PARA EQUAÇÕES DIFERENCIAIS ORDINÁRIAS

Rivaldo Bezerra de Aquino Filho[1], Aparecido Jesuino de Souza[2]

RESUMO

Nesse artigo tratamos de métodos numéricos básicos para Equações Diferenciais Ordinárias. Com o auxilio do sofware Matlab resolvemos numericamente problemas de valor inicial e de contorno. Também foi feita uma análise de erro do método conhecido como Método de Euler Explícito.

Palavras-chave: métodos numéricos, equações diferenciais ordinárias, problemas de contorno.

NUMERICAL METHODS FOR ORDINARY DIFFERENTIAL EQUATIONS

ABSTRACT

In this article we discuss some basic numerical methods for Ordinary Differential Equations. With the aid of the Matlab software we solve numerically some initial and boundary value problems. We have also made the error analysis of the Explicit Euler method.

Keywords: numerical methods, ordinary differential equation, boundary value problems.

INTRODUÇÃO

Métodos numéricos são ferramentas poderosas para se determinar, exata ou aproximadamente, soluções numéricas de problemas modelados matematicamente.

Nesse trabalho fizemos uso dessas ferramentas para determinar numericamente a solução de alguns problemas de valor inicial (PVI) e de contorno (PVC). Para o desenvolvimento da teoria usamos como referências os livros (CUMINATO & MENEGUETTE, 2006; CONTE, 1977). As implementações dos métodos foram feitas usando o software MATLAB e para o melhor uso desse instrumento nos baseamos em (MATSUMOTO, 2001).

Também fizemos uma breve análise de erro para o método conhecido como Euler explícito. Nessa análise nos baseamos exclusivamente no livro (CONTE,1977).

MATERIAL E MÉTODOS

Este trabalho foi desenvolvido no Laboratório de Informática da Unidade Acadêmica de Matemática e Estatística (LIDME).

Material

Foi utilizado o software MATLAB.

1. PROBLEMA DE VALOR INICIAL EM EQUAÇÕES ORDINÁRIAS

Uma equação diferencial na qual a variável dependente é função de apenas uma variável é dita equação diferencial ordinária.

[1] Aluno do Curso de Matemática, Unidade Acadêmica de Matemática e Estatística, UFCG, Campina Gande , PB, E-mail: rivaldo.mat@gmail.com

[2] Prof. Doutor, Unidade Acadêmica de Matemática e Estatística, UFCG, Campina Grande, PB, E-mail: cido@dme.ufcg.edu.br

Um problema de valor inicial (PVI) é um problema de evolução, no qual a informação inicial (conhecida) é propagada para o interior do domínio unidimensional pela equação diferencial.

Matematicamente, o mais simples dos problemas de valor inicial pode ser apresentado na forma:

$$\begin{cases} y' = f(x,y) \\ y(a) = \alpha, \end{cases} \quad (1.1)$$

onde $f : IR^2 \to IR$ é uma função contínua. A função $y = y(x)$ $(x \geq a)$ é a função incógnita e α é o seu valor inicial no ponto a.

Um ponto importante a ser citado é a questão da *existência* e *unicidade* de soluções de problema de valor inicial para uma equação diferencial, visto que só faz sentido buscar a sua solução numérica, ou solução aproximada, se de antemão estiver garantida a existência e unicidade de sua solução, pois ao determinar uma solução numérica de um problema que a princípio não possua solução ou que possua, mas não seja única, o processo numérico poderá não convergir, ou convergir para algo que não seja confiável. No caso de um problema de valor inicial de equações ordinárias esta questão está bem resolvida.

Suponhamos que a função $f(x,y)$ satisfaça as seguintes condições:

- $f : E \to IR^n$ é contínua, onde $E = \{(x,y), x \in [a,b], y \in IR^n\}$;
- Existe uma constante K tal que para todo $x \in [a,b]$ e quais quer dois valores y e y^* em IR^n

$$\left| f(x,y) - f(x,y^*) \right| \leq K \left| y - y^* \right|$$

Teorema 1.1: Seja $f(x,y)$ satisfazendo as condições anteriores e seja α um vetor dado. Então, existe exatamente uma função $y(x)$ com as seguintes propriedades:

i. $y = y(x)$ é contínua e diferenciável para x em $[a,b]$;

ii. $y'(x) = f(x, y(x))$, $x \in [a,b]$;

iii. $y(a) = \alpha$.

Vamos tratar agora de um método conhecido por Método de Diferenças Finitas. O método de diferenças finitas é a discretização do domínio e a substituição das derivadas presentes na equação diferencial, por aproximações destas envolvendo somente valores numéricos da função. Na prática substitui-se as derivadas pela razão incremental que converge para o valor da derivada quando o incremento tende a zero. Dizemos então que o problema foi discretizado.

Vamos utilizar como ferramenta matemática no cálculo de aproximações para as derivadas a série de Taylor que relaciona valores da função e suas derivadas num ponto x com valores dessa mesma função numa vizinhança de x, ou seja, em $(x \pm h)$ com $h > 0$. Se $y(x)$, tem derivadas de ordem $n+1$, em x, podemos escrever :

$$\begin{aligned} y(x+h) \;=\; & y(x) + hy'(x) + \frac{h^2}{2!}y''(x) + \ldots + \frac{h^n}{n!}y^{(n)}(x) + \\ & + \frac{h^{(n+1)}}{(n+1)!}y^{(n+1)}(\xi), \qquad x < \xi < x+h. \qquad (1.2). \end{aligned}$$

O último termo da expressão representa (1.2) representa o erro da aproximação de $y(x+h)$ pelo polinômio de Taylor de ordem n :

$$P_n(h) = y(x) + y'(x)h + \frac{y''(x)}{2!}h^2 + \ldots + \frac{y^{(n)}(x)}{n!}h^n.$$

