RA

stichting
mathematisch
centrum

MC

REKENAFDELING RA NR 20/71 SEPTEMBER

J.P. HOLLENBERG
OPLOSSING VAN LINEAIRE KLEINSTE-KWADRATEN
PROBLEMEN MET FOUTENANALYSE

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

Voorwoord

Dit rapport brengt verslag uit van enig onderzoek over de nauwkeurigheid van verschillende algorithmen voor de oplossing van kleinste-kwadratenproblemen.

De schrijver wil zijn dank betuigen voor de hulp die hij van Drs. W. Hoffmann mocht krijgen en voor de stimulerende leiding, suggesties en kritiek die Dr. T.J. Dekker hem gegeven heeft.

Inhoud

1. Probleemstelling en oplosmethoden

Zij A een gegeven reële m*n-matrix (m*n) en b een gegeven reële m-vector. Dan bestaat er een n-vector x, zodat $||b-Ax||_2$ minimaal is. Deze vector x heet de kleinste-kwadratenoplossing van het stelsel [A:b]. De oplossingsvector x is uniek als rang(A) = n (rang(A) is het maximale aantal onafhankelijke rijen of kolommen van A).

Oplossingsmethoden voor het geval rang(A) = n

De "klassieke" methode

Deze wijze van oplossen berust op het feit dat de vector x de oplossingsvector is van het vierkante stelsel

$$A^TAx = A^Tb \tag{1.1}$$

de zgn. normaalvergelijking (zie Dekker (1971, blz. 57)). Daar de (n*n) matrix M ($=A^TA$) een symmetrische positief definiete matrix is, kunnen we het stelsel (1.1) oplossen met de wortelmethode van Cholesky plus een heen- en een terugsubstitutie. Zie hiervoor bijvoorbeeld Wilkinson (1965).

De QR-methode

Bij de QR-methode ontbinden we A in een orthogonale (m*n)-matrix Q en een (n*n)-matrix R, zodanig dat

$$A = QR. \tag{1.2}$$

(Een orthogonale (m*n)-matrix Q is een matrix waarvoor geldt $Q^TQ = I_n$.) (1.2) gesubstitueerd in (1.1) geeft ons

$$R^TRx = R^TQ^Tb. \tag{1.3}$$

Daar we uit zijn gegaan van de veronderstelling dat rang(A) = n kunnen we stellen dat R^{-1} bestaat. Door een vermenigvuldiging met R^{-1} brengen we (1.3) over in

$$Rx = Q^Tb. \tag{1.4}$$

Aangezien R een bovendriehoeksmatrix is, kan dit stelsel rechtstreeks worden opgelost. Voor de ontbinding (1.2) kan men gebruik maken van verschillende (mathematisch equivalente) methoden. Deze methoden zijn

a) de Gram-Schmidt-orthogonalisatie (klassiek of gewijzigd), hiervoor verwijzen we naar Dekker (1971, blz. 61-64) en Björck (1967, blz. 1-21);
b) de Householder -orthogonalisatie, zie hiervoor Dekker (1971, blz. 64-65) en Businger & Golub (1965, blz. 206-216).

Hoewel de methodes mathematisch equivalent zijn, zijn ze dat numeriek niet. Alleen de klassieke Gram-Schmidt is qua numerieke stabiliteit onbevredigend (zie het voorbeeld van P. Läuchli in Björck (1967, blz. 4). De Householder-methode is voor gebruik op een rekenautomaat te verkiezen boven de gewijzigde Gram-Schmidt orthogonalisatie, omdat er iets minder rekentijd en minder geheugenruimte gevraagd wordt.
Een ALGOL-procedure voor de Householder-methode is gegeven in Businger & Golub (1965), blz. 269-276).

De QR-methode met iteratief verbeteren

Het iteratief verbeteren van de berekende oplossingsvector $\bar{x}$ van het stelsel (1.4) heeft in het algemeen weinig zin daar het residu (dat is: $||A\bar{x}-b||$) niet klein hoeft te zijn. Zie hiervoor Golub & Wilkinson (1966, blz. 139-148). Iteratief verbeteren is wél zinvol als we het stelsel (1.3) oplossen, want $||A^Tb-R^TR\bar{x}||$ zal wel klein zijn (mits het stelsel niet te slecht geconditioneerd is).

Oplossingsmethode voor het geval rang(A) $\leq$ n

Er is nu geen eenduidige oplossing meer voor het gestelde probleem. We zoeken daarom naar de kleinste vector x onder alle vectoren die een kleinste-kwadratenoplossing van het stelsel [A:b] zijn.

Een methode om deze x te bepalen wordt beschreven in Forsythe & Moler (1967).
Deze methode is als volgt:
We ontbinden A in U(m*n), $\sum$(n*n) en V(n*n) en wel zo dat

$$A = U\sum V^T.$$

Waarin $U^TU = V^TV = VV^T = I_n$ en $\sum = \text{diag}(\sigma_1,\ldots,\sigma_n)$,
met

$$\sigma_i > 0 \qquad i \leq r = \text{rang}(A)$$

$$\sigma_i = 0 \qquad r < i \leq n \ .$$

Voor x geldt nu

$$x = V\sum^{+}U^Tb$$

met $\sum^{+} = \text{diag}(\sigma_1^{+},\ldots,\sigma_n^{+})$,
waarin

$$\sigma_i^{+} = \begin{cases} 1/\sigma_i & \sigma_i > 0 \\ 0 & \sigma_i = 0. \end{cases}$$

Een ALGOL-procedure voor deze methode vindt men in Golub & Reinsch (1970, blz. 403-420) en een FORTRAN-procedure in Businger & Golub (1969, blz. 564-565).

2. De conditie van het probleem

Alvorens we de fouten behandelen die gemaakt kunnen worden bij het oplossen, beschouwen we de invloed van fouten die mogelijk reeds in de gegevens aanwezig zijn.

De gegevens, de matrix A en de vector b, kunnen het resultaat zijn van waarnemingen en afrondingen. De gegevens, die we verkregen zouden hebben als we de genoemde fouten niet gemaakt zouden hebben, noemen we $\hat{A}$ en $\hat{b}$. We stellen

$$A = \hat{A} + \delta A \tag{2.1}$$

en

$$b = \hat{b} + \delta b \tag{2.2}$$

Zij nu x de kleinste-kwadratenoplossing van het stelsel $[A:b]$ en $\hat{x}$ van het stelsel $[\hat{A}:\hat{b}]$ dan definiëren we

$$r = Ax - b \tag{2.3}$$

en $\hat{M} = \hat{A}^T\hat{A}$ zodat we analoog aan (1.1) vinden

$$\hat{M}\hat{x} = \hat{A}^T\hat{b}.$$

Uit (1.1) en (2.1) volgt

$$(\hat{A}^T+\delta A^T)Ax = A^Tb$$

of

$$\hat{M}x = \hat{A}^Tb - \delta A^T(Ax-b) - \hat{A}^T\delta Ax.$$

Hieruit volgt met (2.2) en (2.3)

$$\hat{M}(x-\hat{x}) = -\delta A^Tr+\hat{A}^T(\delta b-\delta Ax) \tag{2.4}$$

Als we nu $\hat{A}$ ontbinden (b.v. volgens (1.2)) dan vinden we een bovendriehoeksmatrix $\hat{R}$. Uit (2.4) volgt dan

$$||x-\hat{x}|| \leq ||\hat{M}^{-1}||\ ||\delta A^T||\ ||r|| + ||\hat{R}^{-1}||\ (||\delta A||\ ||x|| + ||\delta b||) \tag{2.5}$$

Ongeacht hoe nauwkeurig men rekent zal men dus niet meer precisie in het antwoord kunnen bereiken dan (2.5) aangeeft.

3. Fouten in floating-point-operaties

Zoals gebruikelijk zullen we de berekende waarde van een grootheid x in het vervolg aanduiden met $\bar{x}$. Deze berekende waarden zijn meestal het resultaat van floating-point-operaties, die we aan zullen geven met "fl".

In Dekker (1971, §4) wordt een floating-point-arithmetiek met grondtal β en een woordlengte van t bits nader geanalyseerd. Enkele van de in de volgende §§ gebruikte resultaten zijn

$$fl(a \pm b) = a(1+\varepsilon) \pm b(1+\varepsilon') \tag{3.1}$$

$$fl(a*b) = (a*b)(1+\varepsilon^*) \tag{3.2}$$

$$fl(a/b) = (a/b)(1+\varepsilon^*) \tag{3.3}$$

$$\left| fl\left(\sum_{i=1}^{n} (a_i*b_i) \right) - \sum_{i=1}^{n} (a_i*b_i) \right| \leq \varepsilon_n \sum_{i=1}^{n} |a_i*b_i| \tag{3.4}$$

hierin zijn $|\varepsilon|$, $|\varepsilon'|$ en $|\varepsilon^*| < c\beta^{-t}$

$$|\varepsilon_n| < n\beta^{-t_1}$$

$$\beta^{-t_1} = 1.06c\beta^{-t}.$$

Hierin is c een getal dat afhangt van de nauwkeurigheid van de gebruikte arithmetiek, de waarde van c is op zijn best $\beta/2$.

4. Foutenanalyse van de "klassieke" methode

Opmerking: Een vereenvoudigde en verkorte vorm van de inhoud van de §§ 4, 5 en 6 is te vinden in Hoffmann & Hollenberg (1971).

Als we de "klassieke" methode volgen dan maken we door afronding fouten in

a) het vormen van de matrix $M = A^TA$, hiervoor stellen we

$$\bar{M} = fl(A^TA) = A^TA + E \tag{4.1}$$

b) het ontbinden van A^TA volgens de wortelmethode, hiervoor stellen we

$$(\bar{U}^T\bar{U}) = \overline{A^TA} + F = A^TA + E + F \tag{4.2}$$

c) het vormen van het rechterlid, hiervoor stellen we $(A^Tb=c)$

$$fl(A^Tb) = A^Tb + e = \bar{c} \tag{4.3}$$

d) de heensubstitutie, hiervoor stellen we

$$(\bar{U}^T+G)\bar{y} = \bar{c} \tag{4.4}$$

e) de terugsubstitutie, hiervoor stellen we

$$(\bar{U}+H)\bar{x} = \bar{y}. \tag{4.5}$$

We zullen beschouwen hoe deze fouten-matrices E, F, e, G en H de gevonden oplossingsvector $\bar{x}$ beïnvloed hebben.
Uit (4.3), (4.4) en (4.5) volgt

$$(\bar{U}+G^T)^T(\bar{U}+H)\bar{x} = (\bar{U}+G^T)^T\bar{y}$$

$$= \bar{c} = A^Tb + e$$

wat na substitutie van (1.1) geeft

$$(\bar{U}^T\bar{U}+\bar{U}^TH+G(\bar{U}+H))\bar{x} = A^TAx + e. \tag{4.6}$$

Uit (4.2) en (4.6) volgt

$$A^TA\bar{x} - A^TAx + (E+F+\bar{U}^TH+G(\bar{U}+H))\bar{x} = e$$

m.a.w.

$$\bar{x} - x = M^{-1}\{e - (E+F+\bar{U}^TH+G(\bar{U}+H))\bar{x}\}. \qquad (4.7)$$

Voor we de normen van de foutenmatrices gaan afschatten kijken we eerst nog even naar M^{-1}. We nemen aan dat $\bar{U}$ niet singulier is, zodat er een S bestaat waarvoor geldt

$$M = \bar{U}^T\bar{U}(I+S) \qquad (4.8)$$

wat met (4.2) geeft

$$S = -(\bar{U}^T\bar{U})^{-1}(E+F). \qquad (4.9)$$

(4.7) met (4.9) geeft ons

$$\bar{x} - x = (I+S)^{-1}\,[(\bar{U}^T\bar{U})^{-1}\,\{e-(E+F+G(\bar{U}+H))\}\bar{x} + \bar{U}^{-1}H\bar{x}]. \qquad (4.10)$$

<u>Afschatting van de normen van de foutenmatrices</u>

We zullen eerst de fout in $\bar{M}$ en $\bar{c}$ bekijken d.w.z. de bovengrenzen voor de normen van E en e bepalen. Voor een element m_{ij} van de matrix M geldt volgens (3.4)

$$m_{ij} = [a_{1i}a_{1j}+\ldots+a_{mi}a_{mj}] + \varepsilon_m[|a_{1i}a_{1j}|+\ldots+|a_{mi}a_{mj}|]$$

hieruit volgt voor E (= $\bar{M}$-M)

$$||\bar{M}-M|| \leq m\beta^{-t_1}||A^T||\;||A||. \qquad (4.11)$$

Evenzo vinden we voor e(= $\bar{c}$-c)

$$||\bar{c}-c|| \leq m\beta^{-t_1}||A^T||\;||b||. \qquad (4.12)$$

Het volgende te behandelen punt is de fout die gemaakt wordt in de terugsubstitutie. De analyse hiervan is te vinden in Dekker (1971, blz. 35) en het gevonden resultaat toegepast op ons probleem (vergelijking (4.5)) is

$$\left.\begin{array}{ll}
|H_{in}| \leq |\bar{U}_{in}|\ (n-i+2)\ \beta^{-t_1} & \\
|H_{ij}| \leq |\bar{U}_{ij}|\ (j-i+3)\ \beta^{-t_1} & i < j < n \\
|H_{ii}| < 2|\bar{U}_{ii}|\ \beta^{-t_1} & \\
|H_{nn}| < |\bar{U}_{nn}|\ \beta^{-t_1} & \\
|H_{ij}| = 0 & j < i \leq n
\end{array}\right\} \qquad (4.13)$$

Voor de heensubstitutie kunnen we een analoge redenering volgen en krijgen als resultaat dat de bovengrens van $||G^T||$ gelijk is aan $||H||$.

