Universidade Estadual de Campinas
Instituto de Matemática, Estatística e Computação Científica
Análise Numérica - Profa.: Sandra Augusta Santos
Alunos:
Beatriz Gomes da Silva, RA: 194631
Julia Machado Moretto, RA: 176953
Pedro Eduardo Faria Pietrafeza, RA: 185616

Projeto 2

Campinas
2020

Sumário

1 Introdução

O objetivo central desse projeto foi analisar três métodos distintos de resolução do problema de quadrados mínimos lineares. Os métodos são:

- Solução das equações normais via fatoração de Cholesky;
- Solução via fatoração QR condensada, obtida pelo processo de Gram-Schmidt modificado;
- Solução via fatoração QR completa usando transformações de Householder.

Para testá-los usamos o problema teste de aproximar a curva $f(x) = exp(sen(6x))$ no intervalo $[0, 1]$ por um polinômio de grau 10.

2 Desenvolvimento

Para estudar o problema, usamos o software GNU Octave.

O problema estudado foi o de roximar a curva $f(x) = exp(sen(6x))$ no intervalo $[0, 1]$ por um polinômio de grau 10.

Os dados de entrada foram os valores computados para $f(x)$ nos pontos igualmente espaçados $x_k = k * h$, com k de 0 a 20 e com $h = 0.05$.

As 20 equações lineares podem ter seus coeficientes obtidos com as primeiras 11 colunas de matriz de Vandermonde.

Portanto temos o sistema da forma $A * X = b$ com A e b calculados da forma:

```
%Matriz de vader
A = fliplr(vander([0, 1/20, 2/20, 3/20, 4/20, 5/20, 6/20, 7/20, 8/20, 9/20, 10/20, 11/20, 12/20, 13/20, 14/20, 15/20, 16/20, 17/20, 18/20, 19/20, 1]));
A = A(:,[1:11]);

%b
b = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]*0.05;
b = exp(sin(6*b));
b = b';
```

2.1 Equações normais via Fatoração de Cholesky

O sistema que temos pode ser escrito como:

$$A * x = b$$

As chamadas equações normais que usaremos para resolver o problema podem ser descritas pelo sistema matricial:

$$A^t * A * x = A^t * b$$

Portanto aplicaremos Cholesky à matriz $A^t * A$ e resolveremos o sistema da forma:

$$R^t * R * x = A^t * b$$
$$R^t * y = A^t * b$$
$$y = R$$

Após rodarmos o algoritmo explicitado no apêndice, obtivemos o seguinte vetor para os coeficientes:

```
>> cholesky

COEF =

   9.9925e-01
   7.5085e+00
  -3.3916e+01
   6.2285e+02
  -3.5278e+03
   7.5176e+03
  -4.1568e+03
  -8.9034e+03
   1.6993e+04
  -1.1235e+04
   2.7153e+03
```

Os valores de $f(x)$ para os coeficientes do polinômio obtido para cada um dos X_k foram:

```
ans =

   0.99925
   1.34797
   1.75130
   2.19083
   2.54630
   2.71040
   2.64134
   2.36863
   1.97015
   1.53846
   1.15024
   0.84836
   0.63960
   0.50605
   0.42319
   0.37592
   0.36421
```

```
    0.39509
    0.46758
    0.57315
    0.75686
```

Além disso, calculamos o número de condição em relação à norma 2 de R:

$$cond(R, 2) = 2.3161e + 07$$

2.2 Fatoração QR condensada, obtida pelo processo de Gram-Schmidt modificado

Supondo o sistema $A * x = b$, com a fatoração $A = Q * R$ temos que é necessário resolver o sistema $R * x = Q^t * b$. Após resolver a fatoração e o sistema com o código do apêndice obtivemos os seguintes valores para os coeficientes:

```
>> GRAM

COEF =

   9.9926e-01
   7.5069e+00
  -3.3865e+01
   6.2222e+02
  -3.5238e+03
   7.5033e+03
  -4.1248e+03
  -8.9477e+03
   1.7031e+04
  -1.1252e+04
   2.7188e+03
```

