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 discplina antes mencionada.

Orientador: Profº. Drº. Paulo Sergio da Motta Pires

Novembro
2018

Sumário

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 serem 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 execuação 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 entra a linguagem humana e matemática e a linguagem binária de máquina. Existem diversas linguagens com os mais diversos propósitos. Umas aplicadas 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

```
program main
    implicit none
    double precision :: x0(3), x(size(x0)), tol, iter
    !Chute inicial
    x0(1) = 0.5
    x0(2) = 0.5
    x0(3) = 0.5
    iter = 100 !Numero de iteracoes
    tol = 1e-12 !Tolerancia

    x = newton_raph(x0, tol, .true., iter)
    write(*,*) x(1), x(2), x(3) !Solucao



contains

    !DECLARACAO DAS FUNCOES
    function f(p) result(res)
        IMPLICIT NONE
        double precision :: p(:), res(size(p))

        !Sistema de equacoes

        !eq1: 1/2*sin(x1*x2)-(x2/(4*pi))-(x1/2),
        !eq2: (1-1/(4*pi))*((exp(2*x1))-exp(1))-((exp(1)*x2)/pi)
           -2*exp(1)*x1)

        !eq3: x2+x3-exp(-x1)
        !eq4: x1+x3-exp(-x3)
        !eq5: x1+x2-exp(-x3)

        res(1) = p(2) + p(3) - exp(-p(1))
        res(2) = p(1) + p(3) - exp(-p(3))
        res(3) = p(1) + p(2) - exp(-p(3))
    end function f
    !MATRIZ JACOBIANA
```

```
37      function jac(x) result(jc)
38          IMPLICIT NONE
39          double precision :: dx=1e-12, x(:), jc(size(x), size(x)),
                xn(size(x)), dy(size(x)), dif(size(x))
40          integer :: n, i, k
41
42          n = size(x)
43          do k = 1,n !Calculo numerico da matriz jacobiana
44              xn = x
45              xn(k) = xn(k)+dx
46              dy = f(xn) - f(x)
47              dif = dy/dx
48              do i = 1,n
49                  jc(i, k) = dif(i)
50              end do
51          end do
52      end function jac
53      ! METODO DA FATORACAO LU
54      function LU(M, b) result(x)
55          IMPLICIT NONE
56          double precision :: b(:), M(:, :), x(size(b)), y(size(b))
               , U(size(b), size(b)), L(size(b), size(b)), c
57          integer :: n, i, j, k
58
59          n = size(b)
60
61          do i = 1,n
62              do j = 1,n
63                  L(i, j) = 0
64                  U(i, j) = 0
65              end do
66          end do
67          do i = 1,n
68              x(i) = 0
69              y(i) = 0
70              L(i, i) = 1 !Preenche L com a Matriz Identidade
71          end do
72
73          U = M !Atribui a Matriz J a Matriz U
74
75          do i = 1,(n-1)
```

```
76              do k = (i+1),n
77                  c = U(i, k) / U(i, i)
78                  L(k, i) = c !Armazena o multiplicador
79                  do j = 1,n
80                      U(k, j) = U(k, j) - c*U(i, j) !Multiplica com
                            o pivo da linha e subtrai
81  end
82                  end do
83              end do
84              do k = (i+1),n
85                  U(k, i) = 0
86              end do
87          end do
88
89          !Resolve o Sistema Ly=b
90          do i = 1,n
91              y(i) = b(i) / L(i, i)
92              do k = 1,(i-1)
93                  y(i) = y(i) - y(k)*L(i, k)
94              end do
95          end do
96
97          x = y
98          !Resolve o Sistema Ux=y
99          do i = n,1,-1
100             do k = (i+1),n
101                 x(i) = x(i) - x(k)*U(i, k)
102             end do
103             x(i) = x(i)/U(i, i)
104         end do
105     end function LU
106     !METODO NEWTON_RAPHSON
107     function newton_raph(x0, tol, iter, n_tot) result(x)
108         IMPLICIT NONE
109         double precision :: x0(:), x(size(x0)), tol, e, func(size
               (x0)), S(size(x0)), jc(size(x0), size(x0))
110         integer :: n_tot, n0, n=0, i
111         logical :: iter
112
113         n0 = size(x0)
114
```

```
115         do i = 1,n0
116             x(i) = x0(i)
117         end do
118         e = maxval(abs(f(x)))
119 
120         do while ((e>tol).and.(n<=n_tot))
121             n = n+1
122             func = f(x)
123             e = maxval(abs(func))
124             jc = jac(x)
125             if (size(func)==1) then
126                 x = x - (func(1)/jc(1, 1))
127             else
128                 S = LU(jc, -func)
129                 x = x+S
130             end if
131         end do
132 
133         if (iter .eqv. .true.) then
134             write(*,*) "Total de iteracoes: ", n
135         end if
136         if (n>=n_tot) then
137             write(*,*) "Processo parou, numero de iteracoes
                    limite atingido", n
138         end if
139 
140     end function newton_raph
141 
142 end program main
```