Er rest ons nog een bovengrens van F in vergelijking (4.2) te vinden. De ontbinding van de symmetrische positief definiete matrix M volgens de wortelmethode van Cholesky geeft ons een matrix U zodat $U^TU = M$. De algorithme hiervoor is op een simultaan uitgevoerde schatting na equivalent met de algorithme voor de Gauss-eleminatie en we kunnen deze dan ook als volgt formuleren:

We beginnen met een matrix $M^{(1)}$ $(=M)$ en in het k^e $(k\leq n)$ stadium voeren we $M^{(k-1)}$ over in $M^{(k)}$ en bepalen de k^e rij van U uit de k^e rij van $M^{(k)}$ door een schaling. De overgang van de $M^{(k)}$ naar $M^{(k+1)}$ wordt gedefinieerd door

$$\left.\begin{array}{ll}
M_{ij}^{(k+1)} = M_{ij}^{(k)} & i < k \\
M_{ij}^{(k+1)} = M_{ij}^{(k)} - U_{ki}U_{kj} & i,j > k \\
M_{ik}^{(k+1)} = 0 & i \geq k \\
U_{kk} = \sqrt{M_{kk}^{(k)}} & \\
U_{kj} = M_{kj}^{k} / \sqrt{M_{kk}^{k}} &
\end{array}\right\} \qquad (4.14)$$

Als we bovenstaand schema volgen dan krijgen we door afronding van de (tussen) resultaten fouten. We zullen de distributie van die fouten bekijken.

We definiëren de matrix $\varepsilon^{(k)}$ door voor elk element te stellen

$$\bar{M}_{ij}^{(k+1)} = \bar{M}_{ij}^{(k)} - \bar{U}_{ki}\bar{U}_{kj} + \varepsilon_{ij}^{(k)} \tag{4.15}$$

en uit (4.1) en (4.2) volgt

$$\bar{M}_{ij}^{(k+1)} = \bar{M}_{ij}^{(k)}(1+\varepsilon) - \bar{U}_{ki}\bar{U}_{kj}(i+\varepsilon^*)(1+\varepsilon') \tag{4.16}$$

met $|\varepsilon|$, $|\varepsilon^*|$ en $|\varepsilon'| \leq c\beta^{-t}$.
Uit (4.5) en (4.6) volgt

$$\varepsilon_{ij}^{(k)} = \bar{M}_{ij}^{(k)}\varepsilon - \bar{U}_{ki}\bar{U}_{kj}[(1+\varepsilon^*)(1+\varepsilon')-1]$$

wat ons geeft

$$|\varepsilon_{ij}^{(k)}| \leq c\beta^{-t}|\bar{M}_{ij}^{(k)}| + 2\beta^{-t}1|\bar{U}_{ki}|\ |\bar{U}_{kj}|$$

of

$$|\varepsilon_{ij}^{(k)}| \leq 3g\beta^{-t}1. \tag{4.17}$$

Hierin is $g = \max_{i,j,k} |\bar{M}_{ij}^{k}|$, en wegens het positief definiet zijn van M geldt

$$g = \max_{i,j} |\bar{M}_{ij}|.$$

Verder volgt er uit (4.14) en (4.15)

$$\bar{M}_{kk}^{k} = M_{kk}^{(1)} - \bar{U}_{1k}\bar{U}_{1k} - \ldots - \bar{U}_{k-1,k}\bar{U}_{k-1,k} + e_{kk}$$

waarin

$$e_{kk} = \varepsilon_{kk}^{(1)} + \ldots + \varepsilon_{kk}^{k-1} \tag{4.18}$$

Voor $\bar{U}_{kk}$ geldt nu wegens het bovenstaande

$$\bar{U}_{kk} = \sqrt{A_{kk}^{(1)} - \bar{U}_{1k}\bar{U}_{1k} - \ldots - \bar{U}_{k-1k}\bar{U}_{k-1k} + e_{kk}}\,(1+\varepsilon)$$

waarin ε afhankelijk is van de gebruikte sqrt-routine; we nemen aan dat $|\varepsilon| < c\beta^{-t}$.
Kwadrateren geeft ons nu

$$\bar{U}_{kk}\bar{U}_{kk} = (A^{(1)}_{kk} - \bar{U}_{1k}\bar{U}_{1k} - \ldots - \bar{U}_{k-1k}\bar{U}_{k-1k} + e_{kk})(1+\varepsilon)^2$$

of

$$A^{(1)}_{kk} - \bar{U}_{1k}\bar{U}_{1k} - \ldots - \bar{U}_{kk}\bar{U}_{kk} = f_{kk}$$

met

$$|f_{kk}| \leq |e_{kk}| + 2\beta^{-t_1}(g+|e_{kk}|). \tag{4.19}$$

Voor $\bar{U}_{kj}$ geldt

$$\bar{U}_{kj} = (A^{(1)}_{kj} - \bar{U}_{1k}\bar{U}_{1j} - \ldots - \bar{U}_{k-1k}\bar{U}_{k-1j} + e_{kj}) \frac{(1+\varepsilon)}{\bar{U}_{kk}}$$

of

$$A^{(1)}_{kj} - \bar{U}_{1k}\bar{U}_{1j} - \ldots - \bar{U}_{kj}U_{kk} = f_{kj}$$

met

$$|f_{kj}| \leq e_{kj} + |\varepsilon|\ (g+|e_{kj}|) \tag{4.20}$$

waarin $|\varepsilon| \leq c\beta^{-t}$.
(4.17), (4.18), (4.19) en (4.20) geven ons

$$|F| = (|f_{ij}|) \leq 3g\beta^{-t_1} \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ \vdots & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & 1 & \cdots & n-1 \end{pmatrix} + g\beta^{-t_1} \begin{pmatrix} 2 & 1 & \cdots & 1 \\ 1 & 2 & & \vdots \\ \vdots & & \ddots & \vdots \\ 1 & \cdots & \cdots & 2 \end{pmatrix} \tag{4.21}$$

Als laatste punt bekijken we de schatting van $||\bar{U}^{-1}||$, zoals gezegd vinden we met floating-point-bewerkingen niet $\bar{U}^{-1}$ doch een matrix B met $B = \overline{\bar{U}^{-1}}$. Stel nu dat $B = \bar{U}^{-1}(I+\gamma)$ dan volgt onmiddellijk

$$||\bar{U}^{-1}|| \leq ||B||/(1-||\gamma||) \qquad (4.22)$$

mits $||.||$ consistent is, $||I|| = 1$ en $||\gamma|| < 1$.

Conclusie

Uit (4.10) kunnen we een bovengrens voor $||\bar{x}-x||$ berekenen waarin de bovengrenzen van $||E||$, $||e||$, $||G^T||$, $||H||$, $||F||$ en $||\bar{U}^{-1}||$ voorkomen, welke we berekenen kunnen met behulp van resp. (4.11), (4.12), (4.13), (4.22). We gebruiken als norm $||.||_\infty$ en vinden dan voor de bovengrens voor $||F||_\infty$, $||G^T||_\infty$ en $||H||_\infty$ resp. $g\beta^{-t_1}\frac{3n^2-n+2}{2}$, $g\beta^{-t_1}\frac{n^2+2n-2}{2}$ en $g\beta^{-t_1}\frac{n^2+2n-2}{2}$. Aldus wordt (4.10)

$$||x-\bar{x}||_\infty \leq \frac{1}{1-||S||_\infty}\Big[||(\bar{U}^T\bar{U})^{-1}||_\infty \{m||A^T||_\infty (||b||_\infty+||A||_\infty)$$

$$+ g\frac{3n^2-n+2}{2} + g\frac{n^2+2n-2}{2}(||\bar{U}||_\infty + g\beta^{-t_1}\frac{n^2+2n-2}{2})\}$$

$$+ ||\bar{U}^{-1}||_\infty g\frac{n^2+2n-2}{2}\Big]||\bar{x}||_\infty \beta^{-t_1}. \qquad (4.23)$$

5. Foutenanalyse van de QR-methode

In het vervolg zullen we onderscheid moeten maken tussen
- exacte resultaten en
- berekende resultaten, maar ook tussen deze en
- resultaten die verkregen zijn door exact rekenen, uitgaande van niet-exacte gegevens.

Deze laatsten zullen we onderscheiden van de andere door een slangetje boven het symbool te plaatsen.

Als we de QR-methode volgen dan maken we o.a. door afronding eventueel fouten in

a) het vormen van het linkerlid, hiervoor stellen we

$$R = Q^T A \quad \text{en} \quad \bar{R} = \tilde{Q}^T(A+E) \tag{5.1}$$

(N.B. $\tilde{Q}$ is exact orthogonaal maar hoeft niet gelijk aan Q te zijn.)

b) het vormen van het rechterlid, hiervoor stellen we

$$c = Q^T b \quad \text{en} \quad \bar{c} = \tilde{Q}^T(b+e) \tag{5.2}$$

c) de terugsubstitutie, hiervoor stellen we

$$Rx = c, \quad \bar{R}\tilde{x} = \bar{c} \quad \text{en} \quad (\bar{R}+F)\bar{x} = \bar{c}. \tag{5.3}$$

We zullen beschouwen hoe de foutenmatrices E, F en e de oplossingsvector $\bar{x}$ hebben beïnvloed.

Uit (5.1), (5.2) en (5.3) volgt dat $\tilde{x}$ de exacte kleinste-kwadraten-oplossing is van het stelsel $[A+E \vdots b+e]$, volgens (1.1) geldt er nu

$$(A+E)^T(A+E)\tilde{x} = (A+E)^T(b+e).$$

Hieruit volgt

$$(A^TA+A^TE+E^T(A+E))\tilde{x} = A^Tb + A^Te + E^T(b+e)$$

en na substitutie van $A^Tb = A^TAx$

$$A^TA\tilde{x} - A^TAx = A^Te + E^T(b+e) - (A^TE+E^T(A+E))\tilde{x}$$

wat geeft

$$\tilde{x} - x = (A^TA)^{-1}\{A^Te+E^T(b+e)-(A^TE+E^T(A+E))\tilde{x}\} \qquad (5.4)$$

Uit (5.3) volgt $(\bar{R}+F)\bar{x} = \bar{R}\tilde{x}$ m.a.w.

$$\bar{x} - \tilde{x} = -\bar{R}^{-1}F\bar{x} \qquad (5.5)$$

Uit (5.4) en (5.5) vinden we tenslotte

$$\bar{x} - x = -\bar{R}^{-1}F\bar{x} + M^{-1}[A^Te+E^T(b+e)-(A^TE+E^T(A+E))\tilde{x}]$$

of

$$\delta x = -\bar{R}^{-1}F\bar{x}$$

$$+ M^{-1}A^T[e-E(I+\bar{R}^{-1}F)\bar{x}]$$

$$+ M^{-1}E^T[b-A\tilde{x}+e-E\tilde{x}] \ .$$

We merken op dat als we definiëren

$$\tilde{r} = b + e - (A+E)\tilde{x}$$

er moet gelden

$$||\tilde{r}|| \leq ||b+e|| \ .$$

Zo vinden we tenslotte

$$||\delta x|| \leq ||\bar{R}^{-1}||\ ||F||\ ||\bar{x}||$$

$$+ ||M^{-1}A^T||\ [||e|| + ||E||\ (||\bar{x}|| + ||\bar{R}^{-1}||\ ||F||\ ||X||)]$$

$$+ ||M^{-1}||\ ||E^T||\ (||b|| + ||e||) \ . \qquad (5.6)$$

Natuurlijk kunnen we i.h.a. M^{-1} en $\bar{R}^{-1}$ niet exact berekenen, wel kunnen we numerieke benaderingen vinden, uitgaande van $\bar{R}$. Uit (5.1) volgt $A = \tilde{Q}\bar{R} - E$ en dus

$$M = \bar{R}^T\bar{R} - \bar{R}^T\tilde{Q}E - E^T\tilde{Q}\bar{R} + E^TE \ . \tag{5.7}$$

We nemen aan dat er een S bestaat, zo dat

$$A^TA = \bar{R}^T\bar{R}(I+S). \tag{5.8}$$

Uit (5.7) en (5.8) volgt nu

$$S = -\bar{R}^{-1}\tilde{Q}^TE + (\bar{R}^T\bar{R})^{-1}[E^TE-E^T\tilde{Q}R]$$

en

$$(A^TA)^{-1} = (I+S)^{-1}(\bar{R}^T\bar{R})^{-1} \ .$$

Mits $||S|| < 1$ en $||I|| = 1$ geldt

$$||M^{-1}|| \leq ||(\bar{R}^T\bar{R})^{-1}||/(1-||S||).$$

Voor $M^{-1}A^T$ vinden we

$$M^{-1}A^T = (I+S)^{-1}\ \bar{R}^{-1}(\bar{R}^T)^{-1}\ [\bar{R}^T\tilde{Q}-E^T]$$

$$= (I+S)^{-1}\ \bar{R}^{-1}\ [\tilde{Q}-(\bar{R}^T)^{-1}E^T]$$

waaruit volgt

$$||M^{-1}A^T|| \leq ||\bar{R}^{-1}||\ [||\tilde{Q}|| + ||(\bar{R}^T)^{-1}||\ ||E^T||]/(1-||S||).$$

De afschatting van $||\bar{R}^{-1}||$ verloopt analoog aan de afschatting van $||\bar{U}^{-1}||$ in de vorige § en we vinden dat (4.22) ook geldt voor $||\bar{R}^{-1}||$.

