



Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Departamento de Engenharia da Computação e Automação
DCA0304 – Métodos Computacionais em Engenharia

**Comparação entre Julia e outras linguagens de programação na
eficiência de execução do método de Newton-Raphson para
solução de sistema de equações não-lineares**

André Rodrigues Bezerra Madruga
Bruno Matias de Sousa
José Ricardo Bezerra de Araújo
Levy Gabriel da Silva Galvão

Novembro
2018



Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Departamento de Engenharia da Computação e Automação
DCA0304 – Métodos Computacionais em Engenharia

Comparação entre Julia e outras linguagens de programação na eficiência de execução do método de Newton-Raphson para solução de sistema de equações não-lineares

Relatório técnico referente à execução prática dos métodos numéricos para a solução de sistemas de equações não-lineares realizado na disciplina de Métodos Computacionais em Engenharia, como requisito parcial para avaliação da terceira unidade da disciplina antes mencionada.

Orientador: Prof^o. Dr^o. Paulo Sergio da Motta Pires

Novembro
2018

Sumário

1	Introdução	2
2	Desenvolvimento	3
2.1	Newton-Raphson para sistema de equações não-lineares	3
2.2	SENL propostos	3
2.3	Fluxograma do pseudocódigo	4
3	Resultados	5
3.1	Raízes	5
3.2	Eficiência na execução	5
4	Conclusões	6
5	Apêndice	8
5.1	Fortran	8
5.2	Julia	12
5.3	Python	15

Resumo

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque et gravida mauris. Phasellus at ipsum in nisl iaculis consequat. Fusce vulputate nisl ipsum, quis egestas justo accumsan et. Morbi consequat tellus a eros eleifend congue. Aenean laoreet mattis nunc, at iaculis orci imperdiet in. Donec a diam in sem auctor fringilla. Aenean euismod odio vel arcu pretium, in vehicula urna ultricies.

Palavras-chaves: métodos. computacionais. engenharia.

1 Introdução

Muitos problemas da ciência da computação e de outras ciências podem ser abstraídos por meio de fórmulas e equações provenientes de uma linguagem matemática. Algumas dessas equações podem ser solucionadas de forma analítica utilizando a conceituação da literatura da área. Porém muitos outros problemas não possuem solução fechada por um método analítico, assim sendo necessário recorrer aos métodos iterativos, na maioria dos casos.

Ao longo dos anos os métodos iterativos solucionam problemas em diversas áreas, tais como: economia, engenharia, física, biologia, etc. Sua aplicação se baseia na aproximações para a solução do problema que melhoram em precisão de acordo com que aumentam o número de iterações. A extensa literatura na área de métodos iterativos só fortalece a importância de estudar a área.

A natureza repetitiva dos métodos iterativos sugere a sua execução em recursos computacionais. Assim, a atividade "manufaturada" de realizar os cálculos é transferida para um computador, capaz de executá-las mais rapidamente.

Dessa forma, surgiu com o tempo uma tendência cada vez maior – de acordo com que a tecnologia se desenvolvia – de aliar a solução de problemas matemáticos aos métodos computacionais. Principalmente aqueles cuja solução analítica é difícil ou impossível. Um exemplo são os problemas de sistemas de equações não lineares que serão abordados no presente trabalho.

Para a obtenção das soluções desejadas foram utilizadas as linguagens de programação para realizar a comunicação entre a linguagem humana e matemática e a linguagem binária de máquina. Existem diversas linguagens com os mais diversos propósitos. Uma aplicada ao gerenciamento de bancos de dados e outras com recursos dedicados aos métodos numéricos.

Com a pluralidade de escolhas, basta ao profissional escolher aquela linguagem que mais se adequa às suas necessidades. O mais procurado nos dias atuais na solução de problemas por métodos numéricos é a linguagem que seja mais rápida, que utilize menos recurso computacional e ofereça a resposta mais precisa.

Uma linguagem tida como forte candidata à preferida no cálculo numérico é a Julia. Uma linguagem bastante recente, cujo desenvolvimento começou em 2009 e teve a primeira versão de código aberto lançada em 2012.

2 Desenvolvimento

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque et gravida mauris. Phasellus at ipsum in nisl iaculis consequat. Fusce vulputate nisl ipsum, quis egestas justo accumsan et. Morbi consequat tellus a eros eleifend congue. Aenean laoreet mattis nunc, at iaculis orci imperdiet in. Donec a diam in sem auctor fringilla. Aenean euismod odio vel arcu pretium, in vehicula urna ultricies.

