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

15 de novembro de 2018

Sumário

- Vários problemas sem solução analítica;

Introdução

- Vários problemas sem solução analítica;
- Necessidade de métodos iterativos;

Introdução

- Vários problemas sem solução analítica;
- Necessidade de métodos iterativos;
- Computação numérica;

Introdução

- Vários problemas sem solução analítica;
- Necessidade de métodos iterativos;
- Computação numérica;
- Várias linguagens de programação;

Introdução

- Vários problemas sem solução analítica;
- Necessidade de métodos iterativos;
- Computação numérica;
- Várias linguagens de programação;
- Julia: recente e eficiente.

Objetivos

- Solução de sistemas de equações não-lineares pelo método de Newton-Raphson;

Objetivos

- Solução de sistemas de equações não-lineares pelo método de Newton-Raphson;
- Utilizar *Fortran 95*, *Julia* e *Python*;

Objetivos

- Solução de sistemas de equações não-lineares pelo método de Newton-Raphson;
- Utilizar *Fortran 95*, *Julia* e *Python*;
- Comparar a eficiência de execução do algoritmo por cada linguagem;

Sistemas de equações não-lineares

Sistema 1:

1. $x_2 + x_3 - e^{-x_1} = 0$
2. $x_1 + x_3 - e^{-x_3} = 0$
3. $x_1 + x_2 - e^{-x_3} = 0$

Sistema 2:

1. $\frac{1}{2} sen(x_1 x_2) - \frac{x_2}{4\pi} - \frac{x_1}{2} = 0$
2. $(1 - \frac{1}{4\pi})(e^{2x_1} - e) - \frac{e x_2}{\pi} - 2e x_1 = 0$

Fluxograma

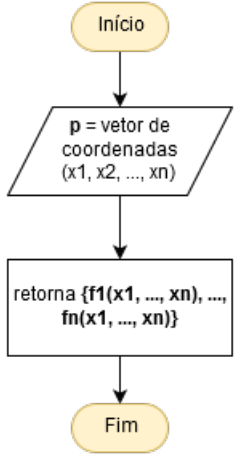


Figura: Fluxograma da função f().

Fluxograma

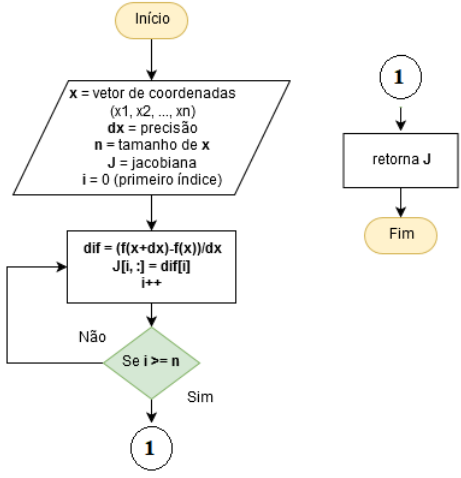


Figura: Fluxograma da função jac().

Fluxograma

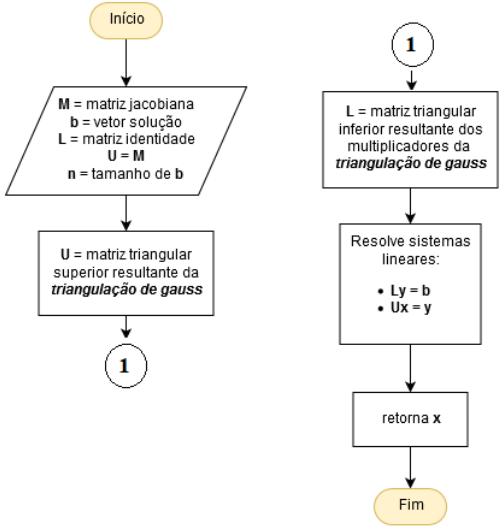


Figura: Fluxograma da função LU().

Fluxograma

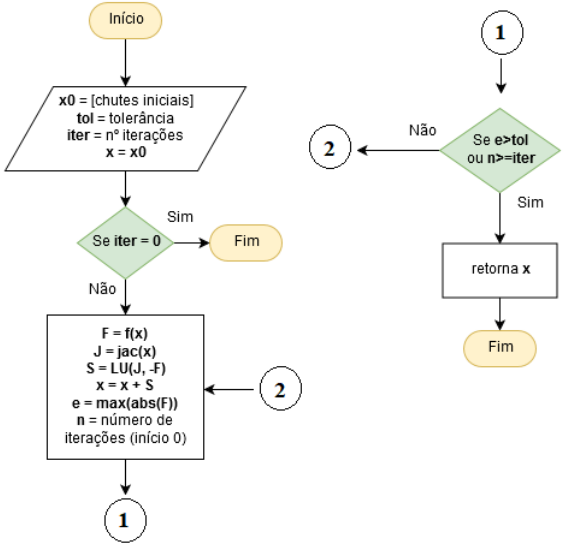


Figura: Fluxograma da função newrap().