Afschatting van de normen van de foutenmatrices en -vectoren

We nemen aan dat de in Businger & Golub (1965) beschreven methode wordt gevolgd om A over te voeren in R. Deze methode komt neer op het voorvermenigvuldigen van A met de matrices $P^{(0)},P^{(1)},\ldots,P^{(n-1)}$. We bekijken eerst

De constructie van $P^{(r)}$

De elementaire hermitische matrix $P^{(r)}$ heeft de vorm

$$P^{(r)} = I - 2w^{(r)}w^{(r)T}$$

waarin $||w^{(r)T}w^{(r)}|| = 1$ en

$$w^{(r)T} = (0,0,\ldots,0,w^{(r)}_{r+1},\ldots,w^{(r)}_{m}).$$

Als we nu een vector a voorvermenigvuldigen met $P^{(r)}$ dan krijgen we het volgende beeld

$$P^{(r)}a = \begin{bmatrix} I & \vdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \vdots & I-2uu^T \end{bmatrix}\begin{bmatrix} b \\ \cdots \\ c \end{bmatrix} = \begin{bmatrix} b \\ \cdots \\ d \end{bmatrix}$$

waarin $d = (I-2uu^T)c$ en $u^T = (w^{(r)}_{r+1},\ldots,w^{(r)}_{m})$. Voor onze doeleinden willen we $w^{(r)}$ en dus u zo kiezen dat $d_i = 0$ voor $i = 2,3,\ldots,k$ $(k=m-r)$, wat we kunnen bereiken door het volgende rekenvoorschrift te volgen

$$s = \left(\sum_{i=1}^{k} c_i^2\right)^{1/2} \tag{5.9}$$

$$H = s^2 + |c_1|s \tag{5.10}$$

$$v^T = (|c_1|+s,c_2,\ldots,c_k) \tag{5.11}$$

$$vv^T/H = 2uu^T \tag{5.12}$$

De fout in $\bar{P}^{(r)}$

Wegens (5.9) en (4.4) geldt

$$\bar{s}^2 = (c_1^2+\ldots+c_k^2)(1+\varepsilon_k)$$

wegens $c_i^2 \geq 0$ en dus

$$\bar{s} = \sqrt{c_1^2+\ldots+c_k^2}\ \sqrt{1+\varepsilon_k}\ (1+\varepsilon) \tag{5.13}$$

hierin hangt ε weer af van de sqrt-routine en net als in §4 stellen we $|\varepsilon| < c\beta^{-t}$. (5.10) kunnen we formuleren als

$$\bar{s} = s(1+\sigma) \tag{5.14}$$

met $|\sigma| \leq \frac{1}{2}|\varepsilon_k| + |\varepsilon| + \frac{1}{2}|\varepsilon_k\varepsilon|$.

Zonder verlies van algemeenheid kunnen we veronderstellen dat c_1 niet-negatief is. Uit (5.10) en (4.4) volgt nu

$$\bar{H} = (s^2+c_1\bar{s})(1+\varepsilon_{k+1}) \tag{5.15}$$

en als we definiëren

$$s^2 + c_1 s(1+\sigma) = (s^2+c_1 s)(1+\alpha) \tag{5.16}$$

dan vinden we voor α

$$\alpha = \frac{c_1\sigma}{s+c_1}$$

en daar $s \geq c_1$ geldt zeker

$$|\alpha| \leq \tfrac{1}{2}|\sigma|. \tag{5.17}$$

Als we nu stellen

$$\bar{H} = H(1+\theta) \tag{5.18}$$

dan vinden we wegens (5.15) en (5.16)

$$1 + \theta = (1+\alpha)(1+\varepsilon_{k+1})$$

en wegens (5.17)

$$|\theta| \leq \tfrac{1}{2}|\sigma| + |\varepsilon_{k+1}| + \tfrac{1}{2}|\sigma|\ |\varepsilon_{k+1}|.$$

De vector $\bar{v}$ wordt gegeven door

$$\bar{v}^T = (fl(c_1+\bar{s}),c_2,\ldots,c_k),$$

er kan dus hoogstens een fout optreden in het eerste element, voor die $\bar{v}_1$ vinden we volgens (5.11)

$$\bar{v}_1 = fl(c_1+\bar{s}) = c_1(1+\varepsilon) + s(1+\sigma)(1+\varepsilon')$$

of

$$\bar{v}_1 = [c_1+s(1+\sigma)]\ (1+\varepsilon^*)$$

waarin $|\varepsilon|$, $|\varepsilon'|$ en $|\varepsilon^*| \leq c\beta^{-t}$.
Weer stellen we

$$c_1 + s(1+\sigma) = (c_1+s)(1+\alpha)$$

en we vinden weer (5.17) wat leidt tot

$$\bar{v}_1 = v_1(1+\psi) \qquad (5.19)$$

met

$$|\psi| \leq \tfrac{1}{2}|\sigma| + |\varepsilon^*| + \tfrac{1}{2}|\sigma\varepsilon^*|.$$

Als we schrijven $\bar{v} = v + \delta v$ dan vinden we, daar $\bar{v}_i = v_i$, $i = 2,3,\ldots,k$,

$$||\delta v||_2 = |v_1|\ |\psi| \leq |\psi|\ ||v||_2.$$

Het linker benedenstuk van $P^{(r)}$ noemen we P ($P=I-vv^T/H$) en we stellen vast dat er geen fouten meer gemaakt worden tot we overgaan tot

voorvermenigvuldigen met $\bar{P}^{(r)}$. We beschouwen eerst de fout in P

$$\bar{P} - P = (vv^T - \bar{v}\bar{v}^T)/H + \bar{v}\bar{v}^T(1/H - 1/\bar{H}) =$$

$$= (v\delta v^T + \delta v v^T + \delta v \delta v^T)/H + (vv^T + v\delta v^T + \delta v v^T + \delta v \delta v^T)\theta/(H(1+\theta)).$$

Daar $||vv^T/H||_2 = 2$ geldt

$$||\bar{P}-P||_2 \leq \pi$$

met

$$\pi = 2(2|\psi| + \psi^2 + |\theta|)/(1-|\theta|)$$

en uit de vorm van $P^{(r)}$ volgt dat

$$||\bar{P}^{(r)} - P^{(r)}||_2 \leq \pi.$$

We beschouwen vervolgens

De fout in de voorvermenigvuldiging met $\bar{P}^{(r)}$

Onze interesse gaat uit naar de foutenmatrix E waarvoor geldt

$$E = fl(\bar{P}^{(r)}A) - \bar{P}^{(r)}A.$$

Het voorvermenigvuldigen van A met $\bar{P}^{(r)}$ geeft

$$\bar{P}^{(r)}A = \begin{bmatrix} I & \vdots & 0 \\ \cdots & \cdot & \cdots \\ 0 & \vdots & \bar{P} \end{bmatrix} \begin{bmatrix} B \\ \cdots \\ C \end{bmatrix} = \begin{bmatrix} B \\ \cdots \\ \bar{P}C \end{bmatrix}$$

waarbij $\bar{P}C$ in de praktijk als volgt wordt berekend

$$\left.\begin{aligned} \tilde{p}^T &= \bar{v}^T C/\bar{H} \\ \tilde{D} &= \bar{P}C = C - \bar{v}\tilde{p}^T \end{aligned}\right\} \qquad (5.20)$$

We beschouwen dus allereerst $\bar{p}$, hiervoor geldt volgens (5.20)

$$\bar{p}_i = \{[\bar{v}_1 c_{1i} + \ldots + \bar{v}_k c_{ki}] + \varepsilon_k[|\bar{v}_1||c_{1i}| + \ldots + |v_k||c_{ki}|]\}\ (1+\varepsilon)/\bar{H}$$

met $|\varepsilon| < c\beta^{-t}$ of

$$\bar{p}_i = \{\tilde{p}_i + \varepsilon_k[|\bar{V}|^T|C|]_i/\bar{H}\}\ (1+\varepsilon). \tag{5.21}$$

Uit (5.21) volgt

$$||\bar{p}-\tilde{p}||_2 \leq c\beta^{-t}||\tilde{p}||_2 + \rho||\bar{v}||_2||C||_E/\bar{H} \tag{5.22}$$

waarin het subscript "E" de euclidische norm aanduidt. Voor ρ geldt

$$|\rho| \leq k\beta^{-t_1}(1+c\beta^{-t}).$$

Als we vervolgens naar $\bar{D}$ kijken dan vinden we uit (5.20)

$$\begin{aligned}\bar{d}_{ij} &= fl((c-\bar{\bar{v}}\bar{\bar{p}}^T)_{ij}) \\ &= c_{ij}(1+\varepsilon'') - \bar{v}_i\bar{p}_j(1+\varepsilon^*)(1+\varepsilon') \\ &= [c_{ij}-\bar{v}_i(\tilde{p}_j+\delta\tilde{p}_j)(1+\varepsilon^*)]\ (1+\varepsilon)\end{aligned}$$

met weer $|\varepsilon|$, $|\varepsilon^*|$, $|\varepsilon'|$ en $|\varepsilon''| \leq c\beta^{-t}$.
Dit geeft

$$\bar{d}_{ij} = [c_{ij}-\bar{v}_i\tilde{p}_j](1+\varepsilon) - [\varepsilon^*\bar{v}_i\tilde{p}_j+\bar{v}_i\delta\tilde{p}_j(1+\varepsilon^*)](1+\varepsilon)$$

waaruit volgt

$$|\bar{d}_{ij}-\tilde{d}_{ij}| \leq |\varepsilon|\ |\tilde{d}_{ij}| + (1+|\varepsilon|)[|\varepsilon^*|\ |\bar{v}_i|\ |\tilde{p}_j| + |\bar{v}_i|\ |\delta\tilde{p}_j|\ (1+|\varepsilon^*|)]$$

en dus

$$||\bar{D}-\tilde{D}||_E \leq c\beta^{-t}||\tilde{D}||_E + (1+c\beta^{-t})\,[c\beta^{-t}||\bar{v}||_2||\tilde{p}||_2 +$$

$$+ ||\bar{v}||_2||\delta\tilde{p}||_2(1+c\beta^{-t})]. \qquad (5.23)$$

Om (5.23) af te schatten berekenen we eerst

$$||\bar{v}||_2^2/\bar{H} \leq ||v||_2^2\,(1+\psi)^2/H(1+\theta) \leq \frac{2(1+|\psi|)^2}{1-|\theta|} \qquad (5.24)$$

wegens (5.18) en (5.19) en dus

$$||\bar{v}||_2||\tilde{p}||_2 \leq ||\bar{v}||_2^2||C||_E/\bar{H} \leq ||C||_E\, 2(1+|\psi|)^2/(1-|\theta|) \qquad (5.25)$$

er geldt wegens (5.22)

$$||\bar{v}||_2||\delta\tilde{p}||_2 \leq ||\bar{v}||_2\,\{c\beta^{-t}||p||+|\rho|\;||\bar{v}||_2||c||_E/\bar{H}\}$$

wat met (5.24) en (5.25) geeft

$$||\bar{v}||_2||\delta\tilde{p}||_2 \leq 2(c\beta^{-t}+|\rho|)(1+|\psi|)^2\;||C||_E/(1-|\theta|) \qquad (5.26)$$

en tenslotte

$$||\tilde{D}||_E = ||\tilde{P}C||_E = ||C||_E\,.$$

Met behulp van (5.24) t/m (5.27) voeren we (5.23) over in

$$||\bar{D}-\tilde{D}||_E \leq \alpha||C||_E$$

met

$$\alpha = c\beta^{-t} + 2(1+c\beta^{-t})\,\frac{(1+|\psi|)^2}{1-|\theta|}\,\{c\beta^{-t}+(c\beta^{-t}+|\rho|)(1+c\beta^{-t})\}\,.$$

Tenslotte vinden we

$$||fl(\bar{P}^{(r)}A)-P^{(r)}A||_E \leq ||\bar{D}-\tilde{D}||_E + ||\tilde{D}-P^{(r)}A||_E$$

$$||E||_E \leq ||\bar{D}-\tilde{D}||_E + ||\bar{P}^{(r)}-P^{(r)}||_E||C||_E$$

$$\leq (\alpha+|\pi|)||C||_E$$

Wat na enig rekenen geeft

$$||fl(\bar{P}^{(r)}A)-P^{(r)}A||_E \leq \beta_k||C||_E \qquad (k=m-r)$$

met $\beta_k \leq (14.12+5.83k)\, c\beta^{-t} + 8(kc\beta^{-t})^2$. (5.28)

Dit resultaat gaan we toepassen in

<u>De fout in een serie van voorvermenigvuldigingen.</u>

Tijdens de Householder triangularisatie wordt A voorvermenigvuldigd met resp. $P^{(0)},P^{(1)},\ldots,P^{(n-1)}$. De vorm en constructie van $P^{(r)}$ hebben we al besproken. De oorspronkelijke matrix A zullen we $A^{(0)}$ noemen en de getransformeerde $A^{(r)}$, $r=1,\ldots,n$.

