NÚMERO DE CONDIÇÃO DE SISTEMAS NÃO LINEARES

ANA CERVIGNI GUERRA

Orientador:
Prof. Dr. José Mário Martínez

Dissertação apresentada no Instituto de Matemática, Estatística e Ciência da Computação, como requisito parcial para obtenção do título de Mestre em Matemática Aplicada.

Outubro - 1980

Ao Beny e à Maitê.

AGRADECIMENTOS

- Ao Martínez, pela paciência, dedicação e seriedade com que conduz seus trabalhos.

- A todos meus amigos que de alguma forma me ajudaram neste trabalho.

- Aos meus pais, que mesmo ausentes, me fazem lembrar suas palavras de incentivo.

- Ao CNPq pela ajuda financeira.

ÍNDICE

INTRODUÇÃO

O objetivo desta tese é analisar sob o ponto de vista numérico se as condições propostas por Bus [1976] para sistemas não lineares são explicativas da performance real do método de Newton.

O número de solubilidade definido por Bus, tenta explicar o comportamento de métodos de "tipo Newton" em sistemas não lineares. Portanto, assemelha-se ao número de condição de sistemas lineares já que este, como mostramos no Capítulo I, também explica performances de métodos para resolver problemas (no caso o método LU).

No Capítulo II introduz-se o conceito de aproximação do Jacobiano. Este conceito é essencial à compreensão da teoria pois Jacobiano analítico exato nunca existe na prática computacional. Tal conceito gera a noção de métodos de "tipo Newton".

No Capítulo III foram obtidas as condições para convergência desses métodos e no Capítulo IV generalizamos a teoria com a consideração do erro de arredondamento. Por fim, neste capítulo define-se o número de solubilidade de Bus.

Baseados nesta teoria escrevemos um programa Fortran para calcular o número de solubilidade de Bus usando precisão dupla para simular aritmética exata. Esse programa é discutido no Capítulo V.

Usando esse programa, calculamos o número de Bus para um conjunto de problemas não lineares com diferentes pontos iniciais. Ao mesmo tempo, resolvemos tais sistemas com o método de

Newton, e comparamos esses resultados com os calculados números de solubilidade de Bus. Tais experiências e as conclusões das mesmas são apresentadas no Capítulo VI.

CAPÍTULO I

NÚMERO DE CONDIÇÃO DE SISTEMAS LINEARES

1.1) CONCEITO DE NÚMERO DE CONDIÇÃO

Para desenvolver um trabalho sobre número de condição de sistemas não lineares, que é o tema desta tese, é preciso maiores informações sobre o que é número de condição no caso do sistema ser linear e também quais as informações que esse número pode nos trazer.

O número de condição é um dado muitas vezes difícil de se obter, mas que tem papel significante quando temos vários sistemas para resolver e a diferença entre seus elementos é pequena. Neste caso, através do número de condição podemos saber se a solução varia muito ou não dependendo da oscilação dos elementos.

Considere o sistema $Ax=b$ onde A é uma matriz não singular de ordem n e que este sistema tem solução única do tipo $x=A^{-1}b$. Se a matriz A permanecer a mesma mas o vetor b sofrer uma variação δb a solução x também terá uma mudança δx que deve satisfazer a equação

$$A(x+\delta x) = b+\delta b$$

desse modo temos:

$$\delta x = A^{-1}\delta b \ ,$$

aplicando norma

$$\|b\| \leq \|A\| \ \|x\| \quad e$$

$$\|\delta x\| \leq \|A^{-1}\| \ \|\delta b\|$$

multiplicando as duas temos:

$$\|b\| \ \|\delta x\| \ \leq \ \|A\| \ \|x\| \ \|A^{-1}\| \ \|\delta b\|$$

$$\frac{\|\delta x\|}{\|x\|} \ \leq \ \|A\| \ \|A^{-1}\| \ \frac{\|\delta b\|}{\|b\|}$$

$$\frac{\|\delta x\|}{\|x\|} \ \leq \ K \ \frac{\|\delta b\|}{\|b\|} \qquad (1)$$

$\frac{\|\delta b\|}{\|b\|}$ pode ser interpretado como a mudança relativa em b.

$\frac{\|\delta x\|}{\|x\|}$ indica a mudança relativa no vetor x.

Se ocorrer do vetor b ser o mesmo e a matriz A sofrer uma perturbação δA de tal modo que $(A+\delta A)^{-1}$ exista, então

$$x = A^{-1}b$$

$$(x+\delta x) = (A+\delta A)^{-1}b$$

$$\delta x = \left[(A+\delta A)^{-1} - A^{-1}\right] b$$

Seja

$$B = A+\delta A$$

então

$$\delta x = \left[B^{-1} - A^{-1}\right] b \ \Rightarrow$$

$$\delta x = \left[A^{-1} A B^{-1} - A^{-1} B B^{-1}\right] b \ \Rightarrow$$

$$\delta x = \left[A^{-1} (A-B) B^{-1}\right] b \ \Rightarrow$$

$$\delta x = \left[- A^{-1} \delta A \ (A+\delta A)^{-1} \right] b \ \Rightarrow$$

$$\delta x = - A^{-1} \delta A \left[(A+\delta A)^{-1} b \right] \ \Rightarrow$$

$$\delta x = - A^{-1} \delta A \ (x+\delta x) \ ;$$

aplicando norma:

$$\|\delta x\| \leq \|A^{-1}\| \ \|\delta A\| \ \|x+\delta x\|$$

$$\frac{\|\delta x\|}{\|x+\delta x\|} \leq \|A^{-1}\| \ \|\delta A\|$$

$$\frac{\|\delta x\|}{\|x+\delta x\|} \leq \|A\| \ \|A^{-1}\| \ \frac{\|\delta A\|}{\|A\|}$$

$$\frac{\|\delta x\|}{\|x+\delta x\|} \leq K \ \frac{\|\delta A\|}{\|A\|} \qquad (2)$$

O número K das expressões anteriores é chamado número de condição, que depende da norma usada. Para norma Euclideana temos:

$$K = \|A\| \ \|A^{-1}\| = \frac{\lambda_1}{\lambda_2}$$

onde λ_1 e λ_2 são respectivamente os módulos do maior e menor valor singular de A.

Quando o número de condição é próximo da unidade dizemos que o sistema é bem condicionado, isto é, uma pequena mudança na matriz A ou no vetor b irá causar uma pequena variação na so-

lução.

Devemos observar que o número K, das expressões anteriores, dá um limite superior da sensibilidade da solução com relação a uma mudança nos dados do sistema inicial. Dessa forma, mesmo que K seja grande (>> 1) a solução ainda pode variar pouco se houver uma pequena mudança nos dados iniciais. Isto ocorre com a matriz

$$\begin{bmatrix} 10^6 & 0 \\ 0 & 1 \end{bmatrix}$$

cujo número de condição é $K = 10^6$ e se $b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ pelo método de Gauss $x_1 = 10^{-6}$ e $x_2 = 1$.

Se a matriz mudar para

$$\begin{bmatrix} 10^6 & 1 \\ 0 & 1 \end{bmatrix} \qquad x_1 = 0 \quad e \quad x_2 = 1.$$

Se a mudança for

$$\begin{bmatrix} 10^6 & 0 \\ 1 & 1 \end{bmatrix} \qquad x_1 = 10^{-6} \qquad x_2 = 0.999$$

1.2) UMA INTERPRETAÇÃO GEOMÉTRICA DO NÚMERO DE CONDIÇÃO

LEMA 1: Seja A uma matriz real nxn $A = (a_1 \ldots, a_n)$ e seja $\alpha_1 = \pi/2$ e α_j, $j = 2 \ldots n$ o ângulo entre a_j e $[a_1 \ldots a_{j-1}]$. Então

$$|\det A| = \prod_{i=1}^{n} \|a_i\|_2 \, |\operatorname{sen} \alpha_i|$$

LEMA 2: Seja A uma matriz mxn de posto completo com $m \geq n$; $A = (a_1 \ldots a_n)$; e define-se $\alpha_j = \alpha_j(A)$ como no lema 1 para $j = 1 \ldots n$. Define-se também $P(A) = \prod_{i=1}^{n} |\text{sen } \alpha_i|$. Então $P(A)$ é invariante a menos de permutações entre as colunas de A.

LEMA 3: Seja A como no lema 2 e seja $\beta_i = \beta_i(A)$, $i = 1 \ldots n$, o ângulo entre a_i e $[a_1 \ldots a_{i-1}, a_{i+1} \ldots a_n]$. Então $|\text{sen } \beta_i| \geq P(A)$.

LEMA 4: Seja A como no lema 2 e define-se $A^+ = (A^t A)^{-1} A^t = (b_1 \ldots b_n)^t$. Então $\|b_i\|_2 \leq 1 / (P(A) \|a_i\|_2)$ para todo $i = 1 \ldots n$.

TEOREMA 1: Seja $\|.\|$ uma norma em R^{mxn}. Então existe $K > 0$, $K = K(m,n)$ tal que para todo A como nas hipóteses do lema 4

$$\|A^+\| \leq K \max \left\{ \frac{1}{\|a_i\|_2} , i = 1 \ldots n \right\} \frac{1}{P(A)}$$

Se K(A) é o número de condição da matriz A nxn, não singular, então do Teorema 1, segue que

$$K(A) \leq K \max \left\{ \|a_i\|_2 , i = 1 \ldots n \right\} \max \left\{ \frac{1}{\|a_i\|_2} , i = 1 \ldots n \right\} \frac{1}{P(A)}$$

Esta desigualdade mostra-nos que quando o número de condição cresce, ou a matriz é mal escalada ou suas colunas são quase dependentes. [5]

1.3) O EFEITO DO ERRO DE ARREDONDAMENTO

Na prática quando resolvemos um sistema, qualquer que

seja o método cometemos um tipo de erro chamado erro de arredondamento, agora vamos ver que tipo de influência terá este erro na solução do problema.

Um sistema do tipo $Ax = b$ pode ser resolvido pelo método da Eliminação de Gauss com pivotamento e a solução encontrada x satisfaz uma equação perturbada do tipo $(A+\delta A)x = b$ onde δA é uma matriz cujos elementos são próximos de zero.

Lembremos que o método da Eliminação de Gauss é baseado na decomposição da matriz A em um produto de duas matrizes triangulares L e U. A decomposição consiste em computar uma sequência de matrizes $A^1, A^2 \ldots A^k \ldots A^n$ a partir da matriz original A, onde A^k é zero abaixo da diagonal nas primeiras k-1 colunas.

A matriz A^{k+1} é obtida de A^k subtraindo um múltiplo da K-ésima linha de cada linha abaixo dela, o resto de A^k não muda. Então, seja A^k com elementos a^k_{ij}, o múltiplo escolhido usando aritmética de ponto flutuante, será:

$$m_{ik} = f\ell\ (a^k_{ik} / a^k_{kk}) \qquad i \geq k+1$$

e

$$a^k_{ij} = \begin{cases} 0 & \text{para } i \geq k+1, \quad j = k. \\ f\ell\ (a^k_{ij} - m_{ik}\ a^k_{kj}) & \text{para } i \geq k+1, \quad j \geq k+1. \\ a^k_{ij} & \text{para outros.} \end{cases}$$

finalmente depois de n passos obtemos a matriz triangular superior que chamamos de U. A matriz L é formada pelos multiplicadores m_{ik} desta maneira

$$L = \begin{bmatrix} 1 & & & 0 \\ m_{21} & 1 & & \\ m_{31} & m_{32} & 1 & \\ \cdots & \cdots & \cdots & \cdots \\ m_{n1} & m_{n2} & m_{n3} & \cdots \quad 1 \end{bmatrix}$$

As matrizes calculadas L e U, através desse processo satisfazem a equação

$$L \cdot U = A + E ,$$

onde E é uma matriz cujos elementos são próximos de zero,que aparecem devido ao erro de arredondamento. Naturalmente, esses erros têm efeito na solução do sistema $Ax = b$. Esse sistema inicial é equivalente aos dois sistemas triangulares e sabemos que as soluções computadas dos sistemas triangulares obedecem a:

$$(L+\delta L)y = b \quad \text{e} \quad (U+\delta U)x = y$$

Portanto a solução computada x é a solução exata de

$$(L+\delta L) \cdot (U+\delta U)x = b$$

Usando a relação $A+E = L.U$, temos:

$$(A + E + \delta L.U + L.\delta U + \delta L\ \delta U)x = b$$

ou

$$(A + \delta A)x = b$$

onde

$$\delta A = E + L\delta L + \delta L.U + \delta L.\delta U.$$

1.4) CONCLUSÃO

Vimos que a solução x, obtida pelo método de Gauss com pivotamento, é a solução exata do sistema perturbado $(A+\delta A)x = b$. O erro relativo entre essa solução e a solução verdadeira do sistema original tem como majorante uma quantidade proporcional ao número de condição.

Desse modo número de condição é a grandeza que limita o crescimento do erro na solução de um sistema resolvido pelo método de Gauss.

CAPÍTULO II

APROXIMAÇÕES CONSISTENTES

Os métodos mais conhecidos para resolver sistemas de equações são os métodos de Newton, Quase-Newton e Secante.

Na prática, quando se trata de resolver um sistema, às vezes é difícil calcular a matriz Jacobiana e se ela for calculada no computador com precisão finita nunca será obtida exatamente. Por isso precisamos analisar situações onde, na fórmula clássica de Newton, o jacobiano é substituído por uma aproximação.

Um exemplo de aproximação para a matriz Jacobiana é usar a fórmula das diferenças finitas, outra aproximação é avaliar as expressões analíticas das derivadas parciais com precisão finita no computador.

Se a aproximação da matriz Jacobiana não for uma boa aproximação, ocorrerão erros que terão muita influência na ordem de convergência do método, por isso devemos escolher esta aproximação com muito cuidado.

Para o sistema $F(x) = 0$ o método de Newton discreto se define por:

$$x_{k+1} = x_k - J(x_k, h_k)^{-1} F(x_k) \qquad (2.1)$$

onde $J(x_k, h_k)$ é uma aproximação por diferenças finitas de $J(x_k)$.

Os métodos do tipo Quase-Newton e Secante têm a forma

$$x_{k+1} = x_k - M_k(x_k, h_k)^{-1} F(x_k) \qquad (2.2)$$

onde $M(x,h)$ é uma outra aproximação de $J(x)$.

Vejamos agora uma definição que tenta exprimir o fato de que uma matriz é uma boa aproximação do Jacobiano.

DEFINIÇÃO 2.1: Seja $F : D \subset R^n \to R^n$ uma função diferenciável em $D_0 \subset D$ e o operador $M : D_M \times D_h \subset R^n \times R^m \to L(R^n)$. Então M é uma *aproximação consistente* de $J(x) = F'(x)$ em $D_0 \subset D_M$ se $0 \in R^m$ é um ponto limite de D_h e

$$\lim_{h \to 0;\ h \in D_h} M(x, h) = F'(x) \qquad \text{para } x \in D_0$$

Se houver constantes c e $r > 0$ tais que

$$\|F'(x) - M(x,h)\| < c\|h\|, \quad \forall\ x \in D_0,\ h \in D_h \cap S(0,r) \tag{2.3}$$

então M se diz uma *aproximação fortemente consistente* de F' em D_0.

Quando $M(x, h)$ é uma aproximação consistente, o método definido em (2.2) converge com uma ordem alta, como expressa o seguinte teorema.

TEOREMA 1: Seja $F : D \subset R^n \to R^n$ diferenciável em uma vizinhança aberta $S_0 \subset D$ de um ponto $Z \in D$ para o qual $F(Z) = 0$ e que F' é contínua em Z e $F'(Z)$ é não singular. Seja $J : D_J \times D_h \subset R^n \times R^m \to L(R^n)$ uma aproximação consistente de F' em S_0. Então existe $\delta > 0$ e $r > 0$ tal que a função

$$G(x,h) = x - J(x,h)^{-1} F(x) \tag{2.4}$$

está definida para todo $x \in S = S(Z,\delta)$, $h \in D_h = D_h \cap S(0,r)$ e satisfaz

$$\|Z-G(x,h)\| \leq W(x,h)\,\|x-Z\| \qquad (2.5)$$

$$\forall\, x \in S,\quad h \in D'h$$

onde

$$W(x,h) \to 0 \qquad x \to Z \qquad e \qquad h \to 0 \qquad (2.6)$$

$$h \in D'h$$

Entretanto, se J é uma aproximação fortemente consistente de F' em S_0 e se

$$\|F'(x)-F'(Z)\| \leq \gamma\,\|x-Z\| \quad \forall\, x \in S_0 \qquad (2.7)$$

então existem constantes α_1 e α_2 tais que

$$\|Z-G(x,h)\| \leq \alpha_1\|x-Z\|^2 + \alpha_2\|h\|\,\|x-Z\| \qquad (2.8)$$

$$\forall\, x \in S,\quad h \in D'h$$

PROVA: Seja $\beta = \|F'(Z)^{-1}\|$ e seja $\varepsilon \in (0, \frac{1}{2}\beta^{-1})$.

Sendo J uma aproximação consistente em S_0, existe um $r > 0$ tal que $D'h$ é não vazio e

$$\|F'(x)-J(x,h)\| \leq \frac{1}{2}\varepsilon \quad \forall\, x \in S_0,\quad h \in D'h$$

Entretanto pela continuidade de F' em Z existe $\delta > 0$ tal que $S = S(Z,\delta) \subset S_0$ e

$$\|F'(x)-F'(Z)\| \leq \frac{1}{2}\varepsilon\,, \quad \forall\, x \in S$$

então

$$\|F'(Z)-J(x,h)\| \leq \varepsilon\,, \quad \forall\, x \in S,\quad h \in D'h$$

e, pelo lema da perturbação (Pág. 45 [2]) segue-se que $J(x,h)^{-1}$ existe e satisfaz

$$\|J(x,h)^{-1}\| \leq N = \frac{\beta}{1-\beta\varepsilon} \qquad \forall\, x \in S, \quad h \in D'h$$

Assim, G é bem definido em $S \times D'h$ e

$$\|G(x,h)-Z\| = \|J(x,h)^{-1}\left[J(x,h)(x-Z)-Fx\right]\|$$

$$\leq N\left[\|J(x,h)-F'(x)\| + \|F'(x)-F'(Z)\|\right]\|x+Z\| +$$

$$N\|Fx - FZ - F'(Z)(x-Z)\|$$

Por (2.5), temos:

$$W(x,h) = N\left[\|J(x,h)-F'(x)\|+\|F'(x)-F'(Z)\|+q(x)\right] \qquad (2.9)$$

onde

$$q(x) = \frac{\|Fx - FZ - F'(Z)(x-Z)\|}{\|x-Z\|} \qquad \text{para } x \neq Z$$

$$q(Z) = 0$$

Para concluir que (2.6) é verdade, devemos observar que a continuidade de F' em Z implica que $q(x)\to 0$ e $F'(x)-F'(Z)\to 0$ se $x\to Z$ enquanto a convergência uniforme da definição 1 implica que $J(x,h)-F'(x) \to 0$ se $h \to 0$ e $x \to Z$.

