Fachbereich Mathematik
Prof. Dr. P. Spellucci
Wintersemester 2007/2008

TECHNISCHE
UNIVERSITÄT
DARMSTADT

12.11.2007

Numerik des Matrizeneigenwertproblems
Übung 4

Präsenzübung

Ü 10 Sei A eine beliebige $n \times n$ nichtzerfallende obere Hessenberg–Matrix. Zeigen Sie, daß man das charakteristische Polynom

$$p_0(\lambda) \;=\; \det(A - \lambda I)$$

und seine Ableitung $p_0'(\lambda)$ bis auf einen gemeinsamen Faktor

$$\frac{1}{(-1)^n \prod_{i=1}^{n-1} a_{i+1,i}}$$

aus folgender Rekursion berechnen kann, die man sowohl mit einem Zahlenwert für λ als auch symbolisch ausführen kann:

$$\begin{aligned}
p_n(\lambda) &= 1 \\
p_n'(\lambda) &= 0 \\
p_{n-i}(\lambda) &= \frac{1}{a_{n-i+1,n-i}}\Big(\lambda p_{n-i+1}(\lambda) - \sum_{k=n-i+1}^{n} a_{n-i+1,k} p_k(\lambda)\Big)\;, \quad i = 1, \ldots, n \\
p_{n-i}'(\lambda) &= \frac{1}{a_{n-i+1,n-i}}\Big(\lambda p_{n-i+1}'(\lambda) + p_{n-i+1}(\lambda) - \sum_{k=n-i+1}^{n} a_{n-i+1,k} p_k'(\lambda)\Big)\;, \quad i = 1, \ldots, n
\end{aligned}$$

wobei noch $a_{1,0} = 1$ gesetzt ist. Ist λ ein Eigenwert, dann ist $(p_1, \ldots, p_n)^T$ ein zugehöriger Eigenvektor. Wie könnte man höhere Ableitungen von p_0 berechnen?
Hinweis: Betrachten Sie das lineare Gleichungssystem

$$(A - \lambda I)\vec{p} \;=\; p_0 e_1 \quad \text{mit} \quad e_1 \;=\; (1, 0, \ldots, 0)^T\;.$$

Ü 11 Es sei A eine nichtzerfallende obere Hessenbergmatrix. Zeigen Sie: jeder Eigenvektor von A hat eine letzte Komponente ungleich null und jeder Linkseigenvektor hat eine erste Komponente ungleich null.

Ü 12 a) Bestimmen Sie eine orthonormale und symmetrische 2×2-Matrix

$$G = \begin{pmatrix} c_1 & c_2 \\ c_2 & c_3 \end{pmatrix}$$

so, daß

$$GA = \begin{pmatrix} c_1 & c_2 \\ c_2 & c_3 \end{pmatrix} \begin{pmatrix} a & * \\ b & * \end{pmatrix} = \begin{pmatrix} * & * \\ 0 & * \end{pmatrix}$$

für jede reelle 2×2-Matrix A (man nennt ein solches G eine Givens-Drehung).

b) Wie kann man das Resultat aus a) verwenden, um eine obere Hessenberg-Matrix H mit einer unitären Matrix G in eine obere Dreiecksmatrix R zu transformieren $GH = R$?
Hinweis: Betten Sie die 2×2-Matrizen aus a) in größere Matrizen ein, so daß diese noch unitär sind, und überlegen Sie sich, welches Element der Subdiagonalen damit eliminiert wird. Wie kann man die Wirkung dieser Rotationen geschickt kombinieren, um das Ziel zu erreichen ?

Hausübung

H 10 Das Resultat von Gruppenübung 10 kann man zum Entwurf eines Eigenwertlösers benutzen, wenn man ein global konvergentes Nullstellenverfahren für Polynome besitzt. Ist der grösste bzw. kleinste Eigenwert einfach und reell, dann konvergiert das Newtonverfahren mit Startwerten rechts vom grössten bzw links vom kleinsten Eigenwert monoton und global. Schreiben Sie ein `MATLAB`–Programm, das zu einem gegebenen Startwert das Newtonverfahren zur Bestimmung eines Eigenwerts und eines zugehörigen Eigenvektors einer nichtzerfallenden oberen Hessenbergmatrix benutzt. Die Form dieses Programms sei eine Funktion

```
function [lambda,x,count,lastp0,resnorm] = newton(A,initlambda,trace)
% Aufruf [lambda,x,count,lastp0,resnorm] = newton(A,initlambda,trace)
% initlambda ist der Startwert
% bei trace=1 werden alle Näherungswerte für lambda und p_0 protokolliert
% count ist die Anzahl der Newtonschritte
% lastp0 ist der letzte Wert des charakteristischen Polynoms
% bis auf einen Faktor ungleich null, siehe Ü10
% lambda ist der berechnete Eigenwert, x der Eigenvektor und
% resnorm die 2-Norm von (A - λI)x
```