Daar $\bar{A}^{(r)} = fl(\bar{P}^{(r-1)}\,\bar{A}^{(r-1)})$ vinden we

$$\bar{A}^{(1)} - P^{(0)}A^{(0)} = F^{(0)} \quad \text{en} \quad \bar{A}^{(r)} - P^{(r)}\bar{A}^{(r-1)} = F^{(r-1)}$$

waarin

$$||F^{(0)}||_E \leq \beta_m||A^{(0)}||_E \quad \text{en} \quad ||F^{(r)}||_E \leq \beta_{m-r}||\bar{A}^{(r)}||, \; r\geq 1 \qquad (5.29)$$

met β_i gedefinieerd in (5.28).
De $\tilde{P}^{(r)}$ zijn exact orthogonaal (hoewel mogelijk ongelijk aan $P^{(r)}$) waardoor we kunnen stellen

$$||\bar{A}^{(r)}||_E \leq ||\bar{A}^{(r-1)}||_E + ||F^{(r-1)}||_E \leq (1+\beta_{m-r+1})\,||\bar{A}^{(r-1)}||_E$$

$$\leq (1+\beta_{m-r+1})\,(1+\beta_{m-r+2})\,||\bar{A}^{(r-2)}||_E$$

$$\leq \prod_{i=0}^{r-1} (1+\beta_{m-i})\,||A^0||_E \,. \qquad (5.30)$$

Tevens geldt

$$\begin{aligned}\bar{A}^{(r)} &= F^{(r-1)} + \tilde{P}^{(r-1)}\bar{A}^{(r-1)} \\ &= F^{(r-1)} + \tilde{P}^{(r-1)}(F^{(r-2)}+\tilde{P}^{(r-2)}\bar{A}^{(r-2)}) \\ &= F^{(r-1)} + \tilde{P}^{(r-1)}F^{(r-2)} + \ldots + \tilde{P}^{(r-1)}\ldots P^{(0)}A^{(0)} \\ &= \tilde{P}^{(r-1)}\ldots P^{(0)}A^{(0)} + G^{(r)} \qquad (5.31)\end{aligned}$$

met

$$G^{(r)} = F^{(r-1)} + \tilde{P}^{(r-1)}F^{(r-2)} + \ldots . \qquad (5.32)$$

We definiëren $\tilde{Q} = \tilde{P}^{(n-1)}\ldots\tilde{P}^{(1)}P^{(0)}$ en vinden uit (5.31)

$$\bar{A}^{(n)} = \tilde{Q}A^{(0)} + G^{(n)} = \tilde{Q}(A+E) \qquad (5.33)$$

met $||E||_E \leq ||G^{(n)}||_E$.
De norm van E schatten we als volgt af

$$\begin{aligned}||E||_E &\leq ||F^{(n-1)}|| + ||F^{(n-2)}|| + \ldots + ||F^{(0)}|| \\ &\leq \beta_{m-n+1}||\bar{A}^{(n-1)}|| + \ldots + \beta_m||A^0||\end{aligned}$$

wegens (5.32) en (5.29) wat met (5.30) geeft

$$||E||_E \leq [\beta_{m-n+1}\prod_{i=0}^{n-2}(1+\beta_{m-i}) + \ldots + \beta_{m-1}(1+\beta_m)+\beta_m]\,||A||_E \qquad (5.34)$$

Iets beter kunnen we de afschatting maken als we bedenken dat we eigenlijk i.p.v. $||A^{(r)}||_E$ de norm van het te transformeren linker benedenstuk van $A^{(r)}$ moeten nemen. De algorithme in Businger & Golub (1967) is zo ingericht dat de kolom met de grootste euclidische lengte steeds naar voren wordt gehaald. Indien nu de transformaties exact zouden plaats vinden (dwz. $\tilde{P}^{(r)} = P^{(r)}$) dan zou de norm van het te transformeren stuk voor zekere r zeker kleiner zijn dan $\sqrt{(n-r)/n}\,||A^{(0)}||$. Op grond van deze redenering voeren we (5.34) over in

$$||E||_E \leq \sum_{i=0}^{n-1} [\beta_{m-i} \sqrt{\frac{i+1}{n}} \prod_{j=0}^{i-1} (1+\beta_{m-j})] \, ||A^0||_E \, . \qquad (5.35)$$

Analoog aan (5.34) kunnen we een bovengrens voor $||e||_2$ vinden.

Wellicht ten overvloede wijzen we erop dat de matrix E in (5.35) dezelfde is als die in (5.1).

Conclusie

Als we de bovengrenzen van de normen van de foutenmatrices hebben bepaald, kunnen we met behulp van (5.6) een bovengrens van $||x-\bar{x}||_2$ bepalen. We vinden dan

$$\begin{aligned} ||x-\bar{x}||_2 \leq \alpha + \frac{1}{1-\gamma} [&||\bar{R}^{-1}||_E^2 ||E||_E \{||b||_2 + 2||e||_2 \\ &+ ||E||_E (||\bar{x}||_2 + \alpha)\} \\ &+ \sqrt{n} \, ||\bar{R}^{-1}||_E (||e||_2 + ||E||_E (||\bar{x}||_2 + \alpha)] \end{aligned} \qquad (5.36)$$

waarin

$$\gamma = ||\bar{R}^{-1}||_E ||E||_E + ||\bar{R}^{-1}||_E^2 ||E||_E (||E||_E + ||\bar{R}||_E)$$

en

$$\alpha = ||\bar{R}^{-1}||_E ||F||_E ||\bar{x}||_2 \, .$$

6. Foutenanalyse met behulp van residuberekening

Zij x de kleinste-kwadratenoplossing van het stelsel [A:b], dan geldt voor de zgn. residuvector r het volgende

$$Ax = b + r \quad \text{en} \quad A^T r = 0 \ .$$

Afschatting van de fout in $\bar{x}$

Als we de vector $\bar{x}$ berekend hebben, dan bestaat er een $\tilde{r}$, zodat

$$A\bar{x} = b + \tilde{r} \ .$$

Na voorvermenigvuldigen met A^T vinden we

$$A^T A\bar{x} = A^T b + A^T\tilde{r} = A^T Ax + A^T\tilde{r} \ . \tag{6.1}$$

Natuurlijk kunnen we in het algemeen $\tilde{r}$ niet berekenen, maar wel $\bar{r}$ met $\bar{r} = fl(A\bar{x}-b)$ en we stellen

$$\tilde{r} - \bar{r} = f \ . \tag{6.2}$$

Vervolgens voeren we de grootheid s in waarvoor geldt

$$s = A^T r, \quad \tilde{s} = A^T\bar{r} \quad \text{en} \quad \bar{s} = fl(A^T\bar{r}) \tag{6.3}$$

en we stellen

$$\tilde{s} - \bar{s} = g \ . \tag{6.4}$$

We vinden nu uit (6.1) t/m (6.4)

$$\begin{aligned} \bar{x} - x &= M^{-1}[A^T(\bar{r}-\bar{r}+\tilde{r})] \\ &= M^{-1}[A^T(\bar{r}+f) \ ] \\ &= M^{-1}[\tilde{s}+A^T f \ ] \\ &= M^{-1}[\bar{s}-\bar{s}+\tilde{s}+A^T f] \\ &= M^{-1}[\bar{s}+g+A^T f \ ] \end{aligned} \tag{6.5}$$

Afschatting van de normen van de foutenmatrices en -vectoren

Het afschatten van $||M^{-1}||$ en $||M^{-1}A^T||$ doet men afhankelijk van de gebruikte methode om M te ontbinden, dit is reeds behandeld, zie §5 en §6.

Volgens §3 vinden we

$$||f|| \leq \varepsilon_{n+1}||A||\ ||\bar{x}|| + c\beta^{-t}||b|| \tag{6.6}$$

en

$$||g|| \leq \varepsilon_m||A^T||\ ||\bar{r}|| \ . \tag{6.7}$$

Conclusie

Uit (6.5), (6.6) en (6.7) volgt

$$||x-\bar{x}|| \leq ||M^{-1}||\ [||\bar{s}|| + \varepsilon_m||A^T||\ ||\bar{r}||]$$

$$+ ||M^{-1}A^T||\ [\varepsilon_{n+1}||A||\ ||x|| + \varepsilon\beta^{-t}||b||] \ .$$

```
procedure lngmatvec(l, u, i, a, b, c, cc, d, dd);
value l, u, i, c, cc; integer l, u, i; real c, cc, d, dd; array a, b;
begin integer k;
    real x, xx, y, yy, z;
    for k:= l step 1 until u do
    begin xx:= a[i,k]; yy:= b[k];
    mul12: x:= xx × 1048577; x:= xx - x + x; xx:= xx - x;
        y:= yy × 1048577; y:= yy - y + y; yy:= yy - y;
        z:= x × yy + xx × y; y:= x × y; x:= y + z;
        y:= y - x + z + xx × yy;
    add2: z:= x + c;
        cc:= if abs(x) > abs(c) then x - z + c + cc + y else c - z
        + x + y + cc; c:= z + cc; cc:= z - c + cc
    end;
    d:= c; dd:= cc
end lngmatvec;

procedure lngtamvec(l, u, i, a, b, c, cc, d, dd);
value l, u, i, c, cc; integer l, u, i; real c, cc, d, dd; array a, b;
begin integer k;
    real x, xx, y, yy, z;
    for k:= l step 1 until u do
    begin xx:= a[k,i]; yy:= b[k];
    mul12: x:= xx × 1048577; x:= xx - x + x; xx:= xx - x;
        y:= yy × 1048577; y:= yy - y + y; yy:= yy - y;
        z:= x × yy + xx × y; y:= x × y; x:= y + z;
        y:= y - x + z + xx × yy;
    add2: z:= x + c;
        cc:= if abs(x) > abs(c) then x - z + c + cc + y else c - z
        + x + y + cc; c:= z + cc; cc:= z - c + cc
    end;
    d:= c; dd:= cc
end lngtamvec;
```

7. De ALGOL 60-procedures

In de volgende procedures worden herhaaldelijk elementaire arithmetische bewerkingen in dubbele precisie uitgevoerd. Procedures hiervoor (add2, sub2, mul12, mul2, div2 en sqrt2) zijn beschreven in Dekker (1970). Overeenkomstig de daar gebruikte notatie zullen we een dubbel-lengtegetal aanduiden als

(kop,staart).

Het berekenen van scalaire produktie in dubbele precisie

lngmatvec

lngmatvec berekent in udbbele precisie de som van (c,cc) en het scalair product van de rijvector, gegeven in array a[i:i,1:u] en de vector, gegeven in array b[1:u].
Het resultaat wordt afgeleverd in (d,dd).

lngtamvec

lngtamvec berekent in dubbele precisie de som van (c,cc) en het scalair product van de kolomvector, gegeven in array a[1:u,i:i] en de vector, gegeven in array b[1:u].
Het resultaat wordt afgeleverd in (d,dd).

```
procedure lngmatmat(l, u, i, j, a, b, c, cc, d, dd);
value l, u, i, j, c, cc; integer l, u, i, j; real c, cc, d, dd;
array a, b;
begin integer k;
    real x, xx, y, yy, z;
    for k:= l step 1 until u do
    begin xx:= a[i,k]; yy:= b[k,j];
    mul12: x:= xx × 1048577; x:= xx - x + x; xx:= xx - x;
        y:= yy × 1048577; y:= yy - y + y; yy:= yy - y;
        z:= x × yy + xx × y; y:= x × y; x:= y + z;
        y:= y - x + z + xx × yy;
    add2: z:= x + c;
        cc:= if abs(x) > abs(c) then x - z + c + cc + y else c - z
        + x + y + cc; c:= z + cc; cc:= z - c + cc
    end;
    d:= c; dd:= cc
end lngmatmat;

procedure lngtammat(l, u, i, j, a, b, c, cc, d, dd);
value l, u, i, j, c, cc; integer l, u, i, j; real c, cc, d, dd;
array a, b;
begin integer k;
    real x, xx, y, yy, z;
    for k:= l step 1 until u do
    begin xx:= a[k,i]; yy:= b[k,j];
    mul12: x:= xx × 1048577; x:= xx - x + x; xx:= xx - x;
        y:= yy × 1048577; y:= yy - y + y; yy:= yy - y;
        z:= x × yy + xx × y; y:= x × y; x:= y + z;
        y:= y - x + z + xx × yy;
    add2: z:= x + c;
        cc:= if abs(x) > abs(c) then x - z + c + cc + y else c - z
        + x + y + cc; c:= z + cc; cc:= z - c + cc
    end;
    d:= c; dd:= cc
end lngtammat;

procedure lngmattam(l, u, i, j, a, b, c, cc, d, dd);
value l, u, i, j, c, cc; integer l, u, i, j; real c, cc, d, dd;
array a, b;
begin integer k;
    real x, xx, y, yy, z;
    for k:= l step 1 until u do
    begin xx:= a[i,k]; yy:= b[j,k];
    mul12: x:= xx × 1048577; x:= xx - x + x; xx:= xx - x;
        y:= yy × 1048577; y:= yy - y + y; yy:= yy - y;
        z:= x × yy + xx × y; y:= x × y; x:= y + z;
        y:= y - x + z + xx × yy;
    add2: z:= x + c;
        cc:= if abs(x) > abs(c) then x - z + c + cc + y else c - z
        + x + y + cc; c:= z + cc; cc:= z - c + cc
    end;
    d:= c; dd:= cc
end lngmattam;
```

lngmatmat

lngmatmat berekent in dubbele precisie de som van (c,cc) en het scalair product van de rijvector, gegeven in array a[i:i,l:u], en de kolomvector, gegeven in array b[l:u,j:j]. Het resultaat wordt afgeleverd in (d,dd).

lngtammat

lngtammat berekent in dubbele precisie de som van (c,cc) en het scalair product van de kolomvectoren, gegeven in array a[l:u,i:i] en array b[l:u,j:j]. Het resultaat wordt afgeleverd in (d,dd).

