UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULADE DE ENGENHARIA MECÂNICA

RELATÓRIO DE ALGORITMO IMPLEMENTADO

CÁLCULO DE SOLUÇÕES DE EQUAÇÕES DE UMA VARIÁVEL

MÉTODO DE NEWTON – RAPHSON

Aluno: Diego Fernando Luque Martin

Matrícula: 191606

Campinas

2017

SUMÁRIO

1. INTRODUÇÃO

Este relatório tem por objetivo demonstrar por meio de um algoritmo implementado em linguagem de programação C, o cálculo de soluções de equações com uma variável utilizando o método de Newton - Raphson com a apresentação numérica dos resultados de raízes de uma função.

2. MÉTODO

Método de Newton – Raphson é um dos métodos mais eficiente e conhecido para a solução de um problema de determinação de uma raiz.

Suponha que f ∈ C2 [a, b]. Seja p0 ∈ [a, b] uma aproximação da solução p de f (x) = 0 tal que f 0 (p0) 6= 0 e |p − p0| seja “pequeno”. Considere o polinômio de Taylor de primeiro grau para f (x), expandido em torno de p0 e calculado em x = p:

$$f(p) = f(p_0) + (p - p_0)f'(p_0) + \frac{(p - p_0)^2}{2} f''\big(\xi(p)\big)$$

Onde ξ(p) está entre p e p_0. Uma vez que f(p) = 0, esta equação fornece:

$$0 = f(p_0) + (p - p_0)f'(p_0) + \frac{(p - p_0)^2}{2} f''\big(\xi(p)\big)$$

Após reduzir a equação, finalmente chegamos em:

$$P_n = P_{n-1} - \frac{f(p_{n-1})}{f'(P_{n-1})}, \text{para } n \geq 1$$

2.1. Algoritmo

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

ENTRADA: aproximação p_0; tolerância TOL; número máximo de iterações N_0.

SAÍDA: solução aproximada p ou mensagem de erro

Passo 1: Faça i = 1;

Passo 2: Enquanto i ≤ N_0, execute os Passos 3 a 6.

Passo 3: Faça P = p_0 – f(p_0) / f’(p_0) (Calcule p_i.)

Passo 4: Se | p – p_0 | < TOL, então

SAÍDA (p); (Procedimento concluído com sucesso.)

PARE.

Passo 5: Faça i = i + 1.

Passo 6: Faça p_0 = p. (Atualize p_0.)

Passo 7: SAÍDA (‘O método falhou após N_0 iterações, N_0= ‘, N_0);
(O procedimento não foi bem-sucedido.)
PARE.

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

- Estimativa Inicial p_0;
- Tolerância ou Erro desejado;
- Número máximo de iterações.

Para fins de facilitação para o usuário, decidiu-se que o valor de tolerância e o número máximo de iterações serão parâmetros internos do algoritmo cabendo ao usuário apenas informar e estimativa do valor p_0.

Após realizadas as iterações de cálculo, o algoritmo deverá apresentar ao final de sua execução uma das seguintes alternativas:

- Resultado da raiz com base na estimativa inicial;
- Erro devido a inexistência de raízes ou o número máximo de iterações atingido.

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

- Linguagem de Programação: C;
- Compilador: Code::Blocks 16.01

2.2. Teste

Para teste do código fonte escrito em linguagem de programação C, foi usado exemplo de cálculo