Se $n = 1$ em (1.2) obtemos uma aproximação da derivada $y'(x)$, conhecida como *fórmula progressiva*, que é dada por :

$$y'(x) = \frac{y(x+h) - y(x)}{h} - \frac{h}{2}y''(\xi).$$

De modo semelhante, tomando $-h$ em (1.2), ainda com $n=1$, obtemos a *fórmula regressiva*, que é dada por :

$$y'(x)=\frac{y(x)-y(x-h)}{h}+\frac{h}{2}y''(\xi).$$

Tomando $n=2$ em (1.2), e reescrevendo (1.2) para h e $-h$, respectivamente, obtemos :

$$y(x+h)=y(x)+hy'(x)+\frac{h^2}{2!}y''(x)+\frac{h^3}{3!}y'''(\xi_1)$$

e

$$y(x-h)=y(x)-hy'(x)+\frac{h^2}{2!}y''(x)-\frac{h^3}{3!}y'''(\xi_2).$$

Subtraindo a última expressão da penúltima obtemos a fórmula centrada ou fórmula de diferenças centradas:

$$y'(x)=\frac{y(x+h)-y(x-h)}{2h}-\frac{h^3}{3!}\left(y'''(\xi_1)+y'''(\xi_2)\right).$$

O primeiro método para solução aproximada de PVI´s que trataremos é o Método de Euler Explícito que utiliza como base a fórmula progressiva para a aproximação da derivada primeira. O primeiro passo é dividir o intervalo $[a,b]$ em N subintervalos iguais, cada um de comprimento h, definindo uma malha $R_h=\{x_i=a+(i-1)h,\ i=1,...,N+1\}$. Com $h=\frac{(b-a)}{N}$.

Sejam $y_i\approx y(x_i)$ aproximações para $y(x_i)$, $i=2,\dots,N+1$ e $y_1=\alpha$. Em cada um dos pontos $x_2,x_3,\dots,x_{N+1}$, aproximamos a equação diferencial (1.1) por :

$$f(x_i,y_i)=\frac{y_{i+1}-y_i}{h},\qquad i=1,\dots,N$$

ou,

$$y_{i+1}=y_i+hf(x_i,y_i),\qquad i=1,\dots,N$$

Perceba que o valor de y_{i+1} é calculado explicitamente em função de y_i, mesmo quando a função f é não linear.

Quando aproximamos a derivada primeira em (1.1) por diferenças regressivas obtemos:

$$f(x_i,y_i)=\frac{y_i-y_{i-1}}{h},\qquad i=2,\dots,N+1 \text{ ou } y_{i+1}=y_i+hf(x_{i+1},y_{i+1}),\qquad i=1,\dots,N.$$

Esse Método é chamado Método de Euler Implícito. Note que se f for uma função não linear teremos que o vetor y_{i+1} será definido implicitamente e, portanto para a sua obtenção deve-se lançar mão de método eficiente para a solução de sistemas de equações algébricas não lineares, como por exemplo o Método de Newton. A comparação entre um método explícito e outro implícito é que em geral o método implícito é mais estável, mas exige um passo h menor para ser obter uma boa aproximação da solução.

Vamos mostrar os Métodos de Euler Explícito e Implícito para resolver numericamente o PVI da forma (1.1) :

Método de Euler Explícito:

Programa Principal: clear % Apaga os dados anteriores; clc % Apaga os comandos da tela; p = 1; % Parâmetro para utilizar os dados iniciais; [a,b,h,y(1)] = dados(p); % Dados iniciais; kmax = (b-a)/h; % Malha y(1);	dados(p) % Dados Iniciais: function [a,b,h,yi] = dados(p) a ; % Extremo inferior de onde x esta definido; b ; % Extremo superior de onde x esta definido; h ; % Passo da malha; yi ;% Dado inicial do PVI;

```
k = 1;
x(1) = a;
while k < (kmax+h)
        y(k+1) = y(k) + h*efe(x(k),y(k));
        x(k+1) = x(k) + h;
        k = k+1;
end
% Gráfico da Solução aproximada;
plot(x,y,'b:*')%Plota o Gráfico ;
```

Função do PVI:

```
function f = efe(x,y)
f ;
```

Método de Euler Implícito:

Programa Principal:

```
clear; % Apaga os dados anteriores;
clc; % Apaga os comandos da tela;
p = 1; % Parâmetro para utilizar os dados iniciais;
[a,b,h,y(1),tol,inte] = dados(p); % Dados iniciais;
kmax = (b-a)/h; % Malha
k=1;
x(1) = a; % primeiro valor de x
% y(1);
while k < (kmax+h) % numero de pontos a ser
calculados
   x(k+1) = x(k) + h;
   %Método de Euler implícito
   y(k+1) =    i   ton(x(k), y(k), x(k+1), h, tol, inte);
   k = k+1;
end
% Gráfico da solução aproximada
plot(x,y, 'r:+');
```

dados(p):

```
% Dados Iniciais:
function [a,b,h,yi,tol,inte]=dados(p)
a = 0; % Extremo inferior de onde x esta definido;
b = 1; % Extremo superior de onde x esta definido;
h = 0.1; % Passo da malha;
yi = 1;% Dado inicial do PVI;
tol = 0.00001; % Tolerância do erro no Método de
Newton;
inte = 10; % numero de iterações máximas
```

Método de Newton:

```
function raiz = newton(xn, yn, xnmais1, h, tol, inte)
inti = 1; % Parâmetro para o numero de iterações;
% Chute inicial para o método de Newton usando o
método de Euler explicito;
yin = yn + h*efe(xn, yn);
yf = yin;
erro=2*tol; % Parâmetro para o erro;
while (abs(erro)>tol) & (inti<inte) % Requisitos para
o Método de Newton (erro e iterações);
   % Método de Newton
   yf = yin-((fnewton(yf, yn, xnmais1, h))/dfnewton(yf,
yn, xnmais1, h));
   erro = (yf-yin);
   yin = yf;
   inti =i nti+1;
end
raiz = yf; % Parâmetro de saída
```

Função do Método de Newton:

```
function saida = fnewton(z, yn, xnmais1, h)
saída = z – yn – h*efe(xnmais1, z); % z incógnita
```

Derivada da Função do Método de Newton na variável z:

```
function saida=dfnewton(z, yn, xnmais1, h) % z
incognita
```

Função do PVI:

```
function f=efe(x,y)
f ;
```

saida = 1 - h*defe(xnmais1,z);	
Derivada da função do PVI em relação a variável y: function df = defe(x,y) df ;	

Vamos ilustrar os resultados obtidos pelos Métodos de Euler Explícito e Implícito para resolver numericamente o seguinte PVI:

$$\begin{cases} y' = x - y \\ y(0) = 1 \end{cases}. \qquad (1.3)$$

A solução exata do PVI é $y(x) = x - 1 + 2e^{-x}$, que por sua vez foi utilizada na comparação da eficiência dos métodos para x entre 0 e 1 e com h = 0,1. Veja comparação na tabela:

