

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULADE DE ENGENHARIA MECÂNICA

**RELATÓRIO DE ALGORITMO IMPLEMENTADO**

**MÉTODO DE NEWTON RHAPSON PARA SOLUÇÃO NUMÉRICA DE SISTEMAS DE EQUAÇÕES NÃO LINEARES**

Aluno: Diego Fernando Luque Martin

Matrícula: 191606

Campinas

2017

Sumário

[1. Introdução 3](#_Toc498179403)

[2. Método 4](#_Toc498179404)

[2.1. Algoritmo 5](#_Toc498179405)

[2.2. Teste 7](#_Toc498179406)

[Considerações sobre o valor de tolerância: 7](#_Toc498179407)

[Considerações sobre o valor do número máximo de iterações: 7](#_Toc498179408)

[Considerações sobre o Critério de Parada: 8](#_Toc498179409)

[Considerações sobre o Delta h: 8](#_Toc498179410)

[Considerações sobre a Aproximação da Derivada Numérica: 8](#_Toc498179411)

[2.3. Validação 9](#_Toc498179412)

[3. Exercícios resolvidos 10](#_Toc498179413)

[Exercício 10.2 – 1c (Burden, 2015) 10](#_Toc498179414)

[Exercício 10.2 – 1d (Burden, 2015) 10](#_Toc498179415)

[Exercício 10.2 – 2a (Burden, 2015) 10](#_Toc498179416)

[Exercício 10.2 – 2b (Burden, 2015) 11](#_Toc498179417)

[Exercício 10.2 – 8a (Burden, 2015) 11](#_Toc498179418)

[Exercício 10.2 – 10a (Burden, 2015) 12](#_Toc498179419)

[4. Conclusão 13](#_Toc498179420)

[5. Referências 14](#_Toc498179421)

[ANEXO A. Código Fonte 15](#_Toc498179422)

1. Introdução

Este relatório tem por fundamento a apresentação do Método de Cálculo para Solução Numérica de Sistemas de Equações Não Lineares utilizando o Método de Newton Rhapson para Sistemas.

1. Método

Para que possamos resolver os Sistemas de Equações Não Lineares, devemos transformar este sistema de equações em um sistema linear através do método de Newton-Raphson, com o auxílio de uma matriz jacobiana. Sabendo que a matriz jacobiana é dada por:

Utilizando o algoritmo que nos direcionou uma solução através do ponto fixo, porém com uma abordagem multidimensional, temos:

Entretanto, um ponto de dificuldade do Método de Newton-Raphson é a resolução da derivada, uma vez que, para muitos casos, não é tão simples e/ou viável de ser encontrada. Com base nisto, devemos aplicar métodos de resoluções numéricos anteriormente estudados na disciplina IM253A. Inicialmente, consideraremos o Método de entrada com 2 pontos, onde:

Adicionalmente, deverá ser definido também o critério de parada do método, uma vez que, o algoritmo operará com iterações. Sendo que, para este relatório fora adotado o erro baseado no resíduo:

* 1. Algoritmo

O livro Analise Numérica (Burden, Análise Numérica, 2015), fornece o seguinte algoritmo (10.2):

ENTRADA: número de incógnitas e equações n; aproximação x = (x1, x2, ..., xn)t; tolerância TOL; número máximo de iterações N.

SAÍDA: solução x = (x1, x2, ..., xn)t ou mensagem de que o número máximo de iterações foi excedido.

Passo 1: Faça k = 1.

Passo 2: Enquanto (k ≤ N) execute Passos 3 a 7

Passo 3: Calcule F(x) e J(x), em que J(x)i,j = (∂fi(x) / (∂xj) para 1 ≤ i, j ≤ n.

Passo 4: Resolva o sistema linear, n x n, J(x)y = -F(x).

Passo 5: Faça x = x + y.

Passo 6: Se ||y|| < TOL então SAÍDA (x);

(O procedimento foi bem-sucedido.)

PARE.

Passo 7: Faça k = k + 1.

Passo 8: SAÍDA (‘Número máximo de iterações excedido’);

(O procedimento não foi bem-sucedido.)

PARE.