Agora, se $F : D \subset R^n \to R^m$ é G-diferenciável num conjunto convexo $D_0 \subset D$ então para qualquer x, y, z $\in D_0$.

$$\|Fy-Fz-F'(x)(y-z)\| \leq \sup_{0\leq t\leq 1} \|F'(z+t(y-z))-F'(x)\|y-z\|$$

(ver [2], pág. 70).

Logo, por isto último, e assumindo (2.6)

$$\|q(x)\| \leq \gamma\, \|x-Z\|, \quad \forall\, x \in S \qquad (2.10)$$

e (2.8) segue-se imediatamente de (2.5) e (2.9) com $\alpha_1 = 2\gamma N$ e $\alpha_2 = Nc$ onde c é a constante de (2.3).

Uma aplicação deste teorema mostra que a ordem de convergência da sequência (2.1) é ordem superlinear quando $\lim_{k\to\infty} h^k = 0$.

O conceito de aproximação consistente será usado no próximo capítulo.

CAPÍTULO III

ANÁLISE DA CONVERGÊNCIA DE MÉTODOS DE "TIPO NEWTON", PARA EQUAÇÕES NÃO LINEARES

O objetivo deste capítulo é encontrar condições para que a sequência de pontos obtidos por métodos do "tipo Newton" convirja para uma solução do sistema.

Seja o sistema de equações

$$F(x) = 0 \tag{3.1}$$

onde $F : \bar{D} \subset R^n \to R^n$ uma função continua numa região $\bar{D}$. Um método iterativo que resolve esse sistema é o método de Newton, mas podemos considerar também os métodos do tipo Newton onde usamos a fórmula definida em (2.2). Para esses métodos deve existir um operador M chamado aproximação consistente e que foi visto no capítulo anterior.

Estudando a convergência dos métodos de "tipo Newton", vamos compará-los com o método de Newton definido em (2.1) e (2.2).

Seja F definida como em (3.1), numa região convexa $D \subset \bar{D}$, F duas vezes diferenciável, J(x) não singular em D, $x_0 \in D$ um ponto inicial e $Z \in D$ a solução de F. Definimos novamente o método de Newton

$$\phi(x) = x - [J(x)]^{-1} F(x) \tag{3.2}$$

e um método de tipo Newton como:

$$\psi(x) = x - [M(x,h)]^{-1} F(x) \tag{3.3}$$

Então usando o teorema do valor médio obtemos a seguinte expressão para o erro em $\emptyset(x)$ como aproximação da solução Z.

$$\|\emptyset(x)-Z\| = \|x-J(x)^{-1}F(x)-Z\| = \|(x-Z)-J(x)^{-1}F(x)\| =$$

$$= \|J(x)^{-1}\left[J(x)(x-Z) - F(x)\right]\| \leq S(x,Z)\|x-Z\|^2 .$$

logo

$$\|\emptyset(x)-Z\| \leq S(x,Z) \|x-Z\|^2 \qquad (3.4)$$

Esta fórmula merece uma explicação detalhada.

Se tomarmos o método de Newton e fizermos seu desenvolvimento em Taylor na vizinhança de x^k, então de:

$$x^{k+1} = x^k-\left[F'(x^k)\right]^{-1} F(x^k) \quad \text{onde } F' = J = \left(\frac{\partial fi}{\partial xj}\right)$$

obtemos:

$$F(x^{k+1}) = F(x^k) + F'(x^k)\left[-F'(x^k)^{-1} F(x^k)\right] + T(x^{k+1})$$

logo:

$$F(x^{k+1}) = 0 + T(x^{k+1})$$

Analisando o termo $T(x^{k+1})$, termo complementar de $2^{\underline{a}}$ ordem temos:

Se $F = (f_1,\ldots,f_n)^T$

$$f_j(x^{k+1}) = f_j(x^k)+f_j(x^k)(x^{k+1}-x^k)+\frac{1}{2}(x^{k+1}-x^k)^T\nabla^2 f_j(\xi)(x^{k+1}-x^k)$$

Mas

$$f_j(x^k) + f'_j(x^k)(x^{k+1}-x^k) = 0 ,$$

$$\nabla^2 f_j(\xi) \quad = \quad \text{hessiano de } f_j \quad \text{e}$$

$$\xi \in \left[x^k,\ x^{k+1}\right] \quad \text{(segmento em } R^n\text{)}$$

$\nabla^2 f_j(\xi)$ é uma matriz simétrica, logo pode-se escrever como:

$$\nabla^2 f_j(\xi) \quad = \quad QDQ^T \quad = \quad Q(\xi)\ D(\xi)\ Q(\xi)^T$$

onde $Q(\xi)$ é ortogonal $(QQ^T = Q^TQ = I)$ e D é a matriz diagonal dos autovalores de $\nabla^2 f_j(\xi)$.

Logo

$$T_j \quad = \quad \frac{1}{2}(x^{k+1}-x^k)^T\ Q(\xi)\ D(\xi)\ Q(\xi)^T(x^{k+1}-x^k)$$

$$= \quad \frac{1}{2}\ y(\xi)^T\ D(\xi)\ y(\xi)$$

onde

$$\|y(\xi)\| \quad = \quad \|x^{k+1}-x^k\| \ ,$$

$$y(\xi) \quad = \quad Q^T(\xi)\ (x^{k+1}-x^k)$$

escrevendo

$$y(\xi) \quad = \quad \begin{pmatrix} y_1(\xi) \\ \vdots \\ y_n(\xi) \end{pmatrix}$$

$$T_j \quad = \quad \frac{1}{2}\ \lambda_1(\xi)\ y_1(\xi)^2 + \ldots + \lambda_n(\xi)\ y_n(\xi)^2 \ .$$

com $\lambda_k(\xi)$ = autovalor j de $\nabla^2 f_j(\xi)$

$$\Rightarrow \quad \|T_j\| \ \leq \ \frac{1}{2}\ M_j\ \|y(\xi)\|^2 \quad = \quad \frac{1}{2}\ M_j \|y(\xi)\|^2 \quad =$$

$$= \quad \frac{1}{2}\ M_j\ \|x^{k+1}-x^k\|^2$$

e M_j é o maior dos módulos dos autovalores de $\nabla^2 f_j(\xi)$ ou seja

$$M_j = \|\nabla^2 f_j(\xi)\|_2 \quad \text{(norma matricial)}$$

Então

$$T(x^{k+1}) = \begin{bmatrix} \frac{1}{2}(x^{k+1}-x^k)^T \nabla^2 f_1(\xi_1)(x^{k+1}-x^k) \\ \vdots \\ \frac{1}{2}(x^{k+1}-x^k)^T \nabla^2 f_n(\xi_n)(x^{k+1}-x^k) \end{bmatrix}$$

e $\|T(x^{k+1})\|_2 =$

$$\sqrt{\sum_{i=1}^{n} \left[\frac{1}{2}(x^{k+1}-x^k)^T \nabla^2 f_j(\xi_j)(x^{k+1}-x^k)\right]^2} \leq$$

$$\sqrt{\sum_{i=1}^{n} \frac{1}{4} M_j^2 \|x^{k+1}-x^k\|^4} \leq \sqrt{\frac{n}{4} M^2 \|x^{k+1}-x^k\|^4} \leq$$

$$\leq \sqrt{\frac{n}{4}}\, M \|x^{k+1}-x^k\|^2$$

onde $M = \max\{\|\nabla^2 f_1(\xi_1)\|_2, \ldots, \|\nabla^2 f_n(\xi_n)\|_2\}$

Se Z é a solução de $F(x) = 0$ e

$$\|x^{k+1}-Z\| \leq \|x^k-Z\|$$

então

a) $\|x^{k+1}-x^k\| \leq \|x^{k+1}-Z\| + \|x^k-Z\| \leq 2\|x^k-Z\|$

Mas tínhamos obtido que:

b) $\|F(x^{k+1})\| = \|T(x^{k+1})\| \leq \sqrt{\frac{n}{4}}\, M\, \|x^{k+1}-x^k\|^2$

Logo substituindo (a) em (b)

$$\|F(x^{k+1})\| = \|T(x^{k+1})\| \leq \sqrt{n}\, M\, \|x^k-Z\|^2$$

Agora, se $F'(Z)$ é inversível

$$\|F(x^{k+1})\| \geq \text{cte} \cdot \|x^{k+1}-Z\|$$

logo

$$\|x^{k+1}-Z\| \leq S(x^k, Z)\, \|x^k-Z\|^2$$

ou, com a notação anterior,

$$\|\phi(x)-Z\| \leq S(x,Z)\, \|x-Z\|^2$$

$\forall$ x suficientemente próximo de Z.

Com isso mostramos que o método de Newton tem convergência quadrática.

Usando o lema da perturbação temos: (secção 10 [1]) .

$$\phi(x)-\psi(x) = \left[-J(x)^{-1}F(x) + M(x,h)^{-1}\, F(x)\right] =$$

$$= \left[-I+J(x)\, M(x,h)^{-1}\right] J(x)^{-1}\, F(x) =$$

$$= \left[I- \sum_{n=0}^{\infty} (I-J(x)^{-1}\, M(x,h))^n\right] J(x)^{-1}\, F(x)$$

Logo,

$$\|\phi(x)-\psi(x)\| \leq C(x,h)\, \|J(x)^{-1}\, F(x)\| \qquad (3.5)$$

onde

$$\bar{C}(x,h) = 2\bar{c}\,\|h\|\,\|J(x)^{-1}\| \qquad (3.6)$$

$\bar{c}$ é o fator de consistência de M e

$$\|h\| \leq 1 / (2\bar{c}\,\|J(x)^{-1}\|)$$

entretanto

$$\|J(x)^{-1}F(x)\| = \|x-\phi(x)\| \leq \|x-Z\| + \|\phi(x)-Z\| \qquad (3.7)$$

$$\|\psi(x)-\phi(x)\| \geq \|\psi(x)-Z\| - \|\phi(x)-Z\|$$

combinando (3.4), (3.5) e (3.7),

$$\|\phi(x)-\psi(x)\| \leq \bar{C}(x,h)\,\|x-Z\| + C(x,h)\,\|\phi(x)-Z\|\,,$$

logo, por (3.4)

$$\|\phi(x)-\psi(x)\| \leq \bar{C}(x,h)\,\|x-Z\| + \bar{C}(x,h)S(x,Z)\,\|x-Z\|^2$$

mas,

$$\|\phi(x)-\psi(x)\| \leq \|\phi(x)-Z\| - \|\psi(x)-Z\| \leq \bar{C}(x,h)\,\|x-Z\| +$$

$$+ \bar{C}(x,h)\,S(x,Z)\,\|x-Z\|^2\,.$$

Logo,

$$\|\psi(x)-Z\| \leq \bar{C}(x,h)\,\|x-Z\| + \bar{C}(x,h)\,S(x,Z)\,\|x-Z\|^2 + \|\phi(x)-Z\|$$

ou seja,

$$\|\psi(x)-Z\| \leq \bar{C}(x,h)\,\|x-Z\| + \bar{C}(x,h)\,S(x,Z)\,\|x-Z\|^2 + S(x,Z)\,\|x-Z\|^2$$

e finalmente

$$\|\psi(x)-Z\| \leq \bar{C}(x,h)\,\|x-Z\| + \left[\bar{C}(x,h)+1\right] S(x,Z)\,\|x-Z\|^2 \qquad (3.8)$$

Observamos, de (3.8) que se $\bar{C}(x,h) = 0\|h\|$ então pode-se esperar uma convergência quadrática.

Os resultados anteriores justificam a seguinte definição:

DEFINIÇÃO 3.1: Seja o sistema não linear definido em (3.1) e seja $x_0 \in D$ uma aproximação da solução Z de (3.1). Então diz-se que o sistema é solúvel por um método de "tipo Newton" se as seguintes condições forem satisfeitas:

a) $J(x)$ e $H(x)$ existem em D e $J(x_0)$ é não singular.

b) Se $\bar{C}(x,h)$ é definido como em (3.6), então h_0 satisfaz $\bar{C}(x_0, h_0) < 1$ e $r_0 = \bar{C}(x_0,h_0)\|\phi(x_0)-x_0\| + \|\phi(x_0)-Z\| < \|x_0-Z\|$

c) $U_0 = \{y \in R^n \;/\; \|y-z\| \leq r_0\} \subset D$ e $J(x)$ é não singular em U_0.

d) $\bar{K} = \sup_{\substack{x \in U_0 \\ k = 1,2...}} \bar{C}\,(x,h_k) < 1$

e) $\bar{\sigma}(F,Z,x_0,M) = \bar{K} + (\bar{K}+1)\, S\, r_0 < 1 \qquad (3.9)$

onde $S = \sup_{x \in U_0} S(x,Z)$

Se todas essas condições forem satisfeitas $\bar{\sigma}(F,Z,x_0,M)$ é chamado de *número de solubilidade* do método que resolve o sistema não linear $F(x) = 0$ com x_0 ponto inicial e Z uma solução.

Se uma dessas condições não for satisfeita o *número de solubilidade* será definido como infinito.

Vejamos agora o seguinte teorema.

TEOREMA 3.1: Se um sistema não linear definido como em (3.1) com uma aproximação inicial x_0 e solução Z é solúvel por um método do tipo Newton, então a sequência de pontos gerados por este método converge para Z. Se entretanto o método é tal que $||h_k|| = 0||x_k - Z||$ para $k \to \infty$ então a convergência é quadrática.

Prova: $$\|\psi(x_0)-Z\| \leq \|\psi(x_0)-\phi(x_0)\| + \|\phi(x_0)-Z\|$$

$$\leq \bar{C}(x_0,h_0)\|\phi(x_0)-x_0\| + \|\phi(x_0)-Z\|$$

mas pela combinação (b) anterior

$$C(x_0h_0)\ \|\phi(x_0)-x_0\| + \|\phi(x_0)-Z\| < \|x_0-Z\|$$

então

$$\|\psi(x_0)-Z\| \leq \|x_0-Z\|$$

por (c), (d) e (e) podemos usar (3.8), desse modo com a condição (3.9) temos o resultado.

Na prática a condição (3.9) é muito forte. Ela nos dá condições de falar sobre o grau de dificuldade para se resolver um sistema com um determinado método. Veremos isso nas experiências numéricas.

CAPÍTULO IV

O EFEITO DO ERRO DE ARREDONDAMENTO

Neste capítulo será estudado o efeito do erro de arredondamento na convergência de métodos de "tipo Newton". Se considerarmos a teoria dada no capítulo anterior para sistemas calculados no computador, teremos os métodos de "tipo Newton numéricos".

Daqui para frente usaremos a seguinte notação:

ε - a precisão do computador.

$f\ell_\varepsilon(.)$ - a expressão entre parênteses calculada com aritmética de ponto flutuante em precisão ε.

Usando o computador, o próprio método de Newton apresenta problemas de erro de arredondamento pois por mais exato que seja calculado o Jacobiano, só se pode garantir que

$$\|M_k - J(x_k)\| \leq \delta \|J(x_k)\| \tag{4.1}$$

sendo

$$M_k = f\ell_\varepsilon (J(x_k))$$

e $\delta \geq \varepsilon$ é um valor que depende de ε e do modo como M_k é calculado.

Faremos uma ampliação da teoria dada anteriormente, começando com um conceito mais geral de aproximação consistente.

DEFINIÇÃO 4.1: Seja F, diferenciável em $D \subset \bar{D} \subset R^n$ e seja um operador M como na definição (2.1), para um número real $r > 0$ e um inteiro $m \geq 0$. Então $M(x,h)$ é chamada aproximação numericamente consistente de $J(x)$ em $D_0 \subset D$, se existe uma constante c_1 e a função $c_0(\varepsilon,h)$ a qual é contínua em ε para $h \neq 0$ fixo e $\varepsilon \geq 0$, tal que as seguintes condições sejam satisfeitas:

$$\|J(x) - f\ell_\varepsilon(M(x,h))\| \leq c_0(\varepsilon,h) + c_1\|h\|$$

$$\text{para todo } x \in D_0, \quad h \in U_r^m \mid \{0\} \tag{4.2}$$

$$\lim_{\varepsilon \to 0} c_0(\varepsilon,h) = 0 \qquad \text{para } h \neq 0 \tag{4.3}$$

e

$$c(\varepsilon,h) = c_0(\varepsilon,h) + c_1\|h\| \tag{4.4}$$

é chamada de fator de consistência.

DEFINIÇÃO 4.2: Chamamos um método de "tipo Newton Numérico" para resolver (3.1) se existe um operador M como na def. 2.1 tal que

$$M_k = f\ell_\varepsilon\,(M(x_k, h_k)$$

e $M(x,h)$ é a aproximação numericamente consistente de $J(x)$ em D_0.

Vamos usar a notação $\bar{\psi}(x)$ para o vetor o qual satisfaz exatamente a equação

$$f\ell_\varepsilon(M(x,h))\,(\bar{\psi}(x) - x) = F(x) \tag{4.5}$$

do mesmo modo que em (3.5) temos:

$$\|\phi(x)-\bar{\psi}(x)\| \leq C(x,h,\varepsilon)\| \left[J(x)\right]^{-1} F(x) \| \qquad (4.6)$$

onde

$$C(x,h,\varepsilon) = 2c(\varepsilon,h) \|J(x)^{-1}\| < 1 \qquad (4.7)$$

e $c(\varepsilon,h)$ é dado por (4.4).

Seja $f\ell_\varepsilon(\bar{\psi}(x))$ uma aproximação numérica de $\bar{\psi}(x)$. Então

$$f\ell_\varepsilon(\bar{\psi}(x)) = f\ell_\varepsilon(f\ell_\varepsilon(\bar{\psi}(x)-x)+x) \qquad (4.8)$$

onde $f\ell_\varepsilon(\bar{\psi}(x)-x)$ é a solução numérica de (4.5) e $F(x)$ é substituída por $f\ell_\varepsilon(F(x))$.