Solução exata:

x	0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
y	1,0	0,9097	0,8375	0,7816	0,7406	0,7131	0,6976	0,6932	0,6987	0,7131	0,7358

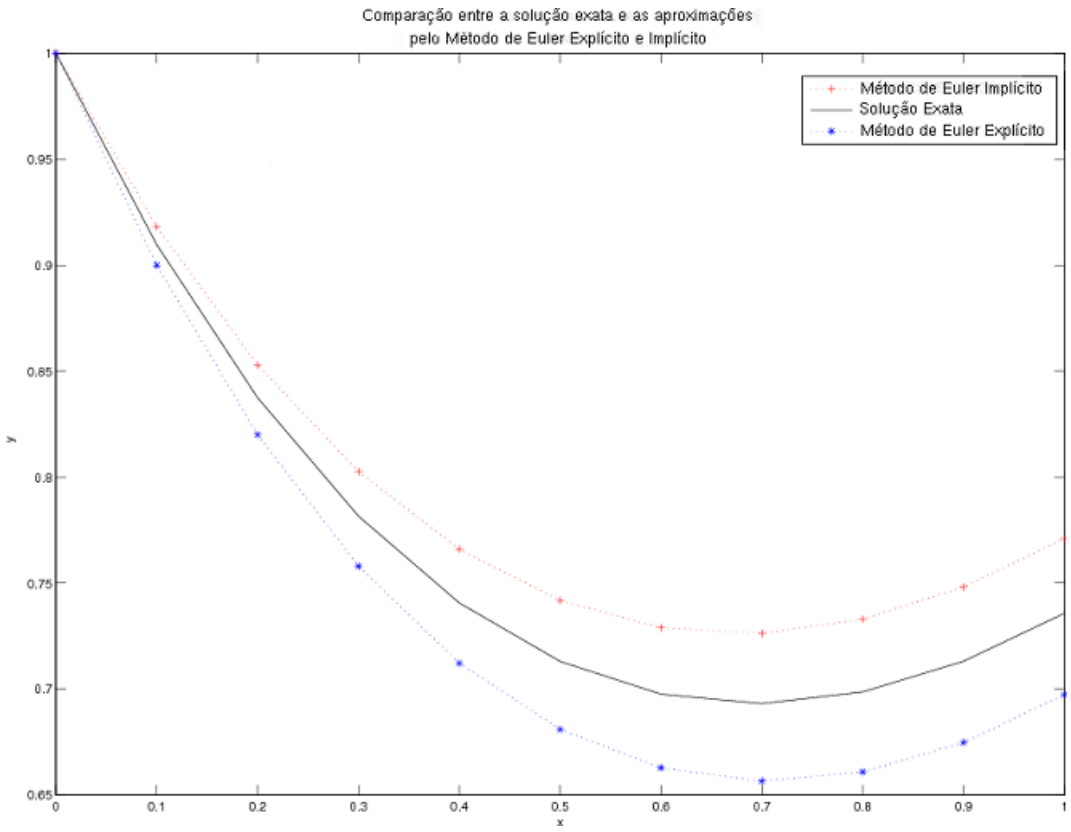
Solução Numérica: Método de Euler Explícito

x	0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
y	1,0	0,9000	0,8200	0,7580	0,7122	0,6810	0,6629	0,6566	0,6609	0,6748	0,6974

Solução Numérica: Método de Euler Implícito

x	0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
y	1,0	0,9182	0,8529	0,8026	0,7660	0,7418	0,7289	0,7263	0,7330	0,7482	0,7711

Graficamente obtivemos :



Também usamos o método das diferenças centradas para obter a solução numérica do PVI (1.3).

Método das Diferenças Centradas:

```
clear; % Apaga os dados anteriores;
clc; % Apaga os comandos da tela;
p=1; % Parametro para utilizar os dados iniciais;
[a,b,h,y1,]=dados(p); % Dados iniciais;
kmax = (b-a)/h; % Malha

x(1) = a; % primeiro valor de x
y(1) = y1; % dado inicial
% primeiro passo de inicializacao
y(2)=y(1)+h*efe(x(1),y(1)); % por euler explicito
k=2;
x(2)=x(1)+h;
while k < (kmax+h) % numero de pontos a ser
calculados
    x(k+1) = x(k) + h;
    %Metodo Euler Centrado
    y(k+1)=y(k-1)+2*h*efe(x(k),y(k));
    k=k+1;
end
f% Grafico da solucao aproximada
plot(x,y, 'b');
```

```
dados(p)
% Dados Iniciais:
function [a,b,h,yi] = dados(p)
a ; % Extremo inferior de onde x esta definido;
b ; % Extremo superior de onde x esta definido;
h ; % Passo da malha;
yi ;% Dado inicial do PVI;
```

```
function w=eulerex(a,b,h,yi)
% Metodo de Euler Explicito
p = 1;
[a,b,h,w(1),tol,inte] = dados(p);
kmax = (b-a)/h;
k = 1;
```

Função do PVI:

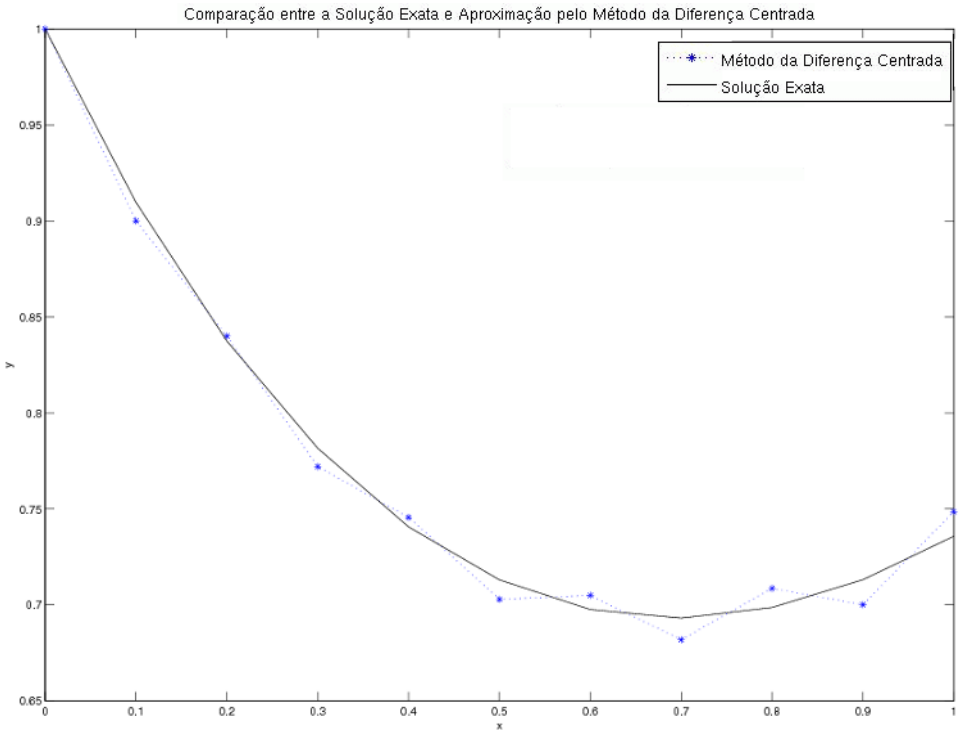
```
function f = efe(x,y)
f ;
```

```
x(1) = a;
while k < (kmax+h)
      w(k+1)=w(k)+h*efe(x(k),w(k));
      x(k+1)=x(k)+h;
      k=k+1;
end
```

