Numerieke Modellering en Benadering: Eigenwaardeproblemen en iteratieve methoden

Sander Prenen, r0701014

20 mei 2020

Inhoudsopgave

Lijst van figuren

1 Inleiding

In dit verslag worden verschillende methodes bestudeerd om eigenwaardeproblemen op te lossen. In sectie 2 wordt de gelijktijdige iteratie bekeken en wordt het algoritme aangepast om de eigenwaarden te vinden dicht bij een gegeven waarde. In sectie 3 wordt zowel de inverse iteratie als de Rayleigh-iteratie bekeken en worden ze vergeleken. Vervolgens wordt in sectie 4 de Jacobi-methode bestudeerd en deze methode wordt in MATLAB geïmplementeerd. Ten slotte wordt in sectie 5 de methode van de geconjugeerde gradiënten en de bijbehorende convergentiesnelheid aangehaald.

2 Gelijktijdige iteratie

Opgave 1a

Bij gelijktijdige iteratie of 'simultaneous iteration' wordt de methode van de machten toegepast op p kolommen tegelijk. De methode convergeert naar de p grootste eigenwaarden en de bijhorende eigenvectoren. Door de methode licht te wijzigen, kunnen de p eigenwaarden van een symmetrische niet-singuliere matrix A, die het dichtst bij een gegeven waarde μ liggen, bepaald worden. Hiervoor wordt verondersteld dat μ geen eigenwaarde is van A. Het algoritme kan hiervoor gebruikt worden als de p eigenwaarden het dichtst bij μ ook de p grootste eigenwaarden zijn. Hiervoor kan gezorgd worden door de matrix A te vervangen door $A - \mu I$. Deze matrix heeft eigenwaarden die μ kleiner zijn dan de eigenwaarden van de originele matrix A. De originele eigenwaarden λ_o zijn de oplossing van volgende vergelijking:

$$\det(A - \lambda_o I) = 0$$

De eigenwaarden λ_n van de nieuwe matrix zijn de oplossingen van onderstaande vergelijking:

$$\det(A - \mu I - \lambda_n I) = 0 \Leftrightarrow \det(A - (\mu + \lambda_n)I) = 0$$

Dit leidt tot volgende gelijkheid $\lambda_o = \mu + \lambda_n$. Dit betekent dat de nieuwe eigenwaarden λ_n inderdaad μ lager liggen dan de originele eigenwaarden λ_o.
Op deze manier zijn de p gezochte eigenwaarden de kleinste van alle eigenwaarden. Om er voor te zorgen dat ze de p grootste eigenwaarden worden, wordt de inverse matrix gebruikt. De inverse matrix heeft namelijk als eigenwaarden de reciproke eigenwaarden van de originele matrix.

$$\begin{aligned} Ax &= \lambda x \\ A\frac{1}{\lambda}x &= x \\ A^{-1}A\frac{1}{\lambda}x &= A^{-1}x \\ \frac{1}{\lambda}x &= A^{-1}x \end{aligned}$$

Nu zijn de p eigenwaarden die het dichtst bij μ liggen de grootste eigenwaarden van de matrix en kan de methode van gelijktijdige iteratie gebruikt worden. In algoritme 1 kan de pseudocode voor deze methode gevonden worden. De kolommen van $\hat{Q}^{(k)}$ convergeren naar de eigenvectoren horende bij de eigenwaarden. Op deze kolommen $q^{(k)}$ wordt het Rayleigh quotiënt toegepast om de bijbehorende eigenwaarden te benaderen.

```
1 Kies Q̂^(0) ∈ ℝ^(m×n) met orthonormale kolommen
2 for k = 1, 2, ... do
3       Z = (A − μI)^(−1) Q̂^(k−1)
4       Q̂^(k) R̂^(k) = Z
5       λ^(k) = q^(k)ᵀ (A − μI)^(−1) q^(k)
6 end
```

Algoritme 1: Gelijktijdige iteratie

Opgave 1b

Wanneer twee eigenwaarden even ver van μ verwijderd liggen, kan de methode in sommige gevallen niet convergeren. De waarde in de verschillende iteratiestappen van de berekende eigenwaarde $\lambda^{(k)}$ voldoet namelijk aan de volgende gelijkheid:

$$|\lambda^{(k)} - \lambda_J| = \mathcal{O}\left(\left|\frac{\mu - \lambda_J}{\mu - \lambda_K}\right|^{2k}\right)$$

Als de eigenwaarden λ_J en λ_K, die het dichtst bij μ liggen gelijk zijn, convergeert de methode niet. Door de eindige precisie van een computer zal de deling in het rechterlid van de vergelijking bijna nooit exact 1 geven. Hierdoor zal de methode in de meeste gevallen wel convergeren.

