Practicum NMB : Eigenwaardenproblemen

Mijn Naam

vrijdag 25 april 2014

Opgave 1

Zij $P \in \mathbb{R}^{n\times n}$ en $x \in \mathbb{R}^n$, dan $Px = \frac{x+Fx}{2}$ met $F \in \mathbb{R}^{n\times n}$ de matrix die de elementen x_1 en x_2, x_3 en x_4, ... omwisselt bij de vermenigvuldiging Fx. Dan:

$$Px = \frac{1}{2}(I+F)x \Leftrightarrow P = \frac{1}{2}(I+F) \tag{1}$$

* P is een projector indien P vierkant(gegeven) en P idempotent ($P^2 = P$). We hebben:

$$P^2 = \frac{1}{4}(I^2 + 2F + F^2) \tag{2}$$

Aangezien F het eerste en tweede element, het derde en vierde element enz... van een vector omwisselt, doet een tweede vermenigvuldiging met F deze permutatie terug teniet. Dus: $F^2 = I$. Dan:

$$P^2 = \frac{1}{4}(I + 2F + I) = \frac{1}{4}(2I + 2F) = \frac{1}{2}(I+F) = P \tag{3}$$

* P is een orthogonale projector $\Leftrightarrow P = P^*$. We hebben:

$$P^* = \frac{1}{2}(I+F)^* = \frac{1}{2}(I^* + F^*) \tag{4}$$

We kunnen nagaan dat:

$$F = \begin{bmatrix} 0 & 1 & & & & & \\ 1 & 0 & & & & & \\ & & 0 & 1 & & & \\ & & 1 & 0 & & & \\ & & & & \ddots & & \\ & & & & & 0 & 1 \\ & & & & & 1 & 0 \end{bmatrix} = F^* \tag{5}$$

zodat:

$$P^* = \frac{1}{2}(I^* + F^*) = \frac{1}{2}(I+F) = P \tag{6}$$

waaruit volgt dat P een orthogonale projector is.

Opgave 2

Een Householder transformatiematrix heeft de volgende structuur:

$$Q_k = \begin{bmatrix} I & \\ & F \end{bmatrix} \tag{7}$$

met I de $(k-1) \times (k-1)$ eenheidsmatrix en F een $(m-k+1) \times (m-k+1)$ unitaire Householder reflector. Deze structuur impliceert een karakteristieke veelterm van de vorm:

$$det(\lambda I - Q_k) = (\lambda - 1)^k \cdot det(\lambda I - F) \tag{8}$$

Voor $k \geq 1$ hebben we dus steeds eigenwaarde 1 in het spectrum van Q_k met algebrasche multipliciteit k. Aangezien dit rechtstreeks voortkomt uit het diagonaal zijn van $\begin{bmatrix} I & 0 \end{bmatrix}^T$, is k eveneens de geometrische multipliciteit. Een diagonale matrix is immers nooit defectief. We kunnen dan k lineair onafhankelijke eigenvectoren vinden voor $\lambda = 1$. Dit is gemakkelijk te verifiren. Neem gewoon:

$$v_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad v_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad v_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \quad \dots \quad v_k = \dots \tag{9}$$

Dan $Q_k v_j = v_j \quad (\forall j \leq k)$. Deze vectoren spannen de ruimte $\mathbb{R}^k$ op waartoe alle vectoren van de eerste k rijen van een matrix $A \in \mathbb{R}^{m \times n}$ behoren (aangevuld tot dimensie m met nulle. Vermits elke kolomvector van A tot en met rij k (en vervolgens aangevuld met nullen tot dimensie m) een lineaire combinatie is van bovenstaande v_j zal een vermenigvuldiging met Q_k neerkomen op een vermenigvuldiging met $\lambda = 1$. Householder laat dus de eerste k rijen van de matrix A ongemoeid zoals men hoopt te verwachten van een Householder transformatiematrix.

De Householder reflector F zal dan zorgen voor de nodige nullen zonder voorheen aangebrachte nullen te vernietigen.

$\kappa \setminus$ n	10	100	1000
1	0.0034s	0.0402s	71.6416s
10^4	0.0026s	0.0342s	66.5048s
10^8	0.0126s	0.0312s	66.1448s

Tabel 1: Householder_explicit

$\kappa \setminus$ n	10	100	1000
1	0.0044	0.0194s	0.0601s
10^4	0.0027s	0.0188s	0.0758s
10^8	0.0154s	0.0137s	0.0590s

Tabel 2: Householder_implicit Apply_Q

Opgave 3

a) De functies Householder_explicit, Householder_implicit en Apply_Q zijn bijgevoegd als MATLAB-bestanden.

b) Indien we een willekeurig stelsel oplossen met Householder_explicit en we variren de dimensie n en het conditiegetal κ, verkrijgen we de resultaten uit Tabel 1. Hetzelfde kunnen we doen voor Householder_implicit gebruik makend van Apply_Q, wat de resultaten uit Tabel 2 oplevert.

Het lijkt erop dat het conditiegetal weinig invloed heeft op de rekentijd. De afmetingen van het matrixprobleem daarentegen wel. $Ax = b$ oplossen m.b.v. Householder_explicit lijkt een zeer slecht idee te zijn bij matrices van een groter formaat. Householder_implicit in combinatie met Apply_Q heeft hier dan weer geen problemen mee.

Opgave 4

Opgave 5

Opgave 6

Opgave 7

Opgave 8

Opgave 9

Het is mogelijk om Jacobi rotaties in parallel uit te voeren door ze op disjuncte rij-/kolomparen. te laten inwerken. Zij A een symmetrische matrix, dan kunnen we dit parallelisme voorstellen door twee rotaties in een enkele rotatiematrix te

steken.

Opgave 10

Het volgende algoritme in pseudocode brengt nullen aan op achtereenvolgens (1,2), (1,3), ... , (1,n); (2,3), (2,4), ... , (2,n); ... ; (n-2,n-1), (n-2,n) ; (n-1,n) en op alle elementen hiermee symmetrisch t.o.v. de hoofddiagonaal.

$Jacobi(A, tolerance)$ {

$n = width(A)$;
$J = I(n, n)$;
$imprecision = 100$;

$while\,(tolerance < imprecision)$ {
$for\,(i = 1;\ i = n - 1;\ i{+}{+})$ {
$for\,(j = i + 1;\ j = m;\ j{+}{+})$ {
$a = A(i, i)$;
$b = A(i, j)$;
$\theta = (1/2) \cdot atan(2d/(b - a))$;
$s = sin(\theta)$;
$c = cos(\theta)$;
$J_k = I(n, n)$;
$J_k(i, i) = c$;
$J_k(j, j) = c$;
$J_k(i, j) = s$;
$J_k(j, i) = -s$
$J = J \cdot J_k$;
$A = (J_k)^T \cdot A \cdot J_k$;
}
}

$offdiagnorm = (sumofsquares(A) - sumofdiagonalsquares(A))^{1/2}$;
$imprecision = |offdiagsum/maxdiagonal(A)|$;
}

$D = A$;
$V = J$;
$return\ V,\ D$;

De implementatie in MATLAB volgt bovenstaande structuur en is te vinden in het bijgeleverde bestand 'jacobi.m'.

Opgave 11

De cursus geeft aan dat de de convergentie van een symmetrische matrix met dimensie $m \leq 1000$ typisch gebeur t in minder dan tien sweeps. Dit lijkt te kloppen. De bijgevoegde matrix mat1 convergeert in 8 sweeps. De norm van de niet-diagonaalelementen, gedeeld door het grootste diagonaalelement, wordt dan immers volledig nul in MATLAB (m.a.w. de fout is van $O(\epsilon_{mach})$).

Voor een willekeurige symmetrische tridiagonale matrix is de rekenkost van dit algoritme dezelfde als van een willekeurige symmetrische niet-tridiagonale matrix, aangezien het algoritme met de structuur van de matrix, afgezien van de noodzakelijk symmetrie, geen rekening houdt. Indien we een variant zouden maken, enkel voor tridiagonale matrices, gebeurt een sweep in m-1 iteraties, nl. het aantal elementen op de diagonaal juist boven of onder de hoofddiagonaal. De rekenkost is dan $O(m)$.