Modellering en simulatie verslag: Practicum 1

Francis Duvivier

December 2013

1 Oplossing opgave 1

Dit tonen we aan door het teschrijven als: $G_{23}^T(G_{23}^T A^T)^T = G_{23}^T(G_{23}^T A)^T$ want A=A^T. Stel we noemen $(G_{23}^T A)^T B$. We weten dat als we c en s van G_{23} berekenen adhv a_{12} en a_{13} dat er op plaats (1,3) van G_{23}^T B of $G_{23}^T A$ een nul kan komen. Als we dus eerst op (3,1) een nul creeren met een givensrotatie op A, en dan het resultaat transponeren dan krijgen we een de matrix B met een nul op (1,3) Als de daar op opniew dezelfde givensrotatie doen zal er een nul komen op (3,1), want de 1e kolom van B is hetzelfde als de eerste kolom van A omdat de eerste rij van A niet aangetast was door de givens rotatie, en aangezien nu ook weer de 1e rij onaangetast blijft zal de nul op (1,3) blijven staan.

Het resultaat is dus dat we matrix kunnen creëren met nullen op (1,3) en (3,1) als we c en s van G_{23} berekenen adhv a_{12} en a_{13}

2 Oplossing opgave 2

De oplossing van deze opgave is geïmplementeerd als P1.opl2(mode) met mode 1,2 en 3 voor respectievelijk gewoon QR, QR met shift en QR met alternatieve shift.

3 Oplossing opgave 3

De matlab code van de oplossing kan men vinden als P1.opl3 en de oplossingen zijn geplot in figuur 1. We zien dat $a_{20,20}$ het snelste convergeert voor QR met alternatieve shift en het traagste voor de gewone QR. Voor $a_{19,19}$ zien we echter dat de QR met gewone shift net iets sneller convergeert dan die met alternatieve shift, maar het verschil is niet groot en kan toevalsmatig zijn. Ook zien we dat het vershil in convergentiesnelheid opvallend veel kleiner is bij $a_{19,19}$ omdat de shift-optimalisaties gericht zijn op het laten convergeren van $a_{20,20}$.