$$f(x) = x^3 + 4x^2 - 10; \quad p_0 = 2; \quad tol. = 10^{-16}$$

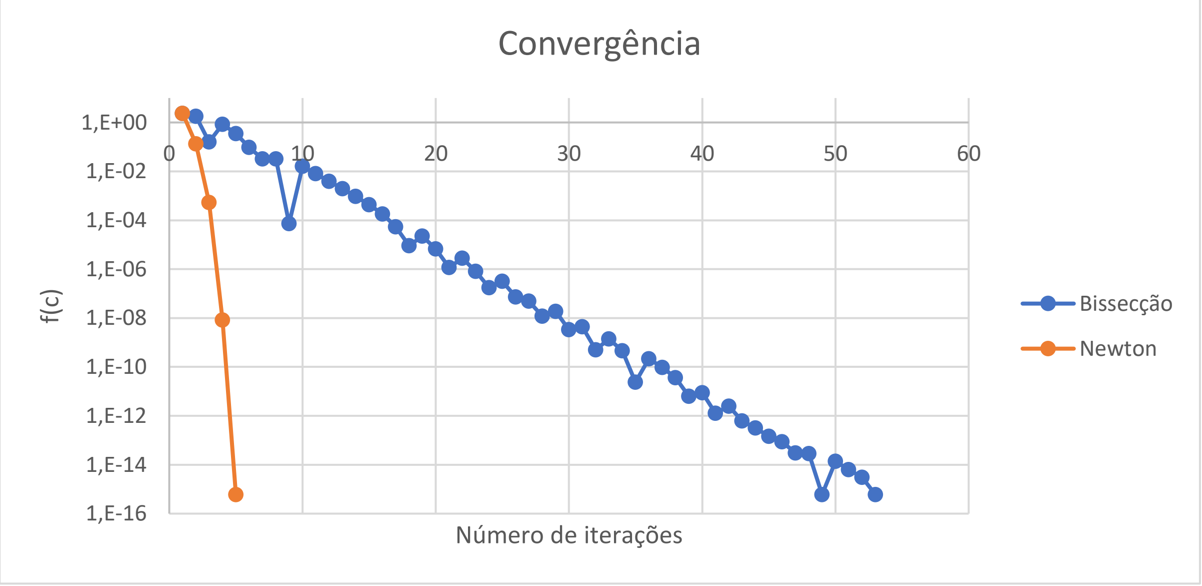
Resultados:

```
Digite um número para a Aproximação P0: 2
i: 1      p0: 2,0000000000000000          p1: 1,5000000000000000          f(p1): 2,3750000000000000
i: 2      p0: 1,5000000000000000          p1: 1,3733333333333333          f(p1): 0,1343454814814808
i: 3      p0: 1,3733333333333333          p1: 1,3652620148746266          f(p1): 0,0005284611795159
i: 4      p0: 1,3652620148746266          p1: 1,3652300139161466          f(p1): 0,0000000082905482
i: 5      p0: 1,3652300139161466          p1: 1,3652300134140969          f(p1): 0,0000000000000006

Raiz encontrada para P0 = 2,0000000000000000: 1,3652300134140969
```

A título de comparação com outros métodos para cálculo de soluções de equações de uma variável, foi possível traçar um gráfico tendo o número de iterações no eixo horizontal e o valor de $f(p_1)$, no eixo vertical em escala logarítmica de base 10. Desta maneira pode se observar a característica de Convergência do resultado do método de cálculo como demostrado na figura 2:

Figura 1



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 e 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 $i = 7$. 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.

2.3. Validação

Para validação do algoritmo escrito foi utilizado o exemplo de cálculo demonstrado no livro Análise Numérica, (Burden, 2015), página 74, onde o autor demostra a resolução de uma função (raiz) para valor inicial e tolerância como dado abaixo:

$$f(x) = \cos(x) - x; \quad p_0 = \frac{\pi}{4}; \quad \text{tol.} = 10^{-16}$$

Os resultados obtidos pelo autor estão na tabela 1 abaixo:

Tabela 1 (Burden, 2011)

Newton's Method	
n	p_n
0	0.7853981635
1	0.7395361337
2	0.7390851781
3	0.7390851332
4	0.7390851332

Utilizando os mesmos critérios de aproximação e tolerância do exemplo acima e aplicando ao algoritmo, obteve se os seguintes resultados:

```
Digite um número para a Aproximação P0: 0,78539816339744830961566084581988
i: 1     p0: 0,7853981633974483          p1: 0,7395361335152383          f(p1): -0,0007548746825027
i: 2     p0: 0,7395361335152383          p1: 0,7390851781060102          f(p1): -0,0000000751298664
i: 3     p0: 0,7390851781060102          p1: 0,7390851332151611          f(p1): -0,0000000000000008

Raiz encontrada para P0 = 0,7853981633974483: 0,7390851332151611
```

Com base nos resultados da iteração 3, o algoritmo foi considerado válido pois o valor de p_n, correspondentes ao valor do resultado do algoritmo p_1 tem igualdade em todas as casas decimais apresentadas pelo autor salvo arredondamentos.

