MODELLERING EN SIMULATIE
Practicum I: Eigenwaardeproblemen
Opgave Januari zittijd academiejaar 2013-2014

Praktische informatie

Dit practicum los je individueel op. Het verslag moet ten laatste afgegeven worden op 10 december 2013. Je kan het in de studentenbrievenbus in gebouw 200A leggen ofwel afgeven tijdens de oefenzitting. Je verslag bevat de oplossingen van de opgaven met bijhorende figuren en een afdruk van je MATLAB-bestanden. De MATLAB-bestanden en het verslag moeten ook opgestuurd worden naar `nico.achtsis@cs.kuleuven.be`.

Vermeld in je verslag hoe lang je aan het practicum gewerkt hebt en zorg dat je verslag duidelijk leesbaar en begrijpbaar is.

Veel succes!

Nico Achtsis en Wim Michiels.

Opgave

We vertrekken van een matrix $A_0 = A$ en construeren achtereenvolgens de matrices A_1, $A_2, \ldots$ als volgt:

$$\begin{aligned} A_k &= Q_k R_k \qquad \text{(QR-factorisatie van } A_k\text{)} \\ A_{k+1} &= R_k Q_k. \end{aligned} \tag{1}$$

We weten dan vanuit de cursus (p38–40) dat A_k convergeert naar een bovendriehoeksmatrix waarbij de eigenwaarden dezelfde zijn als die van A. Met andere woorden, de elementen op de diagonaal van A_k convergeren naar de eigenwaarden van A. Deze convergentie is echter traag, en daarom wordt in de rest van het hoofdstuk het QR-algoritme met shift besproken. Hierbij wordt een shift $\kappa = a_{mm}$ gekozen, waarna het QR-algoritme toegepast wordt op $A - \kappa I$. Vertrekkende van een Hessenbergmatrix $A = A_0$ worden dan achtereenvolgens de (Hessenberg) matrices A_1, $A_2, \ldots$ geconstrueerd als

$$\begin{aligned} \kappa &= a_{mm}^{(k)} \\ A_k - \kappa I &= Q_k R_k \qquad \text{(QR-factorisatie van } A_k - \kappa I\text{)} \\ A_{k+1} &= R_k Q_k + \kappa I. \end{aligned} \tag{2}$$

Let op dat in het bovenstaande algoritme de shift **niet** verandert naargelang het element $a_{m,m-1}^{(k)}$ klein wordt. Het is natuurlijk niet noodzakelijk om de shift $\kappa = a_{mm}$ te kiezen.

Opgave 1 *Neem de willekeurige symmetrische en reële $m \times m$ matrix*

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{12} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1m} & a_{m-1,m} & \cdots & a_{mm} \end{pmatrix}$$

Toon aan dat je met de juiste Givens-rotatie G_{23} een nul creëert in $G_{23}^T A G_{23}$ op de $(1,3)$de en $(3,1)$de plaats.

Je kan hieruit, mits enige redenering, afleiden dat de matrix omgevormd wordt tot een symmetrische en tridiagonale matrix. (Dit moet je niet afleiden voor het practicum.) In dit geval is de Hessenberg matrix van de vorm

$$A = \begin{pmatrix} a_1 & b_1 & 0 & 0 & \cdots & 0 \\ b_1 & a_2 & b_2 & 0 & \cdots & 0 \\ 0 & b_2 & a_3 & b_3 & \cdots & 0 \\ 0 & 0 & b_3 & a_4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & a_m \end{pmatrix}$$

We kunnen dan de shift

$$\kappa = a_m - \frac{\text{sign}(\delta) b_{m-1}^2}{|\delta| + \sqrt{\delta^2 + b_{m-1}^2}} \tag{3}$$

gebruiken, met

$$\delta = \frac{1}{2}(a_{m-1} - a_m).$$

We bekijken de 20×20 matrix

$$A = \begin{pmatrix} 20 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 19 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 18 & -1 & \cdots & 0 \\ 0 & 0 & -1 & 17 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

Je kan in MATLAB het commando `eig` gebruiken om de eigenwaarden van A te vinden. Deze mag je beschouwen als de exacte eigenwaarden van A in de volgende opgaven.