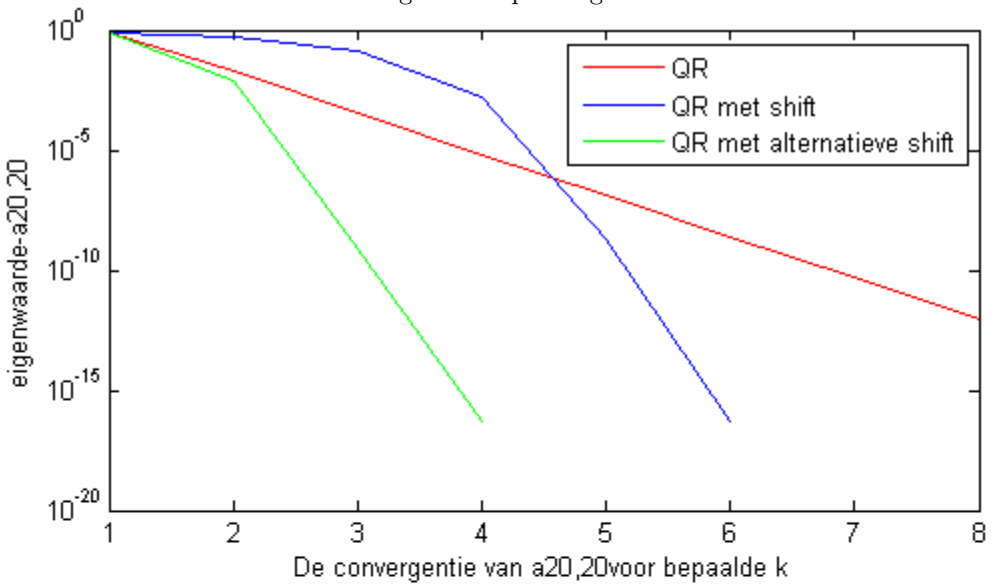
4 Oplossing opgave 4

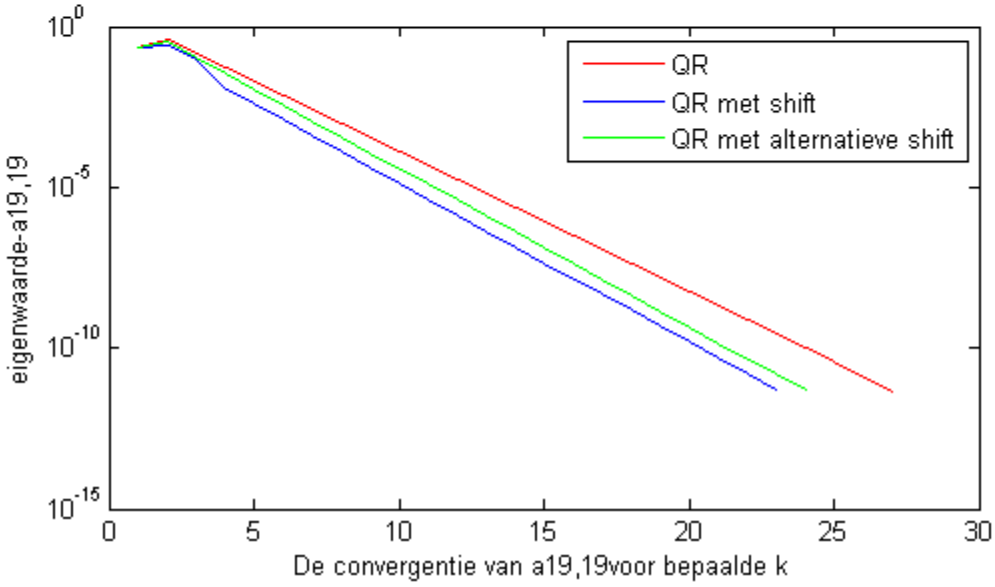
De matlab code van de oplossing kan men vinden als P1.opl4 en de oplossingen zijn geplot in figuur 2. We zien hier duidelijk dat, zoals de theorie zegt, $a_{20,19}$ heel snel convergeert bij QR met shift en nog sneller met QR met alternatieve shift maar dat dit niet veel sneller is bij bv $a_{19,18}$ omdat de shift niet verschuift nadat $a_{20,19}$ praktisch gelijk aan nul geworden is. We zien duidelijk op de plots dat $a_{20-k,19-k}$ voor hogere k trager convergeert.

5 Oplossing opgave 5

De matlab code van de oplossing kan men vinden als P1.opl5 en de oplossingen zijn geplot in figuur 3. We zien hier duidelijk dat, zoals de theorie zegt,dat ook voor de andere waarden op de diagonaal de methoden met shift veel sneller convergeert dan het gewone QR algoritme. Ook zie je dat het algoritme bij de kleinste waarde op de diagonaal, dus vanonder, begint en dan telkens naar boven gaat waardoor de waarden vanonder in de matrix minder iteraties nodig hebben om naar de eigenwaarde te convergeren.

