Numerisk Lineær Algebra F2021
Notesæt 24

Andrew Swann

28. april 2021

Sidst ændret: 28. april 2021.
Versionskode: afe75b2.

Indhold

Indhold **1**

Figurer **1**

24 Potensmetoden og inverspotensmetoden for egenværdier **2**
24.1 Potensmetoden 2
24.2 Page rank 6
24.3 Rayleighs kvotient 11
24.4 Inverspotensmetoden 11

Python indeks **17**

Indeks **17**

Figurer

24.1 En samling websider med links 7

24 Potensmetoden og inverspotensmetoden for egenværdier

For generel $(n \times n)$-matricer findes der ingen algebraiske formel for rødder af det karakteristiske polynomium. Numeriske metoder for egenværdier og egenvektorer er derfor nødt til bruger iteration og kan kun beregne tilnærmelser. I dette kapitel kikker vi på to metoder der tillader beregning af en enkelt egenværdi og en tilhørende egenvektor. Den første kan kun finde den egenværdi der har størst numerisk værdi. Den anden metode finder en egenværdi i nærheden af et første gæt. Vi vil diskutere disse metoder anvendt på symmetriske matricer, men metoderne kan også bruges på andre matricer. Fordelen med symmetriske matricer er at vi har spektralsætningen, sætning 23.9, der garanterer at matricen faktisk har en reel egenværdi.

24.1 Potensmetoden

Lad $A \in \mathbb{R}^{n \times n}$ være en kvadratisk matrix. Potensmetoden er den følgende algoritme:

POTENSMETODEN(A)
1 Begynd med et vilkårligt $w^{(0)}$ af længde 1
2 **for** $k \in \{0, 1, 2, \ldots\}$:
3 $\quad v = Aw^{(k-1)}$
4 $\quad w^{(k)} = v/\|v\|_2$
5 $\quad \lambda^{(k)} = (w^{(k)})^T A w^{(k)}$

Under visse betingelser konvergerer $w^{(k)}$ til en egenvektor for A, og $\lambda^{(k)}$ konvergerer til dens egenværdi.

Eksempel 24.1. Lad os regne i python

```
import numpy as np

rng = np.random.default_rng()

a = np.array([[2., 1., 1.],
              [1., 3., 1.],
              [1., 1., 4.]])
```

```
w = rng.standard_normal((a.shape[0], 1))
n = 20
lambda_out = np.empty(n)

for i in range(n):
    v = a @ w
    w = v / np.linalg.norm(v)
    lambda_out[i] = w.T @ (a @ w)

np.set_printoptions(linewidth = 60)
print(lambda_out)
```

```
[2.54309228 2.79560212 3.51621862 4.48791707 5.01080294
 5.1662295  5.20346163 5.21189398 5.2137791  5.21419931
 5.21429292 5.21431377 5.21431841 5.21431945 5.21431968
 5.21431973 5.21431974 5.21431974 5.21431974 5.21431974]
```

Vi ser at værdierne $\lambda^{(k)}$ = `lambda_out[k]` ser ud til at stabilisere. Hvis vi beregne $Aw^{(k)}$ og sammenligne med $\lambda^{(k)}w^{(k)}$ ser vi at $w^{(k)}$ er tæt på at være en egenvektor for A:

```
print(a @ w)
print(lambda_out[-1] * w)
print(np.allclose(a @ w, lambda_out[-1] * w,
                  atol = np.finfo(float).eps))
```

```
[[-2.07066973]
 [-2.7148674 ]
 [-3.94093292]]
[[-2.0706674 ]
 [-2.71486002]
 [-3.94093922]]
True
```