Os valores de $f(x)$ para os coeficientes do polinômio obtido para cada um dos X_k foram:

```
ans =

   0.99926
   1.34797
   1.75131
   2.19084
   2.54630
   2.71039
   2.64135
   2.36863
   1.97016
   1.53845
   1.15023
```

```
    0.84835
    0.63961
    0.50605
    0.42319
    0.37592
    0.36421
    0.39509
    0.46759
    0.57314
    0.75686
```

Além disso, calculamos o número de condição em relação a norma 2 de Q e R:

$$cond(Q,2) = 1.0000$$
$$cond(R,2) = 2.3175e+$$

2.3 Fatoração QR completa usando transformações de Householder

Supondo o sistema é $A * x = b$, com a fatoração $A = Q * R$ temos que é preciso resolver o sistema $R * x = Q^t * b$. Após resolver a fatoração e o sistema com o código do apêndice obtivemos os seguintes valores para os coeficientes:

```
>> HOUSE

COEF =

   9.9926e-01
   7.5069e+00
  -3.3865e+01
   6.2222e+02
  -3.5238e+03
   7.5033e+03
  -4.1248e+03
  -8.9477e+03
   1.7031e+04
  -1.1252e+04
   2.7188e+03
```

Os valores de $f(x)$ para os coeficientes do polinômio obtido para cada um dos X_k foram:

```
ans =

    0.99926
    1.34797
    1.75131
```

```
2.19084
2.54630
2.71039
2.64135
2.36863
1.97016
1.53845
1.15023
0.84835
0.63961
0.50605
0.42319
0.37592
0.36421
0.39509
0.46759
0.57314
0.75686
```

Além disso, calculamos o número de condição em relação a norma 2 de Q e R:

$$cond(Q,2) = 1.0000$$
$$cond(R,2) = 2.3175e + 07$$

3 Conclusão

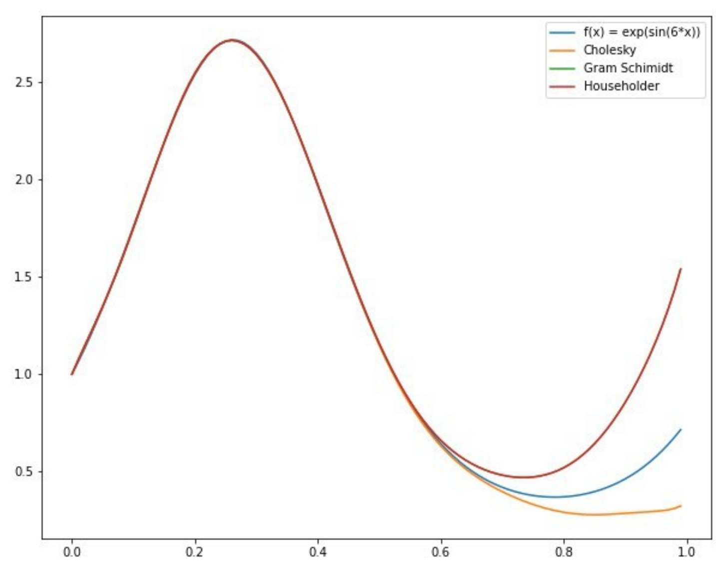
Na tabela abaixo temos os resultados obtidos com os coeficintes para cada método testado:

C_n	Cholesky	GramSchimidt	Householder
0	9.9925e-01	9.9926e-01	9.9926e-01
1	7.5085e+00	7.5069e+00	7.5069e+00
2	-3.3916e+01	-3.3865e+01	-3.3865e+01
3	6.2285e+02	6.2222e+02	6.2222e+02
4	-3.5278e+03	-3.5238e+03	-3.5238e+03
5	7.5176e+03	7.5033e+03	7.5033e+03
6	-4.1568e+03	-4.1248e+03	-4.1248e+03
7	-8.9034e+03	-8.9477e+03	-8.9477e+03
8	1.6993e+04	1.7031e+04	1.7031e+04
9	-1.1235e+04	-1.1252e+04	-1.1252e+044
10	2.7153e+03	2.7188e+03	2.7188e+03

