

# Optimaal onderbepaald

practicum numerieke methoden voor grote lineaire stelsels (wi4010)

Sebastiaan, 9121164, sebastia@ch.twi.tudelft.nl

25 juni 2003

## 1 Onderbepaalde stelsels

In deze opgave beschouwen wij het stelsel

$$Ax = y \tag{1}$$

waarin  $A$  een  $m \times n$  matrix is met  $m \leq n$  met volle rijrank. Dit stelsel is dus onderbepaald, wat evenveel wil zeggen dat er meerdere oplossingen mogelijk zijn. Deze oplossingen kunnen uit de hand lopen, wat niet voor elk probleem gewenst is. We kunnen als extra eis opleggen dat we de kleinste oplossing willen hebben, ofwel:

$$\|\tilde{x}\|_2 = \min_{Ax=y, x \in \mathbb{R}^n} \|x\|_2 \tag{2}$$

We kunnen aantonen dat  $\tilde{x}$  gegeven wordt door

$$\tilde{x} = A^T(AA^T)^{-1}y \tag{3}$$

Stel dat  $\tilde{x}$  inderdaad het minimum is en stel dat  $z$  een andere oplossing is waarvoor geldt dat  $Az = y$  (het stelsel is onderbepaald, dus meerdere oplossingen zijn mogelijk). Zij dan  $\delta = z - \tilde{x}$ . Dan is

$$\begin{aligned} \|z\|_2^2 &= \|\tilde{x} + \delta\|_2^2 \\ &= \langle \tilde{x} + \delta, \tilde{x} + \delta \rangle \\ &= \langle \tilde{x}, \tilde{x} \rangle + 2 \langle \tilde{x}, \delta \rangle + \langle \delta, \delta \rangle \\ &= \|\tilde{x}\|_2^2 + \|\delta\|_2^2 + 2\tilde{x}^T \delta \\ &= \|\tilde{x}\|_2^2 + \|\delta\|_2^2 + 2y^T (AA^T)^{-1}A(z - \tilde{x}) \\ &= \|\tilde{x}\|_2^2 + \|\delta\|_2^2 \\ &\geq \|\tilde{x}\|_2^2 \end{aligned} \tag{4}$$

Aangezien  $A\tilde{x} = y$  en  $Az = y$  is  $A(z - \tilde{x}) = 0$ . Hieruit volgt dat  $\tilde{x}$  inderdaad de minimale oplossing is.

## 2 Choleski als oplosmethode

We kunnen vergelijking 3 oplossen door eerst het stelsel

$$y = AA^T w \quad (5)$$

op te lossen en vervolgens

$$\tilde{x} = A^T w \quad (6)$$

te bepalen.

Voor vergelijking 5 zou de Cholesky decompositie  $AA^T = LL^T$  gebruikt kunnen worden.  $AA^T$  is namelijk symmetrisch ( $(AA^T)^T = AA^T$ ) en positief definit ( $x^T AA^T x = (x^T A)(x^T A)^T > 0 \forall x$ ). Dit is echter niet de meest robuuste manier, ervan uitgaande dat deze decompositie numeriek bepaald wordt op computers die een beperkte nauwkeurigheid hebben. Stel dat  $A$  de volgende vorm heeft:

$$A = \begin{pmatrix} 1 & \varepsilon & 0 & 0 \\ 1 & 0 & \varepsilon & 0 \\ 1 & 0 & 0 & \varepsilon \end{pmatrix} \quad (7)$$

Wanneer we  $AA^T$  berekenen krijgen we

$$AA^T = \begin{pmatrix} 1 + \varepsilon^2 & 1 & 1 \\ 1 & 1 + \varepsilon^2 & 1 \\ 1 & 1 & 1 + \varepsilon^2 \end{pmatrix} \quad (8)$$

We zien dat wanneer  $\varepsilon$  dicht bij de machine precisie ligt (en  $\varepsilon^2$  dus buiten het bereik van de machine precisie valt),  $AA^T$  een singuliere matrix wordt.

De hoeveelheid werk die gedaan moet worden is de som van de volgende stappen:

1. bepaling  $AA^T$
2. Choleski decompositie
3. oplossen van de vergelijking  $LL^T w = y$
4. uitrekenen van  $\tilde{x} = A^T w$

Als we aannemen dat  $A$  elementsgewijs is opgeslagen kunnen we  $B = AA^T$  bepalen met

$$\begin{aligned} &\text{for } i = 1 \text{ to } m \\ &\quad \text{for } j = 1 \text{ to } m \\ &\quad\quad b_{ij} = \sum_{k=1}^n a_{ik} a_{jk} \\ &\quad \text{end} \\ &\text{end} \end{aligned} \quad (9)$$

hetgeen resulteert in  $2nm^2$  operaties. De matrix  $B$  is dan een  $m \times m$  matrix.