Die obere Hessenbergmatrix A mit

$$\begin{aligned} a_{i,i} &= 13 - i \;\; i = 1, \ldots, 12 \\ a_{j+1,j} &= 12 - j, \;\; j = 1, \ldots, 11, \\ a_{i,j} &= 13 - j, \;\; j = 2, \ldots, 12, \;\; i = 1, \ldots, j-1 \end{aligned}$$

hat als grössten bzw. kleinsten Eigenwert 32.22.. bzw. 0.033.... Erproben Sie Ihr Programm nach einem Test an diesem Beispiel.

H 11 Zeigen Sie: Die Matrix

$$C = \begin{pmatrix} 0,\ldots,0 & -c_0 \\ I & \begin{pmatrix} -c_1 \\ \cdot \\ \cdot \\ -c_{n-1} \end{pmatrix} \end{pmatrix}$$

hat das charakteristische Polynom

$$\det(zI - C) = c_0 + c_1 z + c_2 z^2 + \ldots + c_{n-1} z^{n-1} + z^n$$

Sind alle Eigenwerte λ_i paarweise verschieden, dann gilt

$$VCV^{-1} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$$

mit

$$V = \begin{pmatrix} 1 & \lambda_1 & \ldots & \lambda_1^{n-1} \\ & \ldots & \ldots & \\ 1 & \lambda_n & \ldots & \lambda_n^{n-1} \end{pmatrix}$$

H 12 Schreiben Sie eine `MATLAB`–Funktion (M-file), die eine beliebige $n \times n$-Matrix durch Ähnlichkeitstransformation mit Householdermatrizen auf obere Hessenberggestalt transformiert.

```
function [H,U,beta] =  orthhes(A)
% Aufruf [H,U,beta] =  orthhes(A)
% H resultierende obere Hessenbergmatrix, als volle Matrix gespeichert
% U Matrix: speichert in Spalte i ab Zeile i+1 den Vektor u(:,i)
% beta(i) =  2/(u(:,i)'*u(:,i)
```

Testen Sie diese Funktion z.B. mit Matrizen aus der Matrix-Gallerie von Higham. Matrizen können Sie z.B. erhalten mit

A=gallery(.....) % Kleinschreibung

wobei die genauen Parameter von Matrixtyp abhängen. Information mit

help gallery
help/private/matname (% aus help gallery)

Sie können auch Matrizen aus `elmat` nehmen

help elmat

z.B. `hilb(n), hankel(c,r), rosser`. Bestimmen Sie alle Eigenwerte der Matrizen einmal mit `eig(A)` und dann mit `eig(H)` und protokollieren Sie die maximalen Abweichungen in den Eigenwerten.

Numerik des Matrizeneigenwertproblems
Übung 4, Lösungsvorschlag

Präsenzübung

Ü 10 Sei A eine beliebige $n \times n$ nichtzerfallende obere Hessenberg–Matrix. Zeigen Sie, daß man das charakteristische Polynom

$$p_0(\lambda) \;=\; \det(A - \lambda I)$$

und seine Ableitung $p_0'(\lambda)$ bis auf einen gemeinsamen Faktor

$$\frac{1}{(-1)^n \prod_{i=1}^{n-1} a_{i+1,i}}$$

aus folgender Rekursion berechnen kann, die man sowohl mit einem Zahlenwert für λ als auch symbolisch ausführen kann:

$$\begin{aligned}
p_n(\lambda) &= 1 \\
p_n'(\lambda) &= 0 \\
p_{n-i}(\lambda) &= \frac{1}{a_{n-i+1,n-i}} \Big(\lambda p_{n-i+1}(\lambda) - \sum_{k=n-i+1}^{n} a_{n-i+1,k} p_k(\lambda) \Big) \;, \quad i = 1, \ldots, n \\
p_{n-i}'(\lambda) &= \frac{1}{a_{n-i+1,n-i}} \Big(\lambda p_{n-i+1}'(\lambda) + p_{n-i+1}(\lambda) - \sum_{k=n-i+1}^{n} a_{n-i+1,k} p_k'(\lambda) \Big) \;, \quad i = 1, \ldots, n
\end{aligned}$$

wobei noch $a_{1,0} = 1$ gesetzt ist. Ist λ ein Eigenwert, dann ist $(p_1, \ldots, p_n)^T$ ein zugehöriger Eigenvektor. Wie könnte man höhere Ableitungen von p_0 berechnen?
Hinweis: Betrachten Sie das lineare Gleichungssystem

$$(A - \lambda I)\vec{p} \;=\; p_0 e_1 \quad \text{mit} \quad e_1 \;=\; (1, 0, \ldots, 0)^T \;.$$

Mit $p_0(\lambda) = 0$ erhält man aus obiger Darstellung unmittelbar, daß $\vec{p} = (p_1, \ldots, p_n)^T$ ein Eigenvektor ist. Die Rekursion für die p_i ist nichts anderes als die systematische Auflösung des Gleichungssystems nach den p_i nach Setzung von $p_n = 1$. p_0 erhält man dann aus der ersten Zeile. Die Anwendung der Cramerschen Regel für die $n-$te Komponente liefert aber gerade

$$1 \;=\; p_n \;=\; \frac{(-1)^n p_0 \prod_{i=1}^{n-1} a_{i+1,i}}{\det(A - \lambda I)}$$

Die Rekursion für die Ableitung erhält man aus der Differentiation der Rekursionsformel für die p_i. Entsprechend kann man höhere Ableitungen berechnen durch weitere Differentiationen dieser Rekursion.

Ü 11 Es sei A eine nichtzerfallende obere Hessenbergmatrix. Zeigen Sie: jeder Eigenvektor von A hat eine letzte Komponente ungleich null und jeder Linkseigenvektor hat eine erste Komponente ungleich null.

Wir betrachten die Gleichung

$$(A - \lambda I)x \;=\; 0\;, \qquad x \;\neq\; 0\;.$$

Die letzte Zeile lautet hier

$$a_{n,n-1}x_{n-1} + (a_{n,n} - \lambda)x_n \;=\; 0$$

und aus $x_n = 0$ folgt wegen $a_{n,n-1} \neq 0$, daß $x_{n-1} = 0$. Induktiv erhält man dann aus den Zeilen $n-1, \ldots, 1$, daß $x_i = 0 \;\forall\; i$, also einen Widerspruch. Analog geht man über Zeile 1 bis n von

$$y^H(A - \lambda I) \;=\; 0\;, \qquad y \;\neq\; 0$$

vor bei der Annahme $y_1 \;=\; 0$.

Ü 12 a) Bestimmen Sie eine orthonormale und symmetrische 2×2-Matrix

$$G = \begin{pmatrix} c_1 & c_2 \\ c_2 & c_3 \end{pmatrix}$$

so, daß

$$GA = \begin{pmatrix} c_1 & c_2 \\ c_2 & c_3 \end{pmatrix} \begin{pmatrix} a & * \\ b & * \end{pmatrix} = \begin{pmatrix} * & * \\ 0 & * \end{pmatrix}$$

für jede reelle 2×2-Matrix A (man nennt ein solches G eine Givens-Drehung).

b) Wie kann man das Resultat aus a) verwenden, um eine obere Hessenberg-Matrix H mit einer unitären Matrix G in eine obere Dreiecksmatrix R zu transformieren $GH = R$?
Hinweis: Betten Sie die 2×2-Matrizen aus a) in größere Matrizen ein, so daß diese noch unitär sind, und überlegen Sie sich, welches Element der Subdiagonalen damit eliminiert wird. Wie kann man die Wirkung dieser Rotationen geschickt kombinieren, um das Ziel zu erreichen ?