△

Vi analyserer potensmetoden for A en symmetrisk matrix. Så er A diagonaliserbar via en ortogonal matrix, pga. sætning 23.9. Søjlerne af den ortogonale matrix giver en ortonormal basis $v_0, \ldots, v_{n-1}$ af egenvektorer for A. Skriv $\lambda_0, \ldots, \lambda_{n-1}$ for de tilhørende egenværdier, så $Av_i = \lambda_i v_i$ for hvert i. Da vi har en basis, kan vi skrive en vilkårlig $w \in \mathbb{R}^n$ som en lineær kombination af $v_0, \ldots, v_{n-1}$:

$$w = s_0 v_0 + s_1 v_1 + \cdots + s_{n-1} v_{n-1}.$$

Da $Av_i = \lambda_i v_i$, kan vi nu beregne Aw som

$$\begin{aligned} Aw &= s_0 Av_0 + s_1 Av_1 + \cdots + s_{n-1} Av_{n-1} \\ &= s_0 \lambda_0 v_0 + s_1 \lambda_1 v_1 + \cdots + s_{n-1} \lambda_{n-1} v_{n-1}. \end{aligned}$$

I potensmetoden er $w^{(k)}$ en enhedsvektor i retningen $A^k w$. Vi ser at

$$\begin{aligned} A^k w &= s_0 \lambda_0^k v_0 + s_1 \lambda_1^k v_1 + \cdots + s_{n-1} \lambda_{n-1}^k v_{n-1} \\ &= \lambda_0^k \left(s_0 v_0 + s_1 \left(\frac{\lambda_1}{\lambda_0} \right)^k v_1 + \cdots + s_{n-1} \left(\frac{\lambda_{n-1}}{\lambda_0} \right)^k v_{n-1} \right), \end{aligned} \tag{24.1}$$

hvis $\lambda_0 \neq 0$.

Antag nu at $|\lambda_0| > |\lambda_1| \geqslant \cdots \geqslant |\lambda_{n-1}| \geqslant 0$ og at $s_0 \neq 0$. For hvert $i > 0$ er $|\lambda_i / \lambda_0| < 1$, så faktoren $(\lambda_i / \lambda_0)^k$ konvergerer mod 0 når $k \to \infty$. Dette betyder at for stort k domineres $A^k w$ af ledet

$$\lambda_0^k s_0 v_0,$$

som er parallel med v_0. Da $w^{(k)}$ og v_0 er enhedsvektorer, har vi dermed at for alle store k ligger $w^{(k)}$ tæt på $\varepsilon_k v_0$, hvor $\varepsilon_k \in \{\pm 1\}$.

Proposition 24.2. *Antag at $A \in \mathbb{R}^{n \times n}$ symmetrisk, med egenværdier der opfylder $|\lambda_0| > |\lambda_1| \geqslant \cdots \geqslant |\lambda_{n-1}| \geqslant 0$. Lad v_0 være en enhedsvektor, som er en egenvektor for A med egenværdi λ_0, og lad $w \in \mathbb{R}^n$ opfylder $\langle w, v_0 \rangle \neq 0$. Lad $w^{(0)} = w$, og lad $w^{(k)}$, $\lambda^{(k)}$ være frembragt af potensmetoden.*

Så findes der $\varepsilon_k \in \{\pm 1\}$ og konstanter $K_0, K_1 \geqslant 0$ således at for alle k tilstrækkelig stor gælder

$$\| w^{(k)} - \varepsilon_k v_0 \|_2 \leqslant K_0 \left| \frac{\lambda_1}{\lambda_0} \right|^k \quad \text{og} \quad |\lambda^{(k)} - \lambda_0| \leqslant K_1 \left| \frac{\lambda_1}{\lambda_0} \right|^{2k}.$$

□

Dette siger at efter et vis trin, forbedres tilnærmelsen til λ_0 med en faktor der er højst $|\lambda_1/\lambda_0|^2$ for hver iteration af potensmetoden. Med andre ord, har vi kvadratisk konvergens. Vi skriver

$$|\lambda^{(k)} - \lambda_0| \leqslant O\left(\left|\frac{\lambda_1}{\lambda_0}\right|^{2k}\right) \quad \text{for } k \to \infty.$$

Notationen $O(\,\cdot\,)$ kaldes *stor-»O«-notationen*. Det følger at potensmetoden virker godt hvis $|\lambda_1|$ er væsentlig mindre end $|\lambda_0|$.