Com base no exemplo dado acima, o algoritmo deverá exigir informações do usuário:

- Número de Incógnitas do Sistema ou de Equações [n];

- Aproximações Iniciais w[n];

Após realizados os cálculos, o algoritmo deverá apresentar ao final de sua execução uma das seguintes alternativas:

- Vetor Solução Aproximada w[n];

- Erro.

Para execução do programa em linguagem de programação, os seguintes parâmetros foram adotados:

- Linguagem de Programação: C;

- Compilador: Code::Blocks 16.01

* 1. Teste

Para teste do código fonte escrito em linguagem de programação C, foi usado exemplo de cálculo dado pelo Livro Análise Numérica (Burden, 2015), página 717:

Após a implementação do algoritmo obteve se os seguintes resultados:

kDigite o número de incógnitas do sistema [n]: 3

Digite as aproximações iniciais para o sistema w[n]:

w[1]= 0,1

w[2]= 0,1

w[3]= -0,1

k= 0 w[1]= +0,10000000 w[2]= +0,10000000 w[3]= -0,10000000 erro[4]= +0,00000000

k= 1 w[1]= +0,49986967 w[2]= +0,01946685 w[3]= -0,52152047 erro[4]= +0,34438793

k= 2 w[1]= +0,50001424 w[2]= +0,00158859 w[3]= -0,52355696 erro[4]= +0,02588914

k= 3 w[1]= +0,50000011 w[2]= +0,00001244 w[3]= -0,52359845 erro[4]= +0,00020122

k= 4 w[1]= +0,50000000 w[2]= +0,00000000 w[3]= -0,52359878 erro[4]= +0,00000001

O método atingiu a tolerância especificada.

Fim do programa.

Obs: h = 0.00000001

Considerações sobre o valor de tolerância:

Após alguns testes realizados, convencionou-se que o valor de tolerância definido como um parâmetro interno do algoritmo não deve ultrapassar 10-16, pois quando este valor é ultrapassado para baixo, os cálculos exigidos da máquina ou computador apresentam erro ou valores não representativos.

Considerações sobre o valor do número máximo de iterações:

Devido ao valor mínimo de tolerância definido acima, nos exemplos estudados ente relatório, o número máximo de iterações não ultrapassou . Por termos um estudo com poucos exemplos devemos definir também uma margem de segurança para que este valor de iterações não seja impeditivo para outros casos. Com base nessas informações convencionou-se o número máximo de iterações em 100.

Considerações sobre o Critério de Parada:

Para evitar que o algoritmo permaneça em um número de iterações indeterminado, tendo como parâmetros o valor de tolerância e o número máximo de iterações, convencionou se o seguinte critério:

Considerações sobre o Delta h:

Para o cálculo das derivadas dos sistemas, utilizou de uma aproximação numérica dada pela equação:

Inicialmente para o valor de h definiu se como h = tolerância x 10, ou seja 10-15. Porém com valores muito pequenos de h (abaixo de 10-8) observou se que há um aumento do número de iterações para obtenção do resultado. Com isso convencionou a seguinte regra para o algoritmo: h = tolerância x 10 com o valor mínimo de 10-8.

Considerações sobre a Aproximação da Derivada Numérica:

Afim de reduzirmos a influência do erro ocasionado pela aproximação da derivada numérica utilizando um delta h citado anteriormente podemos utilizar formulas com mais pontos de derivação:

Fórmula com 3 pontos centrada:

Fórmula com 5 pontos:

Para a implementação do algoritmo será considerada a fórmula com 5 pontos acima:

for (j = 1; j <= n; j++)

{

for(\_i=1; \_i<=n; \_i++){\_w1[\_i] = w[\_i]; \_w2[\_i] = w[\_i]; \_w3[\_i] = w[\_i]; \_w4[\_i] = w[\_i];}

\_w1[j] = \_w1[j] - 2 \* h;

\_w2[j] = \_w2[j] - h;

\_w3[j] = \_w3[j] + h;

\_w4[j] = \_w4[j] + 2 \* h;

/\*J[i][j] = (funcao(i, \_w3) - funcao(i, w)) / h;\*/

/\*J[i][j] = (funcao(i, \_w3) - funcao(i, \_w2)) / (2 \* h);\*/

J[i][j] = (funcao(i, \_w1) - 8\*funcao(i, \_w2) + 8\*funcao(i, \_w3) - funcao(i, \_w4)) / (12\*h);