Vivamus ut pharetra diam. Aliquam metus sem, tristique ac dignissim eu, pretium id velit. Donec id tincidunt odio. Fusce vehicula ac est quis convallis. Nullam sollicitudin euismod dolor, eget blandit turpis hendrerit at. In bibendum suscipit odio, at consequat erat laoreet id. Donec in tellus at nulla gravida egestas. Suspendisse non elementum leo. Nam viverra sapien sed velit tempor scelerisque. Nunc accumsan odio eget mi vehicula, vel interdum libero gravida. Pellentesque vitae molestie diam, quis vestibulum libero. Aenean finibus sapien diam, ac sagittis magna placerat nec. Ut maximus eros felis, vel egestas diam convallis sit amet. Integer interdum elementum turpis, sit amet luctus nisi scelerisque at. Donec eleifend arcu dictum metus ullamcorper tempor. Aenean placerat, arcu id suscipit vehicula, velit ipsum viverra lorem, et pretium velit tellus quis libero.

2.1 Newton-Raphson para sistema de equações não-lineares

Nunc accumsan odio eget mi vehicula, vel interdum libero gravida. Pellentesque vitae molestie diam, quis vestibulum libero. Aenean finibus sapien diam, ac sagittis magna placerat nec. Ut maximus eros felis, vel egestas diam convallis sit amet. Integer interdum elementum turpis, sit amet luctus nisi scelerisque at. Donec eleifend arcu dictum metus ullamcorper tempor. Aenean placerat, arcu id suscipit vehicula, velit ipsum viverra lorem, et pretium velit tellus quis libero.

2.2 SENL propostos

Nunc accumsan odio eget mi vehicula, vel interdum libero gravida. Pellentesque vitae molestie diam, quis vestibulum libero. Aenean finibus sapien diam, ac sagittis magna placerat nec. Ut maximus eros felis, vel egestas diam convallis sit amet. Integer interdum elementum turpis, sit amet luctus nisi scelerisque at. Donec eleifend arcu dictum metus ullamcorper tempor. Aenean placerat, arcu id suscipit vehicula, velit ipsum viverra lorem, et pretium velit tellus quis libero.

2.3 Fluxograma do pseudocódigo

Nunc accumsan odio eget mi vehicula, vel interdum libero gravida. Pellentesque vitae molestie diam, quis vestibulum libero. Aenean finibus sapien diam, ac sagittis magna placerat nec. Ut maximus eros felis, vel egestas diam convallis sit amet. Integer interdum elementum turpis, sit amet luctus nisi scelerisque at. Donec eleifend arcu dictum metus ullamcorper tempor. Aenean placerat, arcu id suscipit vehicula, velit ipsum viverra lorem, et pretium velit tellus quis libero.

3 Resultados

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque et gravida mauris. Phasellus at ipsum in nisl iaculis consequat. Fusce vulputate nisl ipsum, quis egestas justo accumsan et. Morbi consequat tellus a eros eleifend congue. Aenean laoreet mattis nunc, at iaculis orci imperdiet in. Donec a diam in sem auctor fringilla. Aenean euismod odio vel arcu pretium, in vehicula urna ultricies.

Vivamus ut pharetra diam. Aliquam metus sem, tristique ac dignissim eu, pretium id velit. Donec id tincidunt odio. Fusce vehicula ac est quis convallis. Nullam sollicitudin euismod dolor, eget blandit turpis hendrerit at. In bibendum suscipit odio, at consequat erat laoreet id. Donec in tellus at nulla gravida egestas. Suspendisse non elementum leo. Nam viverra sapien sed velit tempor scelerisque. Nunc accumsan odio eget mi vehicula, vel interdum libero gravida. Pellentesque vitae molestie diam, quis vestibulum libero. Aenean finibus sapien diam, ac sagittis magna placerat nec. Ut maximus eros felis, vel egestas diam convallis sit amet. Integer interdum elementum turpis, sit amet luctus nisi scelerisque at. Donec eleifend arcu dictum metus ullamcorper tempor. Aenean placerat, arcu id suscipit vehicula, velit ipsum viverra lorem, et pretium velit tellus quis libero.

3.1 Raízes

Nunc accumsan odio eget mi vehicula, vel interdum libero gravida. Pellentesque vitae molestie diam, quis vestibulum libero. Aenean finibus sapien diam, ac sagittis magna placerat nec. Ut maximus eros felis, vel egestas diam convallis sit amet. Integer interdum elementum turpis, sit amet luctus nisi scelerisque at. Donec eleifend arcu dictum metus ullamcorper tempor. Aenean placerat, arcu id suscipit vehicula, velit ipsum viverra lorem, et pretium velit tellus quis libero.