Bevis. Vores antagelser sikrer at $\lambda_0^k s_0$, for $s_0 = \langle w, v_0\rangle$, er forskellig fra 0. Lad ε_k være fortegnet af $\lambda_0^k s_0$. Vi har $w^{(k)} = A^k w / \|A^k w\|_2$. Men fra (24.1) har vi

$$\langle w^{(k)}, \varepsilon_k v_0\rangle = \frac{\langle A^k w, \varepsilon_k v_0\rangle}{\|A^k w\|_2} = \frac{\langle \lambda_0^k s_0 v_0, \varepsilon_k v_0\rangle}{\|A^k w\|_2} = \frac{\varepsilon_k \lambda_0^k s_0}{\|A^k w\|_2} > 0,$$

da $\langle v_i, v_0\rangle = 0$, for $i > 0$. Ved at bruge $\langle w^{(k)}, \varepsilon_k v_0\rangle \geqslant 0$, følger det at

$$\begin{aligned}\|w^{(k)} - \varepsilon_k v_0\|_2^2 &= \|w^{(k)}\|_2^2 + \|v_0\|_2^2 - 2\langle w^{(k)}, \varepsilon_k v_0\rangle = 2 - 2\langle w^{(k)}, \varepsilon_k v_0\rangle \\ &\leqslant \big(2 - 2\langle w^{(k)}, \varepsilon_k v_0\rangle\big)\big(1 + \langle w^{(k)}, \varepsilon_k v_0\rangle\big) = 2\big(1 - \langle w^{(k)}, \varepsilon_k v_0\rangle^2\big).\end{aligned}$$

Men

$$1 - \langle w^{(k)}, \varepsilon_k v_0\rangle^2 = 1 - \frac{\langle A^k w, \varepsilon_k v_0\rangle^2}{\|A^k w\|_2^2} = \frac{\|A^k w\|_2^2 - \langle A^k w, \varepsilon_k v_0\rangle^2}{\|A^k w\|_2^2}$$

Da v_i er ortonormal, er

$$\|A^k w\|_2^2 = s_0^2\lambda_0^{2k} + s_1^2\lambda_1^{2k} + \cdots + s_{n-1}^2\lambda_{n-1}^{2k}$$

og $\langle A^k w, \varepsilon_k v_0\rangle^2 = s_0^2\lambda_0^{2k}$, som er netop den første led i $\|A^k w\|_2^2$. Det følger at

$$\begin{aligned}1 - \langle w^{(k)}, \varepsilon_k v_0\rangle^2 &= \frac{s_1^2\lambda_1^{2k} + \cdots + s_{n-1}^2\lambda_{n-1}^{2k}}{s_0^2\lambda_0^{2k} + s_1^2\lambda_1^{2k} + \cdots + s_{n-1}^2\lambda_{n-1}^{2k}} \\ &\leqslant \frac{s_1^2\lambda_1^{2k} + \cdots + s_{n-1}^2\lambda_{n-1}^{2k}}{s_0^2\lambda_0^{2k}} \\ &\leqslant \frac{(s_1^2 + \cdots + s_{n-1}^2)\lambda_1^{2k}}{s_0^2\lambda_0^{2k}} = \frac{1 - s_0^2}{s_0^2}\left(\frac{\lambda_1}{\lambda_0}\right)^{2k},\end{aligned}$$

og den første ulighed i proposition holder med

$$K_0 = \frac{\sqrt{2(1-s_0^2)}}{|s_0|}.$$

Tilsvarende har man

$$\begin{aligned}
\lambda^{(k)} - \lambda_0 = (w^{(k)})^T A w^{(k)} - \lambda_0 &= \frac{(A^k w)^T A^{k+1} w}{\|A^k w\|_2^2} - \lambda_0 \\
&= \frac{w^T A^{2k+1} w}{w^T A^{2k} w} - \lambda_0 \\
&= \frac{s_0^2 \lambda_0^{2k+1} + s_1^2 \lambda_1^{2k+1} + \cdots + s_{n-1}^2 \lambda_{n-1}^{2k+1}}{s_0^2 \lambda_0^{2k} + s_1^2 \lambda_1^{2k} + \cdots + s_{n-1}^2 \lambda_{n-1}^{2k}} - \lambda_0 \\
&= \frac{s_1^2 \lambda_1^{2k} (\lambda_1 - \lambda_0) + \cdots + s_{n-1}^2 \lambda_{n-1}^{2k} (\lambda_{n-1} - \lambda_0)}{s_0^2 \lambda_0^{2k} + s_1^2 \lambda_1^{2k} + \cdots + s_{n-1}^2 \lambda_{n-1}^{2k}}
\end{aligned}$$

