Practicum Numerieke Wiskunde

2e Bachelor Informatica-Wiskunde

Academiejaar 2015-2016

Het doel van dit practicum is het implementeren van het QR-algoritme voor het berekenen van de eigenwaarden van een matrix. Je zal stapsgewijs kennis maken met de bouwstenen die hiervoor nodig zijn. Als finaal resultaat zal je je Matlab functies die de verschillende deelalgoritmen voorstellen moeten combineren tot een code die het uiteindelijke QR-algoritme voorstelt.

1 De QR-decompositie

Elke matrix $A \in \mathbb{R}^{n \times n}$ kan geschreven worden als een product

$$A = \begin{pmatrix} | & & | \\ \mathbf{a}_1 & \dots & \mathbf{a}_n \\ | & & | \end{pmatrix} = QR$$

$$= \begin{pmatrix} | & & | \\ \mathbf{q}_1 & \dots & \mathbf{q}_n \\ | & & | \end{pmatrix} R,$$

met Q orthogonaal en R bovendriehoeks. We noemen deze ontbinding de *QR-factorizatie* van de matrix A. Indien A inverteerbaar is en we eisen dat de diagonaalelementen van R strikt positief zijn, dan zijn de factoren Q en R uniek bepaald.

1. Gegeven een matrix A, berekent Algoritme 1 de kolommen van de matrix Q en de elementen in het bovendriehoeksgedeelte van de matrix R. Toon dit aan.

2. Toon aan dat Algoritme 1 theoretisch equivalent is met Algoritme 2.

3. Implementeer Algoritmen 1 en 2 in Matlab. Beschouw de matrix A uit het bestand `practicum.mat`. Bereken de QR-factorizatie van deze matrix met elk van beide implementaties. Welk van beide algoritmen geeft het meest nauwkeurige resultaat? Kan je dit numeriek verklaren?

Input: $n \times n$ matrix A;
Output: factoren Q en R;
$r_{11} = ||\mathbf{a}_1||$;
$\mathbf{q}_1 := \mathbf{a}_1 / r_{11}$;
for $j = 1, \ldots, n-1$ **do**
  $\mathbf{z} := \mathbf{a}_{j+1}$;
  for $i = 1, \ldots, j$ **do**
    $r_{i,j+1} := \mathbf{q}_i^T \mathbf{z}$;
  end
  $\mathbf{z} = \mathbf{z} - \sum_{i=1}^{j} r_{i,j+1} \mathbf{q}_i$;
  $r_{j+1,j+1} = ||\mathbf{z}||$;
  $\mathbf{q}_{j+1} = \mathbf{z} / r_{j+1,j+1}$;
end

Algorithm 1: Gram-Schmidt algoritme

2 Het basis QR-algoritme

Beschouw de QR-factorizatie van de matrix A,

$$A = Q_0 R_0.$$

We wisselen nu de factoren om en beschouwen het product $A_1 := R_0 Q_0$ met QR-factorizatie

$$A_1 = Q_1 R_1.$$

Opnieuw wisselen we de factoren om en beschouwen het product $A_2 := R_1 Q_1$. Dit proces kunnen we blijven herhalen, zodat we een rij van matrices A_k verkrijgen, beginnende met $A_0 := A$ die gedefinieerd is door

$$A_k := R_{k-1} Q_{k-1},$$

met Q_{k-1} en R_{k-1} de factoren in de QR-factorizatie van A_{k-1}. We noemen één iteratie van dit proces een *QR-stap*.

1. Toon aan dat de matrices $A_0, A_1, \ldots$ allemaal dezelfde eigenwaarden hebben.

2. Het idee achter het QR-algoritme is dat de matrices A_k onder bepaalde voorwaarden convergeren naar een bovendriehoeksmatrix, waarna de eigenwaarden gewoon kunnen worden afgelezen van de hoofddiagonaal. Implementeer het basis QR-algoritme: schrijf een functie `QRstep.m` die voor een gegeven matrix A één QR-stap uitvoert (maak hiervoor gebruik van het Matlab commando `qr`).