Fluxograma

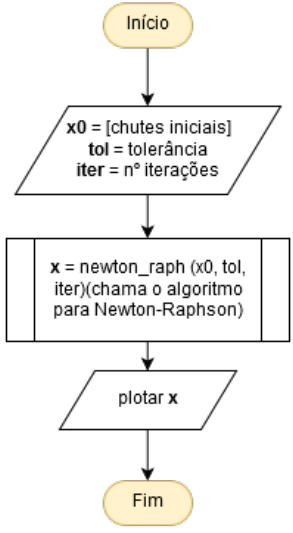


Figura: Fluxograma da main.

Figura: Marca abnTeX2. Fonte: <http://www.abntex.net.br/>

	Solução
sis_1_sol1	[0.35173371, 0.35173371, 0.35173371]
sis_1_sol2	[-0.83202504, 1.14898375, 1.14898375]
sis_2_sol1	[0.11116545 -2.26144905]
sis_2_sol2	[-0.52596071, 0.78465031]
sis_2_soln	[-4.28925635, 24.05879992]

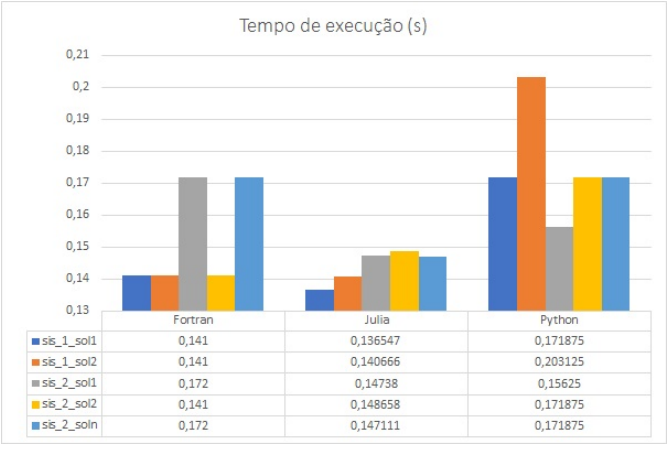
Figura: Tabela de soluções para cada sistema.

Tempo de execução

	Fortran *gcc 5.1.0*	Julia *1.0.1*	Python *3.7*	Chute inicial
sis_1_sol1	1.032	1	1.258	[0.5, 0.5, 0.5]
sis_1_sol2	1.002	1	1.444	[0, 1, 2]
sis_2_sol1	1.167	1	1.060	[0.5, -2]
sis_2_sol2	0.948	1	1.156	[0, 0]
sis_2_soln	1.169	1	1.175	[0.25, 0.25]

Figura: Tabela de comparação de tempo de execução relativo, tendo o tempo de Julia como 1.

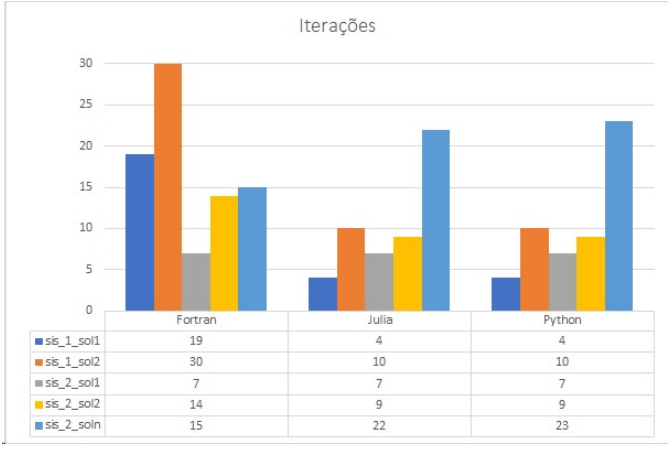
Tempo de execução



	Fortran	Julia	Python
sis_1_sol1	0,141	0,136547	0,171875
sis_1_sol2	0,141	0,140666	0,203125
sis_2_sol1	0,172	0,14738	0,15625
sis_2_sol2	0,141	0,148658	0,171875
sis_2_soln	0,172	0,147111	0,171875

Figura: Gráfico e tabela de comparação do tempo de execução absoluto.

Tempo de execução



	Fortran	Julia	Python
sis_1_sol1	19	4	4
sis_1_sol2	30	10	10
sis_2_sol1	7	7	7
sis_2_sol2	14	9	9
sis_2_soln	15	22	23

Figura: Tabela de comparação de iterações.

Conclusões

- Vários problemas sem solução analítica;
- Necessidade de métodos iterativos;
- Computação numérica;
- Várias linguagens de programação;
- Julia: recente e eficiente.