Suponha que vamos resolver o sistema linear $Ax = b$ pelo método de Gauss no computador com precisão ε, mas o vetor b não é fornecido exatamente. Então seja a perturbação de b dada por $\|\delta b\|$. Assim o erro na solução numérica $\bar{x}$ como aproximação da solução exata x^* é dada por:

$$\frac{\|\bar{x}-x^*\|}{\|x^*\|} = K(A) \left[\frac{\varepsilon g(n)}{1-K(A)\varepsilon g(n)} + \frac{\|\delta b\|}{\|b\|} \right] \qquad (4.9)$$

onde $K(A) = \|A\| \ \|A^{-1}\|$ e $g(n)$ uma função dependendo da ordem n, da norma usada e da estratégia do pivotamento. Ainda deve-se assumir que

$$K(A).\varepsilon.g(n) < 1 .$$

Usando o lema da perturbação (secção 10 [1]) podemos dizer que

$$K(f\ell_\varepsilon(M(x,h))) \leq 3K(J(x))$$

aplicando isso em $f\ell_\varepsilon(\bar{\psi}(x)-x)$ obtemos

$$\|f\ell_\varepsilon(\bar{\psi}(x)-x)-(\bar{\psi}(x)-x)\| \leq \alpha(x,\varepsilon,n)\,\|\bar{\psi}(x)-x\| \tag{4.10}$$

onde

$$\alpha(x,\varepsilon,n) = 3K(J(x))\left[\frac{\varepsilon q(n)}{1-\varepsilon K(J(x))\varepsilon q(n)} + \delta\right] \tag{4.11}$$

e δ satisfaz

$$\|f\ell_\varepsilon(F(x))-F(x)\| \leq \delta\,\|F(x)\| \tag{4.12}$$

Assumimos que $3K(J(x))\,\varepsilon q(n) < 1$.

Combinando (4.8) e (4.10)

$$\|f\ell_\varepsilon(\bar{\psi}(x))-\bar{\psi}(x)\| = \|f\ell_\varepsilon(f\ell_\varepsilon(\bar{\psi}(x)-x)+x)-\bar{\psi}(x)\|$$

$$\|f\ell_\varepsilon(\bar{\psi}(x))-\bar{\psi}(x)\| = \|(1+\varepsilon)\left[f\ell_\varepsilon(\bar{\psi}(x)-x)+x\right]-\bar{\psi}(x)\|$$

$$\|f\ell_\varepsilon(\bar{\psi}(x))-\bar{\psi}(x)\| = \|f\ell_\varepsilon(\bar{\psi}(x)-x)+x+\varepsilon f\ell_\varepsilon(\bar{\psi}(x)-x)+\varepsilon x-\bar{\psi}(x)\|$$

$$\|f\ell_\varepsilon(\bar{\psi}(x))-\bar{\psi}(x)\| = \|f\ell_\varepsilon(\bar{\psi}(x)-x)+\varepsilon f\ell_\varepsilon(\bar{\psi}(x)-x)+\varepsilon x-(\bar{\psi}(x)-x)\|$$

$$\|f\ell_\varepsilon(\bar{\psi}(x))-\bar{\psi}(x)\| = \|(1+\varepsilon)f\ell_\varepsilon(\bar{\psi}(x)-x)+\varepsilon x-(\bar{\psi}(x)-x)\|$$

$$\|f\ell_\varepsilon(\bar{\psi}(x))-\bar{\psi}(x)\| \leq \|(1+\varepsilon)\left[f\ell_\varepsilon(\bar{\psi}(x)-x)-(\bar{\psi}(x)-x)\right]\|+\varepsilon\|\bar{\psi}(x)-x\|+\varepsilon\|x\|$$

Por (4.10) temos:

$$\|f\ell_\varepsilon(\bar{\psi}(x))-\bar{\psi}(x)\| \leq \varepsilon\|x\| + \left[(1+\varepsilon)\alpha(x,\varepsilon,n)+\varepsilon\right]\|\bar{\psi}(x)-x\|$$

(chamando $(1+\varepsilon)\alpha(x,\varepsilon,n)+\varepsilon = \beta(x,\varepsilon,n)$) (4.13)

$$\|f\ell_\varepsilon(\bar{\psi}(x))-\bar{\psi}(x)\| \leq \varepsilon\|x\|+\beta(x,\varepsilon,n)\|\bar{\psi}(x)-x\| \qquad (4.14)$$

Finalmente, combinando (3.4), (4.6) e (4.13) obtemos:

$$\|f\ell_\varepsilon\bar{\psi}(x)-Z\| \leq \|f\ell_\varepsilon(\bar{\psi}(x))-\bar{\psi}(x)\|+\|\bar{\psi}(x)-Z\|$$

Por (4.13) e considerando $\beta = \beta(x,\varepsilon,n)$, $C = C(x, h,\varepsilon)$ e $S = S(x, Z)$:

$$\|f\ell_\varepsilon\bar{\psi}(x)-Z\| \leq \varepsilon\|x\| + \beta\ \|\bar{\psi}(x)-x\| + \|\bar{\psi}(x)-Z\| \ ;$$

$$\|f\ell_\varepsilon\bar{\psi}(x)-Z\| \leq \varepsilon\|x\| + \beta\ \|\bar{\psi}(x)-Z\| + \beta\|x-Z\| + \|\bar{\psi}(x)-Z\| \ ;$$

$$\|f\ell_\varepsilon\bar{\psi}(x)-Z\| \leq \varepsilon\|x\| + (1+\beta)\ \|\bar{\psi}(x)-Z\| + \beta\ \|x-Z\| \qquad (4.15)$$

com

$$\|\bar{\psi}(x)-Z\| \leq \|\bar{\psi}(x)-\phi(x)\| + \|\phi(x)-Z\|$$

Então

$$\|f\ell_\varepsilon\bar{\psi}(x)-Z\| \leq \varepsilon\|x\|+(1+\beta)\|\bar{\psi}(x)-\phi(x)\| + (1+\beta)\|\phi(x)-Z\| + \beta\|x-Z\| \ .$$

Por (3.4),

$$\|f\ell_\varepsilon\bar{\psi}(x)-Z\| \leq \varepsilon\|x\|+(1+\beta)\|\bar{\psi}(x)-\phi(x)\|+(1+\beta)S\ \|x-Z\|^2+\beta\|x-Z\| \ ;$$

$$\|f\ell_\varepsilon\bar{\psi}(x)-Z\| \leq \varepsilon\|x\|+(1+\beta)C\|x-Z\|+(1+\beta)C\|\phi(x)-Z\|+(1+\beta)S\|x-Z\|^2+\beta\|x-Z\|;$$

$$\|f\ell_\varepsilon\bar{\psi}(x)-Z\| \leq \varepsilon\|x\| \ \left|(1+\beta)C+\beta\right|\|x-Z\| + \left|(1+\beta)CS+(1+)S\right|\|x-Z\|^2 \ ,$$

ou seja

$$\boxed{\|f\ell_\varepsilon\bar{\psi}(x)-Z\| \leq \varepsilon\|x\|+L(x,\varepsilon,h,n)\|x-Z\| + Q(x,\varepsilon,h,n,Z)\|x-Z\|^2} \qquad (4.16)$$

onde

$$L(x,\varepsilon,h,n) = \beta(x,\varepsilon,n) + (1+\beta(x,\varepsilon,n))\ C(x,h,\varepsilon) \qquad (4.17)$$

$$Q(x,\varepsilon,h,n,Z) = (1+\beta(x,\varepsilon,n))\ (1+C(x,h,\varepsilon))\ S(x,Z) \qquad (4.18)$$

A equação (4.15) mostra que não devemos esperar que a solução do sistema não linear tenha precisão relativa melhor que a do computador. Mas se existe convergência, ela depende das quantidades:

$S(x,Z)$, fator de convergência do método de Newton.

$C(x,h,\varepsilon)$, é a medida do erro em $f\ell_{\varepsilon}(M(x,h))$ como uma aproximação numérica de $J(x)$.

$\beta(x,\varepsilon,n)$, reflete o número de condição do subproblema linear.

Isto tudo justifica a seguinte definição:

DEFINIÇÃO 4.3: Seja o sistema não linear definido em (3.1) e $x_0 \in D$ uma aproximação da solução Z de (3.1). Então este problema é solúvel por um método de "tipo Newton numérico", se as seguintes condições forem satisfeitas:

a) $J(x)$ e $H(x)$ existem em D e

$$K(J(x_0)) \leq 1 / (3\varepsilon g(n))$$

onde $g(n)$ depende do método usado para resolver o sistema linear.

b) h_0 satisfaz $C(x_0,h_0,\varepsilon) < 1$ e se

$$r_0 = \varepsilon\|x_0\| + \|\phi(x_0)-Z\| + \left[\beta(x_0,\varepsilon,n)+(1+\beta(x_0,\varepsilon,n))C(x_0,h_0,\varepsilon)\right] \cdot \|\phi(x_0)-x_0\|$$

então

$$r_0 < \|x_0 - Z\| .$$

c) $U_0 = \{y \in R^n \ / \ \|y-Z\| \leq r_0\} \subset D$

e

$$\sup_{x \in U_0} K(J(x)) < 1 / (3\varepsilon g(n)) .$$

d) $K = \sup_{\substack{x \in U_0 \\ k=1,2...}} C(x,h_k,\varepsilon)$, então h_k satisfaz $k < 1$.

e) $\sigma(F,Z,x_0,M,\varepsilon) = B+(1+B)\ K+(1+B)(1+k)Sr_0$ (4.19)

$\sigma(F,Z,x_0,M,\varepsilon) < 1$, onde

$S = \sup_{x \in U_0} S(x,Z)$ $\qquad$ $B = \sup_{x \in U_0} \beta(x,\varepsilon,n)$

Assim, se forem satisfeitas estas condições, σ será o número de solubilidade para resolver o sistema não linear $F(x) = 0$ com x_0 como ponto inicial e Z solução, através de um método de "tipo Newton".

Caso não seja satisfeita uma dessas condições, o número σ será definido como infinito.

TEOREMA 4.4: Se um sistema não linear definido como em (3.1) com aproximação inicial x_0 e solução Z é solúvel por um método de "tipo Newton", a sequência de pontos gerada por este método converge ao ponto x* com

$$\|x^* - Z\| \leq \varepsilon \|x^*\| .$$

CAPÍTULO V

PROGRAMA DISCUTIDO

No capítulo anterior vimos as condições para que um sistema seja solúvel por um método de "tipo Newton". Estas condições serão aplicadas em alguns sistemas para que possamos comprová-las ou não.

Com este objetivo foi realizado um programa no computador, usando aritmética de precisão dupla para simular a aritmética exata. Com a execução desse programa obtemos o número σ, também chamado de Bus que nos dará informação sobre o grau de dificuldade na resolução do sistema.

As condições que devem ser testadas são:

a) $K(J(x_0)) \leq 1 / (3\varepsilon g(n))$

b) $C(x_0,h_0,\varepsilon) < 1$ $\quad C(x_0,h_0,\varepsilon)$ dado por (4.6)

$$r_0 = \varepsilon\|x_0\| + \|\phi(x_0)-Z\| + \left[\beta(x_0,\varepsilon,n)+(1+\beta(x_0,\varepsilon,n))C(x_0,h_0,\varepsilon)\right] \|\phi(x_0)-x_0\|$$

$$r_0 < \|x_0-Z\|$$

c) $\sup K(J(x)) < 1 / (3\varepsilon g(n))$

d) $K = \sup C(x,h_k,\varepsilon) < 1.$

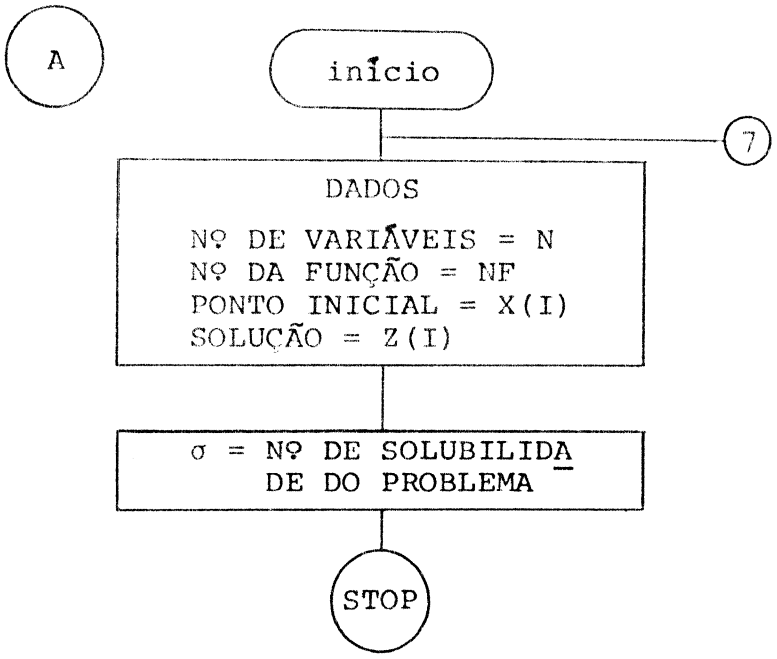
e) $\sigma(F,Z,x_0,M,\varepsilon) = B+(1+B)K+(1+B)(1+K)Sr_0$

onde $S = \text{Sup } S(x,Z)$ $\quad B = \text{Sup } \beta(x,\varepsilon,n)$

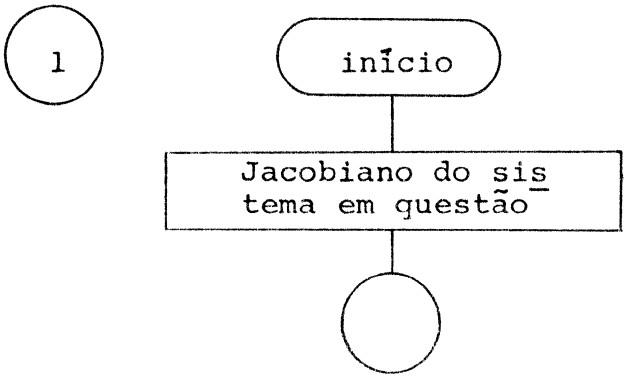
O programa está dividido em várias subroutinas cujos diagramas estão a seguir.