Os resultados com esse método foram:

x	0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
y	1,0	0,9000	0,8200	0,7580	0,7122	0,6810	0,6629	0,6566	0,6609	0,6748	0,6974

Graficamente obtivemos:



Outros dois métodos estudados para solução numérica de PVI's foram os métodos dos Trapézios e o de Runge-Kutta.

Comparando o Método de Euler Explícito e Implícito com a solução exata, observamos que enquanto o Método Explícito erra por falta o Método Implícito erra por excesso. O Método dos Trapézios consiste em uma combinação linear (média aritmética) dos métodos de Euler Explícito e Implícito, afim de que os erros se cancelem e se tenha, portanto uma melhor aproximação.

Equação de diferenças pelo Método dos Trapézios:

$$y_{i+1} - y_i = \frac{h}{2}(f(x_i, y_i) + f(x_{i+1}, y_{i+1}))$$
$$i = 1, 2, ..., N$$

O Método dos Trapézios é implícito, dessa forma utilizamos duas formas de aproximar a solução do PVI utilizando esse método. A primeira forma foi pelo Método de Newton e a segunda foi utilizando iterações pelo Método de Euler Explícito.

Agora mostraremos as duas formas usadas para encontrar a solução numérica do PVI (1.1) pelo Método dos Trapézios.

Método dos Trapézios (Interativo pelo Método de Euler Explícito):

Programa Principal: Método dos Trapézios	dados(p)

```
clear; % Apaga os dados anteriores;
clc; % Apaga os comandos da tela;
p = 1; % Parâmetro para utilizar os dados iniciais;
[a,b,h,y(1)] = dados(p); % Dados iniciais;
kmax = (b-a)/h; % Malha
k = 1;
x(1) = a; % primeiro valor de x
while k < (kmax+h) % numero de pontos a ser
calculados
    x(k+1) = x(k) + h;
    %Método do Trapézio por Euler Interativo
        y(k+1) = teuler(x(k), y(k), x(k+1), h, tol, inte);
    k = k+1;
end
figure(1)
plot(x,y,'g') %Gráfico da solução aproximada
grid on
hold
z = exp(-x.*x); %Solução Exata (Gaussiana)
plot(x,z,'b');
e = abs(z-y); % Erro entre a solução exata e a
numérica
figure(2)
plot(x,e,'g') % Gráfico do erro
```

```
% Dados Iniciais:
function [a,b,h,yi]=dados(p)
a ; % Extremo inferior de onde x esta definido;
b ; % Extremo superior de onde x esta definido;
h ; % Passo da malha;
yi ;  % Dado inicial do PVI;
```

Método Interativo:

```
function raiz = teuler(xn, yn, xnmais1, h, tol, inte)
yin = yn + h*efe(xn, yn);% Chute inicial para o
método do Trapézio usando o método de Euler
explícito;
erro = 2*tol;
inti = 1; % Parâmetro para o numero de iterações;
while (abs(erro)>tol) & (inti<inte);% Requisitos para
o Método do Trapézio utilizando as iterações de
Euler explicito (erro e iterações);

% Método Interativo usando Euler Explicito
    y f = yn + (h/2)*(efe(xn,yn) + efe(xnmais1,yin));
erro = (yf-yin);
    yin = yf;
    inti = inti+1;
end
raiz=yf;
```

Função do PVI:

```
function f=efe(x,y)
f ;
```

Método dos Trapézios (Método de Newton):

Programa Principal: Método dos Trapézios:

```
clear; % Apaga os dados anteriores;
clc; % Apaga os comandos da tela;
p = 1; % Parâmetro para utilizar os dados iniciais;
[a,b,h,y(1),tol,inte] = dados(p); % Dados iniciais;
kmax = (b-a)/h; % Malha
k = 1;
x(1) = a; % primeiro valor de x
while k < (kmax+h) % numero de pontos a ser
calculados
    x(k+1) = x(k) + h;
    %Método do Trapézio pelo Método de Newton
        y(k+1) = tnewton(x(k), y(k), x(k+1), h, tol, inte);
    k = k+1;
```

dados(p)

```
% Dados Iniciais:
function [a,b,h,yi,tol,inte] = dados(p)
a ; % Extremo inferior de onde x esta definido;
b ; % Extremo superior de onde x esta definido;
h ; % Passo da malha;
yi ;  % Dado inicial do PVI;
tol ; % Tolerância do erro no Método de Newton;
inte ; % numero de iterações máximas
```

```
end
figure(1)
plot(x,y,'g') %Gráfico da solução aproximada
grid on
hold
z = exp(-x.*x); %Solução Exata (Gaussiana)
plot(x,z,'b');
e = abs(z-y); % Erro entre a solução exata e a
numérica
figure(2)
plot(x,e,'g') % Gráfico do erro
```

Método de Newton

```
function raiz = tnewton(xn, yn, xnmais1, h, tol, inte)
% Chute inicial para o método de Newton usando o
método de Euler explicito;
yin = yn + h*efe(xn, yn);
yf = yin;
erro = 2*tol;
inti = 1; % Parâmetro para o numero de iterações;
while (abs(erro)>tol) & (inti<inte)% Requisitos para o
Método de Newton (erro e iterações);
%Método de Newton
yf = yin - ((tfnewton(xn,yf, yn, xnmais1,
h))/tdfnewton(xn,yf, yn, xnmais1, h));
    erro = (yf-yin);
    yin = yf;
    inti = inti + 1;
end
raiz=yf;
```