3. EXERCÍCIOS RESOLVIDOS

Exercício 2.3 – 1 (Burden, 2015)

$$f(x) = x^2 - 6; \quad p_0 = 1 \;\; ; \quad p_n = 2$$

```
Digite um número para a Aproximação P0: 1
i: 1     p0: 1,0000000000000000          p1: 3,5000000000000000          f(p1): 6,2500000000000000
i: 2     p0: 3,5000000000000000          p1: 2,6071428571428572          f(p1): 0,7971938775510208

Erro! O número máximo de iterações foi atingido.

Erro! Não encontrada raiz da função!
```

Exercício 2.3 – 6a (Burden, 2015)

$$f(x) = e^x - 2^{-x} + 2\cos x - 6; \quad p_0 = 2; \quad tol. = 10^{-5}$$

```
Digite um número para a Aproximação P0: 1
i: 1     p0: 1,0000000000000000          p1: 3,4697980105110040          f(p1): 24,3272653042279250
i: 2     p0: 3,4697980105110040          p1: 2,7261264691776739          f(p1): 7,5948841802975764
i: 3     p0: 2,7261264691776739          p1: 2,1972944842252793          f(p1): 2,0460522155296847
i: 4     p0: 2,1972944842252793          p1: 1,9142730842143285          f(p1): 0,3737870366932667
i: 5     p0: 1,9142730842143285          p1: 1,8349957966533688          f(p1): 0,0231269497615340
i: 6     p0: 1,8349957966533688          p1: 1,8294098740815743          f(p1): 0,0001077575818359
i: 7     p0: 1,8294098740815743          p1: 1,8293836025124592          f(p1): 0,0000000023731708

Raiz encontrada para P0 = 1,0000000000000000: 1,8293836025124592
```

Exercício 2.3 – 6b (Burden, 2015)

$$f(x) = \ln(x-1) + \cos(x-1); \quad p_0 = 2; \quad tol. = 10^{-5}$$

```
Digite um número para a Aproximação P0: 1,5
i: 1     p0: 1,5000000000000000          p1: 1,3787067743061219          f(p1): -0,0418495131563380
i: 2     p0: 1,3787067743061219          p1: 1,3971358132924878          f(p1): -0,0013043764958595
i: 3     p0: 1,3971358132924878          p1: 1,3977478369912424          f(p1): -0,0000013589627534

Raiz encontrada para P0 = 1,5000000000000000: 1,3977478369912424
```

Exercício 2.3 – 6f (Burden, 2015)

$$f(x) = sen(x) - e^{-x}; \quad p_0 = 0, p_0 = 3, p_0 = 6; \quad tol. = 10^{-5}$$

```
Digite um número para a Aproximação P0: 0
i: 1     p0: 0,0000000000000000          p1: 0,5000000000000000          f(p1): -0,1271051211084304
i: 2     p0: 0,5000000000000000          p1: 0,5856438169664325          f(p1): -0,0040112772059720
i: 3     p0: 0,5856438169664325          p1: 0,5885294126263548          f(p1): -0,0000046202542184
i: 4     p0: 0,5885294126263548          p1: 0,5885327439774188          f(p1): -0,0000000000061610

Raiz encontrada para P0 = 0,0000000000000000: 0,5885327439774188

Digite um número para a Aproximação P0: 3
i: 1     p0: 3,0000000000000000          p1: 3,0971414724372450          f(p1): -0,0007416169800560
i: 2     p0: 3,0971414724372450          p1: 3,0963639608966513          f(p1): -0,0000000271689278

Raiz encontrada para P0 = 3,0000000000000000: 3,0963639608966513

Digite um número para a Aproximação P0: 6
i: 1     p0: 6,0000000000000000          p1: 6,2928317995508252          f(p1): 0,0077968276949756
i: 2     p0: 6,2928317995508252          p1: 6,2850490041427438          f(p1): -0,0000002697412288

Raiz encontrada para P0 = 6,0000000000000000: 6,2850490041427438
```