så

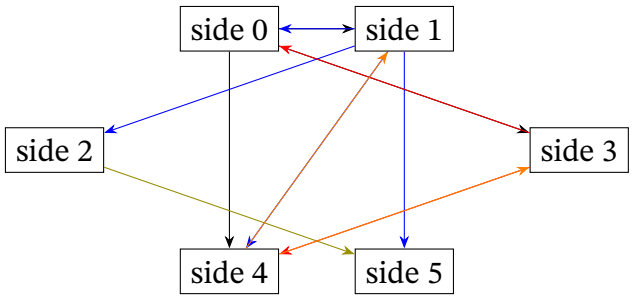
$$\begin{aligned}
|\lambda^{(k)} - \lambda_0| &\leqslant \frac{\left| s_1^2 \lambda_1^{2k} (\lambda_1 - \lambda_0) + \cdots + s_{n-1}^2 \lambda_{n-1}^{2k} (\lambda_{n-1} - \lambda_0) \right|}{s_0^2 \lambda_0^2} \\
&\leqslant \frac{(s_1^2 + \cdots + s_{n-1}^2) \lambda_1^{2k} \max_{i=1,\ldots,n-1} |\lambda_i - \lambda_0|}{s_0^2 \lambda_0^2} \\
&\leqslant \left(\frac{1 - s_0^2}{s_0^2} \max_{i=1,\ldots,n-1} |\lambda_i - \lambda_0| \right) \left(\frac{\lambda_1}{\lambda_0} \right)^{2k},
\end{aligned}$$

som er den anden ønskede ulighed. □

24.2 Page rank

En anvendelse af potensmetoden er til rangering af web sider via page rank, som bruges af Google i forbindelse med prioritering af resultater fra internet søgninger.

Lad os antage at vi vil gerne bestemmer rangering af n websider. En webside j har links til $k(j)$ andre forskellige sider. Så indfører vi en $(n \times n)$-matrix



Figur 24.1: En samling websider med links.

$M = (m_{ij})$ med

$$m_{ij} = \begin{cases} \dfrac{1}{k(j)}, & \text{hvis der er en link fra side } j \text{ til side } i, \\ 0, & \text{hvis der er ingen link } j \text{ til } i \text{ og } k(j) \neq 0, \\ \dfrac{1}{n}, & \text{hvis } j \text{ har ingen links til andre sider.} \end{cases}$$

Vi opstiller en model for hvilket sider en bruger, der læser på nettet, besøger. Brugeren starter på en af siderne. Brugeren går videre til en ny side ved enten (a) at følge en link på den nuværende side eller (b) at springe direkte til en tilfældig side.

Vi antager at tilfældet (a) sker med sandsynlighed $0 < p < 1$, og at tilfældet (b) sker med sandsynlighed $1 - p$. Ofte tages $p = 0{,}85$.

Sandsynligheden for at brugeren går fra side j direkte til side i er nu

$$a_{ij} = pm_{ij} + (1 - p)\frac{1}{n} \tag{24.2}$$

Bemærk at valget af værdien $m_{ij} = 1/n$, når side j har ingen links til andre sider, giver $a_{ij} = p(1/n) + (1 - p)1/n = 1/n$ i sådan en situation. Dette repræsenterer korrekt at brugeren er tvunget til at vælge en tilfældig side, når der er ingen links at følge.