lngmattam

lngmattam berekent in dubbele precisie de som van (c,cc) en het scalair product van de rijvectoren, gegeven in array a[i:i,l:u] en array b[j:j,l:u]. Het resultaat wordt afgeleverd in (d,dd).

```
procedure lngata(a, m, n, ata, d, dd); value m, n; integer m, n;
array a, ata, d, dd;
begin integer i, j;
    for i:= 1 step 1 until n do
    begin lngtammat(1, m, i, i, a, a, 0, 0, d[i], dd[i]);
        for j:= i + 1 step 1 until n do lngtammat(1, m, i, j, a, a,
        0, 0, ata[i,j], ata[j,i])
    end
end lngata;

procedure lngatb(a, m, n, b, c, cc); value m, n; integer m, n;
array a, b, c, cc;
begin integer i;
    for i:= 1 step 1 until n do lngtamvec(1, m, i, a, b, 0, 0, c[i],
    cc[i])
end lngatb;
```

Het opstellen van de normaalvergelijking

lngata

lngata berekent in dubbele precisie de matrix $M = A^TA$, waarbij A de m*n-matrix is die gegeven is in array a[1:m,1:n]. De elementen van de bovendriehoek van M worden in dubbele lengte afgeleverd in array ata[1:n,1:n], d, dd[1:n] als volgt

$$M_{ij} = (\text{ata}[i,j], \text{ata}[j,i]), \quad i < j,$$

$$M_{ii} = (\text{d}[i], \text{dd}[i]).$$

lngata gebruikt lngtammat.

lngatb

lngatb berekent in dubbele precisie de vector $C = A^TB$, waarbij A de m*n-matrix is die gegeven is in array a[1:m,1:n] en B de m-vector, gegeven in array b[1:m].
De elementen van C worden in dubbele lengte afgeleverd in array c,cc[1:n] als volgt

$$C_i = (\text{c}[i], \text{cc}[i]).$$

lngatb gebruikt lngtamvec.

```
procedure lngatarep(a, m, n, ata, d, dd); value m, n; integer m, n;
array a, ata, d, dd;
begin integer i, j;
    for i:= 1 step 1 until n do
    begin lngtammat(1, m, i, i, a, a, d[i], dd[i], d[i], dd[i]);
        for j:= i + 1 step 1 until n do lngtammat(1, m, i, j, a, a,
        ata[i,j], ata[j,i], ata[i,j], ata[j,i])
    end
end lngatarep;

procedure lngatbrep(a, m, n, b, c, cc); value m, n; integer m, n;
array a, b, c, cc;
begin integer i;
    for i:= 1 step 1 until n do lngtamvec(1, m, i, a, b, c[i],
    cc[i], c[i], cc[i])
end lngatbrep;
```

lgnatarep

lngatarep berekent in dubbele precisie de matrix A^TA, waarbij A de m*n-matrix is die gegeven is in array a[1:m,1:n]. De elementen van de bovendriehoek van A^TA worden in dubbele precisie opgeteld bij de overeenkomstige elementen van de (symmetrische) matrix M die gegeven zijn in array ata[1:n,1:n], d,dd[1:n] als volgt

$$M_{ij} = (ata[i,j], ata[j,i]), \qquad i < j,$$

$$M_{ii} = (d[i], dd[i]).$$

De elementen van het resultaat worden op de plaats van de overeenkomstige elementen van M afgeleverd in ata, d en dd.
lngatarep gebruikt lngtammat.

lngatbrep

lngatbrep berekent in dubbele precisie de vector A^TB, waarbij A de m*n-matrix is, gegeven in array[1:m,1:n] en B de m-vector, gegeven in array b[1:m]. De elementen van A^TB worden in dubbele precisie opgeteld bij de overeenkomstige elementen van de vector C die gegeven is in array c,cc[1:n] als volgt

$$C_i = (c[i], cc[i]).$$

De elementen van het resultaat worden op de plaats van de overeenkomstige elementen van C afgeleverd in c en cc.
lngatbrep gebruikt lngtamvec.

```
real procedure lngdetsym(ata, n, d, dd); value n; integer n;
array ata, d, dd;
begin integer k, j;
    real r, rr, de, dde, z, zz;
    de:= 1; dde:= 0;
    for k:= 1 step 1 until n do
    begin sub 2(2 × matmat(1, k - 1, k, k, ata, ata), 0, d[k],
        dd[k], r, rr);
        lngtammat(1, k - 1, k, k, ata, ata, r, rr, r, rr);
        if r > 0 then
        begin lngdetsym:= - k; goto end end;
        mul 2( - r, - rr, de, dde, de, dde);
        sqrt 2( - r, - rr, r, rr); d[k]:= r; dd[k]:= rr;
        for j:= k + 1 step 1 until n do
        begin sub 2(matmat(1, k - 1, k, j, ata, ata) + matmat(1, k -
            1, j, k, ata, ata), 0, ata[k,j], ata[j,k], z, zz);
            lngtammat(1, k - 1, j, k, ata, ata, z, zz, z, zz);
            div 2( - z, - zz, r, rr, ata[k,j], ata[j,k])
        end;
    end;
    lngdetsym:= de;
end:
end lngdetsym;
```

Oplossingsmethode van Cholesky

lngdetsym

lngdetsym:= determinant van de n-de orde positief definiete symmetrische matrix M. De bovendriehoek van M is in dubbele lengte gegeven in array ata[1:n,1:n], d,dd[1:n] als volgt

$$M_{ij} = (\text{ata}[i,j], \text{ata}[j,i]), \qquad i < j,$$

$$M_{ii} = (\text{d}[i], \text{dd}[i]).$$

Bovendien wordt de Cholesky-matrix, U, van M (d.w.z. de bovendriehoeks-matrix die voldoet aan $U^TU = M$) in dubbele precisie berekend en de elementen van U worden op de plaats van de overeenkomstige elementen van M afgeleverd in ata, d en dd. Als M echter niet positief definiet is, wordt de Cholesky-ontbinding onderbroken en lngdetsym:= het tegengestelde van de indexwaarde waarvoor bleek dat M niet positief definiet is, zie Dekker (1968, section 220).
lngdetsym gebruikt sub2, mul2, div2, sqrt2, lngtammat, matmat en indirect mul12.

```
procedure lngsolsym(ata, n, d, dd, b, bb); value n; integer n;
array ata, b, bb, d, dd;
begin integer i;
    real z, zz;
    for i:= 1 step 1 until n do
    begin sub 2(matvec(1, i - 1, i, ata, b) + tamvec(1, i - 1, i,
        ata, bb), 0, b[i], bb[i], z, zz);
        lngtamvec(1, i - 1, i, ata, b, z, zz, z, zz);
        div 2( - z, - zz, d[i], dd[i], b[i], bb[i])
    end;
    for i:= n step - 1 until 1 do
    begin sub 2(tamvec(i + 1, n, i, ata, b) + matvec(i + 1, n, i,
        ata, bb), 0, b[i], bb[i], z, zz);
        lngmatvec(i + 1, n, i, ata, b, z, zz, z, zz);
        div 2( - z, - zz, d[i], dd[i], b[i], bb[i])
    end
end lngsolsym ;
```

lngsolsym

lngsolsym lost het n-de orde lineaire stelsel $U^TUX = B$ in dubbele precisie op. Hierin is U de n*n-bovendriehoeksmatrix, die in dubbele lengte gegeven is in <u>array</u> ata[1:n,1:n], d,dd[1:n] als volgt

$$U_{ij} = (ata[i,j], ata[j,i]), \quad i < j,$$

$$U_{ii} = (d[i], dd[i]),$$

en B de n-vector, die in dubbele lengte gegeven is in <u>array</u> b,bb[1:n] als volgt

$$B_i = (b[i], bb[i]).$$

De in dubbele precisie berekende oplossingsvector, X, wordt over de overeenkomstige elementen van B in b en bb heengeschreven.
lngsolsym gebruikt sub2, div2, lngmatvec, lngtamvec, matvec, tamvec en indirect mul12.

```
real procedure lngdetsolsym(ata, n, d, dd, b, bb); value n; integer n;
array ata, d, dd, b, bb;
begin real det;
    lngdetsolsym:= det:= lngdetsym(ata, n, d, dd);
    if det > 0 then lngsolsym(ata, n, d, dd, b, bb)
end lngdetsolsym;
```

lngdetsolsym

lngdetsolsym:= determinant van de n-de orde positief definiete symmetrische matrix M. De bovendriehoek van M is in dubbele lengte gegeven in array ata[1:n,1:n], d,dd[1:n] als volgt

$$M_{ij} = (\text{ata}[i,j] \quad \text{ata}[j,i]), \qquad i < j,$$

$$M_{ii} = (\text{d}[i], \text{dd}[i]).$$

Bovendien lost lngdetsolsym het n-de orde lineaire stelsel MX = B op in dubbele precisie. Hierin is B de n-vector die in dubbele lengte gegeven is in array b,bb[1:n] als volgt

$$B_i = (\text{b}[i], \text{bb}[i]).$$

De oplossingsvector, X, wordt over de overeenkomstige elementen van B in b en bb heen geschreven. Bovendien wordt de Cholesky-matrix U van M (d.w.z. de bovendriehoeksmatrix die voldoet aan $U^TU = M$) in dubbele precisie berekend en de elementen van U worden op de plaats van de overeenkomstige elementen van M afgeleverd in a, d en dd. Als M echter niet positief definiet is, wordt de Cholesky-ontbinding onderbroken en lngdetsolsym:= het tegengestelde van de indexwaarde waarvoor bleek dat M niet positief definiet is, zie Dekker (1968, section 220). lngdetsolsym gebruikt lngdetsym, lngsolsym en indirect sub2, mul12, mul2, div2, sqrt2, lngtammat, lngmatvec, lngtamvec, matmat, matvec en tamvec.

```
real procedure lnglsqsolcla(a, m, n, b, u, d, dd, x, xx); value m, n;
integer m, n; array a, b, u, d, dd, x, xx;
begin lngata(a, m, n, u, d, dd); lngatb(a, m, n, b, x, xx);
    lnglsqsolcla:= lngdetsolsym(u, n, d, dd, x, xx)
end lnglsqsolcla;
```

Het opstellen en oplossen van de normaalvergelijking (klassieke methode)

lnglsqsolcla

lnglsqsolcla:= determinant van de matrix $M = A^TA$, waarbij A de m*n-matrix is die gegeven is in array a[1:m,1:n] ($m \geq n$). De Cholesky-matrix, U, van M (d.w.z. de bovendriehoeksmatrix die voldoet aan $U^TU = M$) wordt in dubbele precisie berekend en in dubbele lengte afgeleverd in array u[1:n,1:n], d,dd[1:n] als volgt

$$U_{ij} = (u[i,j], u[j,i]), \qquad i < j$$

$$U_{ii} = (d[i], dd[i]).$$

Bovendien wordt in dubbele precisie de kleinste-kwadratenoplossing, X, van het stelsel [A:B] berekend, waarbij B de m-vector is, gegeven in array b[1:m]. X wordt in dubbele lengte afgeleverd in array c,cc[1:n] als volgt

$$X_i = (c[i], cc[i]).$$

lnglsqsolcla gebruikt lngata, lngatb, lngdetsolsym en indirect lngdetsym, lngsolsym, sub2, mul12, mul2, div2, sqrt2, lngtammat, lngtamvec, lngmatvec, matmat, matvec en tamvec.

```
procedure lnginvutm(u, n, du, ddu, iu, diu, ddiu); value n; integer n;
array iu, diu, ddiu, u, du, ddu;
begin real z, zz, x, xx;
    integer i, j;
    for i:= n step - 1 until 1 do
    begin div2(1, 0, du[i], ddu[i], diu[i], ddiu[i]);
        for j:= i + 1 step 1 until n do
        begin lngmatmat(i + 1, j - 1, i, j, iu, u, tammat(i + 1, j -
            1, i, j, iu, u) + mattam(i + 1, j - 1, i, j, iu, u), 0,
            z, zz); mul2(diu[i], ddiu[i], u[i,j], u[j,i], x, xx);
            add2(z, zz, x, xx, z, zz);
            div2( - z, - zz, du[j], ddu[j], iu[i,j], iu[j,i])
        end
    end
end lnginvutm;

procedure lnguut(u, n, d, dd); value n; integer n; array u, d, dd;
begin integer i, j;
    real x, xx, z, zz;
    for i:= 1 step 1 until n do
    begin mul2(d[i], dd[i], d[i], dd[i], x, xx);
        lngmattam(i + 1, n, i, i, u, u, 2 × matmat(i + 1, n, i, i,
        u, u), 0, z, zz); add2(x, xx, z, zz, d[i], dd[i]);
        for j:= i + 1 step 1 until n do
        begin mul2(d[j], dd[j], u[i,j], u[j,i], x, xx);
            lngmattam(j + 1, n, i, j, u, u, matmat(j + 1, n, i, j,
            u, u) + matmat(j + 1, n, j, i, u, u), 0, z, zz);
            add2(x, xx, z, zz, u[i,j], u[j,i])
        end
    end
end lnguut;
```