3 Inverse iteratie en Rayleigh quotiënt iteratie

3.1 Rayleigh quotiënt iteratie

Opgave 2a

Om eigenwaarden te benaderen kan de Rayleigh quotiënt iteratie gebruikt worden. Deze methode convergeert kubisch naar de correct waarde. In elke iteratiestap moet het volgende stelsel opgelost worden:

$$(A - \lambda^{(k-1)} I)w = v^{(k-1)}$$

Indien de benaderende eigenwaarde $\lambda^{(k-1)}$ dicht in de buurt komt van de exacte eigenwaarde, wordt dit stelsel meer en meer singulier. Hierdoor kan de oplossing w van dit stelsel opblazen. Dit vormt echter geen proleem voor de iteratie aangezien w genormeerd wordt. Enkel de richting van de vector w is dus belangrijk en deze wordt niet veranderd.

Opgave 2b

Er kan aangetoond worden dat voor een matrix $A \in \mathbb{R}^{n\times n}$ en een vector $x \in \mathbb{R}^{n\times 1}$, met x een benadering voor een eigenvector van A, de oplossing $\rho \in \mathbb{R}$ van het volgende minimalisatieprobleem

$$\min_{\rho\in\mathbb{R}} \|Ax - \rho x\|_2$$

overeenkomt met het Rayleigh quotiënt van x.

De functie $f(\rho) = \|Ax - \rho x\|_2 = (Ax - \rho x)^\intercal (Ax - \rho x)$ moet geminimaliseerd worden.
De afgeleide $f'(\rho)$ moet dus nul zijn.

$$\begin{aligned}(Ax - \rho x)^\intercal (Ax - \rho x) &= x^\intercal A^\intercal Ax - x^\intercal A^\intercal \rho x - x^\intercal \rho Ax + \rho^2 x^\intercal x \\ &= x^\intercal A^\intercal Ax - 2\rho x^\intercal Ax + \rho^2 x^\intercal x\end{aligned}$$

Deze vergelijking afleiden naar ρ en gelijk stellen aan nul geeft volgende uitdrukking:

$$-2x^\intercal Ax + 2\rho x^\intercal x = 0 \Leftrightarrow \rho = \frac{x^\intercal Ax}{x^\intercal x}$$

Deze laatste uitdrukking is inderdaad het Rayleigh quotiënt.

3.2 Vergelijking inverse iteratie en Rayleigh quotiënt iteratie

Opgave 3

Door de kubische convergentie van de Rayleigh quotiënt iteratie heeft deze methode vaak de voorkeur over de inverse iteratie. Toch zijn er ook gevallen waar de inverse iteratie de voorkeur geniet. Er worden in deze tekst twee gevallen besproken:

- De matrix $A \in \mathbb{R}^{m\times m}$ is (zeer) groot: In elke iteratiestap van de Rayleigh quotiënt iteratie moet een nieuwe matrix geïnverteerd worden. Hierdoor zijn er $\mathcal{O}(m^3)$ flops nodig per stap. Voor de inverse iteratie moet er een lineair stelsel worden opgelost in elke stap. Dit lijkt op het eerste zicht $\mathcal{O}(m^3)$ flops te vragen, maar als de matrix A op QR-gefactoriseerd is, vermindert het aantal flops naar $\mathcal{O}(m^2)$. Hierdoor is de inverse iteratie beter geschikt voor grote matrices.
- De spectrale eigenschappen van A zijn nauwelijks gekend: Het doel is om een eigenpaar te vinden dat het dichtst bij $\mu \in \mathbb{R}$ ligt, maar een goede startwaarde voor de eigenvector is niet gekend. De Rayleigh quotiënt iteratie vertrekt van een startwaarde van een eigenvector. Deze startwaarde is hier niet gekend dus is de inverse iteratie, die vertrekt van een startwaarde voor de eigenwaarde, beter geschikt.

4 Jacobi-methode

De Jacobi-methode steunt op de diagonalisatie van 2×2 symmetrische matrices met behulp van een orthogonale matrix J,

$$J^T \begin{bmatrix} a & d \\ d & b \end{bmatrix} J = \begin{bmatrix} \neq 0 & 0 \\ 0 & \neq 0 \end{bmatrix} \tag{1}$$

Hierbij wordt de orthogonale matrix J gedefinieerd als een rotatie-matrix:

$$J = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \tag{2}$$

4.1 De coëfficiënten c en s

Opgave 4

In deze sectie worden de coëfficiënten c en s van de matrix J uit vergelijking (2) bepaald zodat (1) geldt. Aangezien J een rotatie-matrix is, is de determinant van j gelijk aan 1.

$$\det J = c^2 + s^2 = 1$$

Volgende oplossing is een oplossing van het stelsel:

$$\begin{cases} c = \cos(\theta) \\ s = \sin(\theta) \end{cases}$$

Indien de niet-diagonaal elementen in vergelijking (1) worden bepaald, wordt volgende uitdrukking verkregen:

$$\frac{1}{2}\sin(2\theta)(a-b) + d\cos(2\theta) = 0$$

Hieruit kan θ gehaald worden:

$$\theta = \frac{1}{2}\arctan\left(\frac{2d}{b-a}\right) \tag{3}$$

Als a gelijk is aan b wordt θ gelijk aan $\frac{\pi}{2}$ aangezien voor deze hoek de tangens oneindig wordt.