3.2 Eficiência na execução

Nunc accumsan odio eget mi vehicula, vel interdum libero gravida. Pellentesque vitae molestie diam, quis vestibulum libero. Aenean finibus sapien diam, ac sagittis magna placerat nec. Ut maximus eros felis, vel egestas diam convallis sit amet. Integer interdum elementum turpis, sit amet luctus nisi scelerisque at. Donec eleifend arcu dictum metus ullamcorper tempor. Aenean placerat, arcu id suscipit vehicula, velit ipsum viverra lorem, et pretium velit tellus quis libero.

4 Conclusões

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque et gravida mauris. Phasellus at ipsum in nisl iaculis consequat. Fusce vulputate nisl ipsum, quis egestas justo accumsan et. Morbi consequat tellus a eros eleifend congue. Aenean laoreet mattis nunc, at iaculis orci imperdiet in. Donec a diam in sem auctor fringilla. Aenean euismod odio vel arcu pretium, in vehicula urna ultricies.

Vivamus ut pharetra diam. Aliquam metus sem, tristique ac dignissim eu, pretium id velit. Donec id tincidunt odio. Fusce vehicula ac est quis convallis. Nullam sollicitudin euismod dolor, eget blandit turpis hendrerit at. In bibendum suscipit odio, at consequat erat laoreet id. Donec in tellus at nulla gravida egestas. Suspendisse non elementum leo. Nam viverra sapien sed velit tempor scelerisque. Nunc accumsan odio eget mi vehicula, vel interdum libero gravida. Pellentesque vitae molestie diam, quis vestibulum libero. Aenean finibus sapien diam, ac sagittis magna placerat nec. Ut maximus eros felis, vel egestas diam convallis sit amet. Integer interdum elementum turpis, sit amet luctus nisi scelerisque at. Donec eleifend arcu dictum metus ullamcorper tempor. Aenean placerat, arcu id suscipit vehicula, velit ipsum viverra lorem, et pretium velit tellus quis libero.

Referências

Nenhuma citação no texto.

A. B. de Normas Técnicas. *NBR 10719: Apresentação de relatórios técnico-científicos*. 1989. Nenhuma citação no texto.

M. d. A. Marconi and E. M. Lakatos. *Fundamentos de metodologia científica*. 5. ed.-São Paulo: Atlas, 2003. Nenhuma citação no texto.

5 Apêndice

5.1 Fortran

```

1 program main
2     implicit none
3     double precision :: x0(3), x(size(x0)), tol=1e-12
4
5     x0(1) = 0.5
6     x0(2) = 0.5
7     x0(3) = 0.5
8
9     x = newton_raph(x0, tol, .true., 100)
10    write(*,*) x(1), x(2), x(3)
11
12
13
14 contains
15
16    function f(p) result(res)
17        IMPLICIT NONE
18        double precision :: p(:), res(size(p))
19
20        res(1) = p(2) + p(3) - exp(-p(1))
21        res(2) = p(1) + p(3) - exp(-p(3))
22        res(3) = p(1) + p(2) - exp(-p(3))
23    end function f
24
25    function jac(x) result(jc)
26        IMPLICIT NONE
27        double precision :: dx=1e-12, x(:), jc(size(x), size(x)),
28            xn(size(x)), dy(size(x)), dif(size(x))
29        integer :: n, i, k
30
31        n = size(x)
32        do k = 1,n
33            xn = x
34            xn(k) = xn(k)+dx
35            dy = f(xn) - f(x)
36            dif = dy/dx
37            do i = 1,n

```

```

37         jc(i, k) = dif(i)
38     end do
39 end do
40 end function jac
41
42 function LU(M, b) result(x)
43     IMPLICIT NONE
44     double precision :: b(:), M(:, :), x(size(b)), y(size(b))
45         , U(size(b), size(b)), L(size(b), size(b)), c
46     integer :: n, i, j, k
47
48     n = size(b)
49
50     do i = 1,n
51         do j = 1,n
52             L(i, j) = 0
53             U(i, j) = 0
54         end do
55     end do
56
57     do i = 1,n
58         x(i) = 0
59         y(i) = 0
60         L(i, i) = 1
61     end do
62
63     U = M
64
65     do i = 1,(n-1)
66         do k = (i+1),n
67             c = U(i, k) / U(i, i)
68             L(k, i) = c
69             do j = 1,n
70                 U(k, j) = U(k, j) - c*U(i, j)
71             end do
72         end do
73         do k = (i+1),n
74             U(k, i) = 0
75         end do
76     end do
77
78     do i = 1,n