5.1) DIAGRAMAS DAS SUBROUTINAS

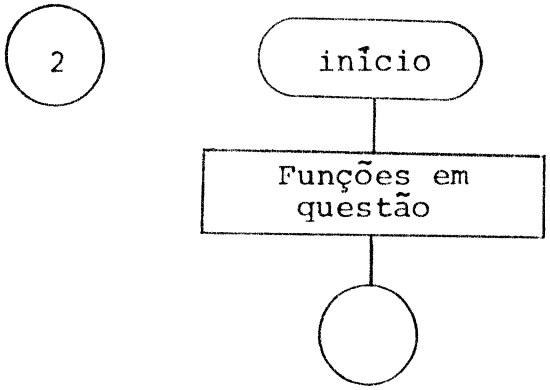
Programa Principal



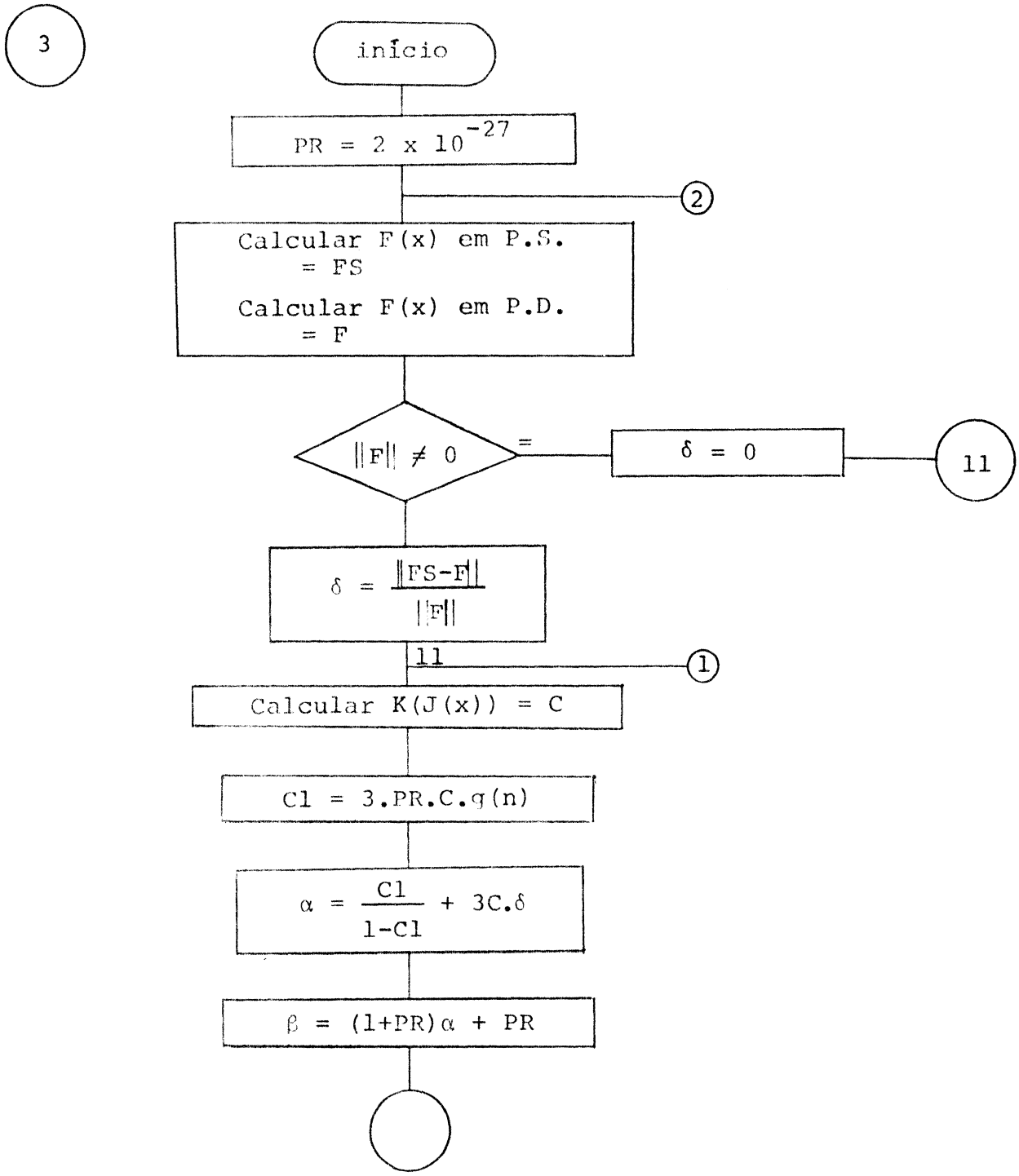
Subroutine Jacobi



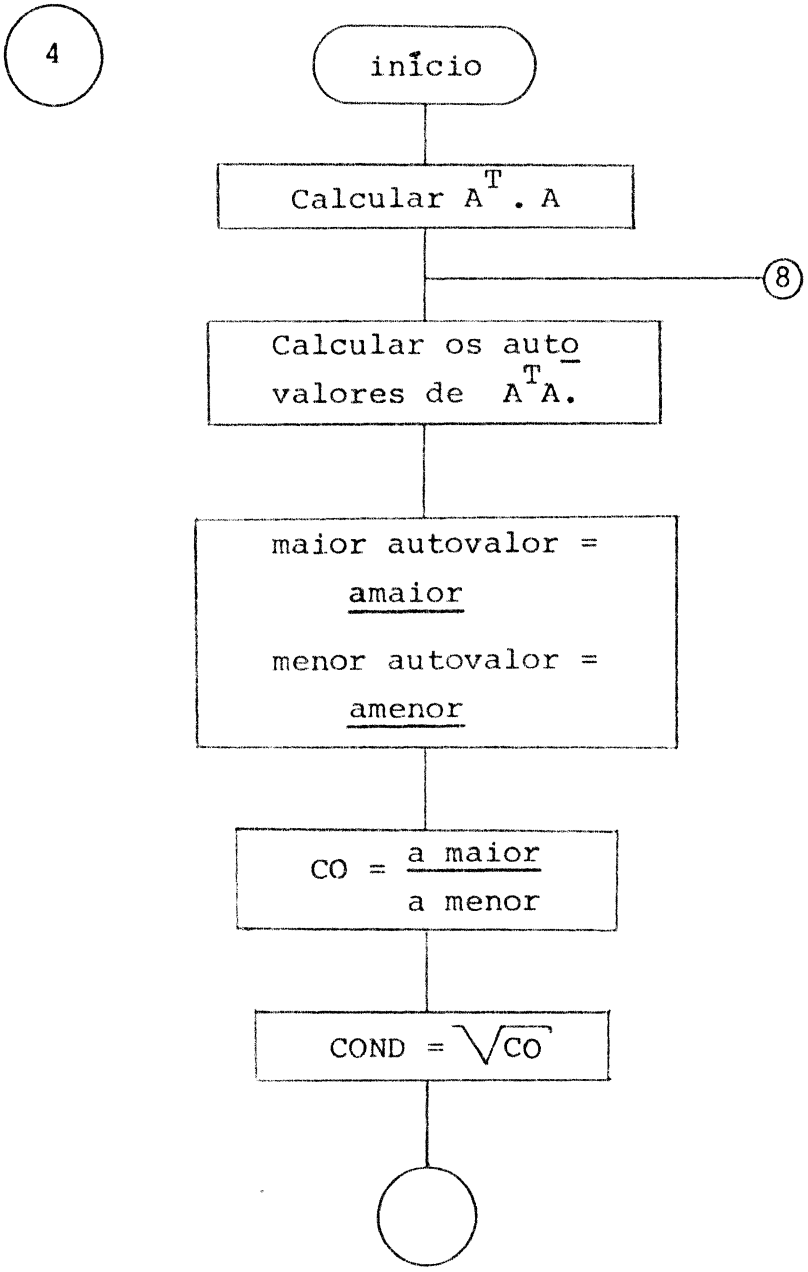
Subroutine Fun



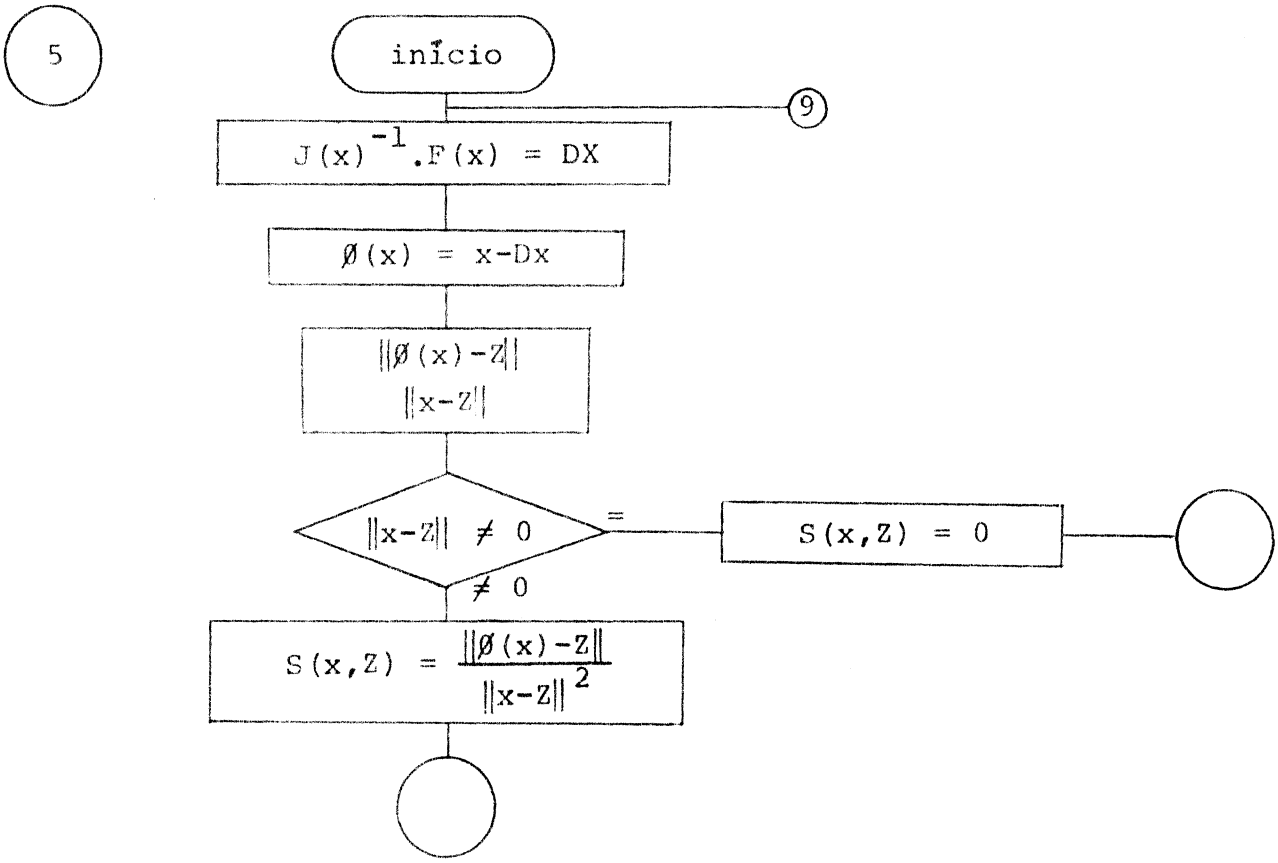
Subroutine Beta



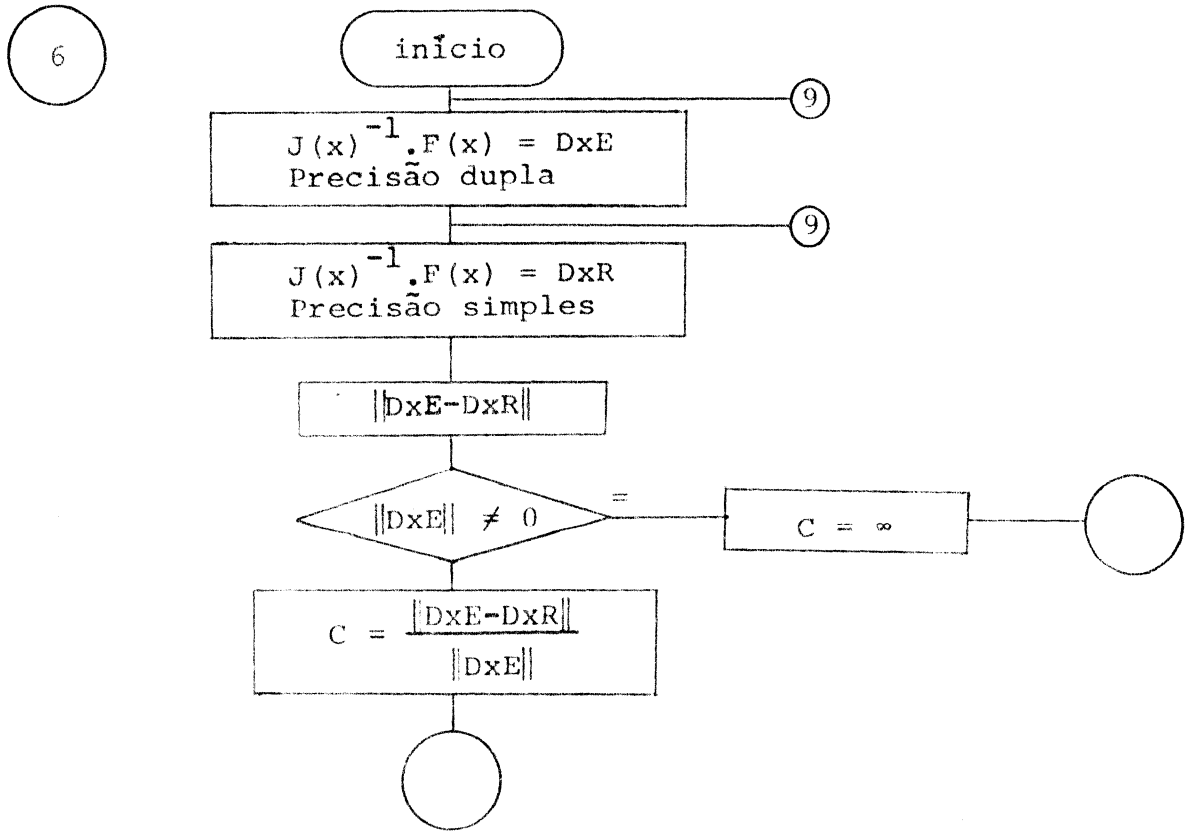
Subroutine Cond



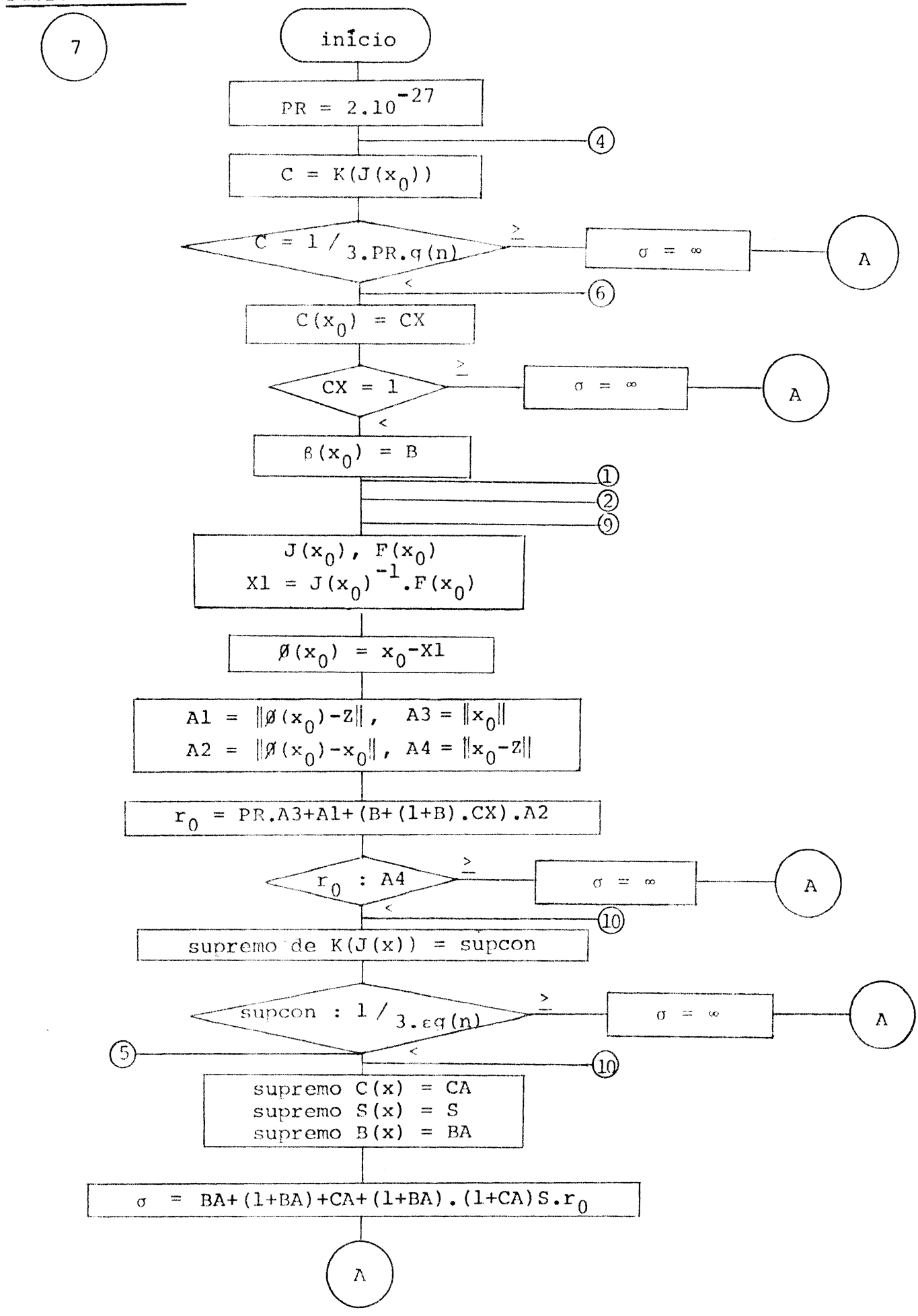
Subroutine Esse



Subroutine Cxeps



Subroutine Bus



8 Subroutine Deigen

Calcula autovalores com precisão dupla de matrizes reais e simétricas.

9 Subroutine Siste

Tirado do Livro - Computer Solucion of Linear Algebraic Systems - Forsythe-Moler pág. 68-69.

10 Subroutine Nelme

Minimiza funções de N variáveis, sem restrições pelo método de "Nelder-Mead" que não usa derivadas.

Tirado do Laboratório de Matemática Aplicada - IMECC - - UNICAMP.

CAPÍTULO VI

6.1) EXPERIÊNCIAS NUMÉRICAS

Usando as subroutinas do capítulo anterior obtivemos o número σ. Para que pudéssemos avaliar a medida em que esse número explica o comportamento de um método numérico, os mesmos sistemas foram resolvidos pelo método de Newton, para os mesmos pontos iniciais e anotamos o número de iterações para convergência.

OBSERVAÇÃO: O número σ pode ser infinito de maneiras diferentes.

a) Quando não obedece as condições de Bus então define-se $\sigma = \infty$.

b) Quando obedecendo as condições de Bus ele é calculado com valor infinito. Exemplo $\sigma = 0.1112968 \times 10^{38}$.

Com esses dados construimos as tabelas:

1 -

$$F_1(x) = 10(x_2 - x_1^2)$$
$$F_2(x) = 1 - x_1.$$

solução = (1,1) J(x) é sempre não singular

Ponto inicial		σ	Nº de iterações Newton
-5	11	∞	2
-4	11	∞	2
-3	11	∞	2
-2	11	0.9153333×10^{1}	2
-1	11	0.4020742×10^{1}	2
0	11	0.1001617×10^{1}	2
1	11	0.1791397×10^{-3}	1
2	11	0.1007956×10^{1}	2
-3	11	∞	2
-1	7	0.4019991×10	2
1	7	0.1791397×10^{-3}	1
2	7	0.1005532×10	2
3	7	0.4031844×10	2
4	7	∞	2
0.9	0.9	0.1791397×10^{-3}	2
1.1	1.2	0.1791397×10^{-3}	2

2 -

$$F_1(x) = x_1^2 - x_2 - 1$$
$$F_2(x) = (x_1-2)^2 + (x_2-0.5)^2 - 1 \ .$$

soluções: A = (1.546, 1.391)
B = (1.067, 0.139)

$J(x)$ singular se $x_1 \ . \ x_2 = 1$

Ponto inicial		σ	Nº de iterações	
3.2	6.0	∞	6	A
3.2	1.1	∞	5	A
1.3	0.2	0.8564867×10^{-5}	3	B
0.8	0.2	0.8564867×10^{-5}	3	B
1.7	1.3	0.1006262×10^{-4}	3	A
1.4	1.2	0.1006262×10^{-4}	3	A
0.3	3.7	∞	10	A
2.0	3.0	0.3606562×10^{6}	5	A
0.8	0.3	0.1303395	3	B
1	-1	0.6585745	4	B
0.73	0.23	0.1724944	4	B
-1.38	1.29	∞	9	B
1.6	1.33	0.1066262×10^{-4}	2	A
0.98	0.12	0.8564867×10^{-5}	2	B
1.13	0.14	0.8564867×10^{-5}	2	B
1.23	0.28	0.8564867×10^{-5}	3	B
0.1	2.0	∞	23	B
1	0.99	∞	13	B

3 -

$$F_1(x) = x_1^2 - 2x_2 + 1$$

$$F_2(x) = x_1 + 2x_2 - 3$$

solução: A = (1, 1)

B = (-1.402, 1.4831)

J(x) singular se $x_1 \cdot x_2 = -1/4$

Ponto inicial		σ	Nº de iterações	
-3.1	3.2	0.3691643×10^{34}	4	B
1.8	2.8	0.9503452	4	A
0.3	4.0	∞	6	A
1.0	1.5	0.6219589×10^{-1}	3	A
-1.8	1.8	0.1056818×10^{-4}	3	B
2.2	-3.1	∞	10	A
1.02	0.98	0.7160059×10^{-5}	2	A
1.13	0.68	0.7160059×10^{-5}	3	A
1.25	0.36	0.4268993	4	A
1.29	0.98	0.7160059×10^{-5}	3	A
0.96	1.3	0.7160059×10^{-5}	3	A
0.68	1.5	0.1104562	4	A
0.4	0.2	∞	5	A
-1.4	1.3	0.1056818×10^{-4}	3	B
-1.36	1.52	0.1056818×10^{-4}	2	B
-0.86	0.89	0.3189728×10^{34}	4	B
-1.26	0.98	0.9861792×10^{33}	4	B
-1.48	1.68	0.1056818×10^{-4}	3	B

4 -

$$F_1(x) = 10^4 x_1 x_2 - 1$$
$$F_2(x) = e^{-x_1} + e^{-x_2} - 1.0001$$

solução: (0.109815, 9.106227).