4.2 Eigenwaardeontbinding

Opgave 5

De Jacobi methode kan gebruikt worden om de eigenwaarden ontbinding VDV^T van een symmetrische matrix A te berekenen. Hierbij is D een diagonaalmatrix die de eigenwaarden bevat en V een orthognale matrix met de overeenkomstige eigenvectoren. In algoritme 2 kan de pseudo-code voor dit algoritme gevonden worden. In dit algoritme wordt de *sweep*-methode gebruikt die in het handboek beschreven staat.

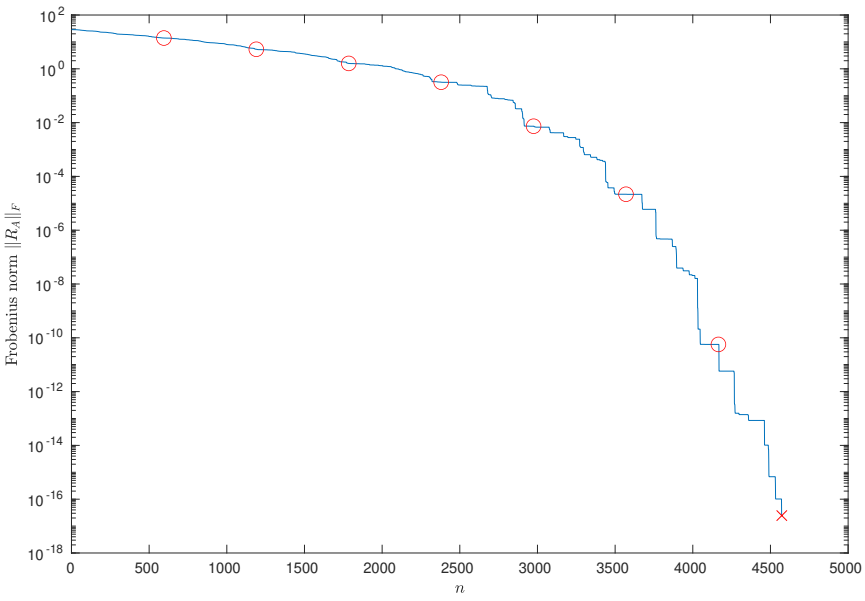
```
1  n = size(A)
2  V = I_n
3  error = ||R_A||_F
4  while error > tol do
5      neem het volgende element van de sweep
6      definieer a, b, d
7      bepaal θ met vergelijking (3)
8      stel de Jacobimatrix J op
9      A = J^T A J
10     V = V J
11     error = ||R_A||_F
12 end
```

Algoritme 2: Jacobi-methode

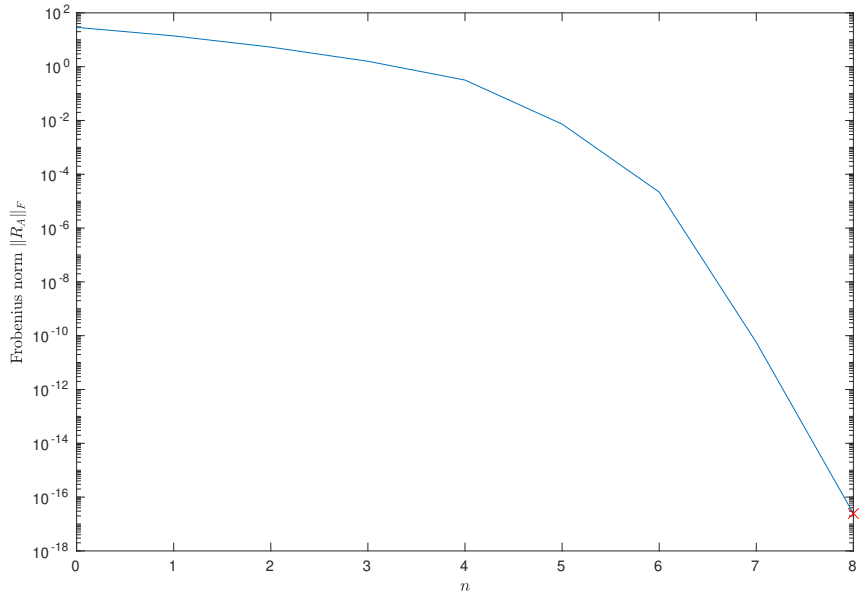
Het algoritme wordt vervolgens geïmplementeerd in MATLAB. In listing 3 kan de code hiervan gevonden worden.