Het berekenen van bovengrenzen voor de fouten

lnginvutm

lnginvutm berekent in dubbele precisie de inverse van de n*n-bovendriehoeksmatrix U. De bovendriehoek van U is in dubbele lengte gegeven in array u[1:n,1:n], du,ddu[1:n] als volgt

$$U_{ij} = (u[i,j],\ u[j,i]), \qquad i < j,$$

$$U_{ii} = (du[i],\ ddu[i]).$$

De elementen van de bovendriehoek van U^{-1} worden op de overeenkomstige plaatsen van die van U afgeleverd in array iu[1:n,1:n], diu,ddiu[1:n]. lnginvutm gebruikt add2, mul2, div2, lngmatmat, lngtammat, mattam, tammat en indirect mul12.

lnguut

lnguut berekent in dubbele precisie de matrix UU^T. Hierin is U de n*n-bovendriehoeksmatrix, waarvan de bovendriehoek in dubbele lengte gegeven is in array u[1:n,1:n], d,dd[1:n] als volgt

$$U_{ij} = (u[i,j],\ u[j,i]), \qquad i < j,$$

$$U_{ii} = (du[i],\ ddu[i]).$$

De elementen van de bovendriehoek van UU^T worden op de overeenkomstige plaats als die van U afgeleverd in u, d en dd.
lnguut gebruikt add2, mul2, lngmattam, matmat en indirect mul12.

```
real procedure infnrmvec(b, n); value n; integer n; array b;
begin real g, h;
    integer i;
    g:= 0;
    for i:= 1 step 1 until n do
    begin h:= abs(b[i]); if g < h then g:= h end;
    infnrmvec:= g
end infnrmvec;

real procedure infnrmmat(i, n, j, a, b, f); value n;
integer i, n, j, a, b; real f;
begin real h, g;
    g:= 0;
    for i:= 1 step 1 until n do
    begin h:= SUM(j, a, b, abs(f)); if g < h then g:= h end;
    infnrmmat:= g
end infnrmmat;

real procedure claerb(a, m, n, b, u, du, ddu, x, g, eps, iu, diu,
ddiu); value m, n, g, eps; integer m, n; real g, eps;
array a, u, du, ddu, x, iu, diu, ddiu;
begin integer i, j;
    real nat, nx, nb, niu, niutu, nd, ne, nf;
    lnginvutm(u, n, du, ddu, iu, diu, ddiu);
    niu:= infnrmmat(i, n, j, i, n, if i = j then diu[i] else
    iu[i,j]); lnguut(iu, n, diu, ddiu);
    niutu:= infnrmmat(i, n, j, 1, n, if j < i then iu[j,i] else (if
    i = j then diu[i] else iu[i,j]));
    nat:= infnrmmat(i, n, j, 1, m, a[j,i]); nx:= infnrmvec(x, n);
    nb:= infnrmvec(b, n);
    nd:= ((n + 1) × (n + 2) / 2 - 2) × eps × g × 1.06;
    ne:= nat × infnrmmat(i, m, j, 1, n, a[i,j]) × m × eps × 1.06;
    nf:= (3 × n × (n - 1) / 2 + n + 1) × eps × g × 1.06;
    claerb:= (niutu × (1.06 × nat × nb × m × eps + (ne + nf + nd ×
    (infnrmmat(i, n, j, i, n, if i = j then du[i] else u[i,j]) +
    nd)) × nx) + nd × nx × niu) / (1 - niutu × (ne + nf))
end claerb;
```

infnrmvec

infnrmvec:= $||B||_\infty = \max_{1 \le i \le n} B_i$, waarbij B de n-vector is die gegeven is in array b[1:n].

infnrmmat

infnrmmat:= $\max_{1 \le i \le n} \left(\sum_{j=a}^{b} |f_{ij}| \right)$, waarbij f_{ij} de met i en j corresponderende waarde van de actuele expressie f voorstelt.

claerb

claerb:= een bovengrens voor $||X-\bar{X}||_\infty$, waarbij X de kleinste-kwadratenoplossing is van het stelsel [A:B]. A is gegeven in array a[1:m,1:n] ($m \ge n$), B in array b[1:m] en $\bar{X}$ in array x[1:n].
De bovendriehoek van de n*n-Cholesky-matrix U van A^TA is in dubbele lengte gegeven in array u[1:n,1:n], du,ddu[1:n] als volgt

$$U_{ij} = (u[i,j], u[j,i]), \qquad i < j,$$

$$U_{ii} = (du[i], ddu[i]).$$

Bovendien is het maximumelement van $M = A^TA$ gegeven in g.
Bij het berekenen van de bovengrens wordt er vanuit gegaan dat X berekend is volgens de klassieke methode (vergl. 4.23), waarbij eps de gebruikte rekenprecisie aangeeft.
De inverse van M wordt in dubbele precisie berekend en de elementen van M^{-1} worden op de overeenkomstige plaatsen van die van U afgeleverd in array iu[1:n,1:n], diu,ddiu[1:n].
claerb gebruikt infnrmvec, infnrmmat, lnginvutm, lnguut en indirect add2, mul12, mul2, div2, lngmatmat, lngmattam, matmat, mattam en tammat.

```
real procedure lnglsqsolclaerb(a, m, n, b, eps, x, xx, iu, diu, ddiu,
ubn); value n, m, eps; integer n, m; real eps, ubn;
array a, b, iu, diu, ddiu, x, xx;
begin integer i, j;
    real z, g;
    array ata[1:n,1:n], d, dd[1:n];
    lngata(a, m, n, ata, d, dd); g:= infnrmvec(d, n);
    lngatb(a, m, n, b, x, xx);
    z:= lnglsqsolclaerb:= lngdetsolsym(ata, n, d, dd, x, xx);
    if z > 0 then ubn:= claerb(a, m, n, b, ata, d, dd, x, g, eps,
    iu, diu, ddiu)
end lnglsqsolclaerb;
```

Klassieke methode met foutenberekening

lnglsqsolclaerb

lnglsqsolclaerb:= determinant van de n-de orde positief definiete symmetrische matrix M, $M = A^TA$, waarin A de m-n-matrix is die gegeven is in array a[1:m,1:n] ($m \geq n$).
Tevens wordt in dubbele precisie de kleinste-kwadratenoplossing, X, van het stelsel [A:B] berekend, waarin B de m-vector is die gegeven is in array b[1:m].
X wordt in dubbele lengte afgeleverd in array x,xx[1:n] als volgt

$$X_i = (x[i], xx[i]).$$

De inverse van M wordt berekend en afgeleverd in array iu[1:n,1:n], diu,ddiu[1:n] als volgt

$$M^{-1}_{ij} = (iu[i,j], iu[j,i]), \qquad i < j$$

$$M^{-1}_{ii} = (diu[i], ddiu[i]).$$

Tenslotte wordt een bovengrens, ubn, voor $||\bar{X}-X||_\infty$ berekend, waarbij eps de gebruikte rekenprecisie aangeeft.
lnglsqsolclaerb gebruikt lngata, lngatb, lngdetsolsym, claerb, infnrmvec en indirect add2, sub2, mul12, mul2, div2, sqrt2, lngmatmat, lngmattam, lngtammat, lngmatvec, lngtamvec, lngdetsym, lngsolsym, lnginvutm, lnguut, infnrmmat, matmat, mattam, tammat, matvec en tamvec.

```
real procedure eclnrmutm(r, n); value n; integer n; array r;
begin integer i;
    eclnrmutm:= sqrt(SUM(i, 1, n, tammat(1, i, i, i, r, r)))
end eclnrmutm;

real procedure eclnrmvec(v, n); value n; integer n; array v;
eclnrmvec:= sqrt(vecvec(1, n, 0, v, v));

real procedure baksuberb(r, diag, n, eps); value n, eps; integer n;
real eps; array r;
begin integer i, j;
    baksuberb:= 1.06 × eps × sqrt(SUM(i, 1, n - 1, SUM(j, 1, i - 1,
    (r[j,i] × (i - j + 3)) ↑ 2) + 4 × diag[i] ↑ 2 + ((n - i + 2) ×
    r[i,n]) ↑ 2) + diag[n] ↑ 2)
end baksuberb;
```

Foutenberekening van de QR-methode

Deze foutenberekening sluit aan op de QR-methode beschreven in Dekker (1968, section 224).

eclnrmutm

eclnrmutm:= $||R||_E$ (d.w.z. de Euclidische norm van R), waarbij R de n*n-bovendriehoeksmatrix is wiens bovendriehoek gegeven is in de bovendriehoek van array u[1:n,1:n].
eclnrmutm gebruikt tammat.

eclnrmvec

eclnrmvec:= $||V||_2$ (d.w.z. de Euclidische norm van V), waarin V de n-vector is die gegeven is in array v[1:n].
eclnrmvec gebruikt vecvec.

baksuberb

baksuberb:= een bovengrens voor de euclidische norm van de fout gemaakt bij een terugsubstitutie, waarbij eps de gebruikte rekenprecisie aangeeft (vergl. (4.13)).
De bovendriehoek van de n*n-bovendriehoeksmatrix U, die mede het linkerlid vormt, is gegeven in array diag[1:n] (de hoofddiagonaal) en de bovendriehoek van array u[1:n,1:n] (de rest).

```
real procedure nrmiup(r, n, diag, pc, ir); value n; integer n;
array r, ir, diag; integer array pc;
begin integer i, j;
    for i:= 1 step 1 until n do
    begin ir[i,i]:= diag[i];
        for j:= i + 1 step 1 until n do ir[i,j]:= r[i,j]
    end;
    invsym20(ir, n, pc); nrmiup:= sqrt(SUM(i, 1, n, ir[i,i]))
end nrmiup;

real procedure decerb(diag, m, n, eps); value m, n, eps; integer m, n;
real eps; array diag;
begin integer i;
    real h, b;
    h:= 0;
    for i:= 1 step 1 until n do
    begin b:= (14.12 + 5.83 × (m - i + 1)) × eps;
        h:= h × (1 + b) + b × abs(diag[i]) × (n - i + 1)
    end;
    decerb:= h
end decerb;
```

nrmiup

nrmiup:= $||R^{-1}||_E$ (d.w.z. de Euclidische norm van R^{-1}). Hierin is R de n*n-bovendriehoeksmatrix, waarvan de bovendriehoek gegeven is in array diag[1:n] (de hoofddiagonaal) en in de bovendriehoek van array r[1:n,1:n] (de rest). De kolomverwisselings indices, gegeven in integer array pc[1:n], definiëren de permutatie matrix P en er geldt $M = P^T R^T R P$, vergelijk Dekker (1968, section 224).
De bovendriehoek van M^{-1} wordt berekend en afgeleverd in de bovendriehoek van array ir[1:n,1:n].
nrmiup gebruikt invsym20 en indirect invsym2, ichcol, ichrow, ichrowcol, matvec en tamvec.

decerb

decerb:= een bovengrens van de euclidische norm van de fout, gemaakt bij de QR-ontbinding door middel van de procedure lsqdec (zie Dekker (1968, mca 2440)) van een m*n-matrix (vergelijk (5.34) en (5.35)). De hoofddiagonaal van de hierbij ontstane bovendriehoeksmatrix is gegeven in array diag[1:n]. De bij de ontbinding gebruikte precisie wordt aangeduid door eps.

```
real procedure decsolerb(r, m, n, diag, pc, x, menca, nb, eps, ir);
value m, n, menca, nb, eps; integer m, n; real menca, nb, eps;
array r, diag, ir, x; integer array pc;
begin integer i;
    real ne, nf, nE, na, nx, nir, z, s;
    nx:= eclnrmvec(x, n); nir:= nrmiup(r, n, diag, pc, ir);
    nE:= decerb(diag, m, n, eps);
    ne:= n × eps × (14.12 + 5.83 × m) × nb; na:= n × menca;
    nf:= baksuberb(r, diag, n, eps); z:= nir × nf × nx;
    s:= nir × nE × (1 + nir × (nE + nE + na));
    decsolerb:= z + (nir × (nir × nE × (nb + ne + nE × (nx + z)) +
    (sqrt(n) + nir × ne) × (ne + nE × (nx + z)))) / (1 - s)
end decsolerb;
```

decsolerb

decsolerb:= een bovengrens voor $||X-\bar{X}||_2$, waarbij X de kleinste-kwadratenoplossing is van het stelsel [A:B]. A is een m*n-matrix en X wordt verondersteld berekend te zijn volgens de QR-methode door middel van lsqdec, zie Dekker (1968, mca 2240) (vergl. (5.36)), waarbij eps de gebruikte precisie aangeeft.
Er geldt

$$M = A^TA = P^TR^TRP,$$

waarin R de n*n-bovendriehoeksmatrix is, waarvan de bovendriehoek gegeven is in array diag[1:n] (de hoofddiagonaal) en in de bovendriehoek van array r[1:n,1:n](de rest) en P is de permutatiematrix, gedefinieerd door de kolomverwisselingsindices gegeven in integer array pc[1:n]; vergelijk Dekker (1968, section 224).
De vector $\bar{X}$ is gegeven in array x[1:n].
Verder is gegeven $||B||_2$ in nb en $||A||_E/n$ in menca.
Bovendien wordt M^{-1} berekend en diens bovendriehoek wordt afgeleverd in de bovendriehoek van array ir[1:n,1:n].
decsolerb gebruikt eclnrmvec, decerb, nrmiup, baksuberb en indirect invsym20, invsym2, ichcol, ichrow, ichrowcol, matvec, tamvec en vecvec.

```
integer procedure lsqdecsolerb(a, m, n, aux, diag, pc, eps, b, ir,
ubn); value n, m, eps; integer n, m; real eps, ubn;
array a, aux, diag, b, ir; integer array pc;
begin integer i;
    real nb;
    lsqdecsolerb:= i:= lsqdec(a, m, n, aux, diag, pc); if i = n then
    begin nb:= eclnrmvec(b, m); lsqsol(a, m, n, diag, pc, b);
        ubn:= decsolerb(a, m, n, diag, pc, b, aux[1], nb, eps, ir)
    end
end lsqdecsolerb;
```