Opgave 2 *Implementeer de drie besproken methodes in MATLAB: QR zonder shift (1), QR met de shift (2) en QR met de alternatieve shift (3). Implementeer ook zelf het QR algoritme. Om de matrix A efficient in MATLAB in te voeren, kan je gebruik maken van de functie* **diag**.

Opgave 3 *Ga na naar welke eigenwaarde het element $a_{20,20}^k$ convergeert. Plot de convergentie $|\lambda - a_{20,20}^k|$ voor de drie methodes op 1 grafiek (tip: gebruik hiervoor* **semilogy**). *Is dit gedrag verwacht? Doe hetzelfde voor het element $a_{19,19}^k$, en leg uit wat je opmerkt.*

Opgave 4 *Ga na naar welke waarde het element $a_{20,19}^k$ convergeert. Plot de convergentie voor de drie methodes op 1 grafiek (gebruik hiervoor weer* **semilogy**). *Is dit gedrag verwacht? Doe hetzelfde voor een aantal andere waarden voor $a_{k,k-1}^k$, en leg uit wat je opmerkt.*

In de vorige oefeningen lag de shift vast rond het element a_{mm}. We kunnen echter de positie van de shift opschuiven als het element $a_{m,m-1}$ nagenoeg nul wordt. Dit geeft het volgend algoritme voor de eerste shift. Vertrekkende van een Hessenbergmatrix $A = A_0$ worden dan achtereenvolgens de (Hessenberg) matrices A_1, $A_2, \ldots$ geconstrueerd als

$$\begin{aligned} \kappa &= a_{\ell\ell}^{(k)} \\ A_k - \kappa I &= Q_k R_k \qquad \text{(QR-factorisatie van } A_k - \kappa I) \\ A_{k+1} &= R_k Q_k + \kappa I \end{aligned}$$

$$\ell = \ell - 1, a_{\ell,\ell-1}^{(k)} = 0 \text{ als } a_{\ell,\ell-1}^{(k)} < \epsilon.$$

Het algoritme begint uiteraard met $\ell = m$, en ϵ is een kleine tolerantiewaarde. Het algoritme met de tweede shift kan analoog aangepast worden.

Opgave 5 *Pas je MATLAB code van de twee algoritmes met shift aan (er verandert uiteraard niets aan de methode zonder shift). Voor de tolerantiewaarde in MATLAB kan je gebruik maken van* **eps**. *Plot de convergentie $|\lambda - a^k_{20,20}|$ voor de drie methodes op 1 grafiek (gebruik hiervoor weer* **semilogy***). Is dit gedrag verwacht? Doe hetzelfde voor de elementen $a^k_{19,19}$ en $a^k_{15,15}$, en leg uit wat je opmerkt.*

Beeldcompressie

Een matrix $A \in \mathbb{R}^{m\times n}$ kan gebruikt worden om een beeld van $m \times n$ pixels weer te geven. Elk element a_{ij} van de matrix specificeert dan de waarde van pixel p_{ij} in het beeld. Door lage rang benaderingen te gebruiken in plaats van de oorspronkelijke matrix A kunnen we het overeenkomstige beeld comprimeren.

In het bestand `A.mat` vind je een 1350×2400-matrix. Met de MATLAB-commando's

```
imagesc(A);
colormap(gray);
```

kan je de figuur bekijken.

Opgave 6

a) Schrijf een programma dat, gegeven een natuurlijk getal $k \leq \text{rang}(A)$, een benadering van rang k berekent met behulp van volgende methodes: i) de singuliere waardenontbinding en ii) de QR-factorisatie. Het programma moet ook de relatieve fout van de benaderingen teruggeven, gemeten met i) de 2-norm en ii) de Frobenius norm.

b) Wat is voor beide methodes de kleinste waarde van k waarmee je in jouw ogen een acceptabele benadering hebt. Plot de benaderde beelden.

c) Maak een grafiek die in functie van k de relatieve fout in 2-norm van beide rang k benaderingen laat zien. Herhaal hetzelfde met de Frobenius norm. Is dit consistent met de theorie?

c) Hoe efficiënt zijn beide methodes om geheugen te besparen? Wat kan je zeggen over de hoeveelheid rekenwerk?