Ponto inicial		σ	Nº de iterações
$0.1098x10^{-4}$	9.1	∞	1
0	9.0	∞	2
$1.x10^{-5}$	3.0	∞	9
0	8.0	∞	3
$1.x10^{-5}$	8.0	∞	3
0	-1	∞	14
0	-2	∞	14
0	-1.5	∞	13
$1.x10^{-5}$	9	$0.6116326x10^{8}$	2
$1.12x10^{-5}$	9	$0.3553555x10^{12}$	2
$1.021x10^{-5}$	9.31	$0.6432029x10^{8}$	2
$0.5x10^{-5}$	8.9	∞	2
$0.8x10^{-5}$	9.1	∞	2
$0.986x10^{-5}$	9.2	$0.4899897x10^{-1}$	2

5 -

$$F_1(x) = 2(x_1-1) - 400x_1(x_2-x_1^2)$$
$$F_2(x) = 200(x_2-x_1^2)$$

solução: (1, 1)

J(x) singular se $x_1^2 = x_2 - 1/200$

Ponto inicial		σ	Nº de iterações
1.1	1.0		4
2.5	3.1		5
3.2	-1.8		4
-0.3	8		4
1.1	-2.1		3
2.5	-3		4
6	-6		5
-3	2	todos os pontos	5
-3.6	5.1	deram $\sigma = \infty$.	5
2.3	3.1		5
-8.2	3.1		5
2.3	3.3		5
1.01	0.99		3
1.0	1		0
-1.2	1.0		5
-1	1		2 .

6 -

$$F_1(x) = x_1 - 1$$

$$F_2(x) = x_1 . x_2 - 1$$

solução: (1 , 1)

J(x) singular se $x_1 = 0$

Ponto inicial		σ	Nº de iterações
1.4	2.2	0.1558685	2
-1.1	-3.2	∞	2
2.3	1.6	0.1541738	2
2.3	1.8	0.2055617	2
6.1	5.0	∞	2
4.2	-3.4	∞	2
0.98	0.84	0.9370357×10^{-5}	2
1.01	0.98	0.9370357×10^{-5}	2
1.08	0.83	0.9370357×10^{-5}	2
1.13	0.96	0.9370357×10^{-5}	2
0.95	1.32	0.9370357×10^{-5}	2
0.84	1.08	0.9370357×10^{-5}	2
0.94	1.07	0.9370357×10^{-5}	2
0.46	1.38	0.2027968	2

7 -

$$F_1(x) = -13+x_1((-x_2+5)x_2-2)x_2$$
$$F_2(x) = -29+x_1((x_2+1)x_2-14)x_2$$

solução: (5, 4)

Ponto inicial		σ	Nº de iterações
3.0	5.0	∞	4
-4.0	7.0	∞	5
4.0	3.0	∞	5
2.0	1.0	∞	38
6.0	5.0	∞	4
8.0	6.0	∞	5
5.23	4.32	∞	3
4.98	3.86	∞	3
5.0	4.0	∞	0
15.0	-2	0.1112968×10^{38}	36
-5.0	0	0.1455062×10^{38}	21
-5.0	3	0.1457622×10^{38}	5
0	2.24	∞	15

8 -

$$F_1(x) = x_1$$

$$F_2(x) = 10x1 \;/\; (x_1+0.1) + 2.x_2^{\,2}.$$

solução: (0, 0)

J(x) singular se $x_1 = 0$ ou $x_2 = 0$

Ponto inicial		σ	Nº de iterações
0.01	-0.001	∞	1
0.032	-0.0021	∞	1
-0.002	-0.008	∞	1
0.3	-0.2	∞	1
1.0	-1.0	∞	1
0.902	0.3	∞	1
-3	1	∞	1
0	1	∞	0
-1	1	∞	1
-0.9	0.24	∞	1

9 -

$$F_1(x) = 3x_1 + x_2 + 2x_3^2 - 3$$
$$F_2(x) = -3x_1 + 5x_2^2 + 2x_1x_3 - 1$$
$$F_3(x) = 25x_2x_1 + 20x_3 + 12$$

solução: A = (0.2960523, 0.6874306, -0.8492385)

B = (1.1, -0.8, 0.5)

Ponto inicial			σ	Nº de iterações	
0	0	0	∞	6	A
1.2	-0.75	0.5	0.1362366×10^{-3}	2	B
0.28	0.67	-0.74	0.747409×10^{-4}	3	A
1.1	2.1	3.1	∞	5	A
1.0	0.5	-1	0.2832019×10^{6}	4	A
0.82	0.51	-0.84	0.7474094×10^{-4}	3	A
2.0	3.0	1.0	∞	11	A
0.1	0.2	0.3	0.8949503×10^{7}	5	A
1.295	0.686	-0.848	∞	2	A
1.15	-0.79	0.48	0.1362366×10^{-3}	2	B
0.3	0.7	-0.9	0.7474094×10^{-4}	2	A
1.3	-0.9	0.6	0.1362366×10^{-3}	3	B

10 -

$$F_1(x) = 2x_1 + 3x_2 - x_3x_4$$
$$F_2(x) = x_1 - 2x_2 + x_3^2x_4 + 3$$
$$F_3(x) = 8x_1^2 + 6x_2 - x_3x_2 + 8x_4$$
$$F_4(x) = x_1x_2 - x_3x_4 + 2x_1$$

solução: A = (1, 0, -2, -1)

B = (3, 0.342, -0.756, -9.2)

Ponto inicial				σ	Nº de iterações	
1	0.1	-2	-1	0.9230821×10^{-2}	1	A
1.2	0.8	-2	-0.9		100	-
1.3	0.9	-2.1	-1.2	∞	6	B
0.9	0.1	-2.2	-0.9		100	-
1.0	0	-1.8	-0.3	∞	6	A
1.3	0.9	-2	-0.8	∞	7	B
2.0	1.0	3.0	1.0		65	-
0.1	0.8	0.3	0.6	∞	40	A .

11 -

$$F_i = \sum_{\substack{j=1 \\ j \neq i}}^{6} \cot(\beta_i x_j) \qquad i = 1 \ldots 6$$

$\beta_1 = 0.02249$ $\qquad$ $\beta_4 = 0.02000$

$\beta_2 = 0.02166$ $\qquad$ $\beta_5 = 0.01918$

$\beta_3 = 0.02083$ $\qquad$ $\beta_6 = 0.01835$

solução: (121.850, 114.161, 93.6488, 62.3186, 41.3219, 30.5027).

Ponto inicial	σ	Nº de iterações
$x_i = 75$	∞	5
121, 114, 93 62, 41, 30	0.316643×10^6	2

12 -

$$F_i(x) = (3-kx_i)x_i+1 - x_{i-1} - 2x_i+1$$

$$k = 0.1 \qquad i = 1,2...10$$

solução: -0.6854835

-0.5516827

-0.5086049

-0.4947104

-0.4902261

-0.4887785

-0.4883113

-0.4881604

-0.481117

-0.480960

Ponto inicial	σ	Nº de iterações
$x_i = -1$	0.1001086×10^{6}	10
-0.6, -0.5, -0.5 -0.4, -0.4, -0.4 -0.4, -0.4, -0.4 -0.4	0.8733329×10^{-3}	8

CONCLUSÕES

Através da teoria desenvolvida temos que, um sistema é fácil de ser resolvido por um método de "tipo Newton", se o número σ, dado por (4.18) tem ordem de grandeza em torno da unidade. Caso contrário o sistema é difícil de se resolver.

Neste trabalho apresentam-se algumas experiências numéricas que nos permitem dizer que do mesmo modo que em sistemas lineares, quando o número σ é pequeno o sistema é facilmente solúvel, porém a recíproca não é verdadeira.

Para alguns casos essa classificação é muito grosseira, pois σ é um majorante e como foi dito no Capítulo I, para sistemas lineares, se o número de condição é pequeno (~ 1) posso afirmar com certeza que a convergência é rápida. Mas em muitos casos onde $\sigma = \infty$ a convergência ainda pode ser obtida com rapidez. É o caso dos problemas 7, 8 e 11.

Além disso devemos observar que no caso da solução pertencer à região onde o Jacobiano é singular podemos afirmar a priori que σ faz uma classificação grosseira. É o caso do problema 8.

PROGRAMA PRINCIPAL, SUBROUTINES:

JACOBI, FUN, SISTE, DECOMP, SING, SOLVE, NELME, BUS, COND, BETA, CXEPS, ESSE, DEIGEN.

```
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(20),Z(20),X1(20)
      COMMON/BFGS/NF
100   TYPE 15
15    FORMAT(////,' DATA "1" SE QUER CONTINUAR BRINCANDO')
      ACCEPT 16,LTT
16    FORMAT(I10)
      IF(LTT.NE.1)STOP
      TYPE 41
41    FORMAT(1X,'NUMERO DE VARIAVEIS:',$)
      ACCEPT 16,N
      TYPE 21
21    FORMAT(/,1X,'NUMERO DA FUNCAO:',$)
      ACCEPT 22,NF
22    FORMAT(I2)
      TYPE 46
46    FORMAT(1X,'PONTO INICIAL')
      DO 47 I=1,N
      TYPE 100,I
100   FORMAT(1X,'X(',I2,')=',$)
      ACCEPT 20,X(I)
20    FORMAT(F)
47    CONTINUE
      WRITE(3,51)
51    FORMAT(1X,/////,'  SISTEMA DE EQUACOES A SER ESTUDADO')
      GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14)NF
1     WRITE(3,30)
30    FORMAT(1X,'F1(X)=10*(X(2)-X(1)**2)',/,1X,'F2(X)=1-X(1)')
      GO TO 40
2     WRITE(3,31)
31    FORMAT(1X,'F1(X)=X1**2-X2-1',/,1X,'F2(X)=(X1-2)**2+(X2-0.
     *5)**2-1')
      GO TO 40
3     WRITE(3,32)
32    FORMAT(1X,'F1(X)=X1**2-2*X2+1',/,1X,'F2(X)=X1+2*X2**2-3')
      GO TO 40
4     WRITE(3,33)
33    FORMAT(1X,'F1(X)=10000.*X1*X2-1',/,1X,'F2(X)=EXP**-X1-EXP
     ***-X2-1.0001')
      GO TO 40
5     WRITE(3,34)
```

```
      FORMAT(1X,'F1(X)=2*(X1-1)-4.0*X1*(X2-X1**2)',/,1X,'F2(
     *X)=2.0*(X2-X1**2)')
      GO TO 40
6     WRITE(3,35)
35    FORMAT(1X,'F1(X)=X1-1',/,1X,'F2(X)=X1*(X2-1')
      GO TO 40
7     WRITE(3,36)
36    FORMAT(1X,'F1(X)=-13+X1+((-X2+5)*X2-2)*X2',/,1X,'F2(X)=-2
     *9+X1+((X2+1)*X2-14*X2)')
      GO TO 40
8     WRITE(3,37)
37    FORMAT(1X,'F1(X)=X1',/,1X,'F2(X)=10*X1/(X1+0.1)+2*X2**2)')
      GO TO 40
9     WRITE(3,43)
43    FORMAT(1X,'F1=3X1+X2+2X3**2-3',/,1X,'F2=-3X1+5X2**2+2X1X3-
     *1',/,1X,'F3=2SX1X2+20X3+12')
      GO TO 40
10    WRITE(3,49)
49    FORMAT(1X,'F1=X1+X2+X3-5',/,1X,'F2=2X1+3X2+2X3-11',/,1X,'F3=
     *X1+X2-X3')
      GO TO 40
11    WRITE(3,50)
50    FORMAT(1X,'F1=2X1+3X2-(3X1',/,1X,'F2=X1-2X2+X3**2X4+3',
     */,1X,'F3=8X1**2+5X2-X3X4+5X4',/,1X,'F4=X1(2-X3X4+2X1')
      GO TO 40
12    WRITE(3,53)
53    FORMAT(1X,' SISTEMA DE 10 VARIAVEIS')
      GO TO 40
13    WRITE(3,54)
54    FORMAT(1X,'SISTEMA DE 6 VARIAVEIS')
      TYPE 25
25    FORMAT(1X,'OBJECAO')
      DO 44 I=1,N
      TYPE 23,I
23    FORMAT(1X,'G(',I2,')=',$)
      ACCEPT 24,Z(I)
24    FORMAT(G)
44    CONTINUE
      WRITE(3,130)(X(I),I=1,N)
130   FORMAT(1X,'VALOR INICIAL',/,6E16.7)
      WRITE(3,17)(Z(I),I=1,6)
17    FORMAT(1X,'OBJECAO:',/,6E16.7)
      CALL DUS(1,X,...,IOUT,NU)
      WRITE(3,18)...
18    FORMAT(1X,'...=',E16.7)
      GOTO 140
      END
```

```
      SUBROUTINE JACISI (S,X,A)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(2),A(2,10),SPEIA(10),DCOEF(10)
      COMMON/SPEC/IE
      GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14)IE
1     A(1,1)=-2.*X(1)
      A(1,2)=10.D0
      A(2,1)=-1.D0
      A(2,2)=1.D0
      RETURN
2     A(1,1)=2.*X(1)
      A(1,2)=-1.
      A(2,1)=2.*(X(1)-2.)
      A(2,2)=2.*(X(2)-0.5)
      RETURN
3     A(1,1)=2.*X(1)
      A(1,2)=-2.
      A(2,1)=1.
      A(2,2)=4.*X(2)
      RETURN
4     A(1,1)=10000.*X(2)
      A(1,2)=10000.*X(1)
      A(2,1)=-EXP**-X(1)
      A(2,2)=-EXP**-X(2)
      RETURN
5     A(1,1)=2.-400.*X(2)+1200.*X(1)**2.
      A(1,2)=-400.*X(1)
      A(2,1)=-100.*X(1)
      A(2,2)=200.
      RETURN
6     A(1,1)=1.
      A(1,2)=0.
      A(2,1)=X(2)
      A(2,2)=X(1)
      RETURN
7     A(1,1)=1.
      A(1,2)=-4.*X(2)**2+10.*X(2)-2.
      A(2,1)=1.
      A(2,2)=3.*X(2)**2+2.*X(2)-14.
      RETURN
8     A(1,1)=1.
      A(1,2)=0.
      A(2,1)=20*X(2)**2+1/((X(1)+0.1)**2+4.*X(2)**2*(X(1)+0.1)
     **+3*X(2)**4)
      A(2,2)=-40*X(1)*X(2)/((X(1)+0.1)**2+4*X(2)**2*(X(1)+0.1)+
     *4*X(2)**4)
      RETURN
```

```
9       A(1,1)=3.
        A(1,2)=1.
        A(1,3)=4.*X(3)
        A(2,1)=-5.+2.*X(3)
        A(2,2)=10.*X(2)
        A(2,3)=2.*X(1)
        A(3,1)=25.*X(2)
        A(3,2)=25.*X(1)
        A(3,3)=20.
        RETURN
1       A(1,1)=1.
        A(1,2)=1.
        A(1,3)=1.
        A(2,1)=2.
        A(2,2)=3.
        A(2,3)=1.
        A(3,1)=1.
        A(3,2)=1.
        A(3,3)=-1.
        RETURN
11      A(1,1)=2.
        A(1,2)=3.
        A(1,3)=-X(4)
        A(1,4)=-X(3)
        A(2,1)=1.
        A(2,2)=-2.
        A(2,3)=2.*X(3)*X(4)
        A(2,4)=X(3)**2
        A(3,1)=10.*X(1)
        A(3,2)=5.-X(3)
        A(3,3)=-X(2)
        A(3,4)=5.
        A(4,1)=X(2)+2.
        A(4,2)=X(1)
        A(4,3)=-X(4)
        A(4,4)=-X(3)
        RETURN
```

```
1       A(1,1)=3.-0.2*X(1)
        DO 101 I=2,4
        DO 101 J=1,4
        IF(I.EQ.J)GO TO 201
        IF(J.EQ.I-1)GO TO 301
        A(I,J)=0.
        GO TO 101
201     A(I,J)=3.-0.2*X(I)
        GO TO 101
301     A(I,J)=-1.
```

```
      A(1,2)=Z.
110   CONTINUE
      RETURN
15    BETA(1)=0.02243
      BETA(2)=0.02166
      BETA(3)=0.02053
      BETA(4)=0.02030
      BETA(5)=0.01913
      BETA(6)=0.01835
      DO 100 I=1,6
      DO 100 J=1,6
      Z=BETA(I)*X(J)
      IF(I.EQ.J)GO TO 200
      DCOT(L)=-1./(DSIN(Z))**2
      A(I,J)=BETA(I)*DCOT(L)
      GO TO 100
200   A(I,J)=0.
100   CONTINUE
      RETURN
      END



      SUBROUTINE FUN(J,X,F)
      IMPLICIT REAL*8 (A-G,O-Z)
      DIMENSION X(20),V(10,10),S(20),BETA(10),COT(10),F(20)
      COMMON/NFUN/NF
      GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13)NF
1     F(1)=10.D0*(X(2)-X(1)**2)
      F(2)=1.D0-X(1)
      RETURN
9     F(1)=3.*X(1)+X(2)+2.*X(3)**2-3.
      F(2)=-3.*X(1)+5.*X(2)**2+2.*X(1)*X(3)-1.
      F(3)=25.*X(1)*X(2)+20.*X(3)+12.
      RETURN
2     F(1)=X(1)**2-X(2)-1.
      F(2)=(X(1)-2.)**2+(X(2)-0.5)**2-1.
      RETURN
3     F(1)=X(1)**2-2.*X(2)+1.
      F(2)=X(1)+2.*X(2)**2-3.




      RETURN
4     F(1)=10000.*X(1)*X(2)-1.
      F(2)=EXP**-X(1)-EXP**-X(2)-1.0001
      RETURN
5     F(1)=2.*(X(1)-1.)-400.*X(1)*(X(2)-X(1)**2)
      F(2)=200.*(X(2)-X(1)**2)
      RETURN
6     F(1)=X(1)-1.
      F(2)=X(1)*X(2)-1
      RETURN
```

```
7       F(1)=-13.+X(1)+((-X(2)+5.)*X(2)-2.)*X(2)
        F(2)=-29.+X(1)+((X(2)+1)*X(2)-14.*X(2))
        RETURN
8       F(1)=X(1)
        F(2)=10.*X(1)/((X(1)+0.1)+2.*X(2)**2)
        RETURN
1       F(1)=X(1)+X(2)+X(3)-6.
        F(2)=2*X(1)+3*X(2)+X(3)-11.
        F(3)=X(1)+X(2)-X(3)
        RETURN
11      F(1)=2.*X(1)+3.*X(2)-X(3)*X(4)
        F(2)=X(1)-2*X(2)+X(3)**2*X(4)+3.
        F(3)=6.*X(1)**2+6.*X(2)-X(3)*X(2)+8.*X(4)
        F(4)=X(1)*X(2)-X(3)*X(4)+2.*X(1)
        RETURN
1       F(1)=(3.-0.1*X(1))*X(1)+1.-2.*X(2)
        DO 101 I=2,N
        F(I)=(3.-0.1*X(I))*X(I)+1.-X(I-1)
11      CONTINUE
        RETURN
1       BETA(1)=0.02243
        BETA(2)=0.04160
        BETA(3)=0.02883
        BETA(4)=0.02920
        BETA(5)=0.01413
        BETA(6)=0.01835
        DO 100 I=1,6
        F(I)=0.
        DO 100 J=1,6
        Z=BETA(I)*X(J)
        IF(I.EQ.J)GO TO 100
        COT(L)=DCOS(Z)/DSIN(Z)
        F(I)=F(I)+COT(L)
10      CONTINUE
        RETURN
        END

        SUBROUTINE SIMEQ(N,A,B,DX)
        IMPLICIT REAL*8 (A-H,O-Z)
        COMMON/NFUN/NF
        DIMENSION A(20,20),UL(20,20),B(20),DX(20)
        CALL DECOMP(N,A,UL)
        CALL SOLVE (N,UL,B,DX)
        RETURN
        END

        SUBROUTINE DECOMP (NN,A,UL)
        IMPLICIT REAL *8(A-H,O-Z)
        DIMENSION A(20,20),UL(20,20),SCALES(20),IPS(20)
        COMMON/NFUN/NF
        COMMON IPS
        N=NN
        DO 5 I=1,N
        IPS(I)=I
        ROWNR=1.0
        DO 2 J=1,N
```

```
      UL(I,J)=A(I,J)
      IF(ROWNRM-DABS(UL(I,J)))1,2,2
1     ROWNRM=DABS(UL(I,J))
2     CONTINUE
      IF(ROWNRM)3,4,3
3     SCALES(I)=1.0/ROWNRM
      GO TO 5
4     CALL SING (1)
      SCALES(I)=0.0
5     CONTINUE
      NM1=N-1
      DO 17 K=1,NM1
      BIG=0.0
      DO 11 I=K,N
      IP=IPS(I)
      SIZE=DABS(UL(IP,K))*SCALES(IP)
      IF(SIZE-BIG)11,11,10
10    BIG=SIZE
      IDXPIV=I
11    CONTINUE
      IF(BIG)13,12,13
12    CALL SING(2)
      GO TO 17
13    IF(IDXPIV-K)14,15,14
14    J=IPS(K)
      IPS(K)=IPS(IDXPIV)
      IPS(IDXPIV)=J
15    KP=IPS(K)
      PIVOT=UL(KP,K)
      KP1=K+1
      DO 16 I=KP1,N
      IP=IPS(I)
      EM=-UL(IP,K)/PIVOT
      UL(IP,K)=-EM
      DO 16 J=KP1,N
      UL(IP,J)=UL(IP,J)+EM*UL(KP,J)
16    CONTINUE
17    CONTINUE
      KP=IPS(N)
      IF(UL(KP,N))19,18,19
18    CALL SING(2)
19    RETURN
      END

      SUBROUTINE SING (IWHY)
11    FORMAT(' MATRIX WITH ZERO ROW IN DECOMPOSE')
12    FORMAT(' SINGULAR MATRIX IN DECOMPOSE ZERO DIVIDE IN SOLVE')
13    FORMAT(' NO CONVERGENCE IN IMPROV MATRIX IS NEARLY  SINGULA
     *R')
```

```
      NOUT=3
      GO TO (1,2,3),IWHY
1     WRITE(NOUT,11)
      GO TO 10
2     WRITE(NOUT,12)
      GO TO 10
3     WRITE(NOUT,13)
10    RETURN
      END


      SUBROUTINE SOLVE(NN,NG,B,X)
      IMPLICIT REAL *8(A-H,O-Z)
      DIMENSION UL(20,20),B(20),X(20),IPS(20)
      COMMON/NRUN/NF
      COMMON IPS
      N=NN
      NP1=N+1
      IP=IPS(1)
      X(1)=B(IP)
      DO 2 I=2,N
      IP=IPS(I)
      IM1=I-1
      SUM=0.0
      DO 1 J=1,IM1
1     SUM=SUM+UL(IP,J)*X(J)
2     X(I)=B(IP)-SUM
      IP=IPS(N)
      X(N)=X(N)/UL(IP,N)
      DO4 IBACK=2,N
      I=NP1-IBACK
      IP=IPS(I)
      IP1=I+1
      SUM=0.0
      DO3 J=IP1,N
3     SUM=SUM+UL(IP,J)*X(J)
4     X(I)=(X(I)-SUM)/UL(IP,I)
      RETURN
      END


      SUBROUTINE NEWTON(N,X,FUN,JTEP,F,EPS,KOV,ITMAX
     *,P,IER)
      IMPLICIT REAL *8(A-H,O-Z)
      COMMON/NRUN/NF
      DIMENSION X(N),JTEP(N),P(N,1),XL(50),XH(50)
     *,XR(50),XB(50),XF(50),XHL(50),XC(50),FFE(51)
      F=1.D30
      N1=N+1
```

```
      ALFA=1.
      BETA=0.5
      GAMA=2.
      KON=0
C     ARMAR O SIMPLEX INICIAL
      DO 1 I=1,N
1     P(I,N1)=X(I)
      DO 2 J=1,N
      DO 3 I=1,N
3     P(I,J)=X(I)
2     P(J,J)=P(J,J)+STEP(J)
C     AVALIAR A FUNCAO NOS VERTICES DO SIMPLEX INICIAL
      DO 4 I=1,N1
      KON=KON+1
      IF(KON.GE.ITMAX)GOTO30
      EFE(I)=FUN(P(1,I))
      IF(EFE(I).GT.F)GOTO4
      F=EFE(I)
      DO 37 KI=1,N
37    X(KI)=P(KI,I)
4     CONTINUE
C     CALCULAR XL E XH
100   FMENOR=1.E30
      FMAIOR=-1.E30
      DO 5 I=1,N1
      IF(EFE(I).GT.FMENOR)GO TO 6
      IL=I
      FXL=EFE(I)
      FMENOR=FXL
      DO 7 J=1,N
7     XL(J)=P(J,IL)
6     IF(EFE(I).LT.FMAIOR)GO TO 5
      IH=I
      FXH=EFE(I)
      FMAIOR=FXH
      DO 8 J=1,N
8     XH(J)=P(J,IH)
5     CONTINUE
C     TESTAR CRITERIO DE PARADA
      A1=0.
      DO 9 J=1,N
      DO 9 I=1,N
9     A1=DMAX1(A1,DABS(P(I,J)-P(I,N1)))
      IF(A1.GT.EPS)GO TO10
      IER=0
      RETURN
10    IF(KON.LT.ITMAX)GO TO 12
30    IER=1
      RETURN
C     CALCULAR O CENTROIDE
12    DO 15 I=1,N
15    XB(I)=0.
      DO 14 J=1,N1
```

```
      IF(J.EQ.IH)GO TO 14
      DO 16 I=1,N
16    XB(I)=XB(I)+P(I,J)
14    CONTINUE
      DO 17 I=1,N
17    XB(I)=XB(I)/DFLOAT(N)
C     CALCULAR XR
      DO 18 I=1,N
18    XR(I)=XB(I)+ALFA*(XB(I)-XH(I))
      FXR=FUN(XR)
      IF(FXR.GT.F)GOTO39
      F=FXR
      DO 38 KI=1,N
38    X(KI)=XR(KI)
39    KON=KON+1
      IF(KON.GE.ITMAX)GOTO36
      IF(FXL.LE.FXR)GO TO 19
C     CALCULAR XE
      DO 20 I=1,N
20    XE(I)=XB(I)+GAMA*(XR(I)-XB(I))
      FXE=FUN(XE)
      IF(FXE.GT.F)GOTO40
      F=FXE
      DO 41 KI=1,N
41    X(KI)=XR(KI)
40    KON=KON+1
      IF(KON.GE.ITMAX)GOTO36
      IF(FXR.LE.FXE)GO TO 21
      DO 22 I=1,N
22    P(I,IH)=XL(I)
      FFE(IH)=FXL
      GO TO 100
21    DO 23 I=1,N
23    P(I,IH)=XR(I)
      FFE(IH)=FXR
      GO TO 100
19    AM=-1.E30
      DO 24 J=1,N1
      IF(J.EQ.IH)GO TO 24
      AM=DMAX1(AM,FFE(J))
24    CONTINUE
      IF(AM.LT.FXR)GO TO25
      DO 26 I=1,N
26    P(I,IH)=XL(I)
      FFE(IH)=FXL
      GO TO 100
C     CALCULAR XBB
25    IF(FXR.LT.FXP)GO TO27
      DO 28 I=1,N
```

```
2       XHL(I)=XR(I)
        FXHL=FXR
        GO TO 29
27      DO 30 I=1,N
3       XHL(I)=XH(I)
        FXHL=FXH
C       CONTRACAO
29      DO 31 I=1,N
31      XC(I)=XB(I)+BETA*(XHL(I)-XB(I))
        FXC=FUN(XC)
        IF(FXC.GT.F)GOTO42
        F=FXC
        DO 43 KI=1,N
4       X(KI)=XC(KI)
42      KON=KON+1
        IF(KON.GE.IIMAX)GOTO36
        IF(FXC.GT.FXHL)GO TO 32
        DO 33 I=1,N

33      P(I,IH)=XC(I)
        EFE(IH)=FXC
        GO TO 100
3       DO 34 J=1,N1
        IF(J.EQ.IL)GO TO 34
        DO 35 I=1,N
35      P(I,J)=P(I,J)+0.5*(XL(I)-P(I,J))
        EFE(J)=FUN(P(1,J))
        IF(EFE(J).GT.F)GOTO34
        F=EFE(J)
        DO 45 KI=1,N
45      X(KI)=P(KI,J)
44      KON=KON+1
        IF(KON.GE.ITMAX)GOTO36
34      CONTINUE
        GO TO 100
        END


        SUBROUTINE BUS(N,X0,Z,BUSNUM)
C       DEFINICAO 3.4 (PAG.279)
        IMPLICIT REAL*8 (A-H,O-Z)
        COMMON/NFUN/NF
        COMMON/EPREC/R0,ZZ(20),NN
        EXTERNAL F1,F2,F3,F4
        DIMENSION XA(20),P(420),Z(20)
        DIMENSION  X0(20),A(20,20),F(20),X1(20),X0LZ(20),X1LZ(20
     *),X1LX0(20)
        DIMENSION STEP(20)
        GN=20.00*DFLOAT(N)**3
```

```
      CALL COND(*,X0,C)
      WRITE(3,15)C
15    FORMAT(1X,'NRO DE CONDICAO DO JACOBIANO EM X0:',E16.7)
      DO16 I=1,N
16    Z0(I)=Z(I)
      N1=1
      PR=2.D0**-27
      IF(C.LT.1.D0/(3.D0*PR*G1))GO TO 3
      BUSNUM=1.D32
      RETURN
3     CALL CXLPG(N,X0,CX)
      WRITE(3,26)CX
26    FORMAT(1X,'CXLPG(X0)=',E16.6)
      IF(CX.LT.1.D0)GO TO 9
      BUSNUM=1.D31
      RETURN
9     CALL BETA(N,X0,B)
      WRITE(3,27)B
27    FORMAT(1X,'BETA(X0)=',E16.8)
C     CALCULO DO PONTO SEGUINTE(A ITERACAO NRO 1)
      CALL JACOBI(N,X0,A)
      CALL FUN(N,X0,F)
      CALL SISTE(N,A,F,X1)
      DO 12 I=1,N
      X1(I)=X0(I)-X1(I)
      WRITE(3,7)X1(I)
7     FORMAT(1X,'F1(X0)=',E16.7)
      X0LZ(I)=X0(I)-Z(I)
      X1LZ(I)=X1(I)-Z(I)
```

```
12    X1LX0(I)=X1(I)-X0(I)
      WRITE(3,16)
16    FORMAT(1X,'PONTO X1=')
      WRITE(3,17)(X1(I),I=1,N)
17    FORMAT(1X,E16.7)
      A1=ANORM2(X1LZ,N)
      A2=ANORM2(X1LX0,N)
      A3=ANORM2(X0,N)
      A4=ANORM2(X0LZ,N)
      WRITE(3,4)A1,A2,A3,A4
4     FORMAT(1X,'//F1(X0)-Z//=',E15.7,3X,'//F1(X0)-X0//=',E16.7,/,
     *1X,'//X0//=',E16.7,3X,'//X0-Z//=',E16.7)
      RO=PR*A3+A1+(B+(1.D0+B)*CX)*A2
      WRITE(3,23)RO
23    FORMAT(1X,'RO=',E16.7)
      IF(RO.LT.A4)GO TO 11
      BUSNUM=1.D32
      RETURN
11    IIMAX=200
      DO 1 I=1,N
```

```
1       SIEP(1)=0.1
        EPS=10.**-4
        DO 19 I=1,N
19      XX(I)=Z(I)
        CALL NELME(N,XX,F4,STEP,SUPCON,EPS,KON,ITMAX,P,IER)
        IF(SUPCON.GT.0.)WRITE(3,28)
28      FORMAT(1X,'HA ERRO EM F4')
        SUPCON=-SUPCON
        WRITE(3,20)SUPCON,IER
        IF(SUPCON.LT.1/(3.D0*PR*GN))GO TO 21
        BUSNUM=1.D33
        RETURN
21      DO 22 I=1,N
22      XX(I)=Z(I)
        CALL NELME(N,XX,F1,STEP,CA,EPS,KON,ITMAX,P,IER)
        IF(CA.GT.0.)WRITE(3,29)
29      FORMAT(1X,'HA ERRO EM F1')
        CA=-CA
        WRITE(3,5)CA
5       FORMAT(1X,'SUPREMO DE C=',E16.7)
        DO 25 I=1,N
25      XX(I)=Z(I)
        CALL NELME(N,XX,F2,STEP,S,EPS,KON,ITMAX,P,IER)
        IF(S.GT.0)WRITE(3,30)
30      FORMAT(1X,'HA ERRO EM F2')
        S=-S
        DO 24 I=1,N
24      XX(I)=Z(I)
        CALL NELME(N,XX,F3,STEP,BA,EPS,KON,ITMAX,P,IER)
        IF(BA.GT.0)WRITE(3,31)
31      FORMAT(1X,'HA ERRO EM F3')
        BA=-BA
        WRITE(3,6)S,BA
6       FORMAT(1X,'SUPREMO DE ESSE=',E16.7,/,1X,'SUPREMO DE BETA=',E16.
        BUSNUM=BA+(1.D0+BA)*CA+(1.D0+BA)*(1.D0+CA)*S*RO
        RETURN
20      FORMAT(1X,'SUPREMO DE NRO COND DO JACOBIANO=',E16.7,3X,'IER='I4
        END

        FUNCTION F1(X)




        IMPLICIT REAL *8 (A-H,O-Z)
        COMMON/NFUN/NF
        COMMON/ERRO/RO,Z(20),N
        DIMENSION X(20),XLZ(20)
        DO 1 I=1,N
1       XLZ(I)=X(I)-Z(I)
        XLZN=ANORM2(XLZ,N)
        IF(XLZN.LE.RO)GO TO 3
```

```
      F1=1.E30
      RETURN
3     CALL CXLPS(N,X,V)
      F1=-V
2     RETURN
      END


      FUNCTION F2(X)
      IMPLICIT REAL *8 (A-H,O-Z)
      COMMON/NFUN/NF
      COMMON/ERREG/R0,Z(20),N
      DIMENSION X(20),XLZ(20)
      DO 3 I=1,N
3     XLZ(I)=X(I)-Z(I)
      XLZN=ANORM2(XLZ,N)
      IF(XLZN.GE.R0)GO TO 4
      F2=1.E30
      RETURN
4     CALL ESSP(N,X,SXZ)
      F2=-SXZ
2     RETURN
      END

      FUNCTION F3(X)
      IMPLICIT REAL *8 (A-H,O-Z)
      COMMON/NFUN/NF
      COMMON/ERREG/R0,Z(20),N
      DIMENSION X(20),XLZ(20)
      DO 4 I=1,N
4     XLZ(I)=X(I)-Z(I)
      XLZN=ANORM2(XLZ,N)
      IF(XLZN.GE.R0)GO TO 5
      F3=1.E30
      RETURN
5     CALL BETA(N,X,B)
      F3=-B
2     RETURN
      END


      FUNCTION F4(X)
      IMPLICIT REAL *8 (A-H,O-Z)
      COMMON/NFUN/NF
      COMMON/ERREG/R0,Z(20),N
      DIMENSION X(20),XLZ(20)
      DO 5 I=1,N
```

```
5       XLZ(I)=X(I)-Z(I)
        XLZN=ANORM2(XLZ,N)
        IF(XLZN.LE.EO)GO TO 8
        F4=1.
        RETURN
8       CALL COND (N,X,F4)
        F4=-F4
2       RETURN
        END


        SUBROUTINE COND(N,X,CONDI)
        IMPLICIT REAL*8 (A-H,O-Z)
        COMMON/NFUN/NF
        DIMENSION A(20,20),B(20,20),X(20),C(210),AUT(20)
        CALL JACOBI(N,X,A)
        DO 1 I=1,N
        DO 1 J=1,N
        B(I,J)=0.D0
        DO 1 K=1,N
1       B(I,J)=B(I,J)+A(K,I)*A(K,J)
        K=1
        DO 2 J=1,N
        DO 2 I=1,J
        C(K)=B(I,J)
2       K=K+1
        CALL DEIGEN(C,B,N,1)
        K=1
        AUT(1)=C(1)
        DO 3 I=2,N
        K=K+I
3       AUT(I)=C(K)
        DO 4 I=1,N
4       AUT(I)=DABS(AUT(I))
        AMAIOR=0.D0
        AMENOR=1.D30
        DO 5 J=1,N
        AMAIOR=DMAX1(AMAIOR,AUT(J))
5       AMENOR=DMIN1(AMENOR,AUT(J))
        CONTINUE
        CO=AMAIOR/AMENOR
        CONDI=DSQRT(CO)
        RETURN
        END

        FUNCTION ANORM2(Z,N)
        IMPLICIT REAL*8(A-G,O-Z)
        COMMON/NFUN/NF
        DIMENSION Z(20)
        A=0.DO
        DO 1 I=1,N
1       A=A+Z(I)**2
        ANORM2=DSQRT(A)
```

```
      RETURN
      END


      SUBROUTINE SOMA (N,A,B)
      IMPLICIT REAL*8 (A-G,O-Z)




      COMMON/NFUN/NF
      DIMENSION X(20),F(20),FS(20),V(20)
      PRE=2.D0**(-27)
C     CALCULO DE DELTA (PAG 276)
      CALL FUN(N,X,F)
      FN=ANORM2(F,N)
      IF(FN.NE.0D0)GO TO 10
      DELTA=0.
      GO TO 11
1     DO 12 I=1,N
      H=F(I)
1     FS(I)=H
      DO 13 I=1,N
1     V(I)=FS(I)-F(I)
      VN=ANORM2(V,N)
      DELTA=VN/FN
C     CALCULO DE ALFA (PAG 278)
1:    CALL COND(N,A,C)
      C1=3.D0*C*PRE*20.D0*DEL0AT(C)**3
      ALFA=C1/(1.D0-C1)+3.D0*C*DELTA
C     CALCULO DE SOMA (PAG 278)
      B=(1.D0+PRE)*ALFA+PRE
      RETURN
      END


      SUBROUTINE CALPS(N,X,V)
C     DEFINICAO 3.7 PAG 277
      IMPLICIT REAL*8(A-G,O-Z)
      COMMON/NFUN/NF
      DIMENSION A(20,20),SA(20,20),X(20),F(20),DXE(20),DXR(20),DIF(20
      CALL JACOBI (N,A,B)
      DO 4 I=1,N
      DO 4 J=1,N
      H=A(I,J)
4     SA(I,J)=H
      CALL FUN(N,X,F)
      CALL SISTE(N,A,F,DXE)
      CALL SISTE(N,SA,F,DXR)
      DO 9 I=1,N
9     DIF(I)=DXE(I)-DXR(I)
```

```
      DIFN=ANORM2(DIF,N)
      IF(DIFN.NE.0.D0)GO TO 6
      V=0.D0
      RETURN
6     DXEN=ANORM2(DXE,N)
      IF(DXEN.NE.0.D0)GO TO 8
      V=1.E33
      RETURN
8     V=DIFN/DXEN
      RETURN
      END



      SUBROUTINE ESSF(N,X,SXZ)
C     DEFINICAO 2.7 PAG 273
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/NFUN/NF
      COMMON /ERRO/RO,Z(20),EN



      DIMENSION X1(20)
      DIMENSION A(20,20),F(20),DX(20),X(20),X1LZ(20),XLZ(20)
      CALL JACOBI(N,X,A)
      CALL FUN(N,X,F)
      CALL SISTE(N,A,F,DX)
      DO 1 I=1,N
      X1(I)=X(I)-DX(I)
      X1LZ(I)=X1(I)-Z(I)
1     XLZ(I)=X(I)-Z(I)
      X1LZNR=ANORM2(X1LZ,N)
      XLZNR=ANORM2(XLZ,N)
      IF(XLZNR.NE.0.D0)GO TO 2
      SXZ=0.D0
      RETURN
2     SXZ=X1LZNR/XLZNR**2
      RETURN
      END



C
C     ..................................................................
C
C        SUBROUTINE EIGEN
C
C        PURPOSE
C           COMPUTE EIGENVALUES AND EIGENVECTORS OF A REAL SYMMETRIC
C           MATRIX
```

```
C
C       USAGE
C          CALL EIGEN(A,R,N,MV)
C
C       DESCRIPTION OF PARAMETERS
C          A - ORIGINAL MATRIX (SYMMETRIC), DESTROYED IN COMPUTATION.
C              RESULTANT EIGENVALUES ARE DEVELOPED IN DIAGONAL OF
C              MATRIX A IN DESCENDING ORDER.
C          R - RESULTANT MATRIX OF EIGENVECTORS (STORED COLUMNWISE,
C              IN SAME SEQUENCE AS EIGENVALUES)
C          N - ORDER OF MATRICES A AND R
C          MV- INPUT CODE
C                    0   COMPUTES EIGENVALUES AND EIGENVECTORS
C                    1   COMPUTE EIGENVALUES ONLY (R NEED NOT BE
C                        DIMENSIONED BUT MUST STILL APPEAR IN CALLING
C                        SEQUENCE)
C
C       REMARKS
C          ORIGINAL MATRIX A MUST BE REAL SYMMETRIC (STORAGE MODE=1)
C          MATRIX A CANNOT BE IN THE SAME LOCATION AS MATRIX R
C
C       SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C          NONE
C
C       METHOD
C          DIAGONALIZATION METHOD ORIGINATED BY JACOBI AND ADAPTED
C          BY VON NEUMANN FOR LARGE COMPUTERS AS FOUND IN 'MATHEMATICA
C          METHODS FOR DIGITAL COMPUTERS', EDITED BY A. RALSTON AND
C          H.S. WILF, JOHN WILEY AND SONS, NEW YORK, 1962, CHAPTER 7
C
C     ---------------------------------------------------------------
C
      SUBROUTINE DEIGEN(A,R,N,MV)
        IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1),R(1)
C
C     ---------------------------------------------------------------
C
C        IF A DOUBLE PRECISION VERSION OF THIS ROUTINE IS DESIRED, THE
C        C IN COLUMN 1 SHOULD BE REMOVED FROM THE DOUBLE PRECISION
C        STATEMENT WHICH FOLLOWS.
C
C     DOUBLE PRECISION A,R,ANORM,ANRMX,THR,X,Y,SINX,SINX2,COSX,
C    1                 COSX2,DIFF,RANGE
C
C        THE C MUST ALSO BE REMOVED FROM DOUBLE PRECISION STATEMENTS
C        APPEARING IN OTHER ROUTINES USED IN CONJUNCTION WITH THIS
C        ROUTINE.
C
C        THE DOUBLE PRECISION VERSION OF THIS SUBROUTINE MUST ALSO
C        CONTAIN DOUBLE PRECISION FORTRAN FUNCTIONS.  SQRTSTATEMENTSEI
C        40, 68, 75, AND 78 MUST BE CHANGED TO DSQRT.  ABS IN STATEMENT
```

```
C          62 MUST BE CHANGED TO DABS. THE CONSTANT IN STATEMENT 5 SHOULD
C          BE CHANGED TO 1.0D-17.
C
C          ...............................................................
C
C          GENERATE IDENTITY MATRIX
C
    5 RANGE=1.D-12
      IF(NV-1) 10,25,10
   10 IQ=-N
      DO 20 J=1,N
      IQ=IQ+N
      DO 20 I=1,N
      IJ=IQ+I
      R(IJ)=0.0
      IF(I-J) 20,15,20
   15 R(IJ)=1.0
   20 CONTINUE
C
C          COMPUTE INITIAL AND FINAL NORMS (ANORM AND ANORMX)
C
   35 ANORM=0.0
      DO 35 I=1,N
      DO 35 J=I,N
      IF(I-J) 30,35,30
   30 IA=I+(J*J-J)/2
      ANORM=ANORM+A(IA)*A(IA)
   35 CONTINUE
      IF(ANORM) 165,165,40
   40 ANORM=1.414*DSQRT(ANORM)
      ANRMX=ANORM*RANGE/DFLOAT(N)
C
C          INITIALIZE INDICATORS AND COMPUTE THRESHOLD, THR
C
      IND=0
      THR=ANORM
   45 THR=THR/DFLOAT(N)
   50 L=1
   55 M=L+1
C
C          COMPUTE SIN AND COS
C
   60 MQ=(M*M-M)/2
      LQ=(L*L-L)/2
      LM=L+MQ
   62 IF( DABS(A(LM))-THR) 130,65,65
   65 IND=1
      LL=L+LQ
      MM=M+MQ
      X=0.5*(A(LL)-A(MM))
   68 Y=-A(LM)/ DSQRT(A(LM)*A(LM)+X*X)
```

```
      IF(X) 70,75,75
   70 Y=-X
   75 SINX=Y/ DSQRT(2.0*(1.0+( DSQRT(1.0-Y*Y))))
      SINX2=SINX*SINX
   78 COSX= DSQRT(1.0-SINX2)
      COSX2=COSX*COSX
      SINCS =SINX*COSX
C
C         ROTATE L AND M COLUMNS
C
      ILQ=N*(L-1)
      IMQ=N*(M-1)
      DO 125 I=1,N
      IQ=(I*I-I)/2
      IF(I-L) 80,115,80
   80 IF(I-M) 85,115,90
   85 IM=I+MQ
      GO TO 95
   90 IM=M+IQ
   95 IF(I-L) 100,105,105
  100 IL=I+LQ
      GO TO 110
  105 IL=L+IQ
  110 X=A(IL)*COSX-A(IM)*SINX
      A(IM)=A(IL)*SINX+A(IM)*COSX
      A(IL)=X
  115 IF(MV-1) 120,125,120
  120 ILR=ILQ+I
      IMR=IMQ+I
      X=R(ILR)*COSX-R(IMR)*SINX
      R(IMR)=R(ILR)*SINX+R(IMR)*COSX
      R(ILR)=X
  125 CONTINUE
      X=2.0*A(LM)*SINCS
      Y=A(LL)*COSX2+A(MM)*SINX2-X
      X=A(LL)*SINX2+A(MM)*COSX2+X
      A(LM)=(A(LL)-A(MM))*SINCS+A(LM)*(COSX2-SINX2)
      A(LL)=Y
      A(MM)=X
C
C         TESTS FOR COMPLETION
C
C         TEST FOR M = LAST COLUMN
C
  130 IF(M-N) 135,140,135
  135 M=M+1
      GO TO 60
C
C         TEST FOR L = SECOND FROM LAST COLUMN
```

```
C
  160 IF(L-(N-1)) 165,150,165
  165 L=L+1
      GO TO 55
  170 IF(IND-1) 180,155,180
  155 IND=0
      GO TO 50
C
C         COMPARE THRESHOLD WITH FINAL NORM
C
  180 IF(IHK-ANORM) 185,185,45
C
C         SORT EIGENVALUES AND EIGENVECTORS
C
  185 IQ=-N
      DO 185 I=1,N
      IQ=IQ+N
      LL=I+(I*I-I)/2
      JQ=N*(I-2)
      DO 185 J=1,N
      JQ=JQ+N
      MM=J+(J*J-J)/2
      IF(A(LL)-A(MM)) 170,185,185
  170 X=A(LL)
      A(LL)=A(MM)
      A(MM)=X
      IF(NV-1) 175,185,175
  175 DO 180 K=1,N
      ILR=IQ+K
      IMR=JQ+K
      X=R(ILR)
      R(ILR)=R(IMR)
  180 R(IMR)=X
  185 CONTINUE
      RETURN
      END
```

