

Algoritm, potensmetoden

Algoritm för att finna största reella egenvärde och tillhörande egenvektor till en reell matris.

givet en startvektor x_0

$i = 0$

repeat

$$y_{i+1} = A x_i$$

$$x_{i+1} = y_{i+1} / \|y_{i+1}\|_2$$

$$\tilde{\lambda}_{i+1} = x_{i+1}^T A x_{i+1}$$

$$i = i + 1$$

until konvergens

Genom att stuva om i snurran kan antalet matris-vektor-multiplikationer halveras

givet en startvektor x_0

$$y_1 = A x_0$$

$i = 1$

repeat

$$x_i = y_i / \|y_i\|_2$$

$$y_{i+1} = A x_i$$

$$\tilde{\lambda}_i = x_i^T y_{i+1}$$

$$i = i + 1$$

until konvergens

Linjär konvergens med konvergensthastighet $|\lambda_2/\lambda_1|$, där λ_1 är det till beloppet största egenvärdet, och λ_2 är det näst största.

Exempel: Potensmetoden

```
»V = randn(3);
»D = diag([2 -1 0.5]);
»A = V*D/V;
»x = ones(3,1);
»y = A*x;
»for k = 1:10
    x = y/norm(y);
    y = A*x;
    la = x'*y
end
    0.83733198519744
    2.87312856029338
    1.64415905951431
    2.19667621471412
    1.90651791043056
    2.04793155270074
    1.97633464522796
    2.01190742351812
    1.99406501908572
    2.00297216759123
»D = D + [0 3 0; -3 0 0 ; 0 0 0];
»A = V*D/V;
»eig(A)'
ans =
    0.5000 - 2.5981i    0.5000 + 2.5981i    0.5000
»x = ones(3,1);
»y = A*x;
»for k = 1:10
    x = y/norm(y);
    y = A*x;
    la = x'*y
end
    0.78207878609794
   -0.94367599040274
    1.38864486739718
   -1.22996860865631
    2.11317602011408
   -0.54512846034724
    2.25980867745654
    0.19684659171458
    0.26764927285825
    0.92675171234019
```

Algorithm: Inversiteration

Om vi byter ut A i potensmetoden mot A^{-1} finner vi istället inversen till det till beloppet minsta egenvärdet $1/\lambda_n$

Om vi byter ut A i potensmetoden mot $(A - \sigma I)^{-1}$ finner vi då inversen till det till beloppet minsta egenvärdet λ_0 av matrisen $A - \sigma I$, men då är $A - (\sigma + \lambda_0) I$ singulär, och λ_0 det till beloppet minsta tal som gör att den blir singulär. Alltså är $\sigma + \lambda_0$ det egenvärde till A som ligger närmast σ .

Eftersom A och $(A - \sigma I)^{-1}$ har samma egenvektorer kan vi bestämma $\sigma + \lambda_0$ direkt från framräknad egenvektor.

givet en startvektor x_0

LU-faktorisera $A - \sigma I = P L U$

$i = 0$

repeat

$$y_{i+1} = (A - \sigma I)^{-1} x_i \quad /* \text{lös system} */$$

$$x_{i+1} = y_{i+1} / \|y_{i+1}\|_2$$

$$\tilde{\lambda}_{i+1} = x_{i+1}^T A x_{i+1}$$

$$i = i + 1$$

until konvergens

Ett σ nära ett egenvärde ger snabb linjär konvergens.

Exempel: Inversiteration

```
»V = randn(3);
»D = diag([2 -1 0.5]);
»A = V*D/V;
»x = ones(3,1);
»[L,U] = lu(A-(-0.9)*eye(3));
»for k = 1:10
    y = U\ (L\x);
    x = y/norm(y);
    la = x'*A*x
end
la =
    -0.89831780060298
la =
    -1.00488224010459
la =
    -0.99974098803177
la =
    -1.00001540316103
la =
    -0.99999900648123
la =
    -1.00000006728530
la =
    -0.99999999532081
la =
    -1.00000000032985
la =
    -0.99999999997659
la =
    -1.00000000000167
```

Algoritm: Ortogonal iteration

Potensmetoden fann det till beloppet största egenvärdet och en tillhörande egenvektor. Vi utökar nu metoden till att hitta rätt på en ON-bas för ett **invariant underrum** som hör till de p största egenvärdena

givet $Z_0 \in \mathbb{R}^{n \times p}$ med ON-kolonner

$i = 0$

repeat

$$Y_{i+1} = A Z_i$$

QR-faktorisera $Y_{i+1} = Z_{i+1} R_{i+1}$
($\text{span}(Z_{i+1})$ approximerar ett
invariant underrum)

$i = i + 1$

until konvergens

Om egenvärdena är ordnade så att $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|$ så konvergerar algoritmen med linjär konvergensthastighet $|\lambda_{p+1}/\lambda_p|$.

Ortogonal iteration

För första kolonnen i Z_i är algoritmen identisk med potensmetoden. Andra kolonnen är beroende av den första, tredje av första och andra, och så vidare.

Exempelvis: vid iteration med tjugo kolonner så beter sig de fem första på samma sätt som om man itererade med fem kolonner. Det innebär att vi utan att blinka kan iterera med n kolonner och välja $Z_0 = I$. Konvergenstakten för kolonn k ges av $|\lambda_{k+1}/\lambda_k|$

Sats 4.8

Vi använder $p = n$ och $Z_0 = I$ i algoritmen för orthogonal iteration.