Exercício 2.3 – 13 (Burden, 2015)

$$f(x) = 230x^4 + 18x^3 + 9x^2 - 221x - 9; \quad p_0 = -1, p_0 = 1; \quad tol. = 10^{-6}$$

```
Digite um número para a Aproximação P0: -1
i: 1     p0: -1,0000000000000000        p1: -0,6081447963800905        f(p1): 156,1398567030929900
i: 2     p0: -0,6081447963800905        p1: -0,2354054694806967        f(p1): 43,9948452117208790
i: 3     p0: -0,2354054694806967        p1: -0,0475910794229643        f(p1): 1,5372523969669381
i: 4     p0: -0,0475910794229643        p1: -0,0406613217777537        f(p1): 0,0004508275807656
i: 5     p0: -0,0406613217777537        p1: -0,0406592883159283        f(p1): 0,0000000000375695

Raiz encontrada para P0 = -1,0000000000000000: -0,0406592883159283

Digite um número para a Aproximação P0: 1
i: 1     p0: 1,0000000000000000         p1: 0,9649805447470817         f(p1): 1,7297027806240781
i: 2     p0: 0,9649805447470817         p1: 0,9624117249792600         f(p1): 0,0088676620196079
i: 3     p0: 0,9624117249792600         p1: 0,9623984191063186         f(p1): 0,0000002370939999

Raiz encontrada para P0 = 1,0000000000000000: 0,9623984191063186
```

Exercício 2.3 – 18 (Burden, 2015)

$$f(x) = \frac{1}{2} + \frac{1}{4}x^2 - x \cdot sen(x) - \frac{1}{2}\cos(2x)\,; \quad p_0 = \frac{\pi}{2}; \quad tol. = 10^{-5}$$

```
Digite um número para a Aproximação P0: 1,5707963267948966192313216916398
i: 1     p0: 1,5707963267948966         p1: 1,#INF000000000000         f(p1): -1,#IND000000000000
i: 2     p0: 1,#INF000000000000         p1: -1,#IND000000000000        f(p1): -1,#IND000000000000
i: 3     p0: -1,#IND000000000000        p1: -1,#IND000000000000        f(p1): -1,#IND000000000000
⋮
i: 100   p0: -1,#IND000000000000        p1: -1,#IND000000000000        f(p1): -1,#IND000000000000

Erro! O número máximo de iterações foi atingido.

Erro! Não encontrada raiz da função!
```

O que pode ser observado é que os resultados não convergem a raiz da função por que a função e sua derivada possuem as mesmas raízes. Conforme o método se aproxima da raiz, a divisão por um número próximo de zero estoura a capacidade de armazenamento da variável tipo double.

4. CONCLUSÃO

O método numérico de Newton - Raphson é relativamente mais complexo se comparado com outros métodos de cálculo de raízes de funções. Requer conhecimento sobre Cálculo Numérico para determinação das derivadas de funções.

Para ser aplicado requer o conhecimento de um valor estimado da raiz da função, valor este que deve ser muito mais próximo da raiz se comparado com os valores estimados de intervalo utilizados no método da bissecção. Tornando assim um método mais utilizado para o refinamento do resultado de outros métodos como, por exemplo, o da bissecção.

Uma das desvantagens analisadas através dos exercícios resolvidos foi que para funções com inclinação com um valor próximo da raiz da função, ou seja, a raiz da função e a raiz da derivada da função tem o mesmo valor ou valores muito próximos, o método diverge da raiz desejada.

Conhecendo esta estimativa e os limitantes de aplicação, o método converge para uma raiz de forma rápida se comparado a outros métodos (tabela 2), pois possui uma convergência de ordem quadrática, ou seja, a quantidade de dígitos significativos duplica à medida que os valores da sequência se aproximam da raiz da função.

