

Método de Newton para Resolver Sistemas de Equações Não-Lineares

Mônica D. S. Zanardini *

junho 2002

1 Método de Newton

Seja o problema de resolver o sistema de equações não-lineares $F(x) = 0$, onde

$$F(x) = (f_1(x) \quad f_2(x) \quad \cdots \quad f_n(x))^t$$

com $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$.

No qual a matriz jacobiana da função F é denotado por:

$$J(x) = (\nabla f'_1(x) \quad \nabla f'_2(x) \quad \cdots \quad \nabla f'_n(x))^t.$$

Um dos processos para resolver $F(x) = 0$ é o método de Newton, desenvolvido por Simpson no século XVIII. Esse método é baseado na resolução aproximada de problemas.

O k -ésimo problema é dado pela aproximação de $F(x)$ pela série de Taylor em torno do ponto x_k :

$$F(x) \approx L_k(x) = F(x_k) + J(x_k)(x - x_k).$$

O ponto seguinte x_{k+1} é a solução de:

$$L_k(x) = F(x_k) + J(x_k)(x - x_k) = 0.$$

*E-mail: monicads@cesec.ufpr.br.

Se $J(x_k)$ é não singular temos que $L_k(x) = 0$ tem uma única solução, portanto o método de Newton em cada interação k consiste em resolver o seguinte sistema linear:

$$J(x_k)S_k = -F(x_k)$$

$$x_{k+1} = x_k + S_k.$$

Dessa forma o processo gera uma sequência de pontos $\{x_k\}$ que converge para a solução do problema.

PROBLEMA 1.1 *Encontre a solução do sistema de equações não-lineares $F(x) = 0$ utilizando o método de Newton, dado:*

$$F(x) = \begin{pmatrix} x_1 + x_2 - 3 \\ x_1^2 + x_2^2 - 9 \end{pmatrix}.$$

Solução:

O programa em Matlab para resolver $F(x) = 0$ através do método de newton está descrito abaixo.

O arquivo P0.m informa os dados do problema ao programa principal:

```

1  function [prob_f,prob_j,prob_x0,prob_eps,prob_B0,...
2      prob_kmax]=P0()
3  %Funcao que fornece os dados necessarios para os
4  %metodos.
5  %Funcao F(x).
6  prob_f='fun';
7  %Jacobiana da funcao F(x).
8  prob_j='jac';
9  %Ponto inicial x0.
10 prob_x0=[1;5];
11 %B0 recebe o valor da jacobiana no ponto inicial x0.
12 %prob_B0=feval(prob_j,prob_x0);
13 %B0 recebe a matriz identidade de ordem 2.
14 %prob_B0=eye(2);
15 %B0 recebe a matriz aproximada da jacobiana de F(x) em x0.
```

```

16 prob_B0=[0.5 0.5; 1.5 9.5];
17 %Numero maximo de iteracoes.
18 prob_kmax=20;
19 %Tolerancia, criterio de parada.
20 prob_eps=1e-5;
21 %Contadores: cont_ext, cont_f, cont_j, cont_it.
22 cont_ext=0;
23 cont_f=0;
24 cont_j=0;
25 cont_it=0;

```

Em *fun.m* é definida $F(x)$:

```

1 function y = fun(x);
2 % y recebe a F avaliada no ponto x
3 y=[x(1)+x(2)-3; x(1)^2+x(2)^2-9];

```

No arquivo *jac.m* temos a matriz jacobiana de $F(x)$:

```

1 function j = jac(x);
2 % j matriz jacobiana avaliada no ponto x
3 j=[1 1; 2*x(1) 2*x(2)];

```

Em *newton.m* é desenvolvida a estrutura do método de Newton (programa principal):

```

1 function [P,x,k]=newton()
2 % O metodo de newton tem por finalidade encontrar
3 % a solucao do sistema nao linear de equacoes
4 % da forma F(x)=0.
5 clear all
6 %O usuario digita o problema escolhido.
7 problema=input('Escolha um entre os problemas P0,P1: ','s');
8 switch problema
9 case {'P0'}
10     %Possibilitando o uso das variaveis definidas em P0.m.

```

```

11     [prob_f,prob_j,prob_x0,prob_eps]=P0;
12     case {'P1'}
13         %Possibilitando o uso das variaveis definidas em P1.m.
14         [prob_f,prob_j,prob_x0,prob_eps]=P1;
15     otherwise
16         %Se nao houver digitacao o problema escolhido sera P0.
17         problema = 'P0';
18     end
19     %O numero de iteracoes inicia com valor zero.
20     k=0;
21     %A variavel x recebe o valor inicial.
22     x=prob_x0;
23     %Armazenando o valor de x inicial.
24     P(:,1)=x;
25     %Definindo a variavel de controle.
26     goon=0;
27     %Especificando quando o loop finaliza.
28     while (goon==0) & (k < 20)
29         %Determinando o valor de F(x) no ponto x em questao.
30         f=feval(prob_f,x);
31         %Determinando o valor da jacobiana de F(x) em x.
32         j=feval(prob_j,x);
33         %Resolvendo o sistema js=-f.
34         s=-j\f;
35         %Armazenando o valor de x_k.
36         x_k=x+s;
37         %Testando a variavel de controle.
38         if norm(x_k-x)<=prob_eps
39             goon=1;
40         end
41         %Atualizando x.
42         x=x_k;
43         %Atualizando o numero de iteracoes.
44         k=k+1;

```

```

45      %Acrescentando o ponto x da iteracao atual na matriz P.
46      P=[P x];
47  end
48  %.....Plotagem dos pontos obtidos em cada iteracao.
49  %Especificando a primeira coordenada do ponto.
50  x1=P(1,1:k+1);
51  %Especificando a segunda coordenada do ponto.
52  y1=P(2,1:k+1);
53  %Fazendo a plotagem dos pontos obtidos nas iteracoes.
54  plot(x1,y1, '-o');
55  %Salvando o grafico com nome P0newton.eps.
56  %print -deps c:/Meusdo~1\P0newton
57  %Salvando o grafico com nome P1newton.eps .
58  %print -deps c:/Meusdo~1\P1newton

```

Executando esse programa obtemos os resultados ilustrados na tabela 1.