5.2 Julia

```
# METODO DA FATORACAO LU
function LU(matriz, vetor_b)
    n = length(vetor_b)
    x = zeros(n)
    L = zeros(n, n)
    for i = 1:n  #Preenche L com a Matriz Identidade
        L[i, i] = 1
    end
    U=zeros(n,n)
    U = matriz #Atribui a Matriz J a Matriz U

    for i = 1:n-1
        for k = i+1:n
            c = U[k, i] / U[i, i]
            L[k, i] = c #Armazena o multiplicador
            for j = 1:n
                U[k, j] = U[k, j] - c*U[i, j] # Multiplica com o
                   pivo da linha e subtrai
            end
        end
        for k = i+1:n
            U[k, i] = 0
        end
    end

    # Resolve o Sistema Ly=b
    y = zeros(n)
    for i = 1:n
        y[i] = vetor_b[i] / L[i, i]

        for k = 1:i-1
            y[i] = y[i]  - y[k]*L[i, k]
        end
    end
    n = length(y)

    # Resolve o Sistema Ux=y
    x = copy(y)
    for i = (n:-1:1)
        for k = 1+i:n
```

```
40              x[i] = x[i] - x[k]*U[i, k]
41          end
42          x[i] = x[i]/U[i, i]
43      end
44
45      return x
46  end
47
48
49  #DECLARACAO DAS FUNCOES
50  function f(p)
51      #Sistema de equacoes
52
53      #eq1: 1/2*sin(x1*x2)-(x2/(4*pi))-(x1/2),
54      #eq2: (1-1/(4*pi))*((exp(2*x1))-exp(1))-((exp(1)*x2)/pi)-2*
            exp(1)*x1)
55
56      #eq3: x2+x3-exp(-x1)
57      #eq4: x1+x3-exp(-x3)
58      #eq5: x1+x2-exp(-x3)
59      a = p[2] + p[3] - exp(-p[1])
60      b = p[1] + p[3] - exp(-p[3])
61      c = p[1] + p[2] - exp(-p[3])
62      return [a c b]
63  end
64  #MATRIZ JACOBIANA
65  function jac(x, dx=1e-10)
66      n = length(x)
67      J = zeros(n, n)
68      for j = 1:n #Calculo numerico da matriz jacobiana
69          xn = copy(x)
70          xn[j] = xn[j] +dx
71          dy = f(xn) - f(x)
72          dif = dy/dx
73          for i = 1:n
74              J[i, j] = dif[i]
75          end
76      end
77      return J
78  end
79  #METODO NEWTON_RAPHSON
```

```
function newton_raph(x0, tol, iter, n_tot)
    #DECLARACAO DAS VARIAVEIS
    tol = abs.(tol) #Modulos de tol e iter
    n_tot = abs.(n_tot)
    x = convert(Array{Float64}, x0) #Cria o vetor x
    e = maximum(abs.(f(x))) #Erro
    n = 0 #Atribue zero a variavel de interacoes totais

    while (e>tol)&(n<=n_tot)
        n += 1
        F = f(x)
        e = maximum(abs.(F))
        J = jac(x)
        if length(F) == 1
            x = x - F / J
        else
            S = LU(J, -F)
            x = x+S
        end
    end
    if iter==true
        print("Total de Iteracoes: ", string(n))
    end
    if n>=n_tot  #Condicao de parada
        print("Processo parou, numero de iteracoes limite
           atingido")
    else
        return x
    end
end


    x0 = [0.5, 1, 5] # Chute inicial
    iter  = 100 # Numero de iteracoes
    tol = 1e-12 # Tolerancia
    @timev x = newton_raph(x0, tol, true, iter)
    print("\nSolucao= ",x, "\n")
```