Eksempel 24.3. For samlingen af 6 websider givet i figur 24.1 har vi

$$M = \begin{bmatrix} 0 & 1/4 & 0 & 1/2 & 0 & 1/6 \\ 1/3 & 0 & 0 & 0 & 1/2 & 1/6 \\ 0 & 1/4 & 0 & 0 & 0 & 1/6 \\ 1/3 & 0 & 0 & 0 & 1/2 & 1/6 \\ 1/3 & 1/4 & 0 & 1/2 & 0 & 1/6 \\ 0 & 1/4 & 1 & 0 & 0 & 1/6 \end{bmatrix}.$$

△

Som eksemplet viser, har hver søjle af M sum 1. Matricer med dette egenskab kaldes *stokastiske matricer*. Formlen (24.2) for $A = (a_{ij})$, giver at A er også en stokastisk matrix. Bemærk desuden at A har at alle a_{ij} opfylder $a_{ij} \geqslant (1-p)/n > 0$.

Proposition 24.4. *Hvis $B \in \mathbb{R}^{n\times n}$ er en stokastisk matrix så er $\lambda = 1$ en egenværdi.*

Bevis. I matricen $B - I_n$ er hvert søljesum lige med $1 - 1 = 0$. Det følger at summen af alle rækker i $B - I_n$ er rækkevektoren 0, så echelonformen for $B - I_n$ indeholder en nulrække. Det følger at $(B - I_n)x = 0$ har en løsning med $x \neq 0$, og dermed $Bx = I_n x = x$. Så x er egenvektor med egenværdi 1. □

Et dybere resultat er

Sætning 24.5 (Perron-Frobenius)**.** *Hvis A er en stokastisk matrix med alle indgange > 0, så har egenværdien $\lambda = 1$ en egenvektor v med positive indganger og alle andre egenværdier λ_i for A har $|\lambda_i| < 1 = \lambda$.* □

Page rank er den positive egenvektor v med $\|v\|_2 = 1$ og egenværdi 1 for matricen A af (24.2). Sætningen sikre at vi kan bruge potensmetoden til at bestemme den.

Eksempel 24.6. Vi fortsætter eksempel 24.3. Vi opstiller matricen `m` og definere `a` via (24.2).

```
import numpy as np

m = np.array([[  0., 1/4.,  0., 1/2.,   0., 1/6.],
              [1/3.,   0.,  0.,   0., 1/2., 1/6.],
              [  0., 1/4.,  0.,   0.,   0., 1/6.],
```

```
              [1/3.,   0.,  0.,   0., 1/2., 1/6.],
              [1/3., 1/4.,  0., 1/2.,   0., 1/6.],
              [  0., 1/4., 1.0,   0.,   0., 1/6.]])

p = 0.85
n = m.shape[0]
a = p * m + ((1-p)/n) * np.ones_like(m)

np.set_printoptions(linewidth = 60)
print(a)
```

```
[[0.025      0.2375     0.025      0.45       0.025
  0.16666667]
 [0.30833333 0.025      0.025      0.025      0.45
  0.16666667]
 [0.025      0.2375     0.025      0.025      0.025
  0.16666667]
 [0.30833333 0.025      0.025      0.025      0.45
  0.16666667]
 [0.30833333 0.2375     0.025      0.45       0.025
  0.16666667]
 [0.025      0.2375     0.875      0.025      0.025
  0.16666667]]
```

Nu bruger vi potensmetoden, skrevet på en måde der kun kræve en matrix-vektormultiplikation per løkke. Bemærk at da vi ønsker en egenvektor med positive indgange og alle indgange i `a` er positive, kan vi begynde med en vektor `v` med positive indgange. Vi vælger at stoppe når de sidste to kandidater for egenvektorer er næste parallelle med hinanden

Koden nedenfor bruger en **`while`** løkke. Gentagelse af løkken kontrolleres af en test, som vi sætter til altid at give sandt, **`True`**. Løkken afbrydes i stedet med **`break`** kommandoen, når indre produktet mellem `v_ny` og `v` er tilstrækkelig tæt på `1.0`.

```
rng = np.random.default_rng()
v = rng.random((n, 1))
v /= np.linalg.norm(v)
```

```
nøjagtighed = 1e-9

while True:
    v_ny = a @ v
    v_ny /= np.linalg.norm(v_ny)
    if np.vdot(v_ny,v) > 1.0 - nøjagtighed:
        print(v_ny)
        break
    else:
        v = v_ny

print('lambda = ', (v_ny.T @ (a @ v_ny))[0, 0])

print(v_ny.argmax())
```

```
[[0.39646858]
 [0.44276128]
 [0.20828898]
 [0.44276128]
 [0.5087964 ]
 [0.38533036]]
lambda =  1.0000009332180497
4
```