Om alla egenvärden har olika belopp, och alla delmatriser $S(1:j, 1:j)$ till egenvektormatrisen S har full rang, så konvergerar $A_i = Z_i^T A Z_i$ mot Shur-formen av A , dvs mot en övertriangulär matris med egenvärdena till A på diagonalen. Egenvärdena är ordnade på diagonalen i avtagande beloppsordning.

Exempel: Orthogonal iteration

```
»V = randn(4);
»D = diag([2 -1 -2.5 0.5]);
»A = V*D/V
A =
    3.4115    -2.3105    -1.9467    -3.0462
   -0.8502     0.4467     0.2103     1.2139
   -1.2146    -0.0369    -1.1866    -1.9019
    4.1922    -2.1263    -1.7968    -3.6716
»Z = eye(4);
»for k = 1:30
    Y = A*Z;
    [Z,R] = qr(Y);
end
»T = Z'*A*Z
T =
   -2.4993   -0.2790    1.7330    4.4337
   -0.0117    1.9993    2.1030    6.0770
   -0.0000    0.0000   -1.0000   -1.4863
    0.0000    0.0000   -0.0000    0.5000

»D = D + [0 3 0 0; -3 0 0 0 ; 0 0 0 0 ; 0 0 0 0];
»A = V*D/V
A =
    2.3439   -3.9400   -2.4228   -0.7738
    1.0039   -1.0983   -0.4980    0.3307
    7.7544   -3.9245   -3.3544   -8.6851
   -1.2149   -0.7696   -0.8362    1.1088
»eig(A)'
ans =
    0.5000 - 2.5981i    0.5000 + 2.5981i   -2.5000    0.5000
»Z = eye(4);
»for k = 1:150
    Y = A*Z;
    [Z,R] = qr(Y);
end
»T = Z'*A*Z
T =
    0.4869   -3.0563    2.0124    1.4392
    2.2089    0.5130   -4.5290   11.4463
    0.0002   -0.0002   -2.4999    4.1713
   -0.0000    0.0000    0.0000    0.5000
```

Algorithm: QR-iteration

Genom att iterera över A istället för över Z erhålls QR-iteration:

$$A_0 = A$$

$$i = 0$$

repeat

$$\text{QR-faktorisera } A_i = Q_i R_i$$

$$A_{i+1} = R_i Q_i$$

$$i = i + 1$$

until konvergens

A_i och A_{i+1} är similära ty $A_{i+1} = Q_i^T A_i Q_i$.

Lemma 4.3

Om A_i beräknas med QR-iteration, så är $A_i = Z_i^T A Z_i$ där Z_i beräknats genom ortogonal iteration med $Z_0 = I$.

Detta ger att A_i konvergerar mot Schurform om alla egenvärden har olika belopp.

Algoritm: QR-iteration med skift

$$A_0 = A$$

$$i = 0$$

repeat

Välj ett skift σ_i i närheten
av ett egenvärde

$$\text{QR-faktorisera } A_i - \sigma_i I = Q_i R_i$$

$$A_{i+1} = R_i Q_i + \sigma_i I$$

$$i = i + 1$$

until konvergens

Lemma 4.4

A_i och A_{i+1} är (ortogonalt) similära.

För reella egenvärden åstadkommer vi **kvadratisk konvergens** genom att välja $\sigma_i = A_i(n, n)$.

När vi har funnit ett egenvärde tar vi bort det från problemet och jobbar vidare med resten av matrisen.

Exempel: QR-iteration

```
»V = randn(4);
»D = diag([2 -1 -2.5 0.5]);
»A = V*D/V
A =
    3.4115    -2.3105    -1.9467    -3.0462
   -0.8502     0.4467     0.2103     1.2139
   -1.2146    -0.0369    -1.1866    -1.9019
    4.1922    -2.1263    -1.7968    -3.6716
»for k = 1:7
    sig = A(4,4); fprintf('%1.2e ',abs(sig-(-2.5)))
    [Q,R] = qr(A - sig*eye(4));
    A = R*Q + sig*eye(4);
end
1.17e+00 2.71e-01 4.82e-02 2.70e-03 5.07e-06 1.67e-11 0.00e+00
»A
A =
    2.0528    -2.3173     1.2981     5.6076
    0.0347     0.4500     0.1303    -4.2637
   -0.0019    -0.0243    -1.0028     3.0696
    0.0000     0.0000    -0.0000    -2.5000
»for k = 1:4
    sig = A(3,3); fprintf('%1.2e ',abs(sig-(-1)))
    [Q,R] = qr(A(1:3,1:3) - sig*eye(3));
    A(1:3,1:3) = R*Q + sig*eye(3);
    A(1:3,4) = Q'*A(1:3,4);
end
2.84e-03 5.32e-06 1.74e-11 3.66e-15
»A
A =
    2.0032    -2.3723     1.2617     5.5145
    0.0020     0.4968     0.1292    -4.4335
   -0.0000    -0.0000    -1.0000     2.9971
    0.0000     0.0000    -0.0000    -2.5000
```

Exempel: QR-iteration, forts

```
»for k = 1:4
    sig = A(2,2); fprintf('%1.2e ',abs(sig-(0.5)))
    [Q,R] = qr(A(1:2,1:2) - sig*eye(2));
    A(1:2,1:2) = R*Q + sig*eye(2);
    A(1:2,3:4) = Q'*A(1:2,3:4);
end
3.20e-03 6.82e-06 3.10e-11 2.66e-15
»A
A =
    2.0000    -2.3743     1.2618     5.5085
    0.0000     0.5000     0.1275    -4.4409
   -0.0000    -0.0000    -1.0000     2.9971
    0.0000     0.0000    -0.0000    -2.5000
```