Opgave 6

Vervolgens wordt het algoritme getest op een 35×35 matrix. De frobeniusnorm $\|R_A\|_F$ van de bovendriehoekselementen wordt na elke iteratiestap berekend. In figuur 1 wordt $\|R_A\|_F$ afgebeeld in functie van het aantal iteratiestappen. De omcirkelde waarden zijn de iteratiestappen waarbij een volledige *sweep* gedaan is. Het kruisje duidt aan waar de norm onder de tolerantiegrens van 10^{-16} komt. Hier is niet noodzakelijk een volledige *sweep* gedaan. Er zijn in totaal dus acht volledige *sweeps* nodig om de opgelegde tolerantie van 10^{-16} te bereiken. Dat wil zeggen dat de convergentie beter dan lineair is zoals in het handboek beschreven staat. In figuur 2 wordt de frobeniusnorm weergegeven in functie van het aantal *sweeps*.



Figuur 1: Frobeniusnorm $\|R_A\|_F$ in functie van het aantal iteratiestappen n



Figuur 2: Frobeniusnorm $\|R_A\|_F$ in functie van het aantal volledige *sweeps* n

```
function [V,D]= jacobi(A,tol)

% Aanname dat A vierkant is
n = size(A,1);

upperTriangularIndexList = nonzeros(triu(reshape(1:numel(A),size(A)),1));
k = upperTriangularIndexList';
err = sqrt(sum(A(k).^2));
V = eye(n);
row = 1;
col = 2;
while err > tol
    a = A(row, row);
    b = A(col, col);
    d = A(row, col);

    % Theta bepalen
    if a == b
        theta = pi / 2;
    else
        theta = 0.5 * atan(2 * d / (b - a));
    end

    % Jacobi matrix opstellen
    J = eye(n);
    c = cos(theta);
    s = sin(theta);
    J(row, row) = c;
    J(col, col) = c;
    J(row, col) = s;
    J(col, row) = -s;

    % Iteratiestap
    A = J' * A * J;
    V = V * J;

    % Fout vinden
    err = sqrt(sum(A(k).^2));

    % row en col ophogen
    if row ≠ n - 1
        if col == n
            row = row + 1;
            col = row + 1;
        else
            col = col + 1;
        end
    else
        if col == n
            row = 1;
            col = 2;
        else
            col = col + 1;
        end
    end
end
D = A;
```

Listing 3: jacobi.m

5 Geconjugeerde gradiënten

Opgave 7a

Bij de methode van de conjugeerde gradiënten toegepast op een positief definiete symmetrische matrix probleem $Ax = b$, met $\kappa(A)$ het conditiegetal van de matrix A, geldt de volgende ongelijkheid:

$$\frac{\|e_n\|_A}{\|e_0\|_A} \leq 2 \Bigg/ \left[\left(\frac{\sqrt{\kappa(A)}+1}{\sqrt{\kappa(A)}-1} \right)^n + \left(\frac{\sqrt{\kappa(A)}+1}{\sqrt{\kappa(A)}-1} \right)^{-n} \right] \leq 2 \left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1} \right)^n \tag{4}$$

Indien de waarden voor $\|e_0\|_A = 1$ en $\|e_{10}\|_A = 2^{-9}$ bekend zijn, kan er een ondergrens voor $\kappa(A)$ bepaald worden door vergelijking (4) in te vullen. Dit geeft volgende oplossing:

$$\begin{aligned}
2 \left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1} \right)^{10} &\geq \frac{\|e_{10}\|_A}{\|e_0\|_A} = 2^{-9} \\
\left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1} \right)^{10} &\geq 2^{-10} \\
\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1} &\geq \frac{1}{2} \\
2\sqrt{\kappa(A)}-2 &\geq \sqrt{\kappa(A)}+1 \\
\sqrt{\kappa(A)} &\geq 3 \\
\kappa(A) &\geq 9
\end{aligned}$$

Opgave 7b

Het conditiegetal $\kappa(A)$ heeft dus een ondergrens van 9. Deze ondergrens geeft geen extra informatie over $\|e_n\|_A$ voor n groter dan 10. Wel kan uit volgende eigenschap:

$$\|e_n\|_A \leq \|e_{n-1}\|_A$$

een bovengrens voor $\|e_{20}\|_A$ bepaald worden.

$$\|e_{20}\|_A \leq \|e_{10}\|_A = 2^{-9}$$

Een striktere bovengrens is niet mogelijk met de gegevens die ter beschikking zijn.