Vi ser at den beregnede λ = `v_ny.T @ (a @ v_ny)` er tæt på 1, og side 4 har fået den største page rank, tæt efterfulgt af sider 1 og 3. Vi kan få rangeringen af sider bestemt af `v_ny` fra `np.argsort()`, som vil giver indekserne for ordningen af indgangerne i `v_ny` i voksende rækkefølger. Vi er mere interesseret i den omvendte rækkefølge, så vi bruger `np.flip()` bagefter.

```
print(np.flip(np.argsort(v_ny, axis=0)))
```

```
[[4]
 [3]
 [1]
 [0]
 [5]
 [2]]
```

Δ

24.3 Rayleighs kvotient

I potensmetoden brugt vi udtrykket $(w^{(k)})^T A w^{(k)}$ som vores tilnærmelse til egenværdien λ_0. Generelt hvis $v \neq 0$, er *Rayleighs kvotient* tallet

$$r(v) = \frac{v^T A v}{v^T v}.$$

Når v er en egenvektor for A med $Av = \lambda v$, er

$$r(v) = \frac{v^T A v}{v^T v} = \frac{v^T \lambda v}{v^t v} = \lambda$$

den tilhørende egenværdi.

Hvis w er tæt på en egenvektor v, er $r(w)$ en god tilnærmelse til egenværdien λ: Skalaen $r(x)$ er den mindste kvadraters løsning på systemet $xr = Ax$, da systemet har normalligninger $x^T xr = x^T Ax$.

For A symmetrisk har vi

$$\begin{aligned}
\frac{\partial r(x)}{\partial x_j} &= \frac{\partial}{\partial x_j} \frac{\sum_{k,\ell=0}^{n-1} x_k a_{k\ell} x_\ell}{\sum_{i=0}^{n-1} x_i^2} \\
&= \frac{\sum_{\ell=0}^{n-1} a_{j\ell} x_\ell + \sum_{k=0}^{n-1} x_k a_{kj}}{x^T x} - \frac{x^T A x \, 2x_j}{(x^T x)^2} \\
&= \frac{1}{x^T x}\big(2(Ax)_j - r(x) 2x_j\big) = \frac{2}{x^T x}\big(Ax - r(x)x\big)_j.
\end{aligned}$$

Det følger at x er et kritisk punkt for $r(x)$ hvis og kun hvis x er en egenvektor. Vi får også, at hvis v er egenvektor for A, så gælder

$$r(x) - r(v) = O(\|x - v\|_2^2) \quad \text{for } x \to v.$$

Dette siger at $r(x)$ giver en estimering af λ, som er korrekt til anden orden i $\|x - v\|_2$.

24.4 Inverspotensmetoden

En stor ulempe med potensmetoden er at den kun finde den egenværdi, der er numerisk størst. Har man interesse i andre egenværdier, eller hvis man har et

godt gæt for den største, kan man bruge inverspotensmetoden, som bruger et gæt μ for den ønskede egenværdi.

INVERSPOTENSMETODEN(A, μ)
1 Begynd med et vilkårligt $w^{(0)}$ af længde 1
2 **for** $k \in \{1, 2, \ldots\}$:
3 Løs $(A - \mu I_n)v = w^{(k-1)}$ for $v \neq 0$
4 $w^{(k)} = v/\|v\|_2$
5 $\lambda^{(k)} = (w^{(k)})^T A w^{(k)}$

Under visse antagelser konvergerer $\lambda^{(k)}$ til egenværdien af A, som er numerisk tættest på μ. Desuden vil $w^{(k)}$ konvergerer mod en egenvektor for denne egenværdi.