```

```

77         y(i) = b(i) / L(i, i)
78         do k = 1,(i-1)
79             y(i) = y(i) - y(k)*L(i, k)
80         end do
81     end do
82
83     x = y
84
85     do i = n,1,-1
86         do k = (i+1),n
87             x(i) = x(i) - x(k)*U(i, k)
88         end do
89         x(i) = x(i)/U(i, i)
90     end do
91 end function LU
92
93 function newton_raph(x0, tol, iter, n_tot) result(x)
94     IMPLICIT NONE
95     double precision :: x0(:), x(size(x0)), tol, e, func(size
96         (x0)), S(size(x0)), jc(size(x0), size(x0))
97     integer :: n_tot, n0, n=0, i
98     logical :: iter
99
100
101     n0 = size(x0)
102
103     do i = 1,n0
104         x(i) = x0(i)
105     end do
106     e = maxval(abs(f(x)))
107
108     do while ((e>tol).and.(n<=n_tot))
109         n = n+1
110         func = f(x)
111         e = maxval(abs(func))
112         jc = jac(x)
113         if (size(func)==1) then
114             x = x - (func(1)/jc(1, 1))
115         else
116             S = LU(jc, -func)
117             x = x+S
118         end if

```

```
117         end do
118
119         if (iter .eqv. .true.) then
120             write(*,*) "Total de iteracoes: ", n
121         end if
122         if (n>=n_tot) then
123             write(*,*) "Processo parou, numero de iteracoes
124                 limite atingido", n
125         end if
126     end function newton_raph
127
128 end program main
129
130
131 }
```

5.2 Julia

```

1 function LU(matriz, vetor_b)
2     n = length(vetor_b)
3     x = zeros(n)
4     L = zeros(n, n)
5     for i = 1:n
6         L[i, i] = 1
7     end
8     U=zeros(n,n)
9     U = matriz
10
11
12     for i = 1:n-1
13         for k = i+1:n
14             c = U[k, i] / U[i, i]
15             L[k, i] = c
16             for j = 1:n
17                 U[k, j] = U[k, j] - c*U[i, j]
18             end
19         end
20         for k = i+1:n
21             U[k, i] = 0
22         end
23     end
24     y = zeros(n)
25     for i = 1:n
26         y[i] = vetor_b[i] / L[i, i]
27
28         for k = 1:i-1
29             y[i] = y[i] - y[k]*L[i, k]
30         end
31     end
32     n = length(y)
33     x = copy(y)
34     for i = (n:-1:1)
35         for k = 1+i:n
36             x[i] = x[i] - x[k]*U[i, k]
37         end
38         x[i] = x[i]/U[i, i]
39     end
40

```

```

41     return x
42 end
43
44 function f(p)
45     # sistema de equacoes
46     # eq1: x + y ^ 2 = 4
47     # eq2: e ^ x + xy = 3
48     a = p[2] + p[3] - exp(-p[1])
49     b = p[1] + p[3] - exp(-p[3])
50     c = p[1] + p[2] - exp(-p[3])
51     return [a c b]
52 end
53
54 function jac(x, dx=1e-10)
55     n = length(x)
56     J = zeros(n, n)
57     for j = 1:n
58         xn = copy(x)
59         xn[j] = xn[j] + dx
60         dy = f(xn) - f(x)
61         dif = dy/dx
62         for i = 1:n
63             J[i, j] = dif[i]
64         end
65     end
66     return J
67 end
68
69 function newton_raph(x0, tol, iter, n_tot)
70     tol = abs.(tol)
71     n_tot = abs.(n_tot)
72     x = convert(Array{Float64}, x0)
73     e = maximum(abs.(f(x)))
74     n = 0
75     while (e>tol)&(n<=n_tot)
76         n += 1
77         F = f(x)
78         e = maximum(abs.(F))
79         J = jac(x)
80         if length(F) == 1
81             x = x - F / J

```



```
82         else
83             S = LU(J, -F)
84             x = x+S
85         end
86     end
87     if iter==true
88         print("Total de Iteracoes: ", string(n))
89     end
90     if n>=n_tot
91         print("Processo parou, numero de iteracoes limite
92             atingido")
93     else
94         return x
95     end
96 end
97
98 x0 = [0.5, 1, 5]
99 x = newton_raph(x0, 1e-12, true, 100)
100 print("\nSolucao= ",x, "\n")
101
102
103 }
```