5.3 Python

```
import numpy as np
import time
from numpy import sin, exp
from math import pi

#DECLARACAO DAS FUNCOES
def func(p):
    '''
    Sistema de equacoes

        eq1: 1/2*sin(x1*x2)-(x2/(4*pi))-(x1/2),
        eq2: (1-1/(4*pi))*((exp(2*x1))-exp(1))-((exp(1)*x2)/pi)
           -2*exp(1)*x1)

        eq3: x2+x3-exp(-x1)
        eq4: x1+x3-exp(-x3)
        eq5: x1+x2-exp(-x3)
    '''

    x1, x2 = p

    return np.array((1/2*sin(x1*x2)-(x2/(4*pi))-(x1/2),
                     (1 - 1 / (4 * pi)) * ((exp(2 * x1)) - exp(1)
                        ) - ((exp(1) * x2) / pi) - 2 * exp(1) * x1
                        ))


#MATRIZ JACOBIANA
def jac(f, x, dx=1e-10):
    x = np.array(x)
    n = len(x)
    J = np.zeros((n, n))
    for j in range(n):          #Calculo numerico da matriz
       jacobiana
        xn = np.copy(x)
        xn[j] = xn[j] + dx
        dy = np.array(f(xn)) - np.array(f(x))
        dif = dy / dx
        for i in range(n):
            J[i, j] = dif[i]
```

```
37      return J
38
39
40  # METODO DA FATORACAO LU
41  def LU(matriz, vetor_b):
42
43      n = len(matriz)
44
45      L = [[0 for i in range(n)] for i in range(n)]
46      for i in range(n):
47          L[i][i] = 1          #Preenche L com a Matriz Identidade
48
49      U = [[0 for i in range(n)] for i in range(n)]
50      for i in range(n):
51          for j in range(n):
52              U[i][j] = matriz[i][j]        #Atribui a Matriz J a
                    Matriz U
53
54
55      # Encontra as Matrizes U e L
56      for i in range(n-1):
57          for k in range(i + 1, n):
58              c = U[k][i] / U[i][i]
59              L[k][i] = c  # Armazena o multiplicador
60              for j in range(n):
61                  U[k][j] -= c * U[i][j]  # Multiplica com o pivo
                        da linha e  subtrai
62
63
64          for k in range(i + 1, n):
65              U[k][i] = 0
66
67
68      # Resolve o Sistema Ly=b
69      y = [0 for i in range(n)]
70
71      for i in range(0, n, 1):
72          y[i] = vetor_b[i] / L[i][i]
73          for k in range(0, i, 1):
74              y[i] -= y[k] * L[i][k]
75
```

```
76
77      # Resolve o Sistema Ux=y
78      x = [y[i] for i in range(n)]
79
80      for i in range(n-1, -1, -1):
81          for k in range(i+1, n):
82              x[i] -= x[k] * U[i][k]
83          x[i] /= U[i][i]
84      return x
85
86  #METODO NEWTON_RAPHSON
87  def newton_raph(func, x0, tol, iter):
88
89      #DECLARACAO DAS VARIAVEIS
90      tol = abs(tol)      #Modulos de tol e iter
91      iter = abs(iter)
92      x = (np.array(x0)).astype(np.float)    #Cria o vetor x
93      e = max(abs(np.array(func(x))))  #Erro
94      n = 0      #Atribue zero a variavel de interacoes totais
95
96      #Print da interacao 0
97      if iter == 0:
98          print('\nTotal de Iteracoes: ' + str(n))
99          return x
100
101     else:
102
103         while (e > tol and n <= iter):
104             n += 1
105             F = np.array(func(x))
106             e = max(abs(F))
107             J = jac(func, x)
108             if len(F) == 1:
109                 x = x - F / J
110             else:
111
112                 S = LU(J, -F)
113                 x = x + S
114
115         print('\nTotal de Iteracoes: ' + str(n))
116         if n >= iter:  #Condicao de parada
```

```
117             print("PROCESSO PAROU, numero de iteracoes limite
                    atingido!")
118             return x
119         else:
120             return x
121
122
123 if __name__ == "__main__":
124
125     inicio = time.time()
126     x0 = [0.5, -2]  #Chute Inicial
127     iter = 100      #Quantidade de Interacoes maximas
128     tol = 1e-12     #tolerancia
129
130     x = newton_raph(func, x0, tol, iter)   #Chama o metodo de
            Newton_Raphson
131
132     print('Solucao: {} ' .format(x))
133
134     fim = time.time()
135     print('Tempo gasto: {}'.format(fim - inicio))
```