For at gøre disse udsagn mere præcis, antager vi at $A \in \mathbb{R}^{n\times n}$ er symmetrisk og at $\mu \in \mathbb{R}$ er givet. Lad λ_p være egenværdien af A, som minimerer $|\mu - \lambda_p|$, og lad λ_q være egenværdien af A med $|\mu - \lambda_q|$ næst mindst. Vi antager så at egenværdierne $\lambda_0, \lambda_1, \ldots, \lambda_{n-1}$ opfylder

$$|\mu - \lambda_p| < |\mu - \lambda_q| \leqslant |\mu - \lambda_i| \qquad \text{for alle } i \neq p.$$

Lad v_p være en egenvektor med egenværdi λ_p og $\|v_p\|_2 = 1$, og antag at begyndelsesvektoren $w^{(0)}$ i inverspotensmetoden opfylder $s_p = \langle w, v_p\rangle \neq 0$. Så har vi:

Proposition 24.7. *For $w^{(k)}$ og $\lambda^{(k)}$ produceret af den inverspotensmetoden, findes der $\varepsilon_k \in \{\pm 1\}$ således at*

$$\|w^{(k)} - \varepsilon_k v_p\|_2 = O\left(\left|\frac{\mu - \lambda_q}{\mu - \lambda_p}\right|^k\right) \quad \text{og} \quad |\lambda^{(k)} - \lambda_0| = O\left(\left|\frac{\mu - \lambda_q}{\mu - \lambda_p}\right|^{2k}\right).$$

□

Vi får igen kvadratisk konvergens for egenværdien, og denne konvergens er hurtig hvis μ ligger væsentlige tættere på λ_p end på λ_q. Dette er en foretrukken metoden for beregning af konkrete egenværdier.

Eksempel 24.8. Lad os kikke igen på eksempel 24.1.

```
import numpy as np

a = np.array([[2., 1., 1.],
```

```
               [1., 3., 1.],
               [1., 1., 4.]])
```

Denne gang bruger vi inverspotensmetoden. Først lad os kikke efter en egenværdi, som er tættest på 1,0

```
mu = 1

rng = np.random.default_rng()

w = rng.standard_normal((a.shape[0], 1))
w /= np.linalg.norm(w)

n = 20
lambda_out = np.empty(n)

for i in range(n):
    v = np.linalg.solve(a - mu * np.eye(a.shape[0]), w)
    w = v / np.linalg.norm(v)
    lambda_out[i] = w.T @ (a @ w)

np.set_printoptions(linewidth = 60)
print(lambda_out)
```

```
[2.46383109 1.40271812 1.32854811 1.32504915 1.32487802
 1.32486957 1.32486915 1.32486913 1.32486913 1.32486913
 1.32486913 1.32486913 1.32486913 1.32486913 1.32486913
 1.32486913 1.32486913 1.32486913 1.32486913 1.32486913]
```

Vi ser at $\lambda^{(k)}$ stabiliserer relativt hurtigt, og vi kan nemt tjekker hvor godt et par λ, w vi har fået

```
print(np.allclose(a @ w, lambda_out[-1] * w,
                  atol = np.finfo(float).eps))
```

```
True
```

Fra eksempel 24.1 ved vi at den største egenværdi ligger tæt på 5,2, dette kan sættes ind i inverspotensmetoden, som værdien for μ

```
mu = 5.2

w = rng.standard_normal((a.shape[0], 1))
w /= np.linalg.norm(w)

n = 20
lambda_out = np.empty(n)

for i in range(n):
    v = np.linalg.solve(a - mu * np.eye(a.shape[0]), w)
    w = v / np.linalg.norm(v)
    lambda_out[i] = w.T @ (a @ w)

np.set_printoptions(linewidth = 60)
print(lambda_out)
```

```
[5.21404585 5.21431974 5.21431974 5.21431974 5.21431974
 5.21431974 5.21431974 5.21431974 5.21431974 5.21431974
 5.21431974 5.21431974 5.21431974 5.21431974 5.21431974
 5.21431974 5.21431974 5.21431974 5.21431974 5.21431974]
```

som giver hurtige konvergens end potensmetoden. △

Hvis vi udnytter at vi ændrer på valget af μ undervejs, kan vi få endnu bedre konvergens. Forfines af inverspotensmetoden på denne måde, far vi Rayleighkvotientmetoden:

RAYLEIGHKVOTIENTMETODEN(A, w)
1 $w^{(0)} = w$ en endhedsvektor
2 $\lambda^{(0)} = w^T A w$
3 **for** $k \in \{1, 2, \ldots\}$:
4 Løs $(A - \lambda^{(k-1)} I_n) v = w^{(k-1)}$ for $v \neq 0$
5 $w^{(k)} = v / \|v\|_2$
6 $\lambda^{(k)} = (w^{(k)})^T A w^{(k)}$

som har tredjeordens konvergens med en faktor der forbedres for hvert skridt:

Proposition 24.9. *For A symmetrisk, konvergerer Rayleighkvotientmetoden til en egenværdi λ_p af A og for stort k gælder*

$$\|w^{(k+1)} - \varepsilon_{k+1} v_p\|_2 = O(\|w^{(k)} - \varepsilon_k v_p\|_2^3) \quad og \quad |\lambda^{(k+1)} - \lambda_p| = O(|\lambda^{(k)} - \lambda_p|^3),$$

med $\epsilon_k \in \{\pm 1\}$. □

Eksempel 24.10. For matricen i (24.1), ser vi konvergens af både egenværdien og egenvektoren indenfor machine epsilon ofte efter kun 3 iterationer:

```
for j in range(5):
    w = rng.standard_normal((a.shape[0], 1))
    w /= np.linalg.norm(w)

    for i in range(20):
        lambda_est = (w.T @ (a @ w))[0,0]
        print(lambda_est)
        if np.allclose(a @ w, lambda_est * w,
                       atol = np.finfo(float).eps):
            print(f'   efter {i} iterationer\n')
            break
        b = a - lambda_est * np.eye(a.shape[0])
        v = np.linalg.solve(b, w)
        w = v / np.linalg.norm(v)
```

```
3.3945234562623323
2.85850771087919
2.4729161909140656
2.4608110770364813
2.460811127189111
   efter 4 iterationer

4.800408617868152
5.199437133141138
5.214319304661917
5.214319743377536
   efter 3 iterationer
```

```
3.202411206465671
2.759637001377854
2.457850359934781
2.460811035908712
2.460811127189111
   efter 4 iterationer

4.269455248018362
5.07300835226735
5.21406918458095
5.214319743376162
   efter 3 iterationer

2.300285638678587
2.4556511204106344
2.46081101904172
2.4608111271891104
   efter 3 iterationer
```

Bemærk at forskellige egenværdier kan beregnes på denne måde. △

Potensmetoden på $A \in \mathbb{R}^{n\times n}$ bruger $O(n^2)$ flops per skridt til at estimere den numerisk størst egenværdi. For inverspotensmetoden, løser vi et lineært ligningssystem med den samme koefficientmatrix $A - \mu I_n$ hver gang, og dette koster $O(n^3)$ flops. Hvis vi først beregne en QR-dekomponering af $A - \mu I_n$ i begyndelsen, kan denne bruges til at reducere arbejdet per skridt fra $O(n^3)$ flops til $O(n^2)$ flops. Tilgengæld, kan sådan en forbedring ikke bruges til Rayleighkvotientmetoden, som bruger $O(n^3)$ flops per skridt. Men dette opvejes af at Rayleighkvotientmetoden bruger meget få skridt til at få meget høj nøjagtighed.

Python indeks

A
`argsort`, 10

B
`break`, 9

F
`flip`, 10

T
`True`, 9

W
`while`, 9

Indeks

I
Inverspotensmetoden, 12

M
matrix
stokastisk, 8

P
page rank, 8
Perron-Frobenius sætning, 8
potensmetoden, 2

R
Rayleighkvotientmetoden, 14
Rayleights kvotient, 11

S
stokastisk matrix, 8
stor-»O«-notationen, 5