Voor de Choleski ontbinding kunnen we de kolomversie gebruiken (zie [2]). Dit algoritme overschrijft de matrix  $B$ . Hierna definiëren we  $L$  met  $l_{ij} = b_{ij}$  voor  $i \geq j$ .

$$\begin{aligned}
 &\text{for } k = 1 \text{ to } m \\
 &\quad b_{kk} = \sqrt{b_{kk} - \sum_{p=1}^{k-1} b_{kp}^2} \\
 &\quad \text{for } i = k + 1 \text{ to } m \\
 &\quad\quad b_{ik} = \frac{b_{ik} - \sum_{p=1}^{k-1} b_{ip}b_{kp}}{b_{kk}} \\
 &\quad \text{end} \\
 &\text{end}
 \end{aligned} \tag{10}$$

Dit algoritme kost ongeveer  $\frac{1}{3}m^3$  operaties.

Het oplossen van  $LL^T w = y$  geschied nu door het oplossen van de stelsels  $Lz = y$  en  $L^T w = z$  met respectievelijk voorwaarde eliminatie en terugsubstitutie. Beide algoritmes zijn hieronder gegeven en kosten beide  $m^2$  operaties.

$$\begin{aligned}
 &\text{for } i = 1 \text{ to } m \\
 &\quad z_i = \frac{y_i - \sum_{j=1}^{i-1} z_j l_{ij}}{l_{ii}} \\
 &\text{end}
 \end{aligned} \tag{11}$$

$$\begin{aligned}
 &\text{for } i = m \text{ to } 1 \\
 &\quad w_i = \frac{z_i - \sum_{j=i+1}^m w_j l_{ji}}{l_{ii}} \\
 &\text{end}
 \end{aligned} \tag{12}$$

Tenslotte rest ons de berekening van  $\tilde{x} = A^T w$ . Dit eenvoudige algoritme kost ons  $2mn$  operaties.

$$\begin{aligned}
 &\text{for } i = 1 \text{ to } n \\
 &\quad \tilde{x}_i = \sum_{j=1}^m a_{ji} w_j \\
 &\text{end}
 \end{aligned} \tag{13}$$

Het totaal aantal operaties dat gedaan moet worden komt dus neer op  $2nm^2 + \frac{1}{3}m^3 + 2m^2 + 2mn$ .

Voor de gemaakte foutanalyse gebruiken we de gevonden analyses uit [2]. We gaan uit van een getalrepresentatie van  $t$  cijfers in de mantisse en grontal  $\beta$ . Voor de meeste computers en programma's is  $\beta = 2$  en  $t = 16$  (Matlab bijvoorbeeld gebruikt  $t = 52$ ). Wanneer we een getal  $a$  in machinerepresentatie hebben is de fout in  $\tilde{a} = a + a\varepsilon$  gelijk aan  $|\varepsilon| \leq u = \frac{1}{2}\beta^{t-1}$  wanneer we ervan uitgaan dat de machine het getal correct afrond.

Voor de eerste stap, het bepalen van  $AA^T$  is de fout  $|E_1| \leq 1.01nu|A|^2$ . We hebben dan  $fl(AA^T) = AA^T + E_1$ . Het maken van de Choleski ontbinding en het doen van voorwaartse eliminatie en terugsubstitutie op deze matrix geeft:

$$(AA^T + E_1 + E_2)w = y \quad (14)$$

met  $\|E_2\|_2 \leq c(n)u\|2\|AA^T + E_1$ . De laatste stap gaat om het uitrekenen van

$$\tilde{x} = A^T(E_1 + E_2)w \quad (15)$$

die een uiteindelijke fout oplevert van ongeveer  $\|E_c\|_2 \leq 1.01nu(\|A\|_2 + \|E_1\|_2 + \|E_2\|_2)$ .