a) *O.B.d.A. nehmen wir an, daß $b \neq 0$, da sonst $G = I$ gesetzt werden kann.*
Aus der Gleichung

$$\begin{pmatrix} c_1 & c_2 \\ c_2 & c_3 \end{pmatrix} \begin{pmatrix} a & * \\ b & * \end{pmatrix} = \begin{pmatrix} * & * \\ 0 & * \end{pmatrix}$$

und den gewünschten Eigenschaften der Matrix G lassen sich dann folgende Gleichungen aufstellen

$$\begin{aligned}
ac_2 + bc_3 &= 0 && \textit{(Transformation auf Dreiecksgestalt),} && \textit{(1)} \\
c_1^2 + c_2^2 &= 1 && \textit{(Spaltenlänge von } G \textit{ ist 1),} && \textit{(2a)} \\
c_2^2 + c_3^2 &= 1 && \textit{(Spaltenlänge von } G \textit{ ist 1),} && \textit{(2b)} \\
c_1c_2 + c_2c_3 &= 0 && \textit{(Spalten von } G \textit{ sind orthogonal).} && \textit{(3)}
\end{aligned}$$

Subtraktion (2a)-(2b) ergibt

$$c_1^2 - c_3^2 = 0 \quad \Leftrightarrow \quad (c_1 + c_3)(c_1 - c_3) = 0 \quad (4),$$

ausklammern von c_2 in (3) liefert

$$c_2(c_1 + c_3) = 0 \quad (5)$$

und nach Auflösen von (1) nach c_3 haben wir

$$c_3 = -c_2 \frac{a}{b}. \quad (6)$$

Aus (4) folgt entweder

$$c_1 = -c_3 \quad oder \quad c_1 = c_3 \quad (7)$$

und aus (5)

$$c_1 = -c_3 \quad oder \quad c_2 = 0.$$

Für den Fall $c_2 = 0$ wäre wegen (6) und (7) aber auch $c_3 = c_1 = 0$, was nach Voraussetzung nicht sein kann. Also ist

$$c_1 = -c_3$$

und folglich

$$c_1 = c_2 \frac{a}{b}. \quad (8)$$

Setzt man nun (8) in (2a) ein, so kommt man zu der Gleichung

$$c_2^2 \frac{a^2}{b^2} + c_2^2 = 1.$$

Durch Auflösen nach c_2 ergibt sich

$$c_2 = \frac{\pm b}{\sqrt{a^2 + b^2}}$$

und somit

$$c_1 = \frac{\pm a}{\sqrt{a^2 + b^2}}.$$

Setzt man das Vorzeichen auf + dann ist

$$G = \frac{1}{\sqrt{a^2 + b^2}} \begin{pmatrix} a & b \\ b & -a \end{pmatrix}.$$

b) *Um eine obere $n \times n$-Hessenberg-Matrix H in eine obere Dreiecksmatrix R zu transformieren, kann man $n - 1$ solcher Givens-Drehungen nacheinander ausführen. Die Rotationsmatrizen haben dabei die Gestalt*

$$U_i = \left(\begin{array}{c|cc|c} I_i & 0 & 0 & 0 \\ \hline 0 & c_i & d_i & 0 \\ 0 & d_i & -c_i & 0 \\ \hline 0 & 0 & 0 & I_{n-i-1} \end{array} \right),$$

wobei I_k die $k \times k$-Einheitsmatrix ist und

$$c_i = \frac{a_i}{\sqrt{a_i^2 + b_i^2}} \quad bzw. \quad d_i = \frac{b_i}{\sqrt{a_i^2 + b_i^2}}.$$

b_i ist das Subdiagonalelement, daß mit G_i zu Null gemacht werden soll und a_i ist das Diagonalelement direkt über b_i. Die Auslöschung beginnt links oben und arbeitet sich nach rechts unten:

$$\begin{aligned} H_1 &= H \\ H_2 &= G_1 H_1 && \Rightarrow (H_2)_{2,1} = 0 \\ H_3 &= G_2 H_2 = G_2 G_1 H && \Rightarrow (H_3)_{3,2} = 0 \\ &\vdots \\ R = H_n &= G_{n-1} H_{n-1} = G_{n-1} \cdots G_1 H. \end{aligned}$$

Die Matrix G ist dann das Produkt von unitären Matrizen

$$G = G_{n-1} G_{n-2} \cdots G_1$$

und als solches selbst unitär.