}

* 1. Validação

Para validação do algoritmo escrito foi utilizado o exemplo de cálculo demonstrado no livro Análise Numérica, (Burden, 2015), página 717, onde o autor demostra o Método de Newton para Sistemas com os seguintes parâmetros:

Resultados obtidos pelo algoritmo implementado:

k= 0 w[1]= +0,10000000 w[2]= +0,10000000 w[3]= -0,10000000 erro[4]= +0,00000000

k= 1 w[1]= +0,49986967 w[2]= +0,01946685 w[3]= -0,52152047 erro[4]= +0,34438793

k= 2 w[1]= +0,50001424 w[2]= +0,00158859 w[3]= -0,52355696 erro[4]= +0,02588914

k= 3 w[1]= +0,50000011 w[2]= +0,00001244 w[3]= -0,52359845 erro[4]= +0,00020122

k= 4 w[1]= +0,50000000 w[2]= +0,00000000 w[3]= -0,52359878 erro[4]= +0,00000001

Obs: h = 0.00000001

Os resultados obtidos pelo autor estão na tabela 1 a seguir:

Tabela 1 (Burden, Numerical Analysis, 2011)



Com base nos resultados da iteração 4, o algoritmo foi considerado válido pois o valor de , correspondentes ao valor do resultado do algoritmo tem igualdade em todas as casas decimais apresentadas pelo autor salvo arredondamentos.

1. Exercícios resolvidos

Exercício 10.2 – 1c (Burden, 2015)



Digite o número de incógnitas do sistema [n]: 2

Digite as aproximações iniciais para o sistema w[n]:

w[1]= 0

w[2]= 0

k= 0 w[1]= +0,00000000 w[2]= +0,00000000 erro[3]= +0,00000000

k= 1 w[1]= -47,99998750 w[2]= +14,99999583 erro[3]= +3203,99829961

k= 2 w[1]= -23,94262015 w[2]= +7,60867810 erro[3]= +797,28333276

k= 3 w[1]= -11,82159961 w[2]= +4,16314442 erro[3]= +194,40596847

k= 4 w[1]= -5,67345492 w[2]= +3,01546133 erro[3]= +43,06840248

k= 5 w[1]= -2,68618849 w[2]= +3,24450938 erro[3]= +9,13361449

k= 6 w[1]= -1,41209067 w[2]= +3,44569135 erro[3]= +1,78522213

k= 7 w[1]= -1,03881250 w[2]= +3,49465183 erro[3]= +0,14892513

k= 8 w[1]= -1,00041570 w[2]= +3,49994324 erro[3]= +0,00158631

k= 9 w[1]= -1,00000005 w[2]= +3,49999999 erro[3]= +0,00000019

O método atingiu a tolerância especificada.

Fim do programa.

Exercício 10.2 – 1d (Burden, 2015)



Digite o número de incógnitas do sistema [n]: 2

Digite as aproximações iniciais para o sistema w[n]:

w[1]= 0

w[2]= 0

k= 0 w[1]= +0,00000000 w[2]= +0,00000000 erro[3]= +0,00000000

k= 1 w[1]= +0,05263158 w[2]= +0,26315789 erro[3]= +0,05540166

k= 2 w[1]= +0,17956147 w[2]= +0,28482464 erro[3]= +0,08008653

k= 3 w[1]= +0,13137338 w[2]= +0,27351818 erro[3]= +0,01148263

k= 4 w[1]= +0,12167049 w[2]= +0,27120685 erro[3]= +0,00046539

k= 5 w[1]= +0,12124275 w[2]= +0,27110535 erro[3]= +0,00000090

k= 6 w[1]= +0,12124191 w[2]= +0,27110516 erro[3]= +0,00000000

O método atingiu a tolerância especificada.

Fim do programa.