### 3 Oplossen met behulp van orthogonaliteit

We beschouwen wederom vergelijking 3. Stel dat we een  $A = LQ$  decompositie konden maken met  $L$  een  $m \times m$  onderdriehoeksmatrix en  $Q$  een  $m \times n$  matrix waarvoor geldt dat  $QQ^T = I$ . Wanneer we  $A = LQ$  gebruiken dan versimpeld de vergelijking tot

$$\begin{aligned} \tilde{x} &= A^T(AA^T)^{-1}y \\ &= Q^T L^T (LQ Q^T L^T)^{-1}y \\ &= Q^T L^T L^{-T} L^{-1}y \\ &= Q^T L^{-1}y \end{aligned} \quad (16)$$

hetgeen een aanzienlijke vereenvoudiging bewerkstelligd.

#### 3.1 Householder transformatie

De Householder transformatie voor een  $m \times n$  matrix is praktisch hetzelfde als voor een vierkante matrix. We maken stapsgewijs van  $A$  een onderdriehoeksmatrix met  $a_{ij} = 0$  voor  $j > i$ . We kunnen in  $m$  stappen matrix  $A$  omvormen. Voor  $A_k$  transformeren we de  $k$ -de rij van  $A_{k-1}$  zodanig dat  $a_{kj} = 0$  voor  $j > k$ . Dit doen we door een orthogonale matrix  $Q_k$  te bepalen. Dan maken we

$$A_{k+1} = A_k Q_k \quad (17)$$

Doen we dit voor alle rijen dan verkrijgen wij:

$$[L \ 0] = A_{m+1} = A Q_1 Q_2 \dots Q_m = A \tilde{Q} \quad (18)$$

We nemen dan  $L$  als de eerste  $m$  kolommen van  $A_{m+1}$  en voor  $Q$  de eerste  $m$  rijen van  $\tilde{Q}$ . Immers,  $[L \ 0] = A \tilde{Q} \sim A = [L \ 0] \tilde{Q}^T$ . Merk op dat de verkregen  $A$  in het onderstaande algoritme niet meer de  $A$  is van de oorspronkelijke vergelijking. Om de leesbaarheid te behouden blijven we echter wel  $A$  schrijven voor de getransformeerde matrix.

Het algoritme voor een dergelijke Householder tranformatie is hetvolgende (zie [2]):

$$\begin{aligned}
 & \tilde{Q} = I_{n \times n} \\
 & \text{for } k = 1 \text{ to } m \\
 & \quad \alpha = 0 \\
 & \quad w = 0 \\
 & \quad M = \max\{|a_k|, \dots, |a_n|\} \\
 & \quad \text{for } i = k \text{ to } n \\
 & \quad \quad w_i = \frac{A_{ki}}{M} \\
 & \quad \quad \alpha = \alpha + w_i^2 \\
 & \quad \text{end} \\
 & \quad \alpha = \sqrt{\alpha} \\
 & \quad \beta = \frac{1}{\alpha(\alpha + |w_k|)} \\
 & \quad w_k = w_k + \text{sign}(w_k)\alpha \\
 & \quad \% \text{ de volgende 3 regels kunnen slimmer} \\
 & \quad K = I - \beta w w^T \\
 & \quad A = AK \\
 & \quad \tilde{Q} = \tilde{Q}K \\
 & \text{end}
 \end{aligned} \quad (19)$$

Dit algoritme kost ongeveer  $4n^3m + 2n^2m + 4nm - 2m^2$  operaties. Merk op dat de matrixvermenigvuldigingen het meeste werk kosten, namelijk  $2n^3$  per keer.

### 3.2 Snelle Householder

Het liefst zouden we deze vermenigvuldigingen willen vermijden. We maken van een vermenigvuldiging van twee vectoren en een matrix een volle matrixvermenigvuldiging. Als we nogmaals terugkijken naar de Householder transformatie dan kunnen

we enige versimpeling aanbrengen:

$$\begin{aligned} AQ &= A(I - \beta ww^T) \\ &= A - \beta(Aw)w^T \end{aligned} \quad (20)$$

Ofwel, we krijgen een matrix-vector vermenigvuldiging. Het mooie is dat we nu  $A$  direct kunnen overschrijven: we hebben dus geen extra geheugen nodig voor de orthogonale matrix  $K$ . Verder kunnen we rekening houden met het feit dat in de  $k$ -de stap de eerste  $k - 1$  elementen van  $w$  gelijk zijn aan 0.