Iteração k	x_k
0	(1, 5)
1	(-0.625, 3.625)
2	(-0.09191176470588, 3.09191176470588)
3	(-0.00265334193724, 3.00265334193724)
4	(-0.00000234259734, 3.00000234259734)
5	(-0.00000000000183, 3.00000000000183)

Tabela 1:

Observe que $x_5 = (-0.00000000000183, 3.00000000000183)$ está bem próximo de uma das soluções do problema proposto dada por $x^* = (0, 3)$.

A sequência de pontos x_0, x_1, \dots, x_5 que se aproxima da solução do problema está ilustrada na Figura 1.

PROBLEMA 1.2 Resolva pelo método de Newton o seguinte sistema de equações não-lineares:

$$F(x) = \begin{pmatrix} (x_1 + 3)(x_2^3 - 7) + 18 \\ \text{sen}(x_2 e^{x_1} - 1) \end{pmatrix} = 0$$

Solução:

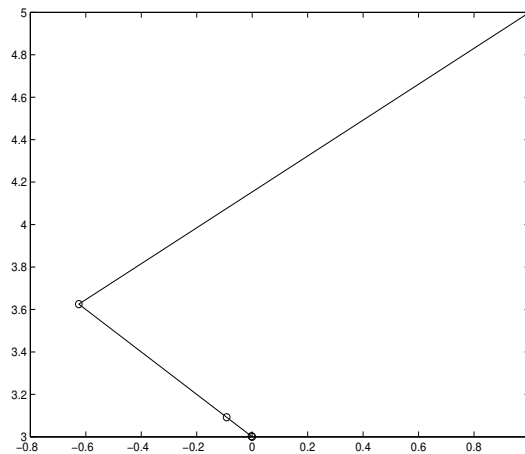


Figura 1:

O programa principal (*newton.m*) que consiste no desenvolvimento do processo de Newton para encontrar a solução do sistema é idêntico ao exercício anterior, e os dados necessários para o método estão especificados nos arquivos *P1.m*, *fun1.m* e *jac1.m*:

```

1  function [prob_f,prob_j,prob_x0,prob_eps,prob_B0,...
2      prob_kmax]=P1()
3  %Funcao que fornece os dados necessarios para
4  %os metodos.
5  %Funcao F(x).
6  prob_f='fun1';
7  %Jacobiana da funcao F(x).
8  prob_j='jac1';
9  %Ponto inicial x0.
10 prob_x0=[-0.5; 1.4];
11 %B0 recebe o valor da jacobiana no ponto inicial x0.
12 %prob_B0=feval(prob_j,prob_x0);
13 %B0 recebe a matriz [1 1 ; 1 1] de ordem 2.
14 prob_B0=ones(2);
15 %Numero maximo de iteracoes.

```

```

16 prob_kmax=20;
17 %Tolerancia, criterio de parada.
18 prob_eps=1e-5;
19 %Contadores: cont_ext, cont_f, cont_j, cont_it.
20 cont_ext=0;
21 cont_f=0;
22 cont_j=0;
23 cont_it=0;

1 function y = fun1(x);
2 % y recebe a F avaliada no ponto x
3 y=[(x(1)+3)*(x(2)^3-7)+18; sin(x(2))...
4     *exp(x(1))-1)];

1 function j = jac1(x);
2 % j matriz jacobiana avaliada no ponto x
3 j=[x(2)^3-7 3*x(1)*x(2)^2+6*x(2); cos(x(2))*...
4     exp(x(1))-1*x(2)*exp(x(1)) cos(x(2))*...
5     exp(x(1))-1*exp(x(1))];

```

A tabela 2 e a figura 2 mostram respectivamente os pontos obtidos em cada iteração k e o comportamento do método. Observe que o valor de x_8 está muito próximo da solução ótima do problema dada por $x^* = (0, 1)$

Iteração k	x_k
0	(-0.5, 1.4)
1	(0.23348022271571, 0.62375308202895)
2	(-0.10822059078402, 1.00747855828610)
3	(0.00485955519113, 1.00069607751840)
⋮	⋮
8	(0.00000064645100, 0.99999935354398)

Tabela 2:

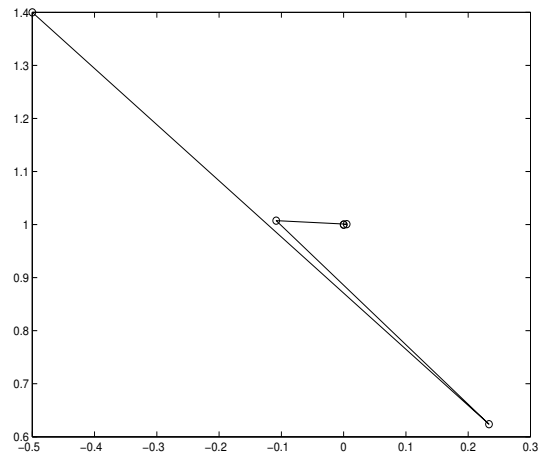


Figura 2:

Referências

- [1] MARTINEZ, José Mário. SANTOS, Sandra Augusta. **20º Colóquio de Matemática - Métodos Computacionais de Otimização** . Rio de Janeiro, IMPA, 1995.