Exercício 10.2 – 2a (Burden, 2015)



Digite o número de incógnitas do sistema [n]: 3

Digite as aproximações iniciais para o sistema w[n]:

w[1]= 0

w[2]= 0

w[3]= 0

k= 0 w[1]= +0,00000000 w[2]= +0,00000000 w[3]= +0,00000000 erro[4]= +0,00000000

k= 1 w[1]= +0,50000000 w[2]= +0,50000155 w[3]= -0,52359877 erro[4]= +155,25096678

k= 2 w[1]= +0,50016669 w[2]= +0,25080442 w[3]= -0,51738741 erro[4]= +38,81200955

k= 3 w[1]= +0,49994493 w[2]= +0,12620664 w[3]= -0,52045511 erro[4]= +9,70287954

k= 4 w[1]= +0,49998624 w[2]= +0,06391324 w[3]= -0,52200313 erro[4]= +2,42529258

k= 5 w[1]= +0,49999467 w[2]= +0,03277689 w[3]= -0,52278013 erro[4]= +0,60592040

k= 6 w[1]= +0,49999742 w[2]= +0,01722924 w[3]= -0,52316840 erro[4]= +0,15108103

k= 7 w[1]= +0,49999860 w[2]= +0,00949623 w[3]= -0,52336156 erro[4]= +0,03737465

k= 8 w[1]= +0,49999916 w[2]= +0,00570988 w[3]= -0,52345614 erro[4]= +0,00896030

k= 9 w[1]= +0,49999942 w[2]= +0,00396594 w[3]= -0,52349971 erro[4]= +0,00190085

k= 10 w[1]= +0,49999951 w[2]= +0,00332332 w[3]= -0,52351576 erro[4]= +0,00025810

k= 11 w[1]= +0,49999953 w[2]= +0,00320354 w[3]= -0,52351875 erro[4]= +0,00000897

k= 12 w[1]= +0,49999953 w[2]= +0,00319907 w[3]= -0,52351886 erro[4]= +0,00000001

k= 13 w[1]= +0,49999953 w[2]= +0,00319906 w[3]= -0,52351886 erro[4]= +0,00000000

O método atingiu a tolerância especificada.

Fim do programa.

Exercício 10.2 – 2b (Burden, 2015)



Digite o número de incógnitas do sistema [n]: 3

Digite as aproximações iniciais para o sistema w[n]:

w[1]= 0

w[2]= 0

w[3]= 0

k= 0 w[1]= +0,00000000 w[2]= +0,00000000 w[3]= +0,00000000 erro[4]= +0,000000000

k= 1 w[1]= +5,00000003 w[2]= +36,99999694 w[3]= -38,9999995 erro[4]= +1368,999770

k= 2 w[1]= +4,35087701 w[2]= +18,49123147 w[3]= -19,84210854 erro[4]= +342,574764

k= 3 w[1]= +5,36382406 w[2]= +9,25545395 w[3]= -11,61927792 erro[4]= +85,2996037

k= 4 w[1]= +5,69605549 w[2]= +4,66532997 w[3]= -7,36138543 erro[4]= +21,0692482

k= 5 w[1]= +5,88282261 w[2]= +2,42728015 w[3]= -5,31010275 erro[4]= +5,00886631

k= 6 w[1]= +5,96609475 w[2]= +1,41264773 w[3]= -4,37874248 erro[4]= +1,02947887

k= 7 w[1]= +5,99518811 w[2]= +1,05856600 w[3]= -4,05375410 erro[4]= +0,12537387

k= 8 w[1]= +5,99987186 w[2]= +1,00155958 w[3]= -4,00143144 erro[4]= +0,00324973

k= 9 w[1]= +5,99999990 w[2]= +1,00000117 w[3]= -4,00000107 erro[4]= +0,00000243

k= 10 w[1]= +6,00000000 w[2]= +1,00000000 w[3]= -4,00000000 erro[4]= +0,00000000

O método atingiu a tolerância especificada.

Fim do programa.