We kunnen de laatste 3 regels van algoritme 19 weglaten en vervangen door het hierna beschreven algoritme. De matrix  $A$  wordt direct overschreven. Vervangen we in onderstaand algoritme  $A$  door  $\tilde{Q}$  en laten we  $p$  lopen tot en met  $n$  dan kunnen we ditzelfde algoritme gebruiken om de uiteindelijke  $Q$  te berekenen.

$$\begin{aligned} &\text{for } p = 1 \text{ to } m \\ &\quad s = 0 \\ &\quad \text{for } i = k \text{ to } n \\ &\quad\quad s = s + w_i a_{pi} \\ &\quad \text{end} \\ &\quad s = \beta s \\ &\quad \text{for } i = k \text{ to } n \\ &\quad\quad a_{pi} = a_{pi} - s w_i \\ &\quad \text{end} \\ &\text{end} \end{aligned} \quad (21)$$

Wanneer we dit algoritme gebruiken in combinatie met 19 dan hebben we ongeveer  $4nm^2 + 4n^2m - 2m^3 - 3\frac{1}{2}m^2 + 3nm$  operaties nodig. We zien dat we nu een orde minder hebben in het aantal operaties door de volle matrixvermenigvuldigingen te vermijden.

Nu zijn we in staat om vergelijking 16 op te lossen. We gaan  $L$  en  $Q$  niet expliciet uitrekenen, we kunnen door gebruik van een aangepast algoritme gewoon  $A$  en  $\tilde{Q}$  blijven gebruiken. We willen de matrix  $L$  uiteraard niet expliciet transformeren. We herschrijven:

$$\begin{aligned} \tilde{x} &= Q^T L^{-1} y \\ LQ\tilde{x} &= y \\ Lz &= y \end{aligned} \quad (22)$$

met  $z = Q\tilde{x}$  en gebruiken het algoritme voor voorwaartse eliminatie om  $z$  uit te rekenen, zie algoritme 11. Vervolgens berekenen we  $\tilde{x} = Q^T z$ , zie algoritme 13. Deze laatste twee stappen voegen nogmaals  $m^2 + 2mn$  aantal operaties toe waardoor het totaal aantal operaties neerkomt op  $4nm^2 + 4n^2m - 2m^3 - 2\frac{1}{2}m^2 + 5nm$ .



## 4.1 Foutschatting

Voor het schatten van de fout heb ik twee manieren kunnen bedenken. De eerste is het uitrekenen van  $A\tilde{\alpha}$  en deze vergelijken met  $q$ , ofwel  $\|A\tilde{\alpha} - q\|_2$ . Aangezien de matrix  $A$  niet meer doet dan het aftrekken van 2 elementen van  $\tilde{\alpha}$  kunnen we aannemen dat de afrondfouten die in deze stap gemaakt worden nihil zijn.

Een andere manier is devolgende: ik heb ontdekt dat de oplossing  $\tilde{\alpha}$  erg regelmatig is. Voor elk element geldt:

$$\tilde{\alpha}_i = \tilde{\alpha}_{i-1} + \frac{i-1}{n} \quad (29)$$

Probleem is nu het bepalen van  $\hat{\alpha}_1$ , met  $\hat{\alpha}$  de echte oplossing. Aangezien de matrix  $A$  erg netjes is ontstond het vermoeden dat  $A^T(AA^T)^{-1}$  misschien ook wel erg netjes zou kunnen zijn. Na wat experimenteren heb ik ontdekt dat de eerste rij van  $-(1-h) \dots - (1-(n-1))h$  loopt met tussenstappen van  $h$ . Op deze manier kunnen we het eerste element van  $\hat{\alpha}$  berekenen door (eerste rij van  $A$  met  $y$ ):

$$\begin{aligned} \hat{\alpha}_1 &= 0 \\ \text{for } i &= 1 \text{ to } n \\ \hat{\alpha}_1 &= \hat{\alpha}_1 - \left(1 - \frac{i}{n}\right) \frac{i}{n} \\ \text{end} \end{aligned} \quad (30)$$

Zodoende krijgen we met  $\|\tilde{\alpha} - \hat{\alpha}\|_2$  een andere foutschatting.

## 4.2 Choleski

Wanneer we Choleski als oplosmethode gebruiken dan kost ons dat  $2nm^2 + \frac{1}{3}m^3 + 2m^2 + 2mn$  operaties. Met  $m = n - 1$  komt dat neer op  $2\frac{1}{3}n^3 - n^2 + O(n)$ .