Na tabela abaixo temos os valores obtidos para os polinômios que aproximam a função $f(x)$ e seu valor para os respectivos pontos do domínio:

X	f(x)	Cholesky	GramSchmidt	Householder
0.00000	1.00000	0.99925	0.99926	0.99926
0.05000	1.34383	1.34797	1.34797	1.34797
0.10000	1.75882	1.75130	1.75131	1.75131
0.15000	2.18874	2.19083	2.19084	2.19084
0.20000	2.53968	2.54630	2.54630	2.54630
0.25000	2.71148	2.71040	2.71039	2.71039
0.30000	2.64811	2.64134	2.64135	2.64135
0.35000	2.37076	2.36863	2.36863	2.36863
0.40000	1.96494	1.97015	1.97016	1.97016
0.45000	1.53323	1.53846	1.53845	1.53845
0.50000	1.15156	1.15024	1.15023	1.15023
0.55000	0.85407	0.84836	0.84835	0.84835
0.60000	0.64242	0.63960	0.63961	0.63961
0.65000	0.50270	0.50605	0.50605	0.50605
0.70000	0.41829	0.42319	0.42319	0.42319
0.75000	0.37624	0.37592	0.37592	0.37592
0.80000	0.36929	0.36421	0.36421	0.36421
0.85000	0.39621	0.39509	0.39509	0.39509
0.90000	0.46173	0.46758	0.46759	0.46759
0.95000	0.57655	0.57315	0.57314	0.57314
1.00000	0.75623	0.75686	0.75686	0.75686

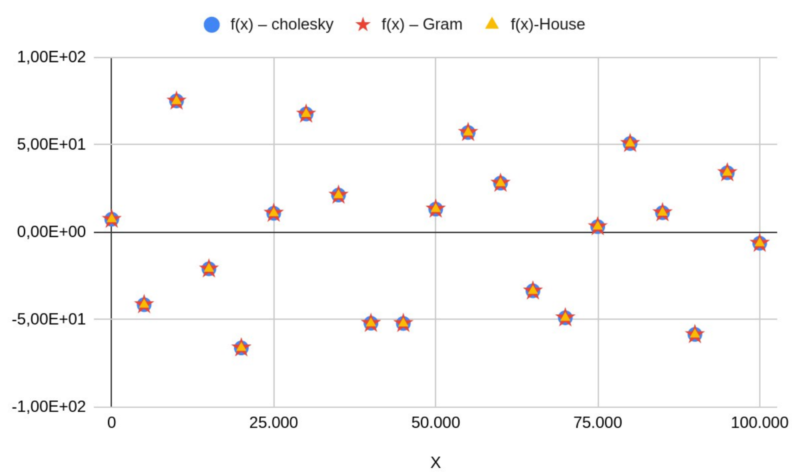
No gráfico abaixo temos a comparação das funções, GramSchimidt e Householder tiveram resultados muito parecidos para precisão de máquina, então estão sobrepostos no gráfico:



Na próxima tabela temos a diferença entre os valores obtidos com $f(x)$ e com os polinômios:

X	f(x)-cholesky	f(x)GS	f(x)-HH
0.00000	7.4514e-04	7.4369e-04	7.4369e-04
0.05000	-4.1487e-03	-4.1423e-03	-4.1423e-03
0.10000	7.5192e-03	7.5117e-03	7.5117e-03
0.15000	-2.0914e-03	-2.0936e-03	-2.0936e-03
0.20000	-6.6206e-03	-6.6153e-03	-6.6153e-03
0.25000	1.0843e-03	1.0887e-03	1.0887e-03
0.30000	6.7705e-03	6.7688e-03	6.7688e-03
0.35000	2.1291e-03	2.1238e-03	2.1238e-03
0.40000	-5.2094e-03	-5.2126e-03	-5.2126e-03
0.45000	-5.2207e-03	-5.2187e-03	-5.2187e-03
0.50000	1.3235e-03	1.3285e-03	1.3285e-03
0.55000	5.7109e-03	5.7139e-03	5.7139e-03
0.60000	2.8116e-03	2.8095e-03	2.8095e-03
0.65000	-3.3522e-03	-3.3572e-03	-3.3572e-03
0.70000	-4.8948e-03	-4.8972e-03	-4.8972e-03
0.75000	3.1980e-04	3.2306e-04	3.2306e-04
0.80000	5.0801e-03	5.0852e-03	5.0852e-03
0.85000	1.1184e-03	1.1174e-03	1.1174e-03
0.90000	-5.8500e-03	-5.8565e-03	-5.8565e-03
0.95000	3.4050e-03	3.4098e-03	3.4098e-03
1.00000	-6.2971e-04	-6.3075e-04	-6.3075e-04

Os valores obtidos pelos 3 métodos foram muito parecidos com o de $f(x)$, para precisão de máquina, tanto GramSchmidit quanto Householder tiveram o mesmo desempenho computacional. Cholesky se saiu levemente melhor, como pode ser percebido pela tabela acima e pelo gráfico abaixo



O número de condição da matriz $A^t * A$ foi calculado:

```
>> p = fliplr(vander([0, 1/20, 2/20, 3/20, 4/20, 5/20, 6/20, 7/20, 8/20, 9/20, 10/20, 11/20, 12/20, 13/20, 14/20, 15/20, 16/20, 17/20, 18/20, 19/20, 1]));
>> MATRIZ = p(:,[1:11]);
>> cond(MATRIZ'*MATRIZ, 2)
ans =    5.3786e+14
```

Pudemos perceber que os números de condição das matrizes Q obtidas por GramSchmidt e por Householder são 1, ou seja as matrizes Q são bem condicionadas. Nas outras matrizes o número de condição foi menor do que 1, o que teoricamente é impossível, os erros da precisão de máquina podem ser os responsáveis pelo ocorrido.

4 Algo a mais

Na máteria de F229 - Física Experimental 2, o métodos dos Mínimos Quadrados é muito utilizado para encontrar constantes físicas. Aqui analisaremos o Experimento 1, proposto no segundo semestre de 2017 nessa disciplina.

Usaremos os dados de um grupo real.

De acordo com o relatório desse grupo "O experimento aqui descrito consistiu em descobrir o raio de giração (k) de um pêndulo composto além do valor da gravidade(g) local. Para isso comparamos nossos resultados experimentais(massa das partes do pêndulo, período de oscilação, distância entre o eixo de rotação e o centro de massa) com $T = \frac{2\pi\sqrt{D+k2/D}}{g}$ que junta o teorema dos eixos paralelos(momento de inércia) com a equação do período do pendulo composto."

Propõe-se a seguinte linearização:

$$T = \frac{2\pi\sqrt{D+k2/D}}{g}$$
$$y = a * x + b$$
$$y = T^2 * D$$
$$x = D^2$$
$$a = \frac{4\pi^2}{g}$$
$$b = \frac{k^2}{g}$$

Aplicando o algoritmo de Cholesky conforme como o do apêndice aos nossos dados:

```
%D em metros e T em segundos
D =[ 0.896,0.896,0.896,0.796,0.796,0.796,0.696, ↩
    0.696,0.696,0.596,0.596,0.596,0.496,0.496,0.496,0.396, ↩
    0.396,0.396];
T = ↩
    [2.076,2.084,2.082,1.831,1.827,1.937,1.957,1.958,1.951,1.905,1.910,1.908,1.8

D = D';
T=T';
A=ones(18,2);
C = ones(18,1);
for i = 1:18
  A(i,1) = D(i)*D(i);
  C(i) = T(i)*T(i)*D(i);
end

%eq normais AtAx = Atb
ATA = A'*A;
ATB = A'* C;

%cholesky
n =2;
A = ATA;
%resto segue-se como no apendice
```

Obtivemos para a e b:

```
COEF =

   3.50585
   0.87908
```

Portanto, após aplicar:

```
g = 4*pi*pi/COEF(1)
k = sqrt(COEF(2)*g)
```

Obtivemos nosso g e nosso k no SI:

```
g =  11.261
k =  3.1463
```

Portanto conseguimos obter valores razoáveis. Escolhemos Cholesky por ser o com menor erro de acordo com a primeira parte do projeto.