Hausübung

H 10 Das Resultat von Gruppenübung 10 kann man zum Entwurf eines Eigenwertlösers benutzen, wenn man ein global konvergentes Nullstellenverfahren für Polynome besitzt. Ist der grösste bzw. kleinste Eigenwert einfach und reell, dann konvergiert das Newtonverfahren mit Startwerten rechts vom grössten bzw links vom kleinsten Eigenwert monoton und global. Schreiben Sie ein MATLAB–Programm, das zu einem gegebenen Startwert das Newtonverfahren zur Bestimmung eines Eigenwerts und eines zugehörigen Eigenvektors einer nichtzerfallenden oberen Hessenbergmatrix benutzt. Die Form dieses Programms sei eine Funktion

```
function [lambda,x,count,lastp0,resnorm] = newton(A,initlambda,trace)
% Aufruf [lambda,x,count,lastp0,resnorm] = newton(A,initlambda,trace)
% initlambda ist der Startwert
% bei trace=1 werden alle Näherungswerte für lambda und p_0 protokolliert
% count ist die Anzahl der Newtonschritte
% lastp0 ist der letzte Wert des charakteristischen Polynoms
% bis auf einen Faktor ungleich null, siehe Ü10
% lambda ist der berechnete Eigenwert, x der Eigenvektor und
% resnorm die 2-Norm von (A - λI)x
```

Die obere Hessenbergmatrix A mit

$$\begin{aligned} a_{i,i} &= 13 - i \;\; i = 1, \ldots, 12 \\ a_{j+1,j} &= 12 - j, \;\; j = 1, \ldots, 11, \\ a_{i,j} &= 13 - j, \;\; j = 2, \ldots, 12, \;\; i = 1, \ldots, j-1 \end{aligned}$$

