Numerieke Methodes in de Algebra

Prof. Dr. Guido Vanden Berghe

Chapter 2

Numerieke oplossing van stelsels lineaire vergelijkingen

Doelstelling

Veel problemen in het numeriek rekenen kunnen herleid worden tot het oplossen van een stelsel lineaire algebraïsche vergelijkingen. We denken aan kleinste kwadratenvraagstukken, het numeriek integreren van differentiaalvergelijkingen, etc,.... In dit hoofdstuk zullen we de bekende Gauss eliminatiemethode bespreken vanuit numeriek standpunt. Topics zoals nauwkeurigheid, aantal rekenkundige bewerkingen, stabiliteit zullen aan de orde komen. Varianten op die methode, heel vaak bruikbaar voor stelsels met coëfficiëntenmatrices met speciale eigenschappen, worden geïntroduceerd. Een reeks iteratieve methoden, zoals de Gauss-Jacobi, de Gauss-Seidel en de SOR methode sluiten het hoofdstuk af.

Zeer veel problemen in de toegepaste wiskunde kunnen worden teruggebracht tot het oplossen van een stelsel lineaire algebraïsche vergelijkingen. In de meeste gevallen worden we geconfronteerd met zgn. stelsels van Cramer van n lineaire vergelijkingen met n onbekenden, nl.

$$Ax = b \tag{2.1}$$

met A een matrix van orde $n \times n$

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \quad \text{of} \quad A = [a_{ij}] \quad (1 \leq i,\ j \leq n) \tag{2.2}$$

en

$$x = [x_1, \dots, x_n]^T \ ,\ b = [b_1, \dots, b_n]^T \tag{2.3}$$

met T de notatie voor transponeren.

Vermits A normalerwijze opgeslagen moet worden in het geheugen van de computer, zullen de geheugenbegrenzingen de orde van de mogelijke stelsels limiteren. De meeste algoritmen, gebruikt voor