QR-methode met foutenberekening

lsqdecsolerb

lsqdecsolerb:= de rang, r, van de m*n-matrix A die gegeven is in array a[1:m,1:n] ($m \geq n$). Indien r = n, wordt de kleinste-kwadraten-oplossing, X, van het stelsel [A:B] berekend, waarbij B de m-vector gegeven in array b[1:m] is. De vector X wordt over B in b heen geschreven. A wordt ontbonden in een orthogonale matrix Q, een bovendriehoeksmatrix R en een permutatiematrix P (zodat A = QRP) die worden afgeleverd in a, array diag[1:n] en integer array pc[1:n], vergelijk Dekker (1968, section 224).
$(A^TA)^{-1}$ wordt berekend en de bovendriehoek daarvan wordt afgeleverd in de bovendriehoek van array ir[1:n,1:n]. Tenslotte wordt een bovengrens voor $||X-\bar{X}||_2$ (vergl. §5) afgeleverd in ubn, waarbij eps de gebruikte rekenprecisie aangeeft.
In array aux[0:2] moet men in aux[0] een relatieve tolerantie voor de bepaling van r meegeven; in dit array worden de volgende resultaten afgeleverd:

aux[1]:= de maximum euclidische norm van de kolommen van A;

aux[2]:= $|R_{rr}|$.

lsqdecsolerb gebruikt decsolerb, lsqdec, lsqsol en indirect eclnrmvec, decerb, nrmiup, baksuberb, invsym20, invsym2, ichcol, ichrow, ichrowcol, elmcol, elmveccol, tammat, matvec, tamvec en vecvec.

```
procedure lngatr(a, m, n, x, b, r); value m, n; integer m, n;
array a, r, x, b;
begin integer i;
    real z;
    array h, hh[1:m];
    for i:= 1 step 1 until m do lngmatvec(1, n, i, a, x, - b[i], 0,
    h[i], hh[i]);
    for i:= 1 step 1 until n do lngtamvec(1, m, i, a, h, tamvec(1,
    m, i, a, hh), 0, r[i], z)
end lngatr;

integer procedure lsqdeciti(a, m, n, aux, r, diag, pc); value n, m;
integer n, m; array a, r, aux, diag; integer array pc;
begin integer i, j;
    for i:= 1 step 1 until m do
    for j:= 1 step 1 until n do r[i,j]:= a[i,j];
    lsqdeciti:= lsqdec(r, m, n, aux, diag, pc);
    for i:= 1 step 1 until n do r[i,i]:= diag[i]
end lsqdeciti;
```

Iteratief verbeteren van de kleinste-kwadratenoplossing

lngatr

lngatr berekent $A^T(AX-B)$ in dubbele precisie. Hierin is A de m*n-matrix die gegeven is in array a[1:m,1:n], B de m-vector die gegeven is in array b[1:m] en X de n-vector die gegeven is in array x[1:n]. Het resultaat wordt (in enkele lengte) afgeleverd in array r[1:n].
lngatr gebruikt lngmatvec, lngtamvec en tamvec.

lsqdeciti

lsqdeciti:= de rang, r, van de m*n-matrix A, gegeven in array a[1:m,1:n]. Indien r = n wordt de QR-ontbinding van A uitgevoerd, resulterende in een bovendriehoeksmatrix R en een permutatiematrix P die (bij benadering) voldoen aan

$$A^TA = P^TR^TRP.$$

De bovendriehoek van R wordt opgeslagen in de bovendriehoek van array r[1:m,1:n], bovendien wordt de hoofddiagonaal extra opgeslagen in array diag[1:n]. De kolomverwisselingsindices die de matrix P definiëren worden afgeleverd in integer array pc[1:n]. In array aux[0:2] moet men in aux[0] een relatieve precisie voor de bepaling van r meegeven; in dit array worden de volgende resultaten afgeleverd:

aux[1]:= de maximum euclidische norm van de kolommen van A;
aux[2]:= $|R_{rr}|$.

lsqdeciti gebruikt lsqdec en indirect elmcol, ichcol en tammat.

```
procedure lsqsoliti(a, m, n, b, qr, pc, eps, aux, x); value n, m, eps;
integer n, m; real eps; array a, b, qr, pc, x, aux;
begin integer i, j, tel;
    real n1, n2, n3, h, hh;
    array r[1:n];
    tel:= 0;
    for i:= 1 step 1 until n do
    begin lngtamvec(1, m, i, a, b, 0, 0, h, hh); r[i]:= - h end;
    solsym20(qr, n, pc, r); n3:= n2:= 0;
    for i:= 1 step 1 until n do
    begin h:= r[i]; n2:= n2 + h × h; h:= x[i]:= - h; n3:= n3 + h × h
    end;
    lngatr(a, m, n, x, b, r);
imp: solsym20(qr, n, pc, r); n1:= n2; n3:= n2:= 0; tel:= tel + 1;
    for i:= 1 step 1 until n do
    begin h:= r[i]; n2:= n2 + h × h; h:= x[i]:= x[i] - h;
        n3:= n3 + h × h
    end;
    lngatr(a, m, n, x, b, r); if n2 < eps × n3 then goto l;
    if n2 > .5 × n1 then
    begin tel:= - tel; goto l end;
    goto imp;
l:  aux[3]:= eclnrmvec(r, n); aux[4]:= tel
end lsqsoliti;
```

lsqsoliti

lsqsoliti berekent de oplossing van het n-de orde lineaire stelsel

$$P^T R^T RPX = A^T B.$$

Hierin is R de n*n-bovendriekhoeksmatrix, gegeven in de bovendriehoek van array qr[1:n,1:n] en P de permutatiematrix gedefinieerd door integer array pc[1:n]. Verder is A de m*n-matrix, gegeven in array a[1:m,1:n] en B de m-vector, gegeven in array b[1:m].
Indien de driehoeksontbinding van A voldoende nauwkeurig blijkt, wordt de oplossing, X, iteratief verbeterd tot dat $||\delta X||_2 < \text{eps}\ ||X||_2$ en afgeleverd in array x[1:n].
In array aux[3:4] wordt afgeleverd

aux[3]:= $||A^T(AX-B)||_2$

aux[4]:= het aantal uitgevoerde iteraties (indien de gevraagde precisie niet bereikt is, een negatief getal).

lsqsoliti gebruikt lngatr, eclnrmvec, lngtamvec, solsym20 en indirect lngmatvec, matvec, tamvec en vecvec.

```
integer procedure lsqdecsoliti(a, m, n, b, eps, aux, r, diag, pc, x);
value m, n, eps; integer m, n; real eps; array a, b, x, aux, diag, r;
integer array pc;
begin integer i;
    i:= lsqdecsoliti:= lsqdeciti(a, m, n, aux, r, diag, pc);
    if i = n then lsqsoliti(a, m, n, b, r, pc, eps, aux, x)
end lsqdecsoliti;
```

lsqdecsoliti

lsqdecsoliti:= de rang, r, van de m*n-matrix A, gegeven in
array a[1:m,1:n].
Indien r = n wordt de QR-ontbinding van A uitgevoerd, resulterende in een bovendriehoeksmatrix R en een permutatie matrix P, zodanig dat (bij benadering) geldt

$$A^TA = P^TR^TRP.$$

De bovendriehoek van R wordt afgeleverd in de bovendriehoek van array r[1:n,1:n], de hoofddiagonaal wordt extra opgeslagen in array diag[1:n]. De kolomverwisselingsindices die P definiëren worden afgeleverd in integer array pc[1:n]. Eveneens wordt de kleinste-kwadratenoplossing, X, van het stelsel [A:B] berekend. Hierin is B de m-vector, gegeven in array b[1:m].
Indien de driehoeksontbinding van A voldoende nauwkeurig blijkt, wordt de oplossing iteratief verbeterd totdat $||\delta X||_2 < \text{eps}\ ||X||_2$ en afgeleverd in array x[1:n].
In array aux[0:4] moet men in aux[0] een relatieve precisie voor de bepaling van r meegeven; in dit array worden de volgende resultaten afgeleverd:

aux[1]:= de maximum euclidische norm van de kolommen van A;
aux[2]:= $|R_{rr}|$;
aux[3]:= $||A^T(AX-B)||_2$;
aux[4]:= het aantal uitgevoerde iteraties (indien de gevraagde precisie niet bereikt is, een negatief getal).

lsqdecsoliti gebruikt lsqdeciti, lsqsoliti en indirect lngatr, eclnrmvec, lngmatvec, lngtamvec, lsqdec, solsym20, elmcol, ichcol, tammat, matvec, tamvec en vecvec.

```
real procedure reserb(a, m, n, b, u, diag, pc, x, natr, menca, eps,
iu); value n, m, natr, menca, eps; integer n, m; real natr, menca, eps;
array a, b, x, diag, u, iu; integer array pc;
begin integer i;
    real ne, na, niu, z;
    array r[1:m];
    ne:= decerb(diag, m, n, eps); na:= n × menca;
    for i:= 1 step 1 until m do r[i]:= matvec(1, n, i, a, x) - b[i];
    niu:= nrmiup(u, n, diag, pc, iu);
    reserb:= niu × (niu × (natr + eclnrmvec(r, m) × na × 1.06 × eps
    × eps × m) + (sqrt(n) + niu × ne) × (1.06 × eps × (n + 1) × na
    × eclnrmvec(x, n) + eps × eclnrmvec(b, m))) / (1 - niu × ne × (1
    + niu × (ne + ne + na)))
end reserb;
```

Foutenschatting m.b.v. residuberekening

reserb

reserb:= een bovengrens van $||X-\bar{X}||_2$, waarbij X de kleinste-kwadraten-oplossing is van het stelsel [A:B] (vergl. §6). Hierin is A de m*n-matrix, gegeven in array a[1:m,1:n] ($m \geq n$) en B de m-vector, gegeven in array b[1:m].
Verder is gegeven de bij de berekening van X gebruikte bovendriehoeks-matrix R en de permutatiematrix P, zodat (bij benadering) geldt

$$M = A^TA = P^TR^TRP.$$

De bovendriehoek van R is gegeven in array diag[1:n] (de hoofddiagonaal) en in de bovendriehoek van array r[1:n,1:n] (de rest). De kolomverwis-selingsindices die P definiëren zijn gegeven in integer array pc[1:n].
Bovendien moet meegegeven worden
$||A^T(A\bar{X}-B)||_2$ in natr
$||A||_E/n$ in menca.
De bovendriehoek van M^{-1} wordt berekend en afgeleverd in de bovendrie-hoek van array ir[1:n,1:n].
reserb gebruikt decerb, eclnrmvec, nrmiup en indirect invsym20, invsym2, ichcol, ichrow, ichrowcol, matvec, tamvec en vecvec.

```
integer procedure lsqdecsolitierb(a, m, n, b, eps1, eps2, aux, x, r,
diag, pc, ir, ubn); value n, m, eps1, eps2; integer n, m;
real eps1, eps2, ubn; array a, b, x, r, diag, aux, ir;
integer array pc;
begin integer i;
    i:= lsqdecsolitierb:= lsqdecsoliti(a, m, n, b, eps1, aux, r,
    diag, pc, x);
    if i = n then ubn:= reserb(a, m, n, b, r, diag, pc, x, aux[3],
    aux[1], eps2, ir)
end lsqdecsolitierb;
```

Iteratief verbeteren plus foutenschatting

lsqdecsolitierb

lsqdecsolitierb:= de rang, r, van de m*n-matrix A, gegeven in array a[1:m,1:n].
Indien r = n wordt de QR-ontbinding van A uitgevoerd, resulterende in een bovendriehoeksmatrix R en een permutatiematrix P die (bij benadering) voldoen aan

$$M = A^TA = P^TR^TRP.$$

De bovendriehoek van R wordt opgeslagen in de bovendriehoek van array r[1:m,1:n], bovendien wordt de hoofddiagonaal extra opgeslagen in array diag[1:n]. De kolomverwisselingsindices die P definiëren worden afgeleverd in integer array pc[1:n].
Eveneens wordt de kleinste-kwadratenoplossing, X, van het stelsel [A:B] berekend, hierin is B de m-vector, gegeven in array b[1:m].
Indien de driehoeksontbinding van A voldoende nauwkeurig blijkt, wordt de oplossing iteratief verbeterd totdat $||\delta X||_2 < eps1\ ||X||_2$ en afgeleverd in array x[1:n].
Een bovengrens voor $||X-\bar{X}||_2$ wordt berekend en afgeleverd in ubn, waarbij eps2 de gebruikte rekenprecisie aangeeft.
In array aux[0:4] moet men in aux[0] een relatieve precisie voor de bepaling van r meegeven; in dit array worden de volgende resultaten afgeleverd:

aux[1]:= de maximum euclidische norm van de kolommen van A;
aux[2]:= $|R_{rr}|$;
aux[3]:= $||A^T(A\bar{X}-B)||_2$;
aux[4]:= het aantal uitgevoerde iteraties (indien de gevraagde precisie niet bereikt is, een negatief getal).