Tabela 2

$\mathrm{f(x)} = \mathrm{x}^3 + 4\mathrm{x}^2 - 10; \quad \mathrm{Intervalo} = [1,2]; \quad \mathrm{tol.} = 10^{-16}$

Método	**Convergência**	**Iterações**
Bissecção	Linear	53
Newton	Quadrática	5

5. 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.

ANEXO A. CÓDIGO FONTE

```
 /*********************************************
 * \brief   Programa realizado para a Aula de Métodos Númericos em Fenômenos de Transporte
 *          Análise Numérica - Burden, Richard - 10ª edição - Pág. 73
 *          Algoritmo 2.3 - Método de Newton Raphson
 *
 * \param   Aproximação Inicial P0
 * \return  Raiz da função
 *
 *********************************************/

#include <stdio.h>
#include <math.h>
#include <locale.h>

#define tol 10e-16  /*valor mínimo de tolerância = 10e-16*/
#define n_max 100   /*valor máximo para iterações = 100*/

double f(double x){
    /*validação ex. 1 no ponto p0 = 0,785398*/  return (cos(x)-x);
    /*comparação com método da bissecção*/      /*return (pow(x,3)+4*pow(x,2)-10);*/
    /*exercicio 2.3-1 no ponto p0=1 */          /*return (pow(x,2) - 6);*/
    /*exercicio 2.3-6a no ponto p0=2 */         /*return (exp(x) + pow(2,-x) + 2 * cos(x) - 6);*/
    /*exercicio 2.3-6b no ponto p0=2 */         /*return (log(x - 1) + cos(x - 1));*/
    /*exercicio 2.3-6f no ponto p0=0, 3 e 6 */  /*return (sin(x) - exp(-x));*/
    /*exercicio 2.3-13 no ponto p0=-1 e 0*/     /*return (230 * pow(x,4) + 18 * pow(x,3) + 9 * pow(x,2)
- 221 * x - 9);*/
    /*exercicio 2.3-18 no ponto p0=pi()/2*/     /*return (1/2 + 1/4 * pow(x,2) - x * sin(x) - 1/2 * cos(2
* x));*/
}
double df(double x){
    /*validação ex. 1 no intervalo [1,2]*/      return (-sin(x)-1);
    /*comparação com método da bissecção*/      /*return (3 * pow(x,2) + 8 * x);*/
    /*exercicio 2.3-1 no ponto p0=1 */          /*return (exp(x) - log(2) / pow(2,x) - 2 * sin(x));*/
    /*exercicio 2.3-6b no ponto p0=2 */         /*return (1 / (x - 1) - sin(x - 1));*/
    /*exercicio 2.3-6f no ponto p0=0, 3 e 6 */  /*return (cos(x) + exp(-x));*/
    /*exercicio 2.3-13 no ponto p0=-1 e 0*/     /*return (920 * pow(x,3) + 54 * pow(x,2) + 18 * x -
221);*/
    /*exercicio 2.3-18 no ponto p0=pi()/2*/     /*return (-1/2 * ( x - 2 * sin(x))*(2 * cos(x) - 1));*/
}

double Newton(double p0){
    double p1;
    int i = 1;

    do{
        p1 = p0 - (f(p0) / df(p0));

        printf("i: %i \t p0: %.16f \t p1: %.16f \t f(p1): %.16f \n", i, p0, p1, f(p1));

        if (fabs(f(p1)) <= tol) {return p1;} else {p0 = p1;}

        i++;
    }while( i <= n_max);

    printf("\nErro! O número máximo de iterações foi atingido.\n\n");
    return p1;
}

int main(void){

    setlocale(LC_ALL,"");

    double p, p0;

    printf("Digite um número para a Aproximação P0: ");
    scanf("%lf", &p0);

    p = Newton(p0);
```

```
    if (fabs(f(p)) <= tol)
        {printf("\nRaiz encontrada para P0 = %.16f: %.16f \n\n", p0, p);}
    else
        {printf("\nErro! Não encontrada raiz da função!\n\n");}

    return 0;
}
```