SUBROUTINE SISTE, SING, SOLVE, NEWTON.

```
      SUBROUTINE SISTE(N,A,B,DX)
      DIMENSION A(10,10),UL(10,10),B(10),DX(10)
      CALL DECOMP(N,A,UL)
      CALL SOLVE (N,UL,B,DX)
      RETURN
      END

      SUBROUTINE DECOMP (NN,A,UL)
      DIMENSION A(10,10),UL(10,10),SCALES(10),IPS(10)
      COMMON IPS
      N=NN
      DO 5 I=1,N
      IPS(I)=I
      ROWNRM=0.0
      DO 2 J=1,N
      UL(I,J)=A(I,J)
      IF(ROWNRM-ABS(UL(I,J)))1,2,2
1     ROWNRM=ABS(UL(I,J))
2     CONTINUE
      IF(ROWNRM)3,4,3
3     SCALES(I)=1.0/ROWNRM
      GO TO 5
4     CALL SING (1)
      SCALES(I)=0.0
5     CONTINUE
      NM1=N-1
      DO 17 K=1,NM1
      BIG=0.0
      DO 11 I=K,N
      IP=IPS(I)
      SIZE=ABS(UL(IP,K))*SCALES(IP)
      IF(SIZE-BIG)11,11,10
10    BIG=SIZE
      IDXPIV=I
11    CONTINUE
      IF(BIG)13,12,13
12    CALL SING(2)
      GO TO 17
13    IF(IDXPIV-K)14,15,14
14    J=IPS(K)
      IPS(K)=IPS(IDXPIV)
      IPS(IDXPIV)=J
15    KP=IPS(K)
      PIVOT=UL(KP,K)
```

```
      KP1=K+1
      DO 16 I=KP1,N
      IP=IPS(I)
      EM=-UL(IP,K)/PIVOT
      UL(IP,K)=-EM
      DO 16 J=KP1,N
      UL(IP,J)=UL(IP,J)+EM*UL(KP,J)
1     CONTINUE
17    CONTINUE
      KP=IPS(N)
      IF(UL(KP,N))19,18,19
1     CALL SING(2)
1     RETURN
      END




      SUBROUTINE SING (IWAY)
1     FORMAT(' MATRIX WITH ZERO ROW IN DECOMPOSE')
1     FORMAT(' SINGULAR MATRIX IN DECOMPOSE ZERO DIVIDE IN SOLVE')
1     FORMAT(' NO CONVERGENCE IN IMPROV MATRIX IS NEARLY  SINGULA
     *R')
      NOUT=3
      GO TO (1,2,3),IWAY
1     WRITE(NOUT,11)
      GO TO 10
2     WRITE(NOUT,12)
      GO TO 10
3     WRITE(NOUT,13)
1     RETURN
      END


      SUBROUTINE SOLVE(NN,UL,B,X)
      DIMENSION UL(10,10),B(10),X(10),IPS(10)
      COMMON IPS
      N=NN
      NP1=N+1
      IP=IPS(1)
      X(1)=B(IP)
      DO 2 I=2,N
      IP=IPS(I)
      IM1=I-1
      SUM=0.0
      DO 1 J=1,IM1
1     SUM=SUM+UL(IP,J)*X(J)
2     X(I)=B(IP)-SUM
      IP=IPS(N)
      X(N)=X(N)/UL(IP,N)
      DO4 IBACK=2,N
      I=NP1-IBACK
```

```
      IP=IPS(I)
      IP1=I+1
      SUM=0.0
      DO3 J=IP1,N
3     SUM=SUM+UL(IP,I)*X(J)
4     X(I)=(X(I)-SUM)/UL(IP,I)
      RETURN
      END
```

```
      SUBROUTINE NEWTON(X,N)
      DIMENSION X(10),A(10,10),F(10),B(10),DX(10)
      KON=0
2     CALL FUN(N,X,F)
      CALL JACOBI(N,X,A)
      WRITE(3,6)KON
6     FORMAT(1X,'KON=',I5)
      WRITE(3,3)
3     FORMAT(1X,'A=',/)
      WRITE(3,4)((A(I,J),J=1,N),I=1,N)
4     FORMAT(1X,6E16.7)
      WRITE(3,5)
5     FORMAT(1X,'F=',/)
      WRITE(3,4)(F(I),I=1,N)
      Z=0.
      DO 7 I=1,N
7     Z=AMAX1(Z,ABS(F(I)))
      IF(Z.GT.1.E-4)GO TO 12
      WRITE(3,13)(X(I),I=1,N)
13    FORMAT(1X,'VALORES OTIMO',/,4E16.7,/)
      WRITE(3,14)KON
14    FORMAT(1X,'NUMERO DE ITERACOES =',I3)
      RETURN
12    IF(KON.GT.100)GO TO 10
      WRITE(3,11)
11    FORMAT(1X,'MAIS DE 100 ITERACOES')
      RETURN
10    KON=KON+1
      DO 8 I=1,N
8     B(I)=-F(I)
      CALL SISTE (N,A,B,DX)
      DO 9 I=1,N
9     X(I)=X(I)+DX(I)
      GO TO 2
      END
```