Input: $n \times n$ matrix A;
Output: factoren Q en R;
$r_{11} = ||\mathbf{a}_1||$;
$\mathbf{q}_1 := \mathbf{a}_1 / r_{11}$;
for $j = 1, \ldots, n-1$ **do**
  $\mathbf{z} := \mathbf{a}_{j+1}$;
  for $i = 1, \ldots, j$ **do**
    $r_{i,j+1} := \mathbf{q}_i^T \mathbf{z}$;
    $\mathbf{z} = \mathbf{z} - r_{i,j+1}\mathbf{q}_i$;
  end
  $r_{j+1,j+1} = ||\mathbf{z}||$;
  $\mathbf{q}_{j+1} = \mathbf{z} / r_{j+1,j+1}$;
end

Algorithm 2: Modified Gram-Schmidt algoritme

3. Veronderstel dat de eigenwaarden van een symmetrische matrix geordend zijn in modulus als

$$|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|.$$

Als $k \to \infty$, dan convergeert A_k lineair naar een diagonaalmatrix $\text{diag}(\lambda_1, \ldots, \lambda_n)$. Het element $a_{i,j}^{(k)}$ met $i > j$ convergeert lineair naar nul met convergentiefactor

$$\min_{i-1 \leq l \leq j} |\lambda_{l+1}| / |\lambda_l|.$$

Maak nu zelf een random symmetrische matrix A met eigenwaarden $\lambda_i = \{1, 2, 4, 8, 16, 256, 512, 2048\}$ en test hierop je functie `QRstep.m`. Het genereren van de matrix kan je doen met de commando's [Q,∼]=qr(rand(8)); A=Q∗diag([1 2 4 8 16 256 512 2048])∗Q'. Maak gebruik van het Matlab commando `imagesc` om de structuur van de matrices A_k te visualiseren. (Hint: om een eenvoudigere kleurenschaal te bekomen, mag je elementen die numeriek nul zijn gelijk stellen aan de machine precisie.) Maak ook enkele plots die het lineair convergentiegedrag illustreren.

Beschouw de matrix B uit het bestand `practicum.mat` en test ook hierop je functie `QRstep.m`. Wat zie je gebeuren en kan je dit verklaren? Kan je de eigenwaarden aflezen?

4. Toon aan dat de complexiteit van je functie `QRstep.m` $\mathcal{O}(n^3)$ is.

3 Het QR-algoritme: twee fasen

Het basis QR-algoritme heeft enkele nadelen: de hoge complexiteit en in het algemeen is de convergentie traag. Om dit op te lossen wordt gebruik gemaakt van twee fases:

Fase 1 In de eerste fase berekenen we een Hessenberg matrix H die orthogonaal equivalent is met A:

$$H = UAU^T,$$

met U orthogonaal.

Fase 2 In de tweede fase passen we het basis QR-algoritme toe op de matrix H.

3.1 Fase 1: Householder transformaties

1. Gegeven een vector $\mathbf{x}$. Beschouw de matrix

$$\mathcal{K} := \begin{pmatrix} | & | & | & & | \\ \mathbf{x} & A\mathbf{x} & A^2\mathbf{x} & \dots & A^{n-1}\mathbf{x} \\ | & | & | & & | \end{pmatrix}$$

en veronderstel dat de kolommen van deze matrix lineair onafhankelijk zijn. Beschouw ook de QR-factorizatie $\mathcal{K} = QR$ van de matrix $\mathcal{K}$. Toon aan dat

$$H := Q^T AQ$$

een Hessenberg matrix is.

2. Puntje 1 geeft ons een manier om gegeven een matrix, een Hessenberg matrix te berekenen die hiermee orthogonaal equivalent is. Numeriek gezien is dit echter niet zo een goed idee. Waarom? (Hint: Vergelijk met de methoden van de machten voor het berekenen van de dominante eigenwaarde.)

   Het is beter om te werken met zogenaamde Householder transformaties. Een matrix $P \in \mathbb{R}^{n\times n}$ van de vorm

$$P := I - 2\mathbf{u}\mathbf{u}^T, \quad \text{met } \mathbf{u} \in \mathbb{R}^n, ||\mathbf{u}|| = 1,$$

   wordt een Householder tranformatie genoemd. Toon aan dat een Householder transformatie zowel symmetrisch als orthogonaal is.