hat als grössten bzw. kleinsten Eigenwert 32.22.. bzw. 0.033.... Erproben Sie Ihr Programm nach einem Test an diesem Beispiel.

```
function [p,dp]=hyman(A,lambda)
%  [p,dp]=hyman(A,lambda)
% berechnet die Rekursion fuer den Wert des
% charakteristischen Polynoms und seiner
% Ableitung einer oberen nichtzerfallenden
% Hessenbergmatrix an der Stelle lambda
siz=size(A);
if siz(1) ~= siz(2)
    error('hyman: matrix must be square');
end
n=siz(1);
% ckeck matrix
sum=0;
for i=1:n-2
```

```
    for j=i+2:n
        sum=sum+abs(A(j,i));
    end
end
if sum~=0
    error('hyman: matrix A is not upper Hessenberg');
end
for i=1:n-1
    if A(i+1,i)==0
        error('hyman: matrix A is reducible');
    end
end
n1=n+1;
p=zeros(n1,1);
dp=zeros(n1,1);
p(n1)=1;
dp(n1)=0;
for i=1:n
    if i==n
        fac=1;
    else
        fac=1/A(n-i+1,n-i);
    end
    p(n1-i)=fac*(lambda*p(n1+1-i)-A(n-i+1,n-i+1:n)*p(n1-i+1:n1));
    dp(n1-i)=fac*(p(n1+1-i)+lambda*dp(n1+1-i)-A(n-i+1,n-i+1:n)*dp(n1-i+1:n1));
end


function [lambda,x,count,lastp0,resnorm] = newton(A,initlambda,trace)
 % Aufruf [lambda,x,count,lastp0,resnorm] = newton(A,initlambda,trace)
 % initlambda ist der Startwert
 % bei trace=1 werden alle N\"aherungswerte f\"ur lambda und p_0
 % protokolliert
 % count ist die Anzahl der Newtonschritte
 % lastp0 ist der letzte Wert des charakteristischen Polynoms
 % bis auf einen Faktor ungleich null, siehe \"U10
 % lambda ist der berechnete Eigenwert, x der Eigenvektor und
 % resnorm die 2-Norm von $(A- \lambda I)x/||x||$ }
 siz=size(A);
 if siz(1) ~= siz(2)
     error('newton hyman: matrix must be square');
 end
 n=siz(1);
 n1=n+1;
```

```
count=0;
p=zeros(n1,1);
lambda=initlambda;
[p,dp]=hyman(A,lambda);
cont=1;
if trace==1
    disp(['iter          lambda                    charpol']);
end
while cont
    count=count+1;
    % compute correction damped newton method
    succ=0;
    sig=2;
    if trace==1
       disp([num2str(count,'%4d'),' ',num2str(lambda,'%23.16e'),' ',num2str(p(1),'
    end
    if abs(dp(1)) <=eps*abs(p(1))
        disp(['newton hyman : small derivative , stop ']);
        break;
    end
    while succ==0
        sig=sig/2;
        nextlambda=lambda-sig*p(1)/dp(1);
        if nextlambda==lambda
            break;
        end
        [nextp,nextdp]=hyman(A,nextlambda);
        if abs(nextp(1)) <= (1-0.01*sig)*abs(p(1))
            succ=1;
            oldlambda=lambda;
            lambda=nextlambda;
        end
    end
    if succ==0
        break;
    end
    if abs(oldlambda-lambda)<= eps*(abs(lambda)+eps)
        cont=0;
    end
    p=nextp;
    dp=nextdp;
end
lastp0=p(1);
```

```
 x=p(2:n1);
 resnorm=norm(A*x-lambda*x);


F=gallery('frank',12);
F

F =

    12    11    10     9     8     7     6     5     4     3     2     1
    11    11    10     9     8     7     6     5     4     3     2     1
     0    10    10     9     8     7     6     5     4     3     2     1
     0     0     9     9     8     7     6     5     4     3     2     1
     0     0     0     8     8     7     6     5     4     3     2     1
     0     0     0     0     7     7     6     5     4     3     2     1
     0     0     0     0     0     6     6     5     4     3     2     1
     0     0     0     0     0     0     5     5     4     3     2     1
     0     0     0     0     0     0     0     4     4     3     2     1
     0     0     0     0     0     0     0     0     3     3     2     1
     0     0     0     0     0     0     0     0     0     2     2     1
     0     0     0     0     0     0     0     0     0     0     1     1

[lambda,x,count,lastp0,resnorm]=newton(F,55.0,1);
iter          lambda                    charpol
1  5.5000000000000000e+01 3.0321032195898281e+12
2  5.1219658530571053e+01 1.0612995865252100e+12
3  4.7793851444841117e+01 3.7072984623962915e+11
4  4.4702198851112691e+01 1.2911387673128160e+11
5  4.1929426145657338e+01 4.4756473273233864e+10
6  3.9466795617779134e+01 1.5395392332810526e+10
7  3.7314715031601025e+01 5.2240556151502476e+09
8  3.5487323608622702e+01 1.7265836425581169e+09
9  3.4019581443181622e+01 5.3910820386403847e+08
10 3.2972297860542760e+01 1.4643503264535809e+08
11 3.2403853154146972e+01 2.7157091267568707e+07
12 3.2240984766344347e+01 1.7512855534502268e+06
13 3.2228953696668533e+01 8.9605649459362030e+03
14 3.2228891503226563e+01 2.3834657669067383e-01
15 3.2228891501572164e+01 8.3446502685546875e-07
lambda

lambda =
```

```
  32.22889150157216

x

x =

   1.0e+07 *

   2.32887381824582
   2.25661338018104
   1.39173022465814
   0.64836416674392
   0.23960247576164
   0.07122821413219
   0.01697733974582
   0.00319012573266
   0.00045727268193
   0.00004715073865
   0.00000312288915
   0.00000010000000

resnorm

resnorm =

     1.819299973202501e-07

lastp0

lastp0 =

    -1.192092895507812e-07

[lambda,x,count,lastp0,resnorm]=newton(F,-1.0,1);
iter          lambda                    charpol
 1 -1.0000000000000000e+00 6.0885516875100208e-02
 2 -8.3333333333333270e-01 2.0845687387980389e-02
 3 -6.8860091883540042e-01 7.1201767815374595e-03
 4 -5.6349025910149986e-01 2.4257287133184553e-03
 5 -4.5588070174554296e-01 8.2405846068712399e-04
 6 -3.6383175363055953e-01 2.7906332587878763e-04
 7 -2.8557219327378164e-01 9.4169436773191466e-05
 8 -2.1948987431209382e-01 3.1649714297470091e-05
```

```
 9 -1.6412214565029759e-01 1.0587907482338468e-05
10 -1.1814685479067583e-01 3.5226247772942791e-06
11 -8.0373957206035712e-02 1.1642070388611349e-06
12 -4.9737844518493508e-02 3.8155521833069103e-07
13 -2.5290665160303521e-02 1.2367654968073569e-07
14 -6.1972165462054821e-03 3.9468628607505948e-08
15  8.2674592281630592e-03 1.2294495266812311e-08
16  1.8716149768726938e-02 3.6676511748860213e-09
17  2.5645989749243359e-02 9.9635351780368363e-10
18  2.9478845985911734e-02 2.1130219788495014e-10
19  3.0848153996437696e-02 2.1784673546238328e-11
20  3.1025222323090448e-02 3.3845634640387880e-13
21  3.1028061336524929e-02 -8.2672866064724476e-17
22  3.1028061314842943e-02 -8.2344151533255807e-17


lambda =

   0.03102806131484

x

x =

  -0.00000000059765
   0.00000001866394
  -0.00000037097677
   0.00000557001080
  -0.00006633973141
   0.00063559406935
  -0.00488250606984
   0.02956829776454
  -0.13659636727844
   0.45393927832221
  -0.96897193868516
   1.00000000000000

resnorm

resnorm =

     3.012627013337273e-16
```

```
lastp0

lastp0 =

    -8.234415153002463e-17
```