Figure 1: Oplossing 3





6 Oplossing opgave 6

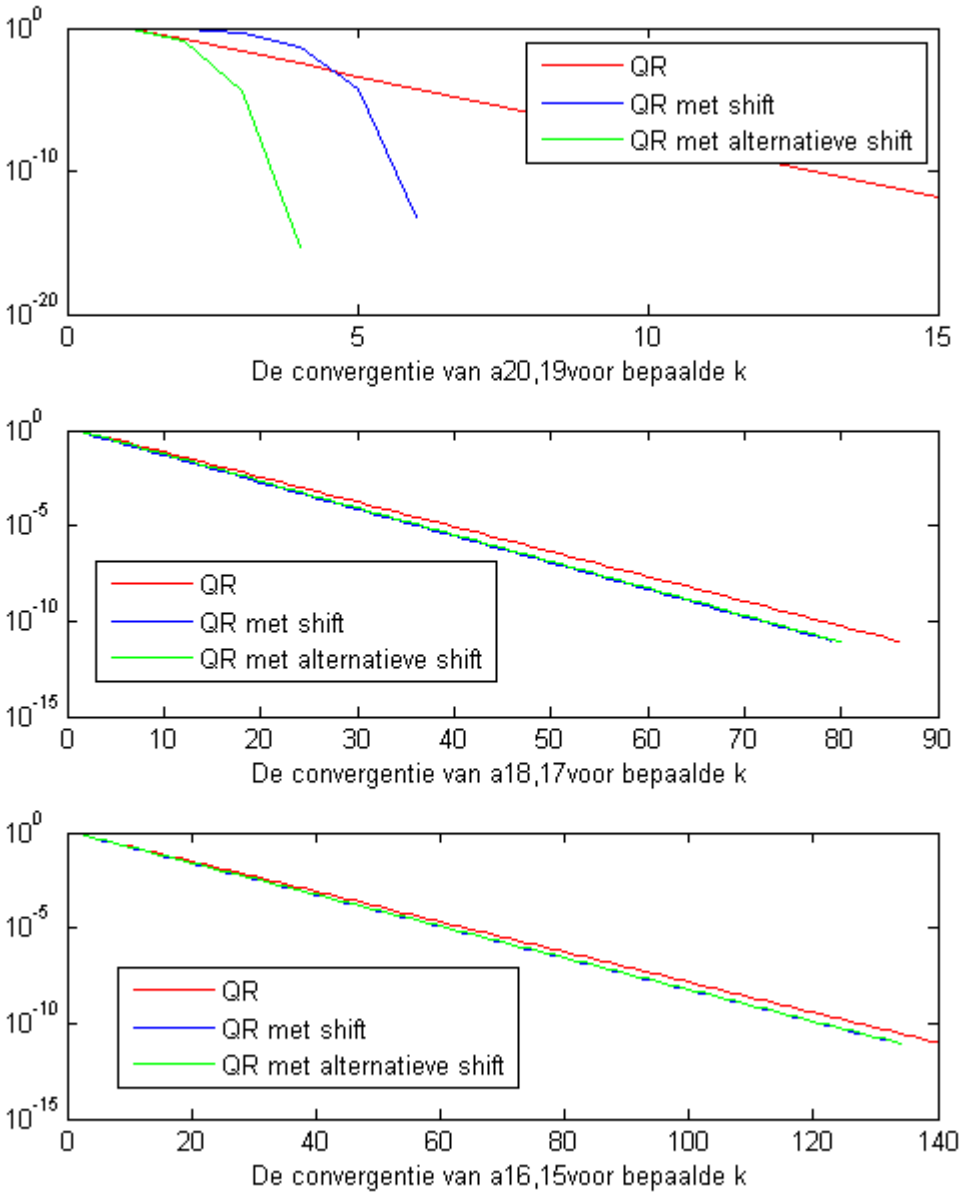
6.1 a)

Het gevraagde programma vindt u in de bijlage met naam opl6a.m.

6.2 b)

De opplossing van deze deelopdracht is geplot in figuur 4, de matlab code voor de generatie vindt u in opl6draw.m. Voor de methode met singuliere waarde ontbinding is de gekozen k rond de 210 en voor de methode met QR-factorisatie is k rond de 450. Opvallend is dat de relatieve fout bij het QR-beeld groter is maar naar mijn mening het beeld er niet veel slechter uitziet. Op de onderstaande figuur kan u de benadering