3. Een Householder transformatie wordt ook wel een Householder reflector genoemd. Dit is omdat het een spiegeling voorstelt: gegeven een vector $\mathbf{x} \in \mathbb{R}^3$ en veronderstel dat $\mathbf{u} \in \mathbb{R}^3$, dan is $P\mathbf{x}$ het spiegelbeeld van $\mathbf{x}$ rond het vlak loodrecht op $\mathbf{u}$. Toon dit aan en maak een bijhorende illustratie.

4. Zij $\mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq 0$ en definieer

$$\mathbf{u} := \frac{\mathbf{x} + \rho||\mathbf{x}||\mathbf{e}_1}{||\mathbf{x} + \rho||\mathbf{x}||\mathbf{e}_1||}, \tag{1}$$

   met $\rho = \text{sign}(x_1)$ (De sign functie duidt het teken van een scalar aan). Dan is $P\mathbf{x} = ||\mathbf{x}||\mathbf{e}_1$ waarbij $P := I - 2\mathbf{u}\mathbf{u}^T$. Toon dit aan.

5. We kunnen nu Householder transformaties gebruiken om een Hessenberg matrix te berekenen die orthogonaal equivalent is met A. Beschouw de orthogonale gelijkvormigheidstransformatie

$$\begin{pmatrix} 1 & 0^T \\ 0 & I - 2\mathbf{u}_1\mathbf{u}_1^T \end{pmatrix} A \begin{pmatrix} 1 & 0^T \\ 0 & I - 2\mathbf{u}_1\mathbf{u}_1^T \end{pmatrix}^T, \tag{2}$$

   waarbij $\mathbf{u}_1$ gedefineerd is als in (1) met $\mathbf{x} := [a_{21}, \ldots, a_{n1}]^T$. Toon aan dat (2) een matrix is waarvan in de eerste kolom overal nullen staan buiten de eerste twee elementen.

   Vul nu zelf de procedure verder aan om door middel van Householder transformaties een Hessenberg matrix H te berekenen die orthogonaal equivalent is met A. Gebruik nu je bevindingen om een functie `Householder.m` te implementeren dat gegeven een input matrix, een Hessenberg matrix berekent die orthogonaal equivalent is met de input matrix.

6. Test je functie `Householder.m` enkele keren uit op een Hermitische matrix. Voor het crëeren van een random Hermitische matrix kan je gebruik maken van de commando's `rand` en `diag(v,k)`. Wat stel je vast? Verklaar.

3.2 Fase 2: QR-stap op Hessenberg matrix

1. Leg uit hoe je de QR-factorizatie van een Hessenberg matrix H kan bepalen door middel van Givens rotaties. Meer specifiek: schrijf H als een product van Givens rotaties en een bovendriehoeksmatrix R. Hoeveel Givens rotaties heb je nodig? (Hint: je kan inspiratie opdoen in Probleem 4 uit Oefenzitting 7).

2. Geef een random Hessenberg matrix H als input aan je functie `QRstep.m` en kijk wat de output is voor toenenemende waarden van k. Je kan voor het creëren van een random Hessenberg matrix bv. gebruik maken van de commando's `diag(rand(n,1))` en `triu(rand(n))`. Welke structuur heeft de output matrix? Toon theoretisch aan dat dit altijd het geval is door gebruik te maken van de QR-factorizatie met Givens rotaties uit het vorige puntje.

3. Gebruik nu je bevindingen om een functie `QRstepHessenberg.m` te implementeren die voor een gegeven Hessenberg matrix H en index k, de matrix H_k teruggeeft.

4. Toon aan dat de complexiteit van één iteratie $\mathcal{O}(n^2)$ is.

5. Maak een functie `generalQRstep.m` die fasen 1 en 2 combineert.

6. Maak je eigen experiment om de complexiteit van de functies `QRstep.m` en `generalQRstep.m` te vergelijken. Laat beide functies 100 iteraties berekenen voor $n \times n$ matrices met `n = round(logspace(1,3,10))`. Je kan gebruik maken van de commando's `tic` en `toc` om timings te maken.

