Resumos MNUM

MIEIC

8 de fevereiro de 2021

Conteúdo

1 Introdução

Este documento contém resumos dos conteúdos lecionados em MNUM no ano letivo 2018/2019. Este documento contém apenas os passos de execução de cada método e não substitui o estudo mais aprofundado dos conteúdos da unidade curricular. Implementações dos métodos presentes neste documento em C++ podem ser vistas **neste repositório**.

2 Erros

Vírgula flutuante: $mantissa * base^{expoente}$

- $\frac{1}{base} \leq |mantissa| < 1$
- $\pm 1.f * 2^e$ - Dois tipos: precisão simples (32 bits) e dupla (64 bits)
  - f: parte fracionária da mantissa
  - $e = expoente * vies$

Erro Absoluto: $x_{exato} - x_{aproximado}$

Erro Relativo: $\frac{x_{exato} - x_{aproximado}}{x_{exato}}$

3 Zeros

3.1 Método da Bisseção

A partir de um intervalo, calcula-se o seu ponto médio. Se $f(\frac{x_1+x_2}{2})$ for nulo, encountrou-se a raíz; se não, reduz-se o intervalo.

C++:

```
for(int n= 0; n < ...; n++) {
    m = (a + b) / 2;
    if(f(a) * f(m) < 0)
        b = m;
    else if(f(a) * f(m) > 0)
        a = m;
    else
        break;
}
```

Critérios de Paragem:

1. $|x_1 - x_2| \leq \epsilon$ - critério de precisão absoluta
2. $\frac{|x_1-x_2|}{x_1} \leq \epsilon$ ou $\frac{|x_1-x_2|}{x_2} \leq \epsilon$ - critério de precisão relativa
3. $|f(x_1) - f(x_2)| \leq \epsilon$ - critério de anulação da função
4. $n = N$ - critério do número de iterações
   - $n \leq (nmerodebitsdamantissa) + \log_2(b-a)$ -> a e b são os extremos do intervalo
   - ou $n \leq 3.3 * (nmerodedgitosdamantissa) + \log_{10}(b-a)$

3.2 Método da Corda ou Falsa Posição

Calcula-se a nova posição traçando-se uma corda entre os dois pontos.

$$m = \frac{a * f(b) - b * f(a)}{f(b) - f(a)}$$

C++:

```
for(int n = 0; n < ...; n++) {
    w = (a * f(b) - b * f(a))/(f(b) - f(a));
    if(f(a) * f(w) < 0)
        b = w;
    else if(f(a) * f(w) > 0)
        a = w;
    else
        break;
}
```

3.3 Método da Tangente ou Newton

Parte apenas de um valor plausível, substituindo este valor pelo zero da tangente neste ponto

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

- Implica conhecimento prévio da derivada e $f'(x_k) \neq 0$

3.4 Método de Picard-Peano

Transforma-se uma equação $f(x) = 0$ em $x = g(x)$, tal que g(x) não seja divergente. Através da equação $x_n = g(x_{n-1})$ calculam-se pontos sucessivos até se chegar perto do 0

- Condição de convergência: $g'(x) < 1$
- Para sistemas de equação: $\begin{cases} \mathrm{x} = \mathrm{g}_1(x, y) \\ \mathrm{y} = \mathrm{g}_2(x, y) \end{cases}$
  Todas as derivadas parcias têm que convergir: $\frac{\delta g_1}{\delta x}$, $\frac{\delta g_1}{\delta y}$, $\frac{\delta g_2}{\delta x}$, $\frac{\delta g_2}{\delta y}$

4 Sistemas de Equações Lineares (A.x = b)

4.1 Método da Eliminação Gaussiana

Divide-se a 1ª equação por a_{11}, para tornar unitário o 1º coeficiente. Multiplica-se esta equação por $-a_{22}$, e soma-se à primeira; resultado será a nova 2ª equação; repete-se este processo para todas as linhas. Repete-se todos os passes utilizando os elementos da diagonal.