Figure 2: Oplossing 4

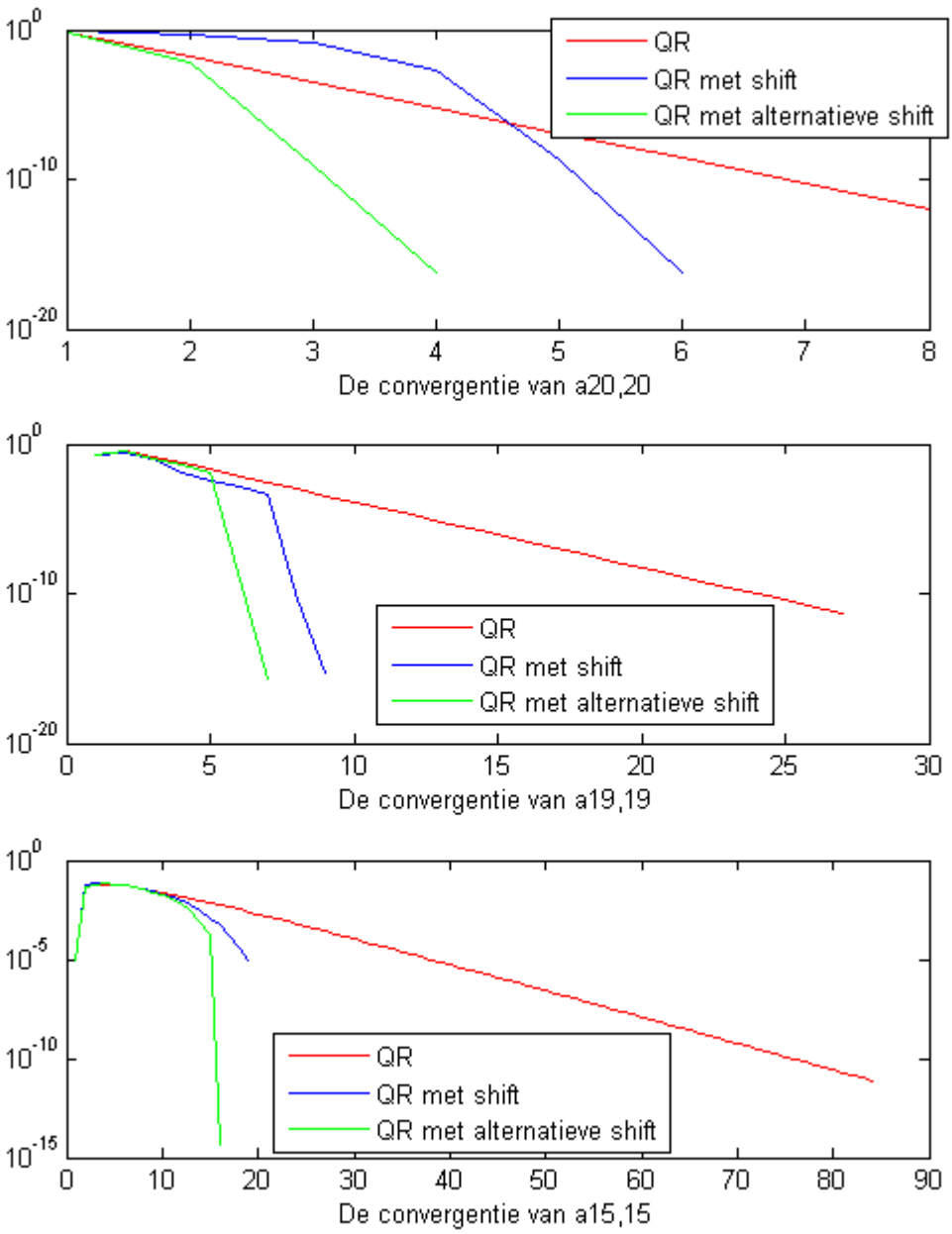


van de beelden zien met de waarde van k en de relatieve fout erbij.

6.3 c)

De opplossing van deze deelopdracht is geplot in figuur 5(2-norm) en 6(Frobenius-norm), de matlab code voor de generatie vindt u in opl6draw.m. Het is consistent met de theorie dat de relatieve fout van de k-de rang benadering gemaakt met behulp van singuliere waardeontbinding overeenstemt met het (k+1)e diagonaal element op de diagonaalmatrix die hierbij bekomen wordt. Ook is het zo dat voor de Frobenius-norm voor rang k geldt dat de de deze overeenstemt met de som van de k-de en alle kleinere singuliere waarden.