4 Praktisch

Groepen

Je werkt in groepjes van twee personen. Ten laatste op **maandag 28 maart** stuur je per groepje een email naar *clara.mertens@cs.kuleuven.be* met hierin de namen en de studentennummers van de leden van je groepje. We raden aan om samen te werken met iemand van een andere studierichting en voordeel te halen uit elkaars expertise.

Tijdsbesteding

Dit practicum heeft een belasting van ongeveer 16 uur per student. Beschouw dit getal als een richtlijn voor je tijdsbesteding. Het is niet de bedoeling om hier ver over te gaan, maar we willen er wel op wijzen dat deze richtlijn ervan uitgaat dat je bij bent met de oefenzittingen (vooral die in MATLAB) en met de nodige leerstof.

Begeleide oefenzitting

In oefenzitting 14 (dit is in de week van maandag 25/4) zal je aan het practicum kunnen werken en kan je ook vragen stellen aan de assistent. Deze oefenzitting hoort niet bij de belasting.

Evaluatie

Het practicum telt mee voor 3 van de 20 punten van het eindexamen. De evaluatie van het practicum is gebaseerd op de ingediende code, het verslag en een demo die je zal moeten geven van je programma's. Tijdens de demo zullen er ook extra vragen gesteld worden en is er ruimte voor feedback. De demo's zullen doorgaan tijdens de tijdsloten van oefenzitting 17 in de week van 16 mei.

Er zal vanaf maandag 2 mei om 12h aan het prikbord naast het secretariaat een blad hangen waar je je moet inschrijven voor de demo. Dit moet ten laatste gebeuren op **vrijdag 6 mei om 14h**.

Code

De code moet geupload worden op Toledo door iemand van het groepje ten laatste op **zondag 8 mei**. Zet al de code in één zip-bestand met jullie achternamen in de bestandsnaam, bv. `codeDeSmetVanIeper.zip`. Het zip-bestand moet minstens de volgende bestanden bevatten:

- `QRstep.m`
- `Householder.m`
- `QRstepHessenberg.m`
- `GeneralQRstep.m`

Vergeet mogelijke hulpfuncties niet toe te voegen die jullie geschreven hebben. (Wij moeten je code kunnen uitvoeren.)

Verslag

Schrijf een duidelijk gestructureerd en bondig verslag waarin zeker het volgende staat:

- Alle figuren en tabellen. Benoem steeds de assen en zorg ervoor dat de grafieken leesbaar zijn. Voeg legendes toe indien nodig. Je hoeft geen uitgebreid onderschrift te voorzien, als je in de tekst duidelijk naar de figuur verwijst.

- Antwoorden op alle opgaves en vragen die gesteld worden.
- Indien je dit nodig acht, kan je in het verslag bepaalde ontwerpkeuzes voor je algoritmes verduidelijken.
- Tijdsbesteding aan de verschillende onderdelen van het practicum:
  - Schrijven code van functies
  - Debuggen
  - Schrijven verslag
  - ...

  Indien jullie het werk verdeeld hebben en de tijdsschattingen voor beide groepsleden verschillen, geef dit dan ook duidelijk aan.

Upload je verslag op Toledo als één pdf-bestand ten laatste op **zondag 8 mei**. Tevens geef je een afgedrukte versie van je verslag af. Geef het pdf-bestand een analoge naam als je code, bv. `verslagDeSmetVanIeper.pdf`. Je kan je verslag deponeren in de studentenbrievenbus ten laatste op **maandag 9 mei om 14h**. De studentenbrievenbus hangt in het printerlokaal op het gelijkvloers van gebouw 200A.

Overzicht deadlines

Wat	**Waar**	**Deadline**
Verdelen in groepjes	e-mail	maandag 28 maart
Uploaden code	Toledo	zondag 8 mei
Uploaden verslag	Toledo	zondag 8 mei
Indienen afgedrukt verslag	Studentenbrievenbus 200A	maandag 9 mei om 14h
Inschrijven demo	Prikbord 200A	vrijdag 6 mei om 14h

Veel succes!

Clara en Marc