5.3 Python

```

1 import numpy as np
2 import time
3 from numpy import sin, exp
4 from math import pi
5
6 def f(p):
7     '''
8     Sistema de equacoes
9
10     eq1: 1/2*sin(x*y)-(y/(4*pi))-(x/2),
11     eq2: (1-1/(4*pi))*((exp(2*x))-exp(1))-((exp(1)*y)/pi)-2*
12           exp(1)*x)
13
14     eq3: y+z-exp(-x)
15     eq4: x+y-exp(-z)
16     eq5: x+z-exp(-z)
17     '''
18     x, y = p
19
20     return np.array(((1/2*sin(x*y)-(y/(4*pi)))-(x/2),
21                      (1 - 1 / (4 * pi)) * ((exp(2 * x)) - exp(1))
22                      - ((exp(1) * y) / pi) - 2 * exp(1) * x))
23
24 def jac(f, x, dx=1e-10):
25     x = np.array(x)
26     n = len(x)
27     J = np.zeros((n, n))
28     for j in range(n):
29         xn = np.copy(x)
30         xn[j] = xn[j] + dx
31         dy = np.array(f(xn)) - np.array(f(x))
32         dif = dy / dx
33         for i in range(n):
34             J[i, j] = dif[i]
35     return J
36
37 def LU(matriz, vetor_b):
38     n = len(matriz)

```

```

39     L = [[0 for i in range(n)] for i in range(n)]
40     for i in range(0, n):
41         L[i][i] = 1
42
43     U = [[0 for i in range(0, n)] for i in range(n)]
44     for i in range(0, n):
45         for j in range(0, n):
46             U[i][j] = matriz[i][j]
47     n = len(U)
48
49
50     for i in range(0, n-1):
51         for k in range(i + 1, n):
52             c = U[k][i] / U[i][i]
53             L[k][i] = c
54             for j in range(0, n):
55                 U[k][j] -= c * U[i][j]
56
57         for k in range(i + 1, n):
58             U[k][i] = 0
59     n = len(L)
60
61     y = [0 for i in range(n)]
62     for i in range(0, n, 1):
63         y[i] = vetor_b[i] / L[i][i]
64         for k in range(0, i, 1):
65             y[i] -= y[k] * L[i][k]
66
67     n = len(U)
68
69     x = [y[i] for i in range(0, n)]
70     for i in range(n-1, -1, -1):
71         for k in range(i+1, n):
72             x[i] -= x[k] * U[i][k]
73         x[i] /= U[i][i]
74     return x
75
76
77 def newton_raph(func, x0, tol, iter):
78     """
79     Parametros:

```

```

80
81         func:          //E uma funcao que deve conter todas as
                        equacoes do sistema.
82         x0:            //Uma lista ou array com chute inicial
                        para cada variavel.
83         tol:           //Determina a tolerancia minima dos
                        valores da matriz F como criterio de parada.
84                        Este valor deve ser proximo de zero,
                        logo e aceitavel tol = 1e-12.
85         iter:          //Representa um numero de iteracoes maximo
                        como criterio de parada.
86
87         """
88         tol = abs(tol)
89         iter = abs(iter)
90         x = (np.array(x0)).astype(np.float)
91         e = max(abs(np.array(func(x))))
92         n = 0
93         if iter == 0:
94             print('\nTotal de Iteracoes: ' + str(n))
95             return x
96         else:
97             while (e > tol and n <= iter):
98                 n += 1
99                 F = np.array(func(x))
100                e = max(abs(F))
101                J = jac(func, x)
102                if len(F) == 1:
103                    x = x - F / J
104                else:
105                    S = LU(J, -F)
106                    x = x + S
107
108                print('\nTotal de Iteracoes: ' + str(n))
109                if n >= iter:
110                    print("PROCESSO PAROU, numero de iteracoes limite
                        atingido!")
111                    return x
112                else:
113                    return x
114

```

```
115
116 if __name__ == "__main__":
117
118     x0 = [0.5, 0.5]
119     iter = 100
120     tol = 1e-12
121
122     x = newton_raph(f, x0, tol, iter)
123
124     print('Solucao: {} ' .format(x))
125
126     fim_sis = time.process_time()
127     print('Tempo gasto: {}'.format(fim_sis))
128
129 }
```