Exercício 10.2 – 8a (Burden, 2015)





Digite o número de incógnitas do sistema [n]: 4

Digite as aproximações iniciais para o sistema w[n]:

w[1]= 1

w[2]= 1

w[3]= 1

w[4]= 1

k= 0 w[1]= +1,00000000 w[2]= +1,00000000 w[3]= +1,00000000 w[4]= +1,00000000 erro= +0,00000000

k= 1 w[1]= +0,00000000 w[2]= +1,16666667 w[3]= +1,16666667 w[4]= +1,00000000 erro= +1,72222222

k= 2 w[1]= +0,00000000 w[2]= +0,90833333 w[3]= +0,90833333 w[4]= +1,00000000 erro= +0,65013889

k= 3 w[1]= -0,00000000 w[2]= +0,79292406 w[3]= +0,79292406 w[4]= +1,00000000 erro= +0,25745714

k= 4 w[1]= -0,00000000 w[2]= +0,74314211 w[3]= +0,74314211 w[4]= +1,00000000 erro= +0,10452039

k= 5 w[1]= -0,00000000 w[2]= +0,72212271 w[3]= +0,72212271 w[4]= +1,00000000 erro= +0,04292242

k= 6 w[1]= -0,00000000 w[2]= +0,71334241 w[3]= +0,71334241 w[4]= +1,00000000 erro= +0,01771479

k= 7 w[1]= -0,00000000 w[2]= +0,70969241 w[3]= +0,70969241 w[4]= +1,00000000 erro= +0,00732664

k= 8 w[1]= +0,00000000 w[2]= +0,70817826 w[3]= +0,70817826 w[4]= +1,00000000 erro= +0,00303289

k= 9 w[1]= +0,00000000 w[2]= +0,70755068 w[3]= +0,70755068 w[4]= +1,00000000 erro= +0,00125594

k= 10 w[1]= +0,00000000 w[2]= +0,70729067 w[3]= +0,70729067 w[4]= +1,00000000 erro= +0,00052017

k= 11 w[1]= -0,00000000 w[2]= +0,70718295 w[3]= +0,70718295 w[4]= +1,00000000 erro= +0,00021545

k= 12 w[1]= +0,00000000 w[2]= +0,70713833 w[3]= +0,70713833 w[4]= +1,00000000 erro= +0,00008924

k= 13 w[1]= -0,00000000 w[2]= +0,70711985 w[3]= +0,70711985 w[4]= +1,00000000 erro= +0,00003696

k= 14 w[1]= +0,00000000 w[2]= +0,70711219 w[3]= +0,70711219 w[4]= +1,00000000 erro= +0,00001531

k= 15 w[1]= -0,00000000 w[2]= +0,70710902 w[3]= +0,70710902 w[4]= +1,00000000 erro= +0,00000634

k= 16 w[1]= +0,00000000 w[2]= +0,70710771 w[3]= +0,70710771 w[4]= +1,00000000 erro= +0,00000263

k= 17 w[1]= -0,00000000 w[2]= +0,70710717 w[3]= +0,70710717 w[4]= +1,00000000 erro= +0,00000109

k= 18 w[1]= +0,00000000 w[2]= +0,70710694 w[3]= +0,70710694 w[4]= +1,00000000 erro= +0,00000045

k= 19 w[1]= +0,00000000 w[2]= +0,70710685 w[3]= +0,70710685 w[4]= +1,00000000 erro= +0,00000019

k= 20 w[1]= -0,00000000 w[2]= +0,70710681 w[3]= +0,70710681 w[4]= +1,00000000 erro= +0,00000008

k= 21 w[1]= +0,00000000 w[2]= +0,70710679 w[3]= +0,70710679 w[4]= +1,00000000 erro= +0,00000003

k= 22 w[1]= -0,00000000 w[2]= +0,70710679 w[3]= +0,70710679 w[4]= +1,00000000 erro= +0,00000001

k= 23 w[1]= +0,00000000 w[2]= +0,70710678 w[3]= +0,70710678 w[4]= +1,00000000 erro= +0,00000001

k= 24 w[1]= -0,00000000 w[2]= +0,70710678 w[3]= +0,70710678 w[4]= +1,00000000 erro= +0,00000000

O método atingiu a tolerância especificada.

Fim do programa.