5 Agradecimentos

Agradecemos Henrique Bazanella Pizzi, Luísa Pires Ferreira, Vitor Hugo Miranda Mourão por nos deixarem usar seus dados do experimento 1 do algo a mais.

Também agradecemos Maria Carolina Volpato por nos ajudar a entender física do experimento e Pedro Ono pela ajuda com os gráficos.

6 Referências

- Compare Gram-Schmidt and Householder Orthogonalization Algorithms - Cleve Moler `https://blogs.mathworks.com/cleve/2016/07/25/compare-gram-schmidt-and-householder-orthogonalization-algorithms`
- `http://www.mat.ufrgs.br/~guidi/grad/MAT01032/calculo_numerico.cap5.pdf`
- Fatoração Cholesky - Profa. Dra. Marli de Freitas Gomes Hernandez CESET-UNICAMP
- Matrix Algorithms, Volume I: Basic Decompositions - G. W. Stewart
- Householder Reflections and the QR Decomposition - Cleve Moler `https://blogs.mathworks.com/cleve/2016/10/03/householder-reflections-and-the-qr-decomposition/`
- `https://www.ufrgs.br/reamat/CalculoNumerico/livro-oct/sdsl-condicionamento_de_sistemas_lineares.html`

Acessos em 06/2020.

7 Apêndice

7.1 Algoritmo - Equações normais via Fatoração de Cholesky

```
%Algoritimo da fatora  o de Cholesky foi adaptado do
%Algoritimo da Profa. Dra. Marli de Freitas Gomes Hernandez
%CESET-UNICAMP

%Matriz de vader
p = fliplr(vander([0, 1/20, 2/20, 3/20, 4/20, 5/20, 6/20, 7/20, 8/20, 9/20, 10/20, 11/20, 12/20, 13/20, 14/20, 15/20, 16/20, 17/20, 18/20, 19/20, 1]));
MATRIZ = p(:,[1:11]);

%F(x)

b = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]*0.05;
```

```
b = exp(sin(6*b));
b = b';

%eq normais AtAx = Atb
ATA = MATRIZ'*MATRIZ;
ATB = MATRIZ'* b;


%cholesky

n = 11;

A = ATA;
% Determinar a Matriz R tal que A=( R  )R, A  s i m trica  positiva definida


% testar simetria da matriz A
for i=1:n
    for j=i+1:n
        if(A(i,j)~=A(j,i))
            disp('ERRO: matriz A  a s s i m trica ')
            c=input('digite CONTROL^C  '); % forma de parar o programa
        end
    end
end
% Determinar G tal que A=G'G
if A(1,1) < 0
    disp('ERRO: raiz quadrada de numero negativo g(i,i).')
    c=input('digite CONTROL^C  '); % forma de parar o programa
else
   if A(1,1) == 0
        disp('ERRO:  d i v i s o  por 0 no  c lculo  de g[i,j]/g(i,i).')
        c=input('digite CONTROL^C  '); % forma de parar o programa
   else
        g(1,1)=sqrt(A(1,1));
        for j=2:n
            g(1,j)= A(1,j)/g(1,1);
        end
        for i=2:n
            g(i,i)=A(i,i);
            for k= 1:i-1
                g(i,i) =g(i,i) - g(k,i)^2;
            end
            if g(i,i) < 0
```

```
                disp('ERRO: raiz quadrada de numero negativo g(i,i).')
                c=input('digite CONTROL^C'); % forma de parar o programa
            else
                if g(i,i) == 0
                    disp('ERRO: divis o por 0 no c lculo de g[i,j]/g(i,i).')
                    c=input('digite CONTROL^C'); % forma de parar o programa
                else
                g(i,i)=sqrt(g(i,i));
                for j=i+1:n
                   g(i,j)=A(i,j);
                   for k=1:i-1
                       g(i,j)=g(i,j)-g(k,i)*g(k,j);
                   end
                   g(i,j)=g(i,j)/g(i,i);
                 end
               end
            end
        end
    end
end
% Outra forma de programar o Cholesky
% Determinar R tal que A=R'R
for i=1:n
  for j=i:n
       r(i,j) = A(i,j);
       if i == j
           if i > 1
               for k=1:i-1
                   r(i,j) = r(i,j) - r(k,i)^2;
               end
           end
           if r(i,i) < 0
                disp('ERRO: raiz quadrada de numero negativo g(i,i).')
                c=input('digite CONTROL^C'); % forma de parar o programa
           else
              if r(i,i) == 0
                  disp('ERRO: divis o por 0 no c lculo de g[i,j]/g(i,i).')
                  c=input('digite CONTROL^C'); % forma de parar o programa
              else
                  r(i,j)=sqrt(r(i,j));
```

```
                end
            end
        else
            if i > 1
                for k=1:i-1
                    r(i,j) = r(i,j) - r(k,i)*r(k,j);
                end
            end
            r(i,j) = r(i,j)/r(i,i);
        end
    end
end

%ATA = R'R
%R'Y=AtB
y = r'\ATB;
%RCOEF = Y
COEF = r\y
%obtem valores do polinomio aplicado nos x abaixo (z = valores nos polinomios)
X = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]*0.05;
for j = 1:21
  x = X(j);
DIAG  = zeros(11);
for i = 1:11
    DIAG(i,i) = x^(i-1);
end
z(j) = sum(COEF'*DIAG);
end
%calcula diferenca entre valor de f(x) e valor obtido por polinomio
D = b-z'
 %calcula numero de condicao
CR = cond(r,inf)
```