H 11 Zeigen Sie: Die Matrix

$$C \;=\; \begin{pmatrix} 0,\ldots,0 & -c_0 \\ I & \begin{pmatrix} -c_1 \\ \cdot \\ \cdot \\ -c_{n-1} \end{pmatrix} \end{pmatrix}$$

hat das charakteristische Polynom

$$\det(zI - C) \;=\; c_0 + c_1 z + c_2 z^2 + \ldots + c_{n-1} z^{n-1} + z^n$$

Sind alle Eigenwerte λ_i paarweise verschieden, dann gilt

$$VCV^{-1} \;=\; \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$$

mit

$$V \;=\; \begin{pmatrix} 1 & \lambda_1 & \ldots & \lambda_1^{n-1} \\ & \ldots & \ldots & \\ 1 & \lambda_n & \ldots & \lambda_n^{n-1} \end{pmatrix}$$

Um die Gleichung für $\det(zI - C)$ *zu zeigen genügt es, nach der ersten Spalte zu entwickeln:*

$$det(zI - C) \;=\; z\det(zI - \tilde{C}) - c_0(-1)^{n-1}(-1)^{n-2} \;=\; z\det(zI - \tilde{C}) + c_0$$

wobei $\tilde{C}$ *eine Matrix der gleichen Struktur ist wie* C*, jedoch in der Dimension um 1 verkleinert, ohne* c_0*. Das Vorzeichen bei* c_0 *entsteht aus dem erstem Minus bei der Entwicklung der ersten Spalte, zweites Element, dem Faktor* $(-1)^{n-1}$ *bei der Entwicklung der Unterdeterminante mit der ersten Zeile* $(0, \ldots, -c_0)$ *der Länge* $n-1$ *und aus der Determinante der verbleibenden Untermatrix der Dimension* $n-2$*, die nun eine obere Dreiecksmatrix mit Diagonale* $-1, \ldots, -1$ *ist. Rekursive Anwendung dieser Beziehung liefert die Behauptung. Die Eigenvektorrelation beweisen wir durch*

$$VC \;=\; diag(\lambda_1, \ldots, \lambda_n)V.$$

Auf der rechten Seite stehen nun zeilenweise die Potenzen 1 bis n *der* λ_i*, da die Diagonalmatrix* V *zeilenweise mit* λ_j *multipliziert. Auf der linken Seite stehen nun in den*

Komponenten 1 bis $n-1$ die $\lambda-$Potenzen 1 bis $n-1$, da C nur auf der Subdiagonalen eine 1 hat in den Spalten 1 bis $n-1$. In der letzten Komponente steht

$$\sum_{i=0}^{n-1} \lambda_j^i(-c_j) \;=\; \lambda_j^n$$

da λ_j ja Nullstelle des Polynoms ist.