Voor het geheugengebruik hebben we de matrix  $A$  met  $n(n-1)$  elementen, de matrix  $AA^T$  met  $(n-1)(n-1)$  elementen die overschreven wordt door  $L$ . Voor het verder oplossen via voorwaartse eliminatie hebben we nog een extra vector nodig van  $n-1$  elementen. Deze gebruikt de vectoren  $\alpha$  en  $q$  ( $n$  en  $n-1$  elementen). Dit komt op een totaal van  $2n^2 + O(1)$  aantal elementen.

Voor  $n = 1000$  kost het Matlab  $2.3368 \cdot 10^9$  flops om tot een antwoord te komen. Dit ligt in de buurt van het verwachte aantal operaties. De gemaakte fout is  $\|A\tilde{\alpha} - q\|_2 = 2.785 \cdot 10^{-10}$  volgens de eerste manier en  $\|\tilde{\alpha} - \hat{\alpha}\|_2 = 2.544 \cdot 10^{-9}$  volgens de tweede manier. Merk op dat Matlab op deze computer met een precisie van  $2.2204 \cdot 10^{-16}$  heeft gerekend.

methode	$n$	flops	$\ A\tilde{\alpha} - q\ _2$	$\ \tilde{\alpha} - \hat{\alpha}\ _2$
Choleski	10	2495	$3.849 \cdot 10^{-15}$	$3.342 \cdot 10^{-15}$
Householder	10	4874	$1.447 \cdot 10^{-15}$	$2.104 \cdot 10^{-15}$
Choleski	100	2348450	$8.991 \cdot 10^{-13}$	$2.949 \cdot 10^{-12}$
Householder	100	4094249	$1.327 \cdot 10^{-13}$	$4.963 \cdot 10^{-13}$
Choleski	1000	$2.3368 \cdot 10^9$	$2.785 \cdot 10^{-10}$	$2.544 \cdot 10^{-9}$
Householder	1000	$4.0115 \cdot 10^9$	$1.227 \cdot 10^{-11}$	$1.311 \cdot 10^{-10}$

Tabel 1: De oplosmethoden vergeleken

### 4.3 Householder

Met behulp van orthogonaliteit kost het  $6n^3 - 6\frac{1}{2}n^2 + O(n)$  iteraties om tot een oplossing te komen.

Ook hier hebben we de matrix  $A$  met  $n(n-1)$  elementen en de vectoren  $\alpha$  en  $q$  met respectievelijk  $n$  en  $n-1$  elementen. De matrix  $A$  wordt door  $L$  overschreven, dus deze stap neemt geen extra geheugen in gebruik, behalve de vector  $w$  met  $n$  elementen. Wel wordt er een nieuwe matrix  $Q$  gemaakt met  $n^2$  elementen. Als laatste wordt nog een vector van  $n-1$  elementen gebruikt. Totaal hebben we dus  $2n^2 + 3n + O(1)$  elementen nodig.

Matlab doet er voor  $n = 1000$   $4.0115 \cdot 10^9$  operaties over. De gemaakte fout volgens de eerste manier is  $\|A\tilde{\alpha} - q\|_2 = 1.227 \cdot 10^{-11}$  en  $\|\tilde{\alpha} - \hat{\alpha}\|_2 = 1.311 \cdot 10^{-10}$  volgens de tweede manier.

### 4.4 Vergelijking

Het geheugengebruik van beide methoden is vergelijkbaar. Zoals verwacht kost de manier met behulp van Householder tweemaal zoveel operaties als met Choleski. Daarentegen is de fout wel 20 keer kleiner voor grote  $n$ . Gezien de orde van de gemaakte fout (oplossing is van orde  $O(100)$ ) kunnen we concluderen dat Choleski de voorkeur heeft. Zie ook tabel 1.

## Referenties

- [1] R.E. Cline en R.J. Plemmons.  $l_2$ -solutions to underdetermined linear systems. *SIAM Review*, 18(1):92–106, januari 1976.
- [2] C. Vuik. *Numerieke methoden voor grote lineaire algebraïsche stelsels*. TUDelft, 2003.