Exercício 10.2 – 10a (Burden, 2015)

Digite o número de incógnitas do sistema [n]: 3

Digite as aproximações iniciais para o sistema w[n]:

w[1]= -0,5

w[2]= -0,5

w[3]= -0,5

k= 0 w[1]= -0,50000000 w[2]= -0,50000000 w[3]= -0,50000000 erro[4]= +0,00000000

k= 1 w[1]= +9,71795255 w[2]= +0,39434527 w[3]= +4,37698811 erro[4]= +29,8529740

k= 2 w[1]= +8,87009480 w[2]= +0,30805014 w[3]= -1,91400310 erro[4]= +1,60820469

k= 3 w[1]= +8,78265907 w[2]= +0,26666311 w[3]= -1,45162432 erro[4]= +0,19108561

k= 4 w[1]= +8,77151073 w[2]= +0,25986052 w[3]= -1,37408620 erro[4]= +0,00454388

k= 5 w[1]= +8,77128657 w[2]= +0,25969554 w[3]= -1,37228235 erro[4]= +0,00000258

O método atingiu a tolerância especificada.

Fim do programa.

1. Conclusão

O Cálculo de Solução Numérica de Sistemas de Equações Não Lineares utilizando o Método de Newton Rhapson mostrou se uma ferramenta poderosa para chegar a aproximações dos valores das incógnitas das equações com uma boa exatidão em relação aos valores reais. O método implementado neste relatório utiliza se de ferramentas estudadas ao longo do curso como Aproximação Numérica de Derivadas de Equações, Eliminação Regressiva de Gauss e Pivotamento Parcial com Escala para Matrizes.

A principal vantagem do Método está na rápida convergência ao valor esperado evoluindo quadraticamente a cada iteração, desde que haja uma boa aproximação inicial e esta seja relativamente próxima da solução verdadeira.

Porém a determinação desta aproximação inicial as vezes pode ser dificultosa pois nem sempre é possível a previsão dos fenômenos físicos envolvidos nos problemas, tornando isto uma desvantagem do método. Outra desvantagem do método é o alto custo computacional dos cálculos da Matriz Jacobiana.

1. Referências

Burden, R. L. (2011). *Numerical Analysis* (9th Edition ed.). Boston: Cengage Learning.

Burden, R. L. (2015). *Análise Numérica* (Tradução da 10ª edição norte-americana ed.). São Paulo: Cengage Learning.

1. Código Fonte

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \brief Programa realizado para a Aula de Métodos Númericos em Fenômenos de Transporte

\* Análise Numérica - Burden, Richard - 10ª edição - Pág. 716

\* Algoritmo 10.1 - Método de Newton-Rhapson para Solução de Sistemas de Equações Não Lineares

\*

\* \param Número de Equações n

\* \param Aproximações Iniciais w[n]

\* \return Solução Aproximada w[n]

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <locale.h>

#define max(a,b) (a > b) ? a : b /\*função máximo\*/

#define tol 10e-16 /\*valor mínimo de tolerância = 10e-16\*/

#define n\_max 100 /\*valor máximo para iterações = 100\*/

/\*declaração das funções\*/

void esc\_piv(int, double \*\*, int \*, double \*);

void sistema(int, double \*\*, int \*, double \*, double \*);

void df(int, int, double \*, double \*\*, double);

double funcao(int, double \*);

int main(void)

{

setlocale(LC\_ALL,"");

double \*\*J, \*w, \*\_w, \*F, \*X;

int \*L;

int k, i, n;

double h, erro, erro\_max;

/\*Definição do h para o calculo da derivação numérica\*/

h = tol \* 10;

if(h <= 1e-8){h = 0.00000001;}

/\*Entrada de parâmetros\*/

printf("Digite o número de incógnitas do sistema [n]: "); scanf("%d", &n);

J = calloc((n+1), sizeof(double \*));

for (i=0; i<=n; i++){J[i] = calloc((n+1), sizeof(double));}

w = calloc((n+1), sizeof(double));

\_w = calloc((n+1), sizeof(double));

F = calloc((n+1), sizeof(double));

X = calloc((n+1), sizeof(double));

L = calloc((n+1), sizeof(int));

printf("Digite as aproximações iniciais para o sistema w[n]: \n");

for (i=1; i<=n; i++){printf("w[%d]= ", i); scanf ("%lf", &w[i]); \_w[i] = w[i];}

printf("\n");

for(k=0; k<=n\_max; k++)

{

/\*Exibição dos valores calculados para o usuário\*/

printf("k= %d", k);

for (i = 1; i <= n; i++){printf("\tw[%d]= %+.8f", i, w[i]);}

printf("\terro= %+.8f", erro\_max);

printf("\n");