Funcao do Metodo de Newton para regra dos trapézios:

```
% y_(i+1)-y_(i)=(h/2)*(f((x_(i),y_(i))+f(x_(i+1),y_(i+1))
function saida = tfnewton(xn, z, yn, xnmais1, h)
saida = z - yn - (h/2)*(efe(xn,yn) + efe(xnmais1, z));
```

Derivada da funcao do metodo de newton na variavel z:

```
function saida=tdfnewton(xn,z, yn, xnmais1, h) % z
é incógnita
saida = 1 - h*(defe(xn,yn)+defe(xnmais1,z));
```

Função do PVI:

```
function f=efe(x,y)
f ;
```

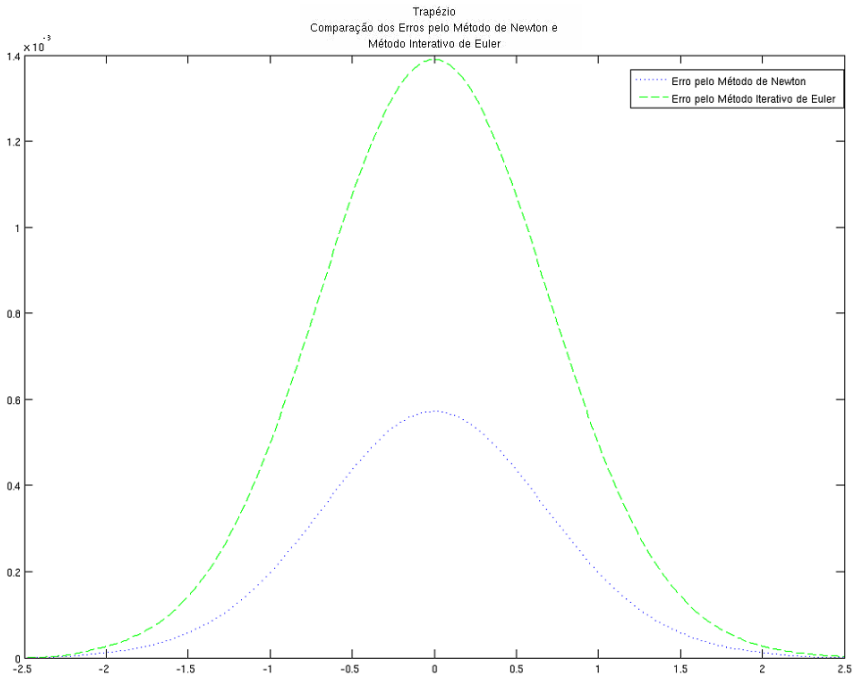
Derivada da função do PVI em relação a y

```
function df = defe(x,y)
df ;
```

Com esse método comparamos as soluções numéricas com a solução exata do seguinte PVI :

$$\begin{cases} y' = -2xy \\ y(-2{,}5) = -6{,}25 \end{cases} \qquad (1.4)$$

No qual tem como solução exata: $y = \exp(-x^2)$. Assim foi gerado o seguinte gráfico de erro:



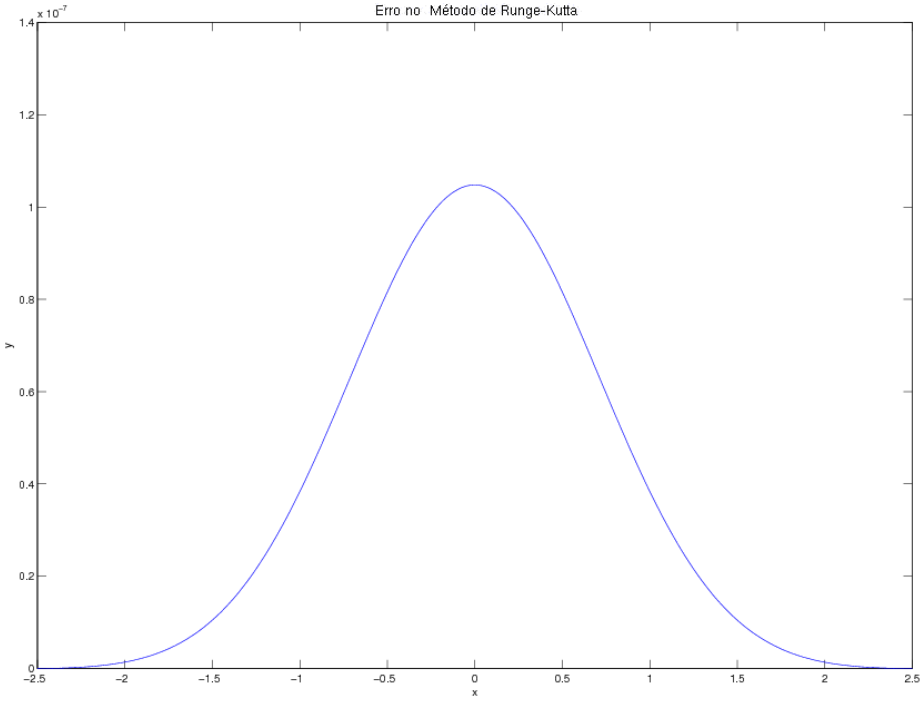
O último método estudado para a solução numérica de PVI's foi o Método de Runge-Kutta de Quarta Ordem. A vantagem deste método, assim como outros métodos de Runge-Kutta, é que além de fornecer uma precisão muito boa são explícitos. Ele está baseado na seguinte fórmula de recorrência:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4).$$

onde

$$k_1 = hf(x_n, y_n);$$
$$k_2 = hf(x_n + \frac{h}{2}, y_n + \frac{1}{2}k_1);$$
$$k_3 = hf(x_n + \frac{h}{2}, y_n + \frac{1}{2}k_2);$$
$$k_4 = hf(x_n + h, y_n + k_3).$$

Esse método também foi utilizado na implementação do PVI (1.4). O erro entre a solução numérica e a solução exata pode ser verificado a seguir:



Mostraremos agora o programa do Método de Runge-Kutta:

Programa Principal:

```
clear; % Apaga os dados anteriores;
clc; % Apaga os comandos da tela;
p=1; % Parâmetro para utilizar os dados iniciais;
[a,b,h,y1] = dados(p); % Dados iniciais;

kmax = (b-a)/h; % Malha
k = 1;
x(1) = a; % primeiro valor de x
y(1) = y1; % dado inicial
while k < (kmax+h) % numero de pontos a ser
calculados
    x(k+1) = x(k) + h;
    %Método de Runge Kutta
   y(k+1) = y(k)+(1/6)*(frugen1(x(k),y(k),h) +
2*frugen2(x(k),y(k),h) + 2*frugen3(x(k),y(k),h) +
frugen4(x(k),y(k),h));

    k=k+1;
end
plot(x,y, 'r'); % Gráfico da solução aproximada
hold
z = exp(-x.*x);
erro = abs(z-y);
plot(x,erro,'b') % Gráfico do erro
```

dados(p)

```
% Dados Iniciais:
function [a,b,h,yi] = dados(p)
a = -2.5; % Extremo inferior de onde x esta definido;
b = 2.5; % Extremo superior de onde x esta definido;
h = 0.01; % Passo da malha;
yi = exp(-6.25);  % Dado inicial do PVI;
```

Função 1:

```
function f1 = frugen1(x,y,h)
f1 = h*efe(x,y);
```

Função 2:

```
function f2 = frugen2(x,y,h)
f1 = frugen1(x,y,h);
f2 = h*efe(x + (h/2),y + 0.5*f1);
```

Função 3:

```
function f3 = frugen3(x,y,h)
f2 = frugen2(x,y,h);
```

Função 4:

```
function f4 = frugen4(x,y,h)
f3 = frugen3(x,y,h);
```

f3 = h*efe(x + (h/2),y + 0.5*f2);	f4 = h*efe(x + h,y + f3);

2. Estimativas de Erro e Convergência do Método de Euler

Por simplicidade decidimos estudar o erro gerado pelo método de Euler Explícito.

Desejamos estimar a grandeza do erro devido ao *truncamento* e_n que é definido por:

$$e_n = y_n - y(x_n). \qquad (2.1)$$

Onde $y(x_n)$ é a verdadeira solução da equação diferencial em x_n e y_n é a solução aproximada através do Método.

Iremos considerar y_1 exato e, portanto $e_1 = 0$.

Admitindo que existam as derivadas apropriadas, podemos expandir $y(x_{n+1})$ para $x = x_n$ usando o Teorema de Taylor com resto:

$$y(x_{n+1}) = y(x_n) + hy'(x_n) + \frac{h^2}{2} y''(\xi_n), \quad x_n \le \xi_n \le x_{n+1} \qquad (2.2)$$

A quantidade $\frac{h^2}{2} y''(\xi_n)$ é denominada o erro de truncamento local, o erro cometido na única etapa de x_n para x_{n+1}, admitindo que y e y' sejam conhecidos,exatamente em $x = x_n$.

Haverá também um erro de arredondamento ao ser calculado y_{n+1} através do computador, usando a fórmula:

$$y_{n+1} = y_n + hf(x_n, y_n). \qquad (2.3)$$

Subtraindo (2.3) de (2.2) e usando (2.1), temos:

$$e_{n+1} = e_n + h[f(x_n, y_n) - f(x_n, y(x_n))] + \\ - \frac{h^2}{2} y''(\xi_n). \qquad (2.4)$$

Pelo Teorema do Valor Médio existe $\overline{y_n} \in (y(x_n), y_n)$, tal que

$$f(x_n, y_n) - f(x_n, y(x_n)) = f_y(x_n, \overline{y_n}) \cdot (y_n - y(x_n)) \\ = f_y(x_n, \overline{y_n})\, e_n.$$

Onde segue de (2.4):

$$e_{n+1} = e_n + h \cdot f_y(x_n, \overline{y_n}) \cdot e_n - \frac{h^2}{2} y''(\xi_n).$$

Vamos supor que ao longo do intervalo $(y(x_n), y_n)$ temos:

$$\left| f_y(x,y) \right| < L \text{ e } \left| y'' \right| < K,$$

onde L e K são constantes positivas.

Assim temos,

$$\left| e_{n+1} \right| \le \left| e_n \right| + h\,L \left| e_n \right| + \frac{h^2}{2} K = (1 + h\,L) \left| e_n \right| + \frac{h^2}{2} K. \qquad (2.5)$$

Mostraremos agora, por indução, que a solução da equação diferença:

$$\xi_{n+1} = (1 + h\,L)\xi_n + \frac{h^2}{2}K, \qquad (2.4)$$

com $\xi_1 = 0$ supera a solução ; ou seja

$$\xi_n \geq \left| e_n \right|.$$

Note que, para $n = 1$ temos $e_1 = 0 = \xi_1$.

Seja $\xi_n \geq \left| e_n \right|$. Temos então

$$\left| e_{n+1} \right| \leq (1 + h\,L)\left| e_n \right| + \frac{h^2}{2}K \leq (1 + h\,L)\xi_n + \frac{h^2}{2}K = \xi_{n+1},$$

em resumo $\xi_{n+1} \geq \left| e_{n+1} \right|$.

Temos que

$$\xi_{n+1} = (1 + h\,L)\xi_n + \frac{h^2}{2}K \;\Rightarrow\; \xi_{n+1} - (1 + h\,L)\xi_n = \frac{h^2}{2}K.$$

Pela teoria das equações diferenças (CONTE, 1977), temos que a solução da equação anterior será:

$$\xi_n = \frac{h\,K}{2L}(1 + h\,L)^n - \frac{h\,K}{2L}. \qquad (2.7)$$

Do cálculo diferencial básico sabemos da aproximação de Mac-Laurin que

$$\exp(h\,L) = 1 + h\,L + \frac{h^2 L}{2!} + \frac{h^3 L}{3!}...$$

Segue-se, portanto, que $1 + h\,L \leq \exp(h\,L)$ e, também, que $(1 + h\,L)^n \leq \exp(n\,h\,L)$.Usando esta expressão em (2.7) concluímos que

$$\xi_n \leq \frac{h\,K}{2L}[\exp(n\,h\,L) - 1] = \frac{h\,K}{2L}[\exp((x_{n+1} - x_1)L) - 1],$$

onde usamos a igualdade $n\,h = x_{n+1} - x_1$. Uma vez que $\left| e_n \right| \leq \xi_n$ provamos o seguinte teorema.

