Practicum NMB : Eigenwaardenproblemen

Matthijs van Keirsblick en Harald Schäfer

vrijdag 25 april 2014

Opgave 1

We beginnen door een QR-factorisatie te berekenen van A_0 met eender welke methode. Vervolgens berekenen we $b = Q^* * b_0$. Met deze waarden kunnen we de iteratieve berekening starten die hieronder beschreven staat. De G_x matrices zijn givens transformaties om de toegevoegde rijen van K terug op nul te stellen om zo terug een bovendriehoeksmatrix R te bekomen waarin de nieuwe waarden in verwerkt zijn.

$Q^{(0)} * R^{(0)} = A_0$
$b = Q^{(0)*} * b_0$
for *i = 1 to k* **do**
  $f = n * d$
  $R^{(i)} = G_f^* * ... * G_1^* * \begin{bmatrix} R^{(i-1)} \\ K \end{bmatrix}$
  $R^{(i)} = R^{(i)}(: n, :)$
  $Q^{(i)} = I * G_1 * ... * G_f$
  $b^{(i)} = Q^{(i)*} * \begin{bmatrix} b^{(i-1)} \\ c \end{bmatrix}$
  $b^{(i)} = b^{(i)}(: n, :)$
end

Aan het einde van elke iteratie is het mogelijk om $x^{(i)}$ te berekenen door achterwaardse substitutie toe te passen op de vergelijking $R^{(i)} * x^{(i)} = b^{(i)}$. Omdat na elke iteratie maar $R^{(i)}$ en $b^{(i)}$ opgeslagen moet worden is het duidelijk dat het gebruikte geheugen niet toeneemt. Omdat de grootte van de matrices R en b niet toeneemt neemt het rekenwerk ook niet toe met elke iteratie. Voor het berekenen van een Givens-transformatie zijn 2 delingen, 2 vermenigvuldigen, een optelling en een vierkwantswortel nodig. Er moeten n*d Givens-transformaties berekend worden per iteratie. Een vermenigvuldiging met een rotatiematrix van grootte n+d zoals in dit geval vraagt 4*(n+d) vermenigvuldigingen en 2*(n+d) optellingen. Er gebeuren 2*n*d -1 van die matrix vermenigvuldiginge per iteratie. Tot slot gebeurt er nog voor de berekening van de $b^{(i)}$ een matrix verme-

Methode	1	2	3	5	10	20	25
QRzonder	0.01836	0.02411	0.03212	0.06223	0.0106	1.498×10^{-6}	2.059×10^{-8}
QRrayleigh	0.01836	1.234×10^{-5}	9.558×10^{-15}				
QRwilkinson	0.01836	9.833×10^{-7}	1.138×10^{-18}				

Tabel 1: convergentie QR methodes voor eigenwaardenberekening

nigvuldiging waarvoor $(n+d)^2$ vermenigvuldigingen gebeuren en $(n+d-1)^2$ optellingen.

- $n * d * 2$ delingen
- $n * d$ vierkantswortels
- $n * d * 2 + 2 * (n+d) * (2 * n * d - 1) + (n+d-1)^2 \approx 4 * n^2 * d$ optellingen
- $n*d*4+4*(n+d)*(2*n*d-1)+(n+d)^2 \approx 8*n^2*d$ vermenigvuldigingen