PROGRAMAS PRINCIPAIS E SUAS RESPECTIVAS SUBROUTINE FUN, JACOBI.

```
      DIMENSION X(10)
      N=2
4     TYPE 5
5     FORMAT(' DATE "1" SE VOUS CONTINUER BRINCADEIRA')
      ACCEPT 6,LTT
6     FORMAT(I10)
      IF(LTT.NE.1)STOP
      TYPE 10
1     FORMAT(' PONTO INICIAL',/' X(1)=',$)
      ACCEPT 20,X(1)
2     FORMAT(FR.0)
      TYPE 30
3     FORMAT(' X(2)=',$)
      ACCEPT 20,X(2)
      WRITE(3,1)
1     FORMAT(1X,'SISTEMA DE EQUACOES A SER RESOLVIDO')
      WRITE(3,2)
2     FORMAT(1X,'F1(X)=10*(X(2)-X(1)**2)',/,1X,'F2(X)=1-X(1)')
      WRITE(3,3)(X(I),I=1,N)
3     FORMAT(1X,'VALOR INICIAL',/,4E16.7)
      CALL NEWTON (X,N)
      GOTO4
      END
```

```
      SUBROUTINE JACOBI (N,X,A)
      DIMENSION X(10),A(10,10)
      A(1,1)=-20*X(1)
      A(1,2)=10.00
      A(2,1)=-1.00
      A(2,2)=0.00
      RETURN
      END
```

```
      SUBROUTINE FUN(N,X,F)
      DIMENSION X(2),F(2)
      F(1)=10.0*(X(2)-X(1)**2)
      F(2)=1.00-X(1)
      RETURN
      END
```

```
      DIMENSION X(10)
      N=2
4     TYPE 5
5     FORMAT(' DATA "1" SE DESEJA CONTINUAR BRINCANDO')
      ACCEPT 6,LTS
6     FORMAT(I1)
      IF(LTS.EQ.1)STOP
      TYPE 10
1     FORMAT(' VALOR INICIAL',/' X(1)=',$)
      ACCEPT 20,X(1)
2     FORMAT(G)
      TYPE 30
3     FORMAT(' X(2)=',$)
      ACCEPT 20,X(2)
      WRITE(3,1)
1     FORMAT(1X,'SISTEMA DE EQUACOES A SER RESOLVIDO')
      WRITE(3,2)
2     FORMAT(1X'X1**2-2X2+1',/,1X,'X1+2X2**2-3')
      WRITE(3,3)(X(I),I=1,N)
3     FORMAT(1X,'VALOR INICIAL',/,4E16.7)
      CALL NEWTON (X,N)
      GOTO4
      END



      SUBROUTINE JACOBI (I,X,A)
      DIMENSION X(10),A(10,10)
      A(1,1)=2.*X(1)
      A(1,2)=-2.
      A(2,1)=1.
      A(2,2)=4.*X(2)
      RETURN
      END



      SUBROUTINE FU(I,X,F)
      DIMENSION X(2),F(2)
      F(1)=X(1)**2-2.*X(2)+1.
      F(2)=X(1)+2.*X(2)**2-3.
      RETURN
      END
```

```
      DIMENSION X(10)
      N=2
4     TYPE 5
5     FORMAT(' DATA "1" SE QUER CONTINUAR NEWTON')
      ACCEPT 6,LTT
6     FORMAT(I10)
      IF(LTT.NE.1)STOP
      TYPE 10
10    FORMAT(' PONTO INICIAL',/' X(1)=',$)
      ACCEPT 20,X(1)
20    FORMAT(G)
      TYPE 30
30    FORMAT(' X(2)=',$)
      ACCEPT 20,X(2)
      WRITE(3,1)
1     FORMAT(1X,'SISTEMA DE EQUACOES A SER RESOLVIDO')
      WRITE(3,2)
2     FORMAT(1X,'X1**2-X2- 1',/,1X,'(X1-2)**2+(X2-0.5)**2-1')
      WRITE(3,3)(X(I),I=1,N)
3     FORMAT(1X,'VALOR INICIAL',/,4F16.7)
      CALL NEWTON (X,2)
      GOTO4
      END



      SUBROUTINE JACOBI (N,X,A)
      DIMENSION X(10),A(10,10)
      A(1,1)=2.*X(1)
      A(1,2)=-1.
      A(2,1)=2.*(X(1)-2.)
      A(2,2)=2.*(X(2)-0.5)
      RETURN
      END



      SUBROUTINE FUN(N,X,F)
      DIMENSION X(2),F(2)
      F(1)=X(1)**2-X(2)-1.
      F(2)=(X(1)-2.)**2+(X(2)-0.5)**2-1.
      RETURN
      END
```

```
      DIMENSION X(10)
      N=2
4     TYPE 5
5     FORMAT(' BATA "1" SE QUER CONTINUAR BRINCANDO')
      ACCEPT 6,LTT
6     FORMAT(I10)
      IF(LTT.NE.1)STOP
      TYPE 10
1     FORMAT(' PONTO INICIAL',/' X(1)=',$)
      ACCEPT 20,X(1)
2     FORMAT(G)
      TYPE 30
3     FORMAT(' X(2)=',$)
      ACCEPT 20,X(2)
      WRITE(3,1)
1     FORMAT(1X,'SISTEMA DE EQUACOES A SER RESOLVIDO')
      WRITE(3,2)
2     FORMAT(1X,'10000*X1*X2-1',/,1X,'EXP(-X1)+EXP(-X2)-1.0001')
      WRITE(3,3)(X(I),I=1,N)
3     FORMAT(1X,'VALOR INICIAL',/,4E16.7)
      CALL NEWTON (X,N)
      GOTO4
      END



      SUBROUTINE JACOBI (N,X,A)
      DIMENSION X(10),A(10,10)
      A(1,1)=10000.*X(2)
      A(1,2)=10000.*X(1)
      A(2,1)=-EXP(-X(1))
      A(2,2)=-EXP(-X(2))
      RETURN
      END



      SUBROUTINE FUN(N,X,F)
      DIMENSION X(2),F(2)
      F(1)=10000.*X(1)*X(2)-1.
      F(2)=EXP(-X(1))+EXP(-X(2))-1.0001
      RETURN
      END
```

```
      DIMENSION X(10)
      N=2
4     TYPE 5
5     FORMAT(' DATE "1" SE QUER CONTINUAR BRI-CANDO')
      ACCEPT 6,LT1
6     FORMAT(I10)
      IF(LT1.NE.1)STOP
      TYPE 10
1     FORMAT(' PONTO INICIAL',/' X(1)=',$)
      ACCEPT 20,X(1)
2     FORMAT(G)
      TYPE 30
3     FORMAT(' X(2)=',$)
      ACCEPT 20,X(2)
      WRITE(3,1)
1     FORMAT(1X,'SISTEMA DE EQUACOES A SER RESOLVIDO')
      WRITE(3,2)
2     FORMAT(1X,'2(X1-1)-400X1(X2-X1**2)',/,1X,'200(X2-X1**2)')
      WRITE(3,3)(X(I),I=1,N)
3     FORMAT(1X,'VALOR INICIAL',/,4E16.7)
      CALL NEWTON (X,N)
      GOTO4
      END



      SUBROUTINE JACOBI (N,X,A)
      DIMENSION X(10),A(10,10)
      A(1,1)=2.-400.*X(2)+1200.*X(1)**2
      A(1,2)=-400.*X(1)
      A(2,1)=-400.*X(1)
      A(2,2)=200.
      RETURN
      END



      SUBROUTINE FUN(I,X,F)
      DIMENSION X(2),F(2)
      F(1)=2.*(X(1)-1.)-400.*X(1)*(X(2)-X(1)**2)
      F(2)=200.*(X(2)-X(1)**2)
      RETURN
      END
```

```
      DIMENSION X(10)
      N=2
4     TYPE 5
5     FORMAT(' DIGITE "1" SE DESEJA CONTINUAR COM O MESMO')
      ACCEPT 6,IP
6     FORMAT(I10)
      IF(IP.EQ.1)GO TO 10
      TYPE 10
1     FORMAT(' VALORES INICIAIS',/' X(1)=',$)
      ACCEPT 20,X(1)
2     FORMAT(F)
      TYPE 30
3     FORMAT(' X(2)=',$)
      ACCEPT 20,X(2)
      WRITE(3,1)
1     FORMAT(1X,'SISTEMA DE EQUACOES A SER RESOLVIDO')
      WRITE(3,2)
2     FORMAT(1X,'X1-1',/,1X,'X1*X2-1')
      WRITE(3,3)(X(I),I=1,N)
3     FORMAT(1X,'VALORES INICIAIS',/,4E16.7)
      CALL NEWTON (X,N)
      GOTO4
      END



      SUBROUTINE JACOBI (X,A,N)
      DIMENSION X(10),A(10,10)
      A(1,1)=1
      A(1,2)=0
      A(2,1)=X(2)
      A(2,2)=X(1)
      RETURN
      END



      SUBROUTINE FU(I,X,F)
      DIMENSION X(2),F(2)
      F(1)=X(1)-1
      F(2)=X(1)*X(2)-1
      RETURN
      END
```

```
      DIMENSION X(10)
      N=2
4     TYPE 5
5     FORMAT(' DIGITE "1" SE VOCE QUER CONTINUAR BRANCANDO')
      ACCEPT 6,LTI
6     FORMAT(I10)
      IF(LTI.EQ.1)STOP
      TYPE 10
10    FORMAT(' PONTO INICIAL',/' X(1)=',$)
      ACCEPT 20,X(1)
20    FORMAT(G)
      TYPE 30
30    FORMAT(' X(2)=',$)
      ACCEPT 20,X(2)
      WRITE(3,1)
1     FORMAT(1X,'SISTEMA DE EQUACOES A SER RESOLVIDO')
      WRITE(3,2)
      WRITE(3,7)
2     FORMAT(1X,'-13+X(1)+((-X(2)+5)*X(2)-2)*X(2)',/)
7     FORMAT(1X,'-29+X(1)+((X(2)+1)*X(2)-14*X(2)')
      WRITE(3,3)(X(I),I=1,N)
3     FORMAT(1X,'VALOR INICIAL',/,4F16.7)
      CALL NEWTON (X,N)
      GOTO4
      END



      SUBROUTINE JACOBI (N,X,A)
      DIMENSION X(10),A(10,10)
      A(1,1)=1.
      A(1,2)=-3*X(2)**2+10.*X(2)-2.
      A(2,1)=1.
      A(2,2)=3.*X(2)**2+2.*X(2)-14.
      RETURN
      END



      SUBROUTINE FUN(N,X,F)
      DIMENSION X(2),F(2)
      F(1)=-13.+X(1)+((-X(2)+5.)*X(2)-2.)*X(2)
      F(2)=-29.+X(1)+((X(2)+1.)*X(2)-14)*X(2)
      RETURN
      END
```

```
      DIMENSION X(10)
      N=10
4     TYPE 5
5     FORMAT(' DATA "1" SE VOCE CONTINUAR BRINCANDO')
      ACCEPT 6,LTT
6     FORMAT(I10)
      IF(LTT.NE.1)STOP
      TYPE 10
1     FORMAT(1X,' PONTO INICIAL')
      DO 42 I=1,N
      TYPE 43,I
43    FORMAT(1X,'X(',I2,')=',$)
      ACCEPT 20,X(I)
2     FORMAT(G)
42    CONTINUE
      WRITE(3,3)(X(I),I=1,N)
3     FORMAT(1X,'VALOR INICIAL DO SISTEMA DE 10 VARIAVEIS',/,10E16.7)
      CALL NEWTON (X,N)
      GOTO4
      END
```

```
      SUBROUTINE JACOBI (N,X,A)
      DIMENSION A(10,10),X(10)
      A(1,1)=3.-0.2*X(1)
      DO 1 I=2,N
      DO 1 J=1,N
      IF(I.EQ.J)GO TO 2
      IF(J.EQ.I-1)GO TO 3
      A(I,J)=0.
      GO TO 1
2     A(I,J)=3.-0.2*X(I)
      GO TO 1
3     A(I,J)=-1.
1     CONTINUE
      A(1,2)=2.
      RETURN
      END
```

```
      SUBROUTINE FUN(N,X,F)
      DIMENSION X(20),F(20)
      F(1)=(3.-0.1*X(1))*X(1)+1.-2.*X(2)
      DO 1 I=2,N
      F(I)=(3.-0.1*X(I))*X(I)+1.-X(I-1)
1     CONTINUE
      RETURN
      END
```

```
      DIMENSION X(10)
      N=2
4     TYPE 5
5     FORMAT(' DATA "1" SE QUER CONTINUAR BRINCANDO')
      ACCEPT 6,LTT
6     FORMAT(I10)
      IF(LTT.NE.1)STOP
      TYPE 10
10    FORMAT(' PONTO INICIAL',/,1X,' X(1)=',$)
      ACCEPT 20,X(1)
20    FORMAT(G)
      TYPE 30
30    FORMAT(' X(2)=',$)
      ACCEPT 20,X(2)
      WRITE(3,1)
1     FORMAT(1X,'SISTEMA DE EQUACOES A SER RESOLVIDO')
      WRITE(3,2)
2     FORMAT(1X,'F1(X)=X1',/,'F2(X)=10.*X(1)/(X(1)+0.1:2.*X(2)**2)')
      WRITE(3,3)(X(I),I=1,N)
3     FORMAT(1X,'VALOR INICIAL',/,4E16.7)
      CALL NEWTON (X,N)
      GOTO4
      END



      SUBROUTINE JACOBI (N,X,A)
      DIMENSION X(10),A(10,10)
      A(1,1)=1.
      A(1,2)=0.
      A(2,1)=20.*X(2)**2+1./((X(1)+0.1)**2+4.*X(2)**2*(X(1)+0.1)+4.*X
     ***4)                                                          (3)
      A(2,2)=-40.*X(1)*X(2)/((X(1)+0.1)**2+4.*X(2)**2*(X(1)+0.1)+4.*X
     ***4)                                                          (4)
      RETURN
      END



      SUBROUTINE FUN(N,X,F)
      DIMENSION X(2),F(2)
      F(1)=X(1)
      F(2)=10*X(1)/(X(1)+0.1+2.*X(2)**2)
      RETURN
      END
```

```
      DIMENSION X(10)
      N=3
4     TYPE 5
5     FORMAT(' DATA "1" SE QUER CONTINUAR SENÃO CANCELE')
      ACCEPT 6,LTI
6     FORMAT(I10)
      IF(LTI.NE.1)STOP
      TYPE 10
10    FORMAT(' PONTO INICIAL',/' X(1)=',$)
      ACCEPT 20,X(1)
20    FORMAT(G)
      TYPE 30
30    FORMAT(' X(2)=',$)
      ACCEPT 20,X(2)
      TYPE 31
31    FORMAT(1X,'X(3)=',$)
      ACCEPT 20,X(3)
      WRITE(3,1)
1     FORMAT(1X,'SISTEMA DE EQUAÇÕES A SER RESOLVIDO')
      WRITE(3,2)
2     FORMAT(1X,'F1=3X1+X2+2X3-3',/,1X,'F2=-3X1+5X2**2+2X1X3-1',
     */,1X,'F3=25X2X1+20X3-17')
      WRITE(3,3)(X(I),I=1,N)
3     FORMAT(1X,'VALOR INICIAL',/,4G16.7)
      CALL NEWTON (X,N)
      GOTO4
      END



      SUBROUTINE JACOBI (N,X,A)
      DIMENSION X(10),A(10,10)
      A(1,1)=3.
      A(1,2)=1.
      A(1,3)=4.*X(3)
      A(2,1)=-3.+2*X(3)
      A(2,2)=10.*X(2)
      A(2,3)=2.*X(1)
      A(3,1)=25.*X(2)
      A(3,2)=25.*X(1)
      A(3,3)=20.
      RETURN
      END



      SUBROUTINE FUN(N,X,F)
      DIMENSION X(2),F(2)
      F(1)=3.*X(1)+X(2)+2.*X(3)**2-3.
      F(2)=-3.*X(1)+5.*X(2)**2+2.*X(1)*X(3)-1.
      F(3)=25.*X(2)*X(1)+20.*X(3)+17.
      RETURN
      END
```

```
      DIMENSION X(10)
      N=4
4     TYPE 5
5     FORMAT(' DATA "1" SE QUER CONTINUAR BRINCANDO')
      ACCEPT 6,LTT
6     FORMAT(I10)
      IF(LTT.NE.1)STOP
      TYPE 10
10    FORMAT(' PONTO INICIAL',/' X(1)=',$)
      ACCEPT 20,X(1)
20    FORMAT(G)
      TYPE 30
30    FORMAT(' X(2)=',$)
      ACCEPT 20,X(2)
      TYPE 31
31    FORMAT(1X,'X(3)=',$,3X,'X(4)=',$)
      ACCEPT 20,X(3),X(4)
      WRITE(3,1)
1     FORMAT(1X,'SISTEMA DE EQUACOES A SER RESOLVIDO')
      WRITE(3,2)
2     FORMAT(1X,'F1=2X1+3X2-X3X4',/,1X,'F2=X1-2X2+X3**2X4
     **3',/,1X,'F3=6X1**2+6X2-X3X2+6X4',/,1X,'F4=X1X2-X3X4+2X1')
      WRITE(3,3)(X(I),I=1,N)
3     FORMAT(1X,'VALOR INICIAL',/,4E16.7)
      CALL NEWTON (X,N)
      GOTO4
      END
```

```
      SUBROUTINE JACOBI (N,X,A)
      DIMENSION X(10),A(10,10)
      A(1,1)=2.
      A(1,2)=3.
      A(1,3)=-X(4)
      A(1,4)=-X(3)
      A(2,1)=1.
      A(2,2)=-2.
      A(2,3)=2.*X(3)*X(4)
      A(2,4)=X(3)**2
      A(3,1)=12.*X(1)
      A(3,2)=6.-X(3)
      A(3,3)=-X(2)
      A(3,4)=0.
      A(4,1)=X(2)+2.
      A(4,2)=X(1)
      A(4,3)=-X(4)
      A(4,4)=-X(3)
      RETURN
      END
```

```
      SUBROUTINE FUN(N,X,F)
      DIMENSION X(20),F(20)
      F(1)=2.*X(1)+3.*X(2)-X(3)*X(4)
      F(2)=X(1)-2.*X(2)+X(3)**2*X(4)+3.
      F(3)=8.*X(1)**2+6*X(2)-X(3)*X(2)+8.*X(4)




      F(4)=X(1)*X(2)-X(3)*X(4)+2*X(1)
      RETURN
      END




      DIMENSION X(10)
      N=6
4     TYPE 5
5     FORMAT(' DATA "1" SE QUER CONTINUAR BRINCANDO')
      ACCEPT 6,LTT
6     FORMAT(I10)
      IF(LTT.NE.1)STOP
      TYPE 10
1.    FORMAT(' PONTO INICIAL',/' X(1)=',$)
      ACCEPT 20,X(1)
2.    FORMAT(3)
      TYPE 30
3.    FORMAT(' X(2)=',$)
      ACCEPT 20,X(2)
      TYPE 31
31    FORMAT(1X,'X(3)=',$)
      ACCEPT 20,X(3)
      TYPE 32
3.    FORMAT(1X,'X(4)=',$)
      ACCEPT 20,X(4)
      TYPE 33
3.    FORMAT(1X,'X(5)=',$)
      ACCEPT 20,X(5)
      TYPE 34
3.    FORMAT(1X,'X(6)=',$)
      ACCEPT 20,X(6)
      WRITE(3,3)(X(I),I=1,6)
3     FORMAT(1X,'VALOR INICIAL DO SISTEMA DE 6 VARIAVEIS',/,6F10.7)
      CALL NEWTON (X,6)
      GOTO4
      END
```

```
      SUBROUTINE JACOBI (X,A,N)
      DIMENSION X(10),A(10,10),BETA(10),DCOT(10)
      BETA(1)=0.2244
      BETA(2)=0.0210
      BETA(3)=0.0208
      BETA(4)=0.0200
      BETA(5)=0.0191
      BETA(6)=0.0185
      DO 1 I=1,6
      DO 1 J=1,6
      Z=BETA(I)*X(J)
      IF(I.EQ.J)GO TO 2
      DCOT(L)=-1./(SIN(Z))**2
      A(I,J)=BETA(I)*DCOT(L)
      GO TO 1
2     A(I,J)=0.
1     CONTINUE
      RETURN
      END


      SUBROUTINE FUN(X,A,F)
      DIMENSION X(20),BETA(20),COT(20),F(20)



      BETA(1)=0.2244
      BETA(2)=0.0210
      BETA(3)=0.0208
      BETA(4)=0.0200
      BETA(5)=0.0191
      BETA(6)=0.0185
      DO 1 I=1,6
      F(I)=0.
      DO 1 J=1,6
      Z=BETA(I)*X(J)
      IF(I.EQ.J)GO TO 1
      COT(L)=COS(Z)/SIN(Z)
      F(I)=F(I)+COT(L)
1     CONTINUE
      RETURN
      END
```

REFERÊNCIAS

[1] RALL, L.B.: "Computational Solution of Nonlinear Operator Equations". New York, Wiley 1969.

[2] ORTEGA, J.M.; RHEINBOLDT, W.C.: "Iterative Solution of Nonlinear Equations in Several Variables". New York Academic Press, 1970.

[3] WILKINSON, J.H.: "The Algebraic Eigenvalue Problem". Oxford: Clarendon Press, 1965.

[4] A.K. CLINE; C.B. MOLER; G.W. STEWART and J.H. WILKINSON: "An Estimate for the Condition Number of A Matrix". Siam Journal on Numerical Analysis.Abril 1979, número 2, volume 16.

[5] J.M. MARTINEZ: "A Bound of the Moore-Penrose Pseudoinverse of A Matrix". Commentationes Mathematicae Universitatis Carolinae 20, 2, 1979.

[6] BUS, J.C.P.: "An Analysis of Convergence of Newton-like Methods for Solving Systems of Nonlinear Equations". Mathematisch Centrum, report N W20/75 Amsterdam 1975.

[7] FORSYTHE, G.; MOLER, C.B.: "Computer Solution of Linear Algebraic Systems". Prentice-Hall, Inc.. Englewood Cliffs, N.J. 1967.