7.2 Algoritmo -Fatoração QR condensada, obtida pelo processo de Gram-Schmidt modificado

```
%Implementacao de GramSchimidt
%Algoritmo de fatoracao QR foi pego numa publicacao de
%  Cleve Moler, July 25, 2016
%Com referencias para G. W. (Pete) Stewart
%referencias para publicacao completa na bibliografia do projeto

%Matriz de vader
```

```
p = fliplr(vander([0, 1/20, 2/20, 3/20, 4/20, 5/20, 6/20, 7/20, 8/20, 9/20, 10/20, 11/20, 12/20, 13/20, 14/20, 15/20, 16/20, 17/20, 18/20, 19/20, 1]));
MATRIZ = p(:,[1:11]);

%F(x)

b = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]*0.05;
b = exp(sin(6*b));

%Implementacao da fatoracao QR de GRAMSCHIMIDT Modificado

function [Q,R] =  mgs(X)
    % Modified Gram-Schmidt.  [Q,R] = mgs(X);
    % G. W. Stewart, "Matrix Algorithms, Volume 1", SIAM, 1998.
    [n,p] = size(X);
    Q = zeros(n,p);
    R = zeros(p,p);
    for k = 1:p
        Q(:,k) = X(:,k);
        for i = 1:k-1
            R(i,k) = Q(:,i)'*Q(:,k);
            Q(:,k) = Q(:,k) - R(i,k)*Q(:,i);
        end
        R(k,k) = norm(Q(:,k))';
        Q(:,k) = Q(:,k)/R(k,k);
    end
end

[Q,R] = mgs(MATRIZ);
%Resolucao do problema de quadrados minimos com QR
COEF = R \ (Q'*b')

%obtem valores do polinomio aplicado nos x abaixo
X = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]*0.05;

for j = 1:21
  x = X(j);
DIAG  = zeros(11);
for i = 1:11
    DIAG(i,i) = x^(i-1);
end
z(j) = sum(COEF'*DIAG);
end
z'
%calcula diferenca entre valor de f(x) e valor obtido por polinomio
D = b'-z'
```

```
%calcula numeros de condicao
CQ = cond(Q,2)
CR = cond(R,2)
```