Teorema 2.1: Seja y_n a solução aproximada de (1.1) gerada pelo Método de Euler Explícito. Se a solução exata $y(x)$ de (1.1) possui uma derivada de segunda ordem contínua no intervalo $[x_1, x_{n+1}]$, e se neste intervalo as desigualdades

$$\left| f_y(x,y) \right| < L \text{ e } \left| y''(x) \right| < K$$

forem satisfeitas para constantes positivas L e K limitadas, então o erro $e_n = y_n - y(x_n)$ é limitado como segue:

$$\left| e_n \right| \leq \frac{h\,K}{2L}[\exp((x_{n+1} - x_1)L) - 1].$$

3. PROBLEMAS DE VALORES DE CONTORNO EM EQUAÇÕES ORDINÁRIAS

Agora apresentaremos um método para obter a solução aproximada do problema de valor de contorno (PVC) a seguir:

$$\begin{cases} y''(x)+f(x)y'(x)+g(x)y=q(x) \\ a_1y(a)+a_2y'(a)=c \\ b_1y(b)+b_2y'(b)=d \end{cases}, \qquad (3.1)$$

onde $[a,b]$ é o intervalo que está definida a variável independente do problema, a_1, a_2, b_1, b_2, c, d são constantes e $f(x)$, $g(x)$, $q(x)$ são funções de x contínuas no intervalo $[a,b]$ satisfazendo com $g(x)<\delta<0$ em $[a,b]$.

Para discretizar o PVC (3.1) usamos o método das diferenças finitas centradas para aproximar as derivadas no interior do domínio, diferenças progressivas no contorno inferior e diferenças regressivas no contorno superior usando uma malha uniforme com N+1 pontos e passo h > 0 satisfazendo a condição de estabilidade

$$\frac{h}{2}\left| f(x) \right| \le 1 \text{ em } [a,b].$$

Assim, no interior da malha temos que a equação $y''(x)+f(x)y'(x)+g(x)y=q(x)$ tem a seguinte discretização:

$$\frac{y_{n+2}-2y_{n+1}+y_n}{h^2}+\frac{f(x_{n+1})(y_{n+2}-y_n)}{2h}+g(x_{n+1})y_{n+1}=q(x_n), \text{ com } n=1,...,N-1.$$

Ou seja,

$$(1-\frac{h}{2}f(x_{n+1}))y_n+(-2+h^2g(x_{n+1}))y_{n+1}+(1+\frac{h}{2}f(x_{n+1}))y_{n+2}=h^2q_{n+1}, \text{ com } n=1,...,N-1.$$

Usando as fórmulas progressivas no extremo a e regressivas no extremo b as condições de contorno em (3.1) têm as seguintes discretizações:

$$(h\,a_1-a_2)y_1+a_2\,y_2=h\,c;$$
$$(h\,b_1+b_2)y_{N+1}-b_2\,y_N=h\,d.$$

Mostraremos agora o programa implementado para calcular numericamente a solução do PVC:

Método das diferenças finitas para PVC (3.1):

Programa Principal

```
clear
clc
p=1; % parametro de entrada dos dados
[a,b,a1,a2,b1,b2,h,N,c,d] = dadosPSL(p); %dados
x = [a:h:b]; % Malha
%Construção da matriz dos coeficientes do sistema
Ay=B
A = zeros(N+1,N+1);
A(1,1) = h*a1-a2;
A(1,2) =a2;
for i = 2:N
   A(i,i-1) = 1 - (h/2)*f(x(i+1));
   A(i,i) = -2 + (h^2)*g(x(i+1));
   A(i,i+1) = 1 + (h/2)*f(x(i+1));
end
A(N+1,N) = -b2;
A(N+1,N+1) = h*b1+b2;
%construção da matriz dos termos independentes
B = zeros(N+1,1);
```

dadosPSL(p)

```
%Dados Iniciais:
function [a,b,a1,a2,b1,b2,h,N,c,d] = dados(p)
%Todos os dados devem ser fornecidos e
representam valores de saída.
a  ; % Extremo Inferior do intervalo [a,b]
b  ; % Extremo Superior do intervalo [a,b]
a1 ;%Coeficiente da condição à esquerda
a2 ;%Coeficiente da condição à esquerda
b1 ; %Coeficiente da condição à direita
b2 ; %Coeficiente da condição à direita
c ; % condição de contorno inferior
d ;% condição de contorno superior
h ; % Passo da malha
N = floor((b-a)/h); % Número de pontos
h = (b-a)/N; % Passo da malha recalculado
```

```
B(1,1) = c*h;
for i = 2:N
      B(i,1) = (h^2)*q(x(i+1));
end
B(N+1,1) = d*h;
% Solução do Sistema Ay=B
y = A\B; % Resolução no interior da malha (y=A^(-1)*B)
```

Função f:
```
function f=f(x)
f ;
```

Função g:
```
function g=g(x)
g;
```

```
function q=q(x)
q ;
```

O esquema numérico está ilustrado para os dois exemplos a seguir.

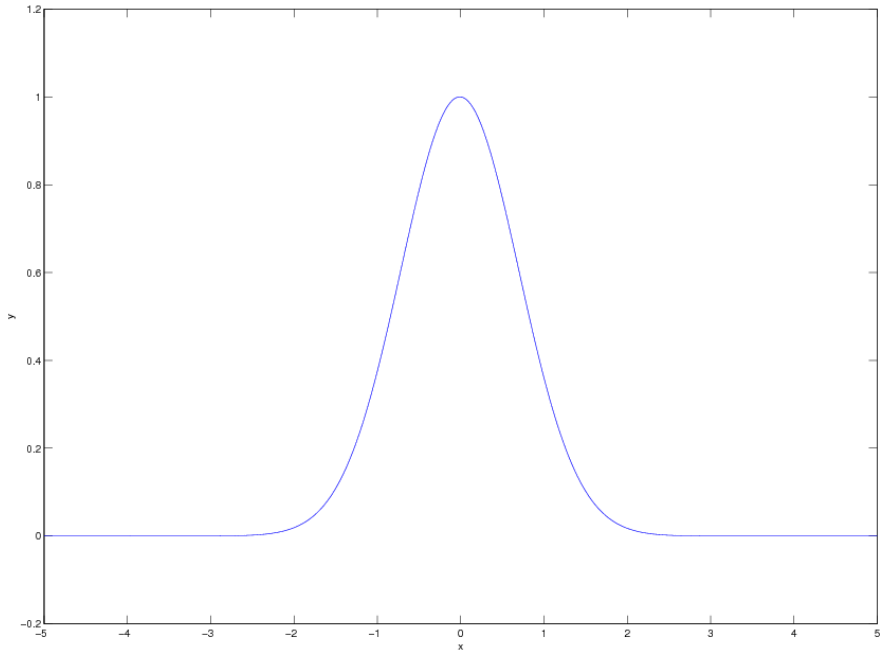
Exemplo 1:

$$\begin{cases} y'' + 2xy' - 2y = -4\exp(-x^2) \\ y(-5) + y'(-5) = 11\exp(-25) \\ y(5) + y'(5) = -9\exp(-25) \end{cases} \quad (3.2)$$

cuja solução exata é $y = \exp(-x^2)$.

Testamos o método das diferenças finitas para PVC (3.2) no intervalo $[-5,5]$, com passo $h = 0{,}01$, $a_1 = 1$, $a_2 = 1$, $b_1 = 1$, $b_2 = 1$, $c = 11\exp(-25)$ e $d = -9\exp(-25)$:

O gráfico da solução numérica está exibido a seguir:



Exemplo2:

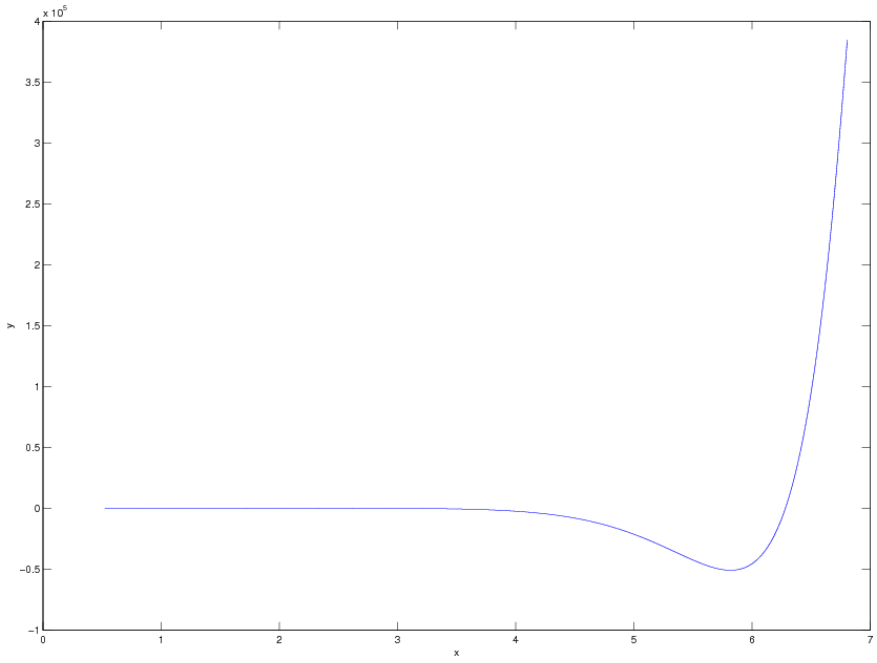
$$\begin{cases} y'' - xy' - x^2 y = (3 - 2x - x^2)\exp(2x)\operatorname{sen}(x) + (4 - x)\exp(2x)\cos(x) \\ y\left(\frac{\pi}{6}\right) + y'\left(\frac{\pi}{6}\right) = \frac{1}{2}\exp\left(\frac{\pi}{3}\right) + \exp\left(\frac{\pi}{3}\right)\cos\left(\frac{\pi}{6}\right) \\ y\left(\frac{13\pi}{6}\right) + y'\left(\frac{13\pi}{6}\right) = \frac{1}{2}\exp\left(\frac{13\pi}{3}\right) + \exp\left(\frac{13\pi}{3}\right) + \exp\left(\frac{13\pi}{3}\right)\cos\left(\frac{13\pi}{3}\right) \end{cases} . \quad (3.3)$$

cuja solução exata é $y = \exp(2x)\operatorname{sen}(x)$.

Neste caso executamos o método das diferenças finitas para o PVC (3.3) no intervalo $[\frac{\pi}{6}, \frac{13\pi}{6}]$, com passo $h = 0{,}001$, $a_1 = 1$, $a_2 = 1$, $b_1 = 1$, $b_2 = 1$ e com

$$c = \frac{1}{2}\exp\left(\frac{\pi}{3}\right) + \exp\left(\frac{\pi}{3}\right)\cos\left(\frac{\pi}{6}\right) \text{ e } d = \frac{1}{2}\exp\left(\frac{13\pi}{3}\right) + \exp\left(\frac{13\pi}{3}\right) + \exp\left(\frac{13\pi}{3}\right)\cos\left(\frac{13\pi}{3}\right)$$

Cujo o gráfico da solução numérica está exibido a seguir :



CONCLUSÕES

Diante dos resultados obtidos neste trabalho concluímos que:

- O melhor método numérico em termos de precisão, para o cálculo de soluções aproximadas de PVI's foi o método de Runge-Kutta;
- No Método de Euler Explícito conseguimos estabelecer a convergência do método, fornecendo um limite superior para o erro.
- No Método das Diferenças Finitas para PVC o método implementado usando diferenças centradas no interior da malha e diferenças progressivas e regressivas no contorno proporcionou resultados numéricos com ótima concordância com as soluções analíticas.

AGRADECIMENTOS

Ao CNPq pela bolsa de Iniciação Científica através do Proc. 504274/2007-4.
.

REFERÊNCIAS BIBLIOGRÁFICAS

CONTE, S.D. **Elementos de Análise Matemática, 3ª Edição.** Editora Globo, 1977. p. 244-317.

CUMINATO, J.A.; MENEGUETTE, M.J. **DISCRETIZACÃO DE EQUAÇÕES DIFERENCIAIS PARCIAIS: Técnicas de Diferenças Finitas.** Disponível em: <http://www.icmc.usp.br/~jacumina/> Acesso em: 27 jul. 2009.

MATSUMOTO, E.Y. **MATLAB6: FUNDAMENTOS DE PROGRAMAÇÃO**. São Paulo, Editora Érica, 2001.