lsqdecsolitierb gebruikt lsqdecsoliti, reserb en indirect lsqdeciti, lsqsoliti, lngatr, eclnrmvec, nrmiup, decerb, lngmatvec, lngtamvec, invsym20, invsym2, ichcol, ichrow, ichrowcol, tammat, matvec, tamvec en vecvec.

```
procedure tfmbidiag(a, m, n, q, e, b, aux); value m, n; integer m, n;
array a, q, e, aux, b;
begin integer i, j, k, l;
    real x, y, s, f, g, h, tol;
    g:= x:= 0; tol:= aux[1];
    for i:= 1 step 1 until n do
    begin e[i]:= g; l:= i + 1; s:= tammat(i, m, i, i, a, a);
        if s < tol then g:= 0 else
        begin f:= a[i,i]; g:= if f < 0 then sqrt(s) else - sqrt(s);
            h:= f × g - s; a[i,i]:= f - g;
            for j:= l step 1 until n do elmcol(i, m, j, i, a, a,
            tammat(i, m, i, j, a, a) / h);
            elmveccol(i, m, i, b, a, tamvec(i, m, i, a, b) / h)
        end;
        q[i]:= g; s:= mattam(l, n, i, i, a, a);
        if s < tol then g:= 0 else
        begin f:= a[i,i + 1];
            g:= if f < 0 then sqrt(s) else - sqrt(s); h:= f × g - s;
            a[i,i + 1]:= f - g;
            for j:= l step 1 until n do e[j]:= a[i,j] / h;
            for j:= l step 1 until m do
            begin s:= mattam(l, n, j, i, a, a);
                for k:= l step 1 until n do a[j,k]:= a[j,k] + s ×
                e[k]
            end
        end;
        y:= abs(q[i]) + abs(e[i]); if y > x then x:= y
    end;
    aux[2]:= x × aux[0];
    for i:= n step - 1 until 1 do
    begin if g ≠ 0 then
        begin h:= a[i,i + 1] × g;
            for j:= l step 1 until n do a[j,i]:= a[i,j] / h;
            for j:= l step 1 until n do elmcol(l, n, j, i, a, a,
            matmat(l, n, i, j, a, a))
        end;
        for j:= l step 1 until n do a[i,j]:= a[j,i]:= 0; a[i,i]:= 1;
        g:= e[i]; l:= i
    end
end tfmbidiag;
```

Berekening van kleinste-kwadratenoplossing van minimale lengte

tfmbidiag

tfmbidiag brengt de m*n-matrix gegeven in array a[1:m,1:n] met behulp van Householder-transformaties terug tot een bidiagonale bovendriehoeksmatrix R. De hoofddiagonaal van R wordt afgeleverd in array q[1:n] en de subdiagonaal in de laatste n-1 elementen van array e[1:n] (e[1] bevat geen subdiagonaal element doch wordt nul). De rechtertransformaties vormen de n*n-orthogonale matrix V die overschreven wordt in a. De vector gegeven in array b[1:m] wordt voorvermenigvuldigd met de linkertransformaties, het resultaat wordt overschreven in b.
In array aux[0:2] moet men in aux[0] een relatieve precisie meegeven en in aux[1] een getal ter grootte van β/aux[0], waarin β het kleinste positieve getal ongelijk nul dat in de machine representeerbaar is, voorstelt.
In aux[2] wordt de waarde van $||R||_\infty$ * aux[0] afgeleverd.
tfmbidiag gebruikt elmcol, elmveccol, mattam, tammat en tamvec.

```
procedure eigbidiag(v, n, q, e, b, aux); value n; integer n;
array v, q, e, b, aux;
begin integer i, j, k, l, l1;
    real c, s, f, g, h, x, y, z, eps;
    eps:= aux[2];
    for k:= n step - 1 until 1 do
    begin
    tfs: for l:= k step - 1 until 1 do
        begin if abs(e[l]) < eps then goto tfc;
            if abs(q[l - 1]) ≤ eps then goto ca
        end;
    ca: c:= 0; s:= 1; l1:= l - 1;
        for i:= l step 1 until k do
        begin f:= s × e[i]; e[i]:= c × e[i];
            if abs(f) < eps then goto tfc; g:= q[i];
            q[i]:= h:= sqrt(f × f + g × g); c:= g / h; s:= - f / h;
            y:= b[l1]; z:= b[i]; b[l1]:= c × y + s × z;
            b[i]:= - s × y + c × z
        end;
    tfc: z:= q[k]; if l = k then goto co; x:= q[l]; y:= q[k - 1];
        g:= e[k - 1]; h:= e[k];
        f:= ((y - z) × (y + z) + (g - h) × (g + h)) / (2 × h × y);
        g:= sqrt(f × f + 1);
        f:= ((x - z) × (x + z) + h × (y / (if f < 0 then f - g else
        f + g) - h)) / x; c:= s:= 1;
        for i:= l + 1 step 1 until k do
        begin g:= e[i]; y:= q[i]; h:= s × g; g:= c × g;
            e[i - 1]:= z:= sqrt(f × f + h × h); c:= f / z; s:= h / z;
            f:= x × c + g × s; g:= - x × s + g × c; h:= y × s;
            y:= y × c; rotcol(1, n, i - 1, i, v, c, s);
            q[i - 1]:= z:= sqrt(f × f + h × h); c:= f / z; s:= h / z;
            f:= c × g + s × y; x:= - s × g + c × y; y:= b[i - 1];
            z:= b[i]; b[i - 1]:= c × y + s × z;
            b[i]:= - s × y + c × z
        end;
        e[l]:= 0; e[k]:= f; q[k]:= x; goto tfs;
    co: if z < 0 then
        begin q[k]:= - z;
            for j:= 1 step 1 until n do v[j,k]:= - v[j,k]
        end
    end
end eigbidiag;
```

eigbidiag

eigbidiag berekent de eigenwaarden van de n*n-bidiagonale bovendriehoeksmatrix R d.m.v. QR-iteratie. De hoogddiagonaal van R is gegeven in array q[1:n] en de subdiagonaal in de laatste n-1 elementen van array e[1:n] (e[1] moet nul zijn). In array aux[2:2] moet men een absolute tolerantie meegeven. De eigenwaarden worden afgeleverd in q en de inhoud van e gaat verloren. De rotaties die R in de diagonaalmatrix der eigenwaarden transformeren worden ook uitgevoerd op de vector gegeven in array b[1:n] en de matrix gegeven in array v[1:n,1:n].
eigbidiag gebruikt rotcol.

```
procedure sngvalsol(v, n, s, x, aux); value n; integer n;
array v, s, x, aux;
begin integer i;
    array h[1:n];
    real eps;
    eps:= aux[3];
    for i:= 1 step 1 until n do h[i]:= if s[i] < eps then 0 else x[i]
    / s[i];
    for i:= 1 step 1 until n do x[i]:= matvec(1, n, i, v, h);
end sngvalsol;

procedure minlsqsol(a, m, n, b, aux); value m, n; integer m, n;
array a, b, aux;
begin array q, e[1:n];
    tfmbidiag(a, m, n, q, e, b, aux); eigbidiag(a, n, q, e, b, aux);
    sngvalsol(a, n, q, b, aux)
end minlsqsol;
```

sngvalsol

sngvalsol bepaalt de pseudo-inverse, P, van de n*n-matrix diag(σ_i), hiervoor moet men in array aux[3:3] een absolute precisie meegeven. De hoofddiagonaal van diag(σ_i) is gegeven in array s[1:n].
Vervolgens wordt de vector VPX bepaald, hierin is V de n*n-matrix, gegeven in array v[1:n,1:n] en X de n-vector, gegeven in array x[1:n]. Het resultaat wordt overschreven in x.
sngvalsol gebruikt matvec.

minlsqsol

minlsqsol bepaalt de kleinste-kwadratenoplossing van het stelsel [A:b] met de kleinste euclidische lengte. De m*n-matrix A is gegeven in array a[1:m,1:n] en de vector b is gegeven in array b[1:m]. De oplossingsvector wordt overschreven in b.
In array[0:3] moet men meegeven
in aux[0] een relatieve tolerantie,
in aux[1] een getal ter grootte β/aux[0], waarin β het kleinste positieve getal ongelijk nul dat in de machine representeerbaar is, voorstelt,
in aux[3] een getal dat de minimum waarde van de singuliere waarden ongelijk nul aangeeft.
minslqsol gebruikt tfmbidiag, eigbidiag, singvalsol en indirect elmveccol, elmcol, rotcol, mattam, tammat, matvec en tamvec.

8. Voorbeelden

Het geval rang(A) = n.

matrix A

+5	+30	+70	+70	+42
+5	+40	+105	+112	+70
+5	+45	+126	+140	+90
+5	+48	+140	+160	+105
+3	+30	+90	+105	+70
−0	−1	−1	−1	−1
+1	−0	−1	−1	−1
+1	+1	−0	−1	−1
+1	+1	+1	−0	−1
+1	+1	+1	+1	−0
+1	+1	+1	+1	+1

x(exact)	b	r
−1	−14	+3
+1	−45	−17
−1	+5	+41
+1	−85	−43
−1	−1	+27
	+1	+1
	−1	−1
	+1	+1
	−1	−1
	+1	+1
	−2	−1

klassieke methode (dubbele lengte) ||.|| duidt de oneindig-norm aan

x(berekend)	delta x
$-.99999999999999999940464_{10}+0$	$-.5953580496533_{10}-19$
$+.99999999999999999904174_{10}+0$	$+.9582553294309_{10}-19$
$-.99999999999999999895366_{10}+0$	$-.10463392557\,49_{10}-18$
$+.99999999999999999897326_{10}+0$	$+.1026743879695_{10}-18$
$-.99999999999999999942095_{10}+0$	$-.5790533283563_{10}-19$

de berekende bovengrens voor ||delta x|| is $+.15885465617\,28_{10}-15$
||delta x|| is $+.10463392557\,49_{10}-18$

qr-methode ||.|| duidt de euclidische norm aan

x(berekend)	delta x
$-.99999999554\,53_{10}-0$	$-.44547050492\,84_{10}-8$
$+.99999999346\,71_{10}-0$	$+.6532900442835_{10}-8$
$-.99999999158\,35_{10}-0$	$-.8416463970207_{10}-8$
$+.99999999004\,92_{10}-0$	$+.9950781532098_{10}-8$
$-.99999999328\,25_{10}-0$	$-.6717527867295_{10}-8$

de berekende bovengrens voor ||delta x|| is $+.13960565284\,73_{10}-4$
||delta x|| is $+.16658429841\,42_{10}-7$

matrix A

+3	+6	+10
+3	+8	+15
+1	+3	+6
−0	−1	−1
+1	−0	−1
+1	+1	−0
+1	+1	+1

x(exact)	b	r
−0	+13	+1
+2	+15	−1
−0	+7	+1
	−1	+1
	−1	−1
	+3	+1
	+1	−1

klassieke methode (dubbele lengte) ||.|| duidt de oneindig-norm aan

x(berekend)	delta x
$+.258082641344445553908989_{10}-22$	$-.25808264134444_{10}-22$
$+.19999999999999999999996_{10}+1$	$+.4178977606954_{10}-22$
$+.17131456341199760982107 6_{10}-22$	$-.1713145634119_{10}-22$

de berekende bovengrens voor ||delta x|| is $+.8493440022604_{10}-20$
||delta x|| is $+.4178977606954_{10}-22$

qr-methode (met iteratief verbeteren)
||.|| duidt de euclidisch norm aan

x(berekend)	delta x
$+.1694065894509_{10}-20$	$-.1694065894509_{10}-20$
$+2$	-0
$-.1270549420881_{10}-20$	$+.1270549420881_{10}-20$

aantal iteraties is $+1$
||residu|| is $+.3749001219622_{10}-18$
de berekende bovengrens van ||delta x|| is $+.1553791135406_{10}-9$
||delta x|| is $+.2117582368136_{10}-20$

matrix A

+3	+6	+10	+3
+3	+8	+15	+3
+1	+3	+6	+1
+5	+48	+140	+5
+3	+30	+90	+3
+14	+144	+945	+14
+2	+21	+140	+2

x(exact)	b	r
+1	−424	−474
+4	+1589	+1521
+2	−3129	−3155
+1	+483	+1
	+305	−1
	+2495	+1
	+367	−1

x(berekend)		delta x	
$+.1000000001904_{10}+$	1	$-.1904481905513_{10}-$	8
$+.3999999999058_{10}+$	1	$+.9422365110368_{10}-$	9
$+.2000000000080_{10}+$	1	$-.8003553375602_{10}-$	10
$+.1000000001904_{10}+$	1	$-.1904481905513_{10}-$	8

9. Literatuur

P.A. Businger & G.H. Golub
Linear least squares solution by Householder transformations,
Numerische Mathematik 7 (1965), blz. 206-216 en 269-276.

P.A. Businger & G.H. Golub
Singular value decomposition of a complex matrix,
Algorithm 358 (FORTRAN),
CACM 12 (1969), blz. 564-565.

T.J. Dekker
ALGOL 60-procedures in numerical algebra,
MC tract 22 (1968).

T.J. Dekker
A floating-point technique for extending the available precision,
MC, MR 118 (1970).

T.J. Dekker
Numerieke Algebra,
MC Syllabus 12 (1971).

G.H. Golub & C. Reinsch
Singular value decomposition and least squares solutions,
Numerische Mathematik 14 (1970), blz. 403-420.

G.H. Golub & J.H. Wilkinson
Note on the iterative refinement of least squares solutions,
Numerische Mathematik 9 (1966), blz. 199-248.

W. Hoffmann & J.P. Hollenberg
Foutenberekening voor lineaire kleinste-kwadratenproblemen,
MC, NR 19 (1971).

J.H. Wilkinson
The Algebraic eigenvalue problem,
Clarendon-Press (1965).