7.3 Algoritmo - Fatoração QR completa usando transformações de Householder

```

%Implementacao de Householder
%Algoritmo de fatoracao QR foi pego em duas publicacoes de
%  Cleve Moler,
%Com referencias para G. W. (Pete) Stewart
%referencias para publicacao completa na bibliografia do projeto

%Matriz de vader
p = fliplr(vander([0, 1/20, 2/20, 3/20, 4/20, 5/20, 6/20, 7/20, 8/20, 9/20, 10/20, 11/20, 12/20, 13/20, 14/20, 15/20, 16/20, 17/20, 18/20, 19/20, 1]));
MATRIZ = p(:,[1:11]);

%F(x)

b = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]*0.05;
b = exp(sin(6*b));

%Implementacao da fatoracao QR de HOUSEHOLDER
% This program does not actually compute the QR orthogonalization, but rather computes
%R and a matrix U containing vectors that generate the Householder reflectors
%whose product is Q.
function [u,nu] = housegen(x)
    % [u,nu] = housegen(x)
    % Generate Householder reflection.
    % G. W. Stewart, "Matrix Algorithms, Volume 1", SIAM, 1998.
    % [u,nu] = housegen(x).
    % H = I - uu' with Hx = -+ nu e_1
    %    returns nu = norm(x).
    u = x;
    nu = norm(x);
    if nu == 0
        u(1) = sqrt(2);
        return
    end
    u = x/nu;
```

```
    if u(1) >= 0
        u(1) = u(1) + 1;
        nu = -nu;
    else
        u(1) = u(1) - 1;
    end
    u = u/sqrt(abs(u(1)));
end

function [U,R] = hqrd(X)
    % Householder triangularization.  [U,R] = hqrd(X);
    % Generators of Householder reflections stored in U.
    % H_k = I - U(:,k)*U(:,k)'.
    % prod(H_m ... H_1)X = [R; 0]
    % where m = min(size(X))
    % G. W. Stewart, "Matrix Algorithms, Volume 1", SIAM, 1998.
    [n,p] = size(X);
    U = zeros(size(X));
    m = min(n,p);
    R = zeros(m,m);
    for k = 1:min(n,p)
        [U(k:n,k),R(k,k)] = housegen(X(k:n,k));
        v = U(k:n,k)'*X(k:n,k+1:p);
        X(k:n,k+1:p) = X(k:n,k+1:p) - U(k:n,k)*v;
        R(k,k+1:p) = X(k,k+1:p);
    end
end

[U,R] =hqrd(MATRIZ);

function Z = house_apply(U,X)
    % Apply Householder reflections.
    % Z = house_apply(U,X), with U from house_qr
    % computes Q*X without actually computing Q.
    H = @(u,x) x - u*(u'*x);
    Z = X;
    [~,n] = size(U);
    for j = n:-1:1
        Z = H(U(:,j),Z);
    end
end

function Z = house_apply_transpose(U,X)
    % Apply Householder transposed reflections.
    % Z = house_apply(U,X), with U from house_qr
    % computes Q'*X without actually computing Q'.
    H = @(u,x) x - u*(u'*x);
    Z = X;
```

```
    [~,n] = size(U);
    for j = 1:n
        Z = H(U(:,j),Z);
    end
end

I = eye(size(U));
   Q = house_apply(U,I)
%contrucao da Q
 % H_k = I - U(:,k)*U(:,k)'.
 %
%Resolucao do problema de quadrados minimos com QR
COEF = R \ (Q'*b')(1:11)

%obtem valores do polinomio aplicado nos x abaixo
X = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]*0.05;

for j = 1:21
  x = X(j);
DIAG  = zeros(11);
for i = 1:11
    DIAG(i,i) = x^(i-1);
end
z(j) = sum(COEF'*DIAG);
end
z'
%calcula diferenca entre valor de f(x) e valor obtido por polinomio
D = b'-z'
%calcula numeros de condicao
CQ = cond(Q,2)
CR = cond(R,2)
```