/\*Cálculo do vetor -F(x)\*/

for (i=1; i<=n; i++){F[i] = - funcao(i, w);}

/\*Cálculo da Matriz Jacobiana\*/

df(k, n, w, J, h);

/\*Cálculo da solução do Sistema Linear\*/

esc\_piv(n, J, L, F);

sistema(n, J, L, F, X);

/\*Cálculo do novo w\*/

for (i=1; i<=n; i++){w[i] = X[i] + w[i];}

/\*Cálculo do Resíduo\*/

erro\_max = 0;

for (i=1; i<=n; i++){erro = fabs(funcao(i, w)); erro\_max = max(erro\_max, erro);}

/\*Critérios de parada\*/

if(erro\_max<=tol){printf("\nO método atingiu a tolerância especificada.\n"); break;}

if(k>=n\_max){printf("\nO método atingiu o número máximo de iterações especificado.\n"); break;}

}

printf("\nFim do programa.\n\n");

for (i=0; i<=n; i++){free(J[i]);}

free(J); free(w); free(\_w); free(F); free(X); free(L);

system("PAUSE");

return 0;

}

/\*Cálculo dos valores da função\*/

double funcao(int f, double \*w)

{

switch (f)

{

case 1 :

/\*validação ex. 2 com valores iniciais [0.1,0.1,-0.1]\*/

return(3 \* w[1] - cos(w[2] \* w[3]) - 0.5);

/\*Exercício 10.2-1c com valores iniciais [0,0,0]

return(w[1] \* (1 - w[1]) + 4 \* w[2] - 12);\*/

/\*Exercício 10.2-1d com valores iniciais [0,0,0]

return(5 \* pow(w[1], 2) - pow(w[2], 2));\*/

/\*Exercício 10.2-2a com valores iniciais [0,0,0]

return(3 \* w[1] - cos(w[2] \* w[3]) - 0.5);\*/

/\*Exercício 10.2-2b com valores iniciais [0,0,0]

return(pow(w[1], 2) + w[2] - 37);\*/

/\*Exercício 10.2-8a com valores iniciais [1,1,1,1]

return(4 \* w[1] - w[2] + w[3] - w[1] \* w[4]);\*/

/\*Exercício 10.2-10a com valores iniciais [0,0,0]

return(w[1] \* exp(w[2] \* 1) + w[3] \* 1 - 10);\*/

break;

case 2:

/\*validação ex. 2 com valores iniciais [0.1,0.1,-0.1]\*/

return(pow(w[1], 2) - 81 \* pow((w[2] + 0.1), 2) + sin(w[3]) + 1.06);

/\*Exercício 10.2-1c com valores iniciais [0,0,0]

return(pow(w[1] - 2, 2) + pow(2 \* w[2] - 3, 2) - 25);\*/

/\*Exercício 10.2-1d com valores iniciais [0,0,0]

return(w[2] - 0.25 \* (sin(w[1]) + cos(w[2])));\*/

/\*Exercício 10.2-2a com valores iniciais [0,0,0]

return(4 \* pow(w[1], 2) - 625 \* pow(w[2], 2) + 2 \* w[2] - 1);\*/

/\*Exercício 10.2-2b com valores iniciais [0,0,0]

return(w[1] - pow(w[2], 2) - 5);\*/

/\*Exercício 10.2-8a com valores iniciais [1,1,1,1]

return(w[1] - 2 \* w[2] + 3 \* w[3] - w[3] \* w[4]);\*/

/\*Exercício 10.2-10a com valores iniciais [0,0,0]

return(w[1] \* exp(w[2] \* 2) + w[3] \* 2 - 12);\*/

break;

case 3:

/\*validação ex. 2 com valores iniciais [0.1,0.1,-0.1]\*/

return(exp(-w[1] \* w[2]) + 20 \* w[3] + (10 \* M\_PI - 3) / 3);