H 12 Schreiben Sie eine `MATLAB`–Funktion (M-file), die eine beliebige $n \times n$-Matrix durch Ähnlichkeitstransformation mit Householdermatrizen auf obere Hessenberggestalt transformiert.

```
function [H,U,beta] =  orthhes(A)
% Aufruf [H,U,beta] =  orthhes(A)
% H resultierende obere Hessenbergmatrix, als volle Matrix gespeichert
% U Matrix: speichert in Spalte i ab Zeile i+1 den Vektor u(:,i)
% beta(i) =  2/(u(:,i)'*u(:,i)
```

Testen Sie diese Funktion z.B. mit Matrizen aus der Matrix-Gallerie von Higham. Matrizen können Sie z.B. erhalten mit

A=gallery(.....) % Kleinschreibung

wobei die genauen Parameter von Matrixtyp abhängen. Information mit

help gallery
help/private/matname (% aus help gallery)

Sie können auch Matrizen aus `elmat` nehmen

help elmat

z.B. `hilb(n), hankel(c,r), rosser`. Bestimmen Sie alle Eigenwerte der Matrizen einmal mit `eig(A)` und dann mit `eig(H)` und protokollieren Sie die maximalen Abweichungen in den Eigenwerten.

```
function [H,U,beta]=orthhes(A)
%[H,U,beta]=orthhes(A)
% transforms a general quadratic matrix A into upper
% hessenberg form using a similarity transform
% by n-2 Householder reflectors represented by
% the columns U(i+1:n,i) of U , i=1,..,n-2
```

```
% and the vector beta, storing beta(i)=2/(U(i+1:n,i)'*U(i+1:n,i))
%
siz=size(A);
if siz(1) ~= siz(2)
    error('orthhes: matrix A must be square');
end
n=siz(1);
H=zeros(n,n);
U=zeros(n,n);
beta=zeros(n,1);
H=A;
v=zeros(n,1);
w=zeros(1,n);
x=zeros(n,1);
for i=1:n-2
    % compute information about Householder reflector
    v(i+1:n)=H(i+1:n,i);
    sigma=norm(v(i+1:n));
    if v(i+1)==0
        v(i+1)=sigma;
        ss=1;
    else
        ss=v(i+1)/abs(v(i+1));
        v(i+1)=ss*(abs(v(i+1))+sigma);
    end
    if sigma~= 0
      beta(i)=2/(v(i+1:n)'*v(i+1:n));
      U(i+1:n,i)=v(i+1:n);
      % multiply current H from the left
      H(i+1,i)=-ss*sigma;
      if i<= n-2
        H(i+2:n,i)=zeros(n-1-i,1);
      end
      w(i+1:n)=v(i+1:n)'*H(i+1:n,i+1:n);
      H(i+1:n,i+1:n)=H(i+1:n,i+1:n)-beta(i)*(v(i+1:n)*w(i+1:n));
      % multiply current H from the right
      x=H(:,i+1:n)*v(i+1:n);
      H(:,i+1:n)=H(:,i+1:n)-beta(i)*(x*v(i+1:n)');

    end
end


function errmax=orthhestest(A)
```

```
siz=size(A);
n=siz(1);
la1=zeros(n,1);
la2=zeros(n,1);
la1=eig(A);
[H,U,beta]=orthhes(A);
la2=eig(H);
error=0;
for i=1:n
    v=la1(i)*ones(n,1);
    error=max(error,min(abs(la2-v)));
end
disp('orthhestest: maximale relative abweichung in den eigenwerten von A und H');
error=error/norm(A);
disp(num2str(error,14));
errmax=error;
```

```
TEST von ORTHHES

A=rosser;
orthhestest(A);
orthhestest: maximale relative abweichung in den eigenwerten von A und H
2.7680300168318e-12

A=gallery('hanowa',20);
orthhestest(A);
orthhestest: maximale relative abweichung in den eigenwerten von A und H
4.4408920985006e-16


A=hilb(10);
orthhestest(A);
orthhestest: maximale relative abweichung in den eigenwerten von A und H
3.3715255991366e-13


c=[1;2;3;4;5;6;7;8;9;10];
r=[10,-9,8,-7,6,-5,4,-3,2,-1];
A=hankel(c,r);
orthhestest(A);
orthhestest: maximale relative abweichung in den eigenwerten von A und H
1.1580222681641e-15
```