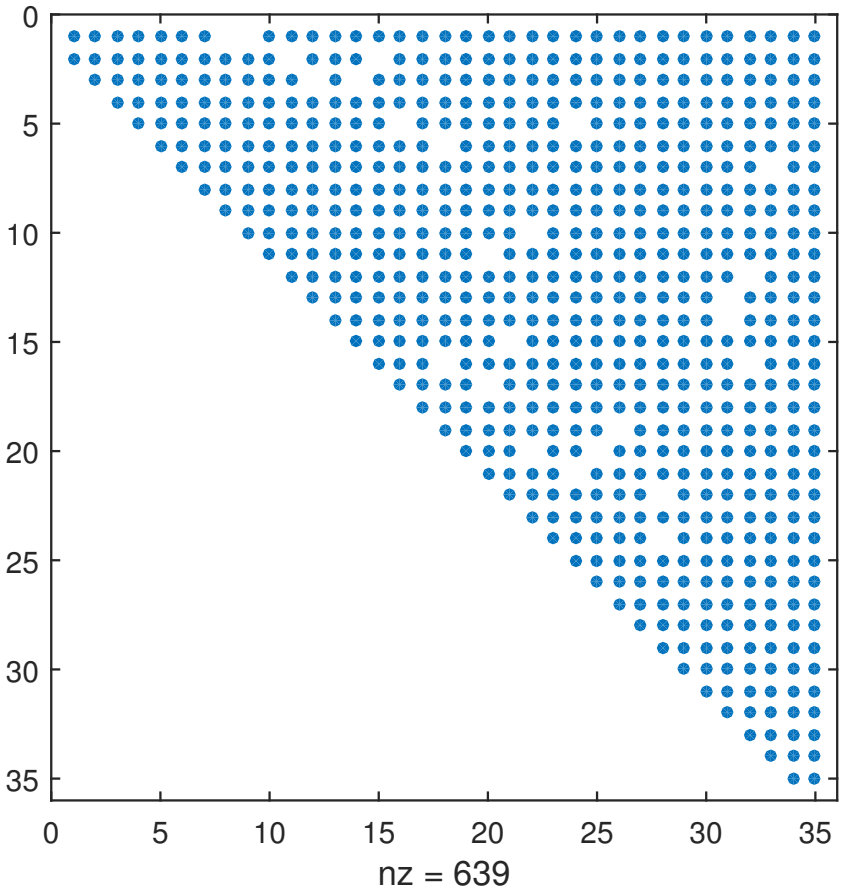
Opgave 3

Alvorens de QR methode, al dan niet met shifts, toe te passen op de matrix, moeten we hem omzetten naar Hessenberg vorm. Deze omzetting kost ons éénmaal $O(n^3)$ rekenwerk.
Omdat de Hessenberg vorm al bijna bovendriehoeks is, kan de QR factorisatie in minder stappen uitgevoerd worden, namelijk in $O(n^2)$ in plaats van in $O(n^3)$. Deze factorisatie moet in iedere iteratie van de QR methode uitgevoerd worden, dus de vermindering in rekenwerk is erg significant.