/\*Exercício 10.2-2a com valores iniciais [0,0,0]

return(exp(-w[1] \* w[2]) + 20 \* w[3] + (10 \* M\_PI - 3) / 3);\*/

/\*Exercício 10.2-2b com valores iniciais [0,0,0]

return(w[1] + w[2] + w[3] - 3);\*/

/\*Exercício 10.2-8a com valores iniciais [1,1,1,1]

return(- w[1] + 3 \* w[2] - 2 \* w[3] - w[2] \* w[4]);\*/

/\*Exercício 10.2-10a com valores iniciais [0,0,0]

return(w[1] \* exp(w[2] \* 3) + w[3] \* 3 - 15);\*/

break;

case 4:

/\*Exercício 10.2-8a com valores iniciais [1,1,1,1]

return(pow(w[1], 2) + pow(w[2], 2) + pow(w[3], 2) - 1);\*/

break;

default:

return 0;

}

}

/\*Cálculo da Matriz Jacobiana\*/

void df(int k, int n, double \*w, double \*\*J, double h)

{

double \*\_w1, \*\_w2, \*\_w3, \*\_w4;

\_w1 = calloc((n+1), sizeof(double \*));

\_w2 = calloc((n+1), sizeof(double \*));

\_w3 = calloc((n+1), sizeof(double \*));

\_w4 = calloc((n+1), sizeof(double \*));

int i, j, \_i;

for (i = 1; i <= n ; i++)

{

for (j = 1; j <= n; j++)

{

for (\_i = 1; \_i <= n; \_i++){\_w1[\_i] = w[\_i]; \_w2[\_i] = w[\_i]; \_w3[\_i] = w[\_i]; \_w4[\_i] = w[\_i];}

\_w1[j] = \_w1[j] - 2 \* h;

\_w2[j] = \_w2[j] - h;

\_w3[j] = \_w3[j] + h;

\_w4[j] = \_w4[j] + 2 \* h;

J[i][j] = (funcao(i, \_w3) - funcao(i, w)) / h;

/\*J[i][j] = (funcao(i, \_w3) - funcao(i, \_w2)) / (2 \* h);\*/

/\*J[i][j] = (funcao(i, \_w1) - 8 \* funcao(i, \_w2) + 8 \* funcao(i, \_w3) - funcao(i, \_w4)) / (12 \* h);\*/

}

}

}

/\*Pivotamento parcial de colunas com escalonamento\*/

void esc\_piv(int n, double \*\*A, int \*L, double \*B)

{

int i, j, k, \_i, \_k;

double S[n+1];

double xmult, smax, rmax, ratio;

/\*Definição do fator de escala s\*/

for (i = 1; i <= n; i++)

{

L[i] = i;

smax = 0.0;

for (j = 1; j <= n; j++){smax = max (smax, fabs(A[i][j]));}

S[i] = smax;

}

/\*Pivotamento de Colunas com escala \*/

for (k = 1; k < n; k++)

{

rmax = 0.0;

for (i = k ; i <= n; i++)

{

\_i = L[i];

ratio = fabs(A[\_i][k]) / S[\_i];

if (ratio > rmax)

{

rmax = ratio;

j = i;

}

}

/\*Troca a referência das linhas\*/

\_k = L[j];

L[j] = L[k];

L[k] = \_k;

for (i = k+1; i <= n; i++)

{

\_i = L[i];

xmult = A[\_i][k] / A[\_k][k];

/\*printf("\n\tmultiplicador[%d,%d]= %+.8f", i, k, xmult);\*/

A[\_i][k] = 0.0;

for (j = k+1; j<= n; j++){A[\_i][j] -= xmult \* A[\_k][j];}

B[\_i] -= xmult \* B[\_k];

}

}

}

/\*Cálculo da solução do sistema linear X\*/

void sistema(int n, double \*\*A, int \*L, double \*B, double \*X)

{

int i, j, k, \_i, \_k, \_n;

double soma;

for (k = 1; k < n ; k++)

{

\_k = L[k];

for (i = k+1; i <= n; i++)

{

\_i = L[i];

B[\_i] -= A[\_i][k] \* B[\_k];

}

}

\_n = L[n];

X[n] = B[\_n] / A[\_n][n];

for (i = n-1; i>= 1; i--)

{

\_i = L[i];

soma = B[\_i];

for (j = i+1; j <= n ; j++){soma -= A[\_i][j] \* X[j];}

X[i] = soma / A[\_i][i];

}

}