C++: (útil definir rowop como no Maxima

```
void rowop(vector<vector<double>> &matrix, size_t i, size_t j, double value) {
    for(size_t k = 0; k < matrix[0].size(); k++) {
        matrix[i][k] -= value * matrix[j][k];
    }
}

...

    for(size_t i = 0; i < matrix.size(); i++) {
        rowop(matrix, i, i, 1 - 1/matrix[i][i]);
        for(size_t j = 0; j < matrix.size(); j++) {
            if(i != j)
                rowop(matrix, j, i, matrix[j][i]);
        }
    }
```

O Erro no Método de Gauss:

- **Estabilidade Externa** (potenciais erros dos coeficientes e dos termos constantes): $A.\delta x = \delta b - \delta A.x$
- **Estabilidade Interna** (erros de arredondamento no decorrer do cálculo): $A.\delta = b - A.x_0 = \epsilon$ (ϵ = coluna dos resíduos)

Mínimização dos erros:

- **Pivotagem Parcial** (eliminar coluna com a equação com maior coeficiente nessa coluna)
- **Pivotagem Total** (eliminar com a equação não tratada com maior coeficiente)

- **Escalagem de Linhas**
- **Escalagem de Colunas**

4.2 Método de Khaletsky

Dado um sistema $A.x = b$, representa-se A por um produto $L.U$, $A = L.U$, e calcula-se x através dos sistems $L.y = b$ e $U.x = y$

Coeficientes LU:

- $l_{i,1} = a_{i,1}$
- $l_{i,j} = a_{i,j} - \sum_{k=1}^{j-1} l_{i,k}.u_{k,j} (i \geq j)$
- $u_{1,i} = \frac{a_{1,i}}{l_{1,1}}$
- $u_{i,j} = \frac{a_{i,j} - \sum_{k=1}^{j-1} l_{i,k}.u_{k,j}}{l_{i,i}} (i < j)$

4.3 Método de Gauss-Jacobi

$$x_i^{(k)} = \frac{1}{a_{i,i}}[- \sum_{j=1, j \neq i}^{n} a_{i,j} x_j^{(k-1)} + b_i], (1 \leq i \leq n)$$

4.4 Método de Gauss-Seidel

$$x_i^{(k)} = \frac{1}{a_{i,i}}[- \sum_{j=1, j<i}^{n} a_{i,j} x_j^{(k)} - \sum_{j=1, j>i}^{n} a_{i,j} x_j^{(k-1)} + b_i], (1 \leq i \leq n)$$

5 Quadratura e Cubatura

5.1 Regra dos Trapézios - 2ª ordem

Substitui-se, em cada intervalo, o arco da curva pela sua corda, calculando, em seguida, a área sob a poligonal assim definida.

$$\int_{x_0}^{x_n} y.\delta x = \frac{h}{2}.[y_0 + 2y_1 + ... + 2y_{n-1} + y_n]$$

5.2 Controlo do erro

Quociente de Convergência: $\frac{S' - S}{S'' - S'} \approx 2^{ordemdomtodo}$

Erro: $\epsilon'' = \frac{S'' - S'}{2^{ordem} - 1}$

5.3 Regra de Simpson - 4ª ordem

Em vez de substituir a curva por cordas, substitui-a pelas parábolas definidas por cada trio de pontos.

$$\int_{x_0}^{x_{2n}} y.\delta x = \frac{h}{3}.[y_0 + 4y_1 + 2y_2 + 4y_3 + ... + 2y_{2n-2} + 4y_{2n-1} + y_{2n}]$$

Fórmula de Simpson - Cubatura:

$$h_x = \frac{A-a}{2}$$

$$h_y = \frac{B-b}{2}$$

$$\int\int f(x,y)\delta x\delta y = \frac{h_x.h_y}{9}.[\sum_0 + 4\sum_1 + 16\sum_2]$$

$\sum_0$: Valores de f nos vértices
$\sum_1$: Valores de f nos pontos médios dos lados
$\sum_0$: Valores de f no centro

6 Equações Diferenciais Ordinárias

6.1 Método de Euler - 1ª ordem

Calcula-se $y'_n = f(x_n, y_n)$, que é a inclinação da curva no ponto (x_n, y_n); calcula-se o ponto seguinte, sendo h o passo :

$$\begin{cases} \mathrm{y}_{n+1} = y_n + h * y'_n \\ \mathrm{x}_{n+1} = x_n + h \end{cases}$$

6.2 Método de Euler Melhorado

A partir de 2 pontos (x_{n-1}, y_{n-1}) e (x_n, y_n), calcula-se (x_{n+1}, y_{n+1}) do seguinte modo:

- Calcula-se um valor previsto $p_{n+1} = y_{n-1} + 2h.y'_n$
- Com este ponto, calcula-se $p_{n+1} = f(x_{n+1}, p_{n+1})$
- Calcula-se também o valor corrigido $y_{n+1} = y_n + \frac{p'_{n+1}+y'_n}{2}$
- Por fim, prepara-se o passo seguinte: $y'_{n+1} = f(x_{n+1}, y_{n+1})$

$$x_i + \frac{1}{2} = x_i + \frac{h}{2}$$

$$y_i + \frac{1}{2} = y_i + \frac{h}{2}.f'(x_i, y_i)$$

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

$$x_{i+1} = x_i + h$$

6.3 Runge-Kutta 2ª ordem

$$y' = f(x_n, y_n)$$

$$\begin{cases} \mathrm{y}_{n+1} = y_n + h.f(x_n + \frac{h}{2}, y_n + \frac{h}{2}.f(x_n, y_n)) \\ \mathrm{x}_{n+1} = x_n + h \end{cases}$$

6.4 Runge-Kutta 4ª ordem

$$\delta y_1 = h.f(x_n, y_n)$$

$$\delta y_2 = h.f(x_n + \frac{h}{2}, y_n + \frac{\delta y_1}{2})$$

$$\delta y_3 = h.f(x_n + \frac{h}{2}, y_n + \frac{\delta y_2}{2})$$

$$\delta y_4 = h.f(x_n + h, y_n + \delta y_3)$$

$$\begin{cases} \mathrm{y}_{n+1} = y_n + \frac{1}{6}.\delta y_1 + \frac{1}{3}.\delta y_2 + \frac{1}{3}.\delta y_3 + \frac{1}{6}.\delta y_4 \\ \mathrm{x}_{n+1} = x_n + h \end{cases}$$

7 Otimização

7.1 Método dos terços

A partir de x_1 e x_2, calculam-se x_3 e x_4 (pontos que dividem o intervalo $[x_1, x_2]$ em 3 partes iguais).

Se $f(x_4) < f(x_3)$, então $x_1 = x_3$; senão se $f(x_4) > f(x_3)$, então $x_2 = x_4$.

7.2 Regra Áurea

A partir de x_1 e x_2, calculam-se $x_3 = x_1 + A.(x_2 - x_1)$ e $x_4 = x_1 + B.(x_2 - x_1)$, tais que $B = \frac{\sqrt{5}-1}{2}$ e $A = B^2$.

Se $f(x_3) < f(x_4)$, então $x_2 = x_4$; senão se $f(x_3) > f(x_4)$, então $x_1 = x_3$.

7.3 Método do Gradiente

$$x_j^{(i+1)} = x_j^{(i)} - h.\frac{\delta f^{(i)}}{\delta x_j}(j = 1, 2, ..., n),$$

em que h é o passo.

Se $f(x^{(i+1)}) < f(x^{(i)})$, dá-se novo passo com $h = 2 * h$.

Se $f(x^{(i+1)}) > f(x^{(i)})$, não se efetua o passo e faz-se nova tentativa com $h = \frac{h}{2}$.

7.4 Método da Quádrica

Só é aplicável nas vizinhanças imediatas do mínimo (ou máximo).

$$x_i^{n+1} = x_n - H^{-1}x\nabla f(x_i^n),$$

sendo H^{-1} o inverso do determinante da matriz hessiana.

7.5 Método de Levenberg-Marquardt

O passo é a soma dos passos dos 2 método anteriores:

$$x_{n+1} = x_n - h_{L.M},$$

tal que $h_{L.M} = H^{-1}\nabla + \lambda\nabla$, sendo λ o parâmetro a determinar mediante a evolução do método:

- Se $f(x_{n+1}) < f(x_n)$, $\lambda = \frac{\lambda}{2}$
- Senão se $f(x_{n+1}) > f(x_n)$, $h = h * 2$

8 Maxima: Comandos Úteis

8.1 Método de Gauss

```
m: matrix([x1, y1, z1, b1], [x2, y2, z2, b2], [x3, y3, z3, b3])$

for i:1 thru 3 do (
    m: rowop(m, i, i, 1 - 1/m[i][i]),
    for j:1 thru 3 do (
        if(i#j) then m: rowop(m, j, i, m[j][i]))) $
```

Este método pressupõe que m[i][i] não é nulo.

8.2 Runge-Kutta 4

```
rk(f', y, 1, [x, 0, 4, 1]);
```

f': derivada da função
y: variável
1: valor inicial de y
[x, 0, 4, 1]: [x, x inicial, x final, h]

Sistemas:

```
rk([x', y'], [x, y], [-1.25, 0.75], [t, 0, 4, h]);
```

8.3 Khaletsky

```
A: matrix([1, 2, 3], [4, 5, 6], [7, 8, 9]);
b: [10, 11, 12];
[P, L, U]: get_lu_factors(lu_factor(A));
Y: invert(L).b;
X: invert(U).Y;
```

8.4 Hessiana

hessian(função, [lista de variáveis]);