Figure 3: Oplossing 5



6.4 d)

De methode met QR-factorisatie is minder efficiënt voor geheugenbesparing dan die met singuliere waardeontbinding. Dit is dus duidelijk beter bij singuliere waardeontbinding. voor grote waarden van k is voor dezelfde k de geheugen besparing van QR ongeveer dezelfde als die voor SVD. Maar aangezien we bij de SVD-methode een veel kleinere waarde van k nodig hebben om een evengoed uitziende benadering te bekomen is SVD in dat opzicht veel efficienter. Over het rekenwerk is het duidelijk dat bij singuliere waarde-ontbinding veel

Figure 4: Oplossing 6 b)

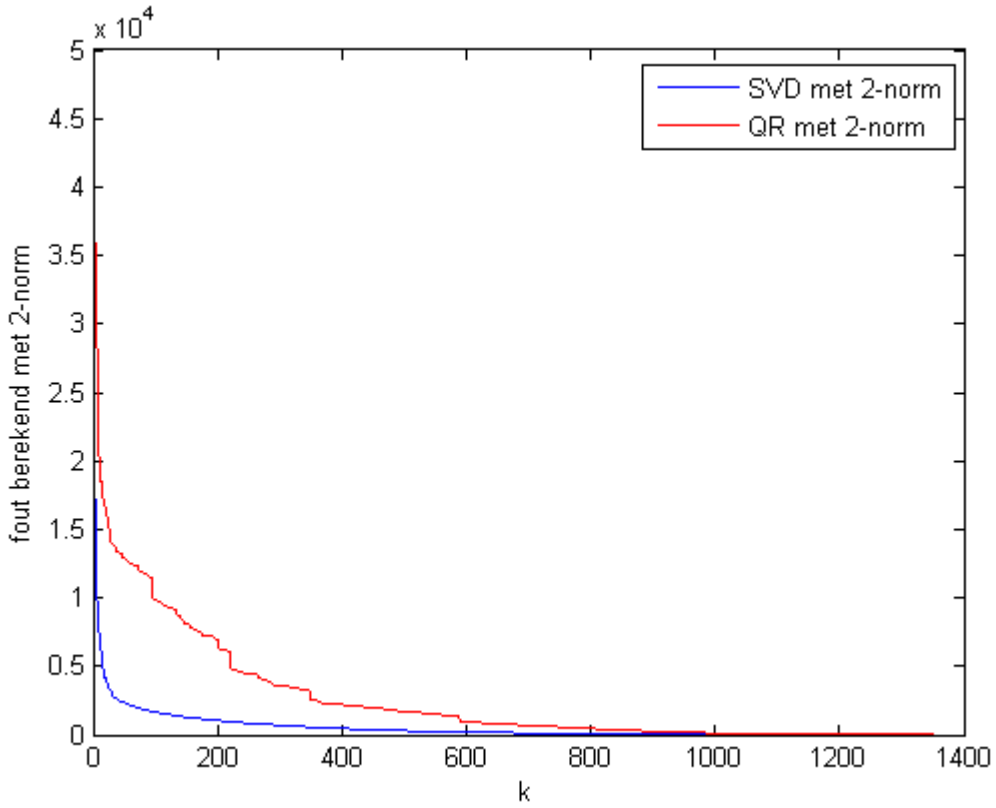
svd

k=218, 2-norm fout=970

qr

k=456, 2-norm fout=1936

Figure 5: Oplossing 6 c) voor 2-norm



meer rekenwerk gedaan moet worden omdat er daarbij k keer een ontbinding berekend moet worden. Bij QR-factorisatie moet er maar 1 keer een factorisatie berekend worden voor eender welke k.

Figure 6: Oplossing 6 c) voor Frobenius-norm