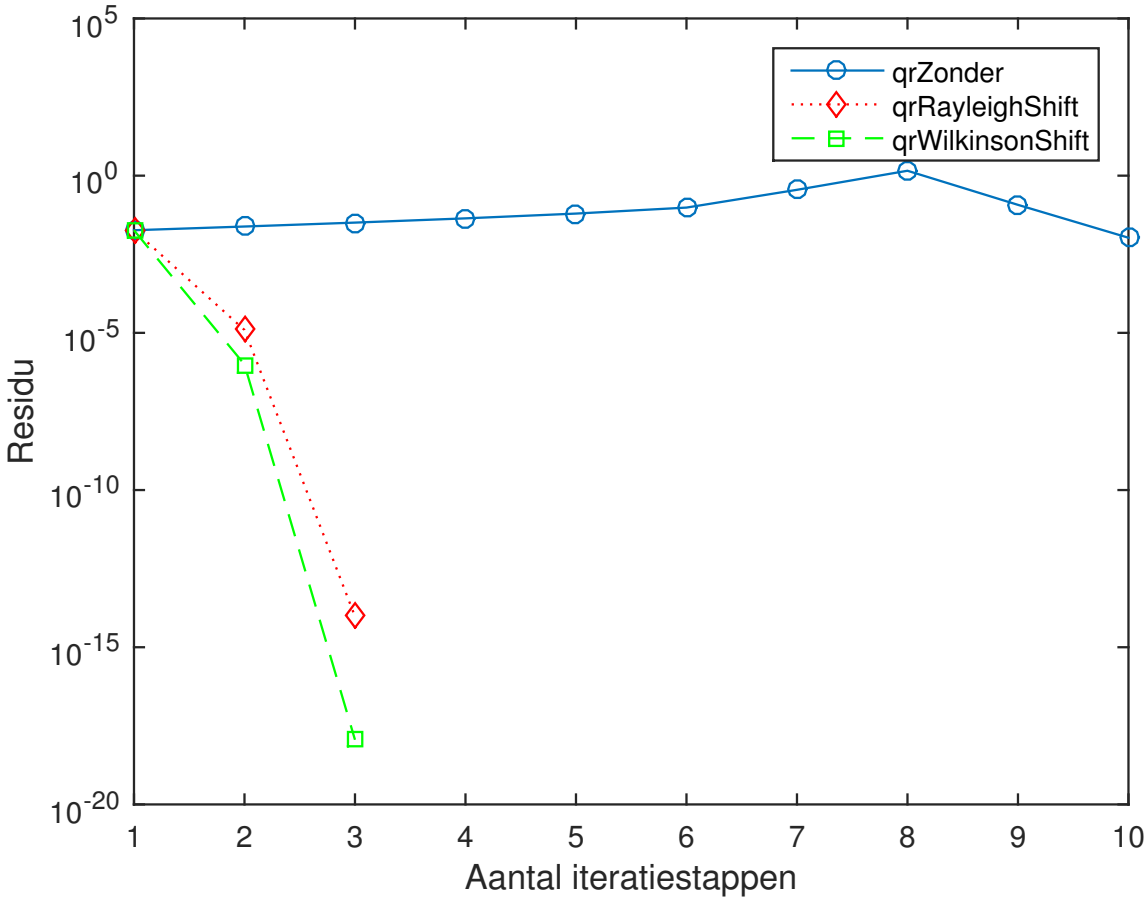
Opgave 4

De verwachting vanuit de theorie is dat de QR methode zonder shifts traag convergeert, zoals power- iteratie. De QR methode met shifts zou in het slechtste geval niet(Rayleigh shifts) of kwadratisch (Wilkinson shifts) convergeren. Hieronder worden het benodigde aantal iteratiestappen per methode in grafieken tabelvorm weergegeven, samen met de fout op het tussenliggende resultaat. We merken op dat zowel de Rayleigh Shift als de Wilkinson Shift zorgen voor een kubische convergentie (aantal beduidende cijfers $*3$ in iedere stap). De Wilkinson Shift heeft in dit geval een snellere convergentie door een (toevallig) goedgekozen eerste shiftwaarde. De convergentiefactor voor de eerste stap van de Wilkinson shift methode is hier 3.46, voor de Rayleigh shift methode is hij 2.827 en voor de QR methode zonder shift is hij zelfs kleiner dan 1. Het valt op dat de QR methode zonder shift niet lijkt te convergeren op deze grafiek. Dit is wel zo, maar pas na een groot aantal stappen (niet weergegeven). Na 40 iteratiestappen is de machinenauwkeurigheid bereikt.

Een aantal zaken zijn nog op te merken:



Figuur 1: De structuur van de Hessenberg- vorm van mat1



Figuur 2: De fout op het berekenen van één eigenwaarde.

- Voor de 2 methoden die gebruik maken van shifts, wordt in iedere stap een aantal keren een geshifte inverse iteratie gedaan, om zo het element op positie (k, k) te laten convergeren naar de $k - de$ eigenwaarde. Opmerkzaam is dat de $i - de$ eigenwaarde met $i = 1, \ldots, k - 1$ als k het aantal eigenwaarden is dat we nog moeten berekenen, ook al genaderd wordt, terwijl het algoritme nog bezig is eigenwaarden verderop in de matrix te berekenen m.b.v shift- iteraties. Hoe kleiner i, hoe minder sterk dit effect. De oorzaak hiervan is dat de QR methode, naast inverse iteratie, ook simultane iteratie toepast.

- De methode met Wilkinson- shifts doet er 66 stappen over om alle eigenwaarden van mat1 te vinden tot op machineprecisie. De methode met Rayleigh- shifts doet er 83 stappen over, en die zonder shifts maar liefst 686 stappen. Wilkinson shifts zijn beter dan de Rayleigh shifts, omdat zij rekening houden met het symmetrische eigenwaarden. Rayleigh shifts kunnen in zo'n gevallen zorgen voor een trage (of zelfs onbestaande) convergentie.