## A Choleski

```
1 clear all;
2 flops(0)
3 n=1000;
4 A=[-eye(n-1)+diag(ones(n-2,1),1) [zeros(1,n-2) 1]'];
5 y=1/n*[1:(n-1)]';
6 C=A; % voor controle
7
8 [m n]=size(A);
9
10 % bepaal AA'
11 B=zeros(m,m);
12 for i=1:m
13     for j=1:m
14         for k=1:n
15             B(i,j)=B(i,j)+A(i,k)*A(j,k);
16         end
17     end
18 end
19
20 % choleski
21 for k=1:m
22     s=0;
23     for p=1:k-1
24         s=s+B(k,p)^2;
25     end
26     B(k,k)=sqrt(B(k,k)-s);
27     for i=k+1:m
28         s=0;
29         for p=1:k-1
30             s=s+B(i,p)*B(k,p);
31         end
32         B(i,k)=(B(i,k)-s)/B(k,k);
33     end
34 end
35
36 % voorwaartse eliminatie
37 z=zeros(m,1);
38 for i=1:m
39     s=0;
40     for j=1:i-1
```

```

41         s=s+z(j)*B(i,j);
42     end
43     z(i)=1/B(i,i)*(y(i)-s);
44 end
45
46 % terugsstitutie
47 w=zeros(m,1);
48 for i=m:-1:1
49     s=0;
50     for j=i+1:m
51         s=s+w(j)*B(j,i);
52     end
53     w(i)=1/B(i,i)*(z(i)-s);
54 end
55
56 % vermenigvuldiging
57 x=zeros(n,1);
58 for i=1:n
59     for j=1:m
60         x(i)=x(i)+A(j,i)*w(j);
61     end
62 end
63
64 flops
65
66 % controle
67 q=zeros(n,1);
68 %q(1)=[(-(1-1/n):1/n:-((1-(n-1)*1/n)))]*y;
69 for i=1:n
70     q(1)=q(1)-(1-i/n)*i/n;
71 end
72
73 for i=2:n
74     q(i)=q(i-1)+(i-1)/n;
75 end
76
77
78 x;
79 fprintf('Gemaakte |A*x-y| fout is %e\n',norm(C*x-y));
80 fprintf('Gemaakte |x-q| is %e\n',norm(x-q));

```

## B Householder

```
1 clear all;
2 flops(0)
3 n=1000;
4 A=[-eye(n-1)+diag(ones(n-2,1),1) [zeros(1,n-2) 1]'];
5 y=1/n*[1:(n-1)]';
6 B=A; % voor controle
7
8 [m n]=size(A);
9 Q=eye(n,n);
10
11 % slim of dom
12 slim=1;
13
14
15 % LQ decompositie
16 for k=1:m
17     alpha=0;
18     M=max(abs(A(k,[k:n])));
19     w=zeros(n,1);
20     for i=k:n
21         w(i)=A(k,i)/M;
22         alpha=alpha+w(i)^2;
23     end
24     alpha=sqrt(alpha);
25     beta=1/(alpha*(alpha+abs(w(k))));
26     w(k)=w(k)+sign(w(k))*alpha;
27
28     if slim
29
30         % bereken nieuwe A
31         for p=1:m
32             s=0;
33             for i=k:n
34                 s=s+w(i)*A(p,i);
35             end
36             s=beta*s;
37             for i=k:n
38                 A(p,i)=A(p,i)-s*w(i);
39             end
40         end
```

```

41
42         % bereken nieuwe Q
43         for p=1:n
44             s=0;
45             for i=k:n
46                 s=s+w(i)*Q(p,i);
47             end
48             s=beta*s;
49             for i=k:n
50                 Q(p,i)=Q(p,i)-s*w(i);
51             end
52         end
53     else
54         K=eye(n)-beta*w*w';
55         A=A*K;
56         Q=Q*K;
57     end
58
59 end
60
61 % voorwaartse eliminatie
62 z=zeros(m,1);
63 for i=1:m
64     s=0;
65     for j=1:i-1
66         s=s+z(j)*A(i,j);
67     end
68     z(i)=1/A(i,i)*(y(i)-s);
69 end
70
71
72 % vermenigvuldiging
73 x=zeros(n,1);
74 for i=1:n
75     for j=1:m
76         x(i)=x(i)+Q(i,j)*z(j);
77     end
78 end
79
80 flops
81
82 % controle
83 q=zeros(n,1);

```

```

84 %q(1)=[(-(1-1/n):1/n:-(1-(n-1)*1/n))*y;
85 for i=1:n
86     q(1)=q(1)-(1-i/n)*i/n;
87 end
88
89 for i=2:n
90     q(i)=q(i-1)+(i-1)/n;
91 end
92
93 x;
94 fprintf ('Gemaakte |A*x-y| fout is %e\n',norm(B*x-y));
95 fprintf ('Gemaakte |x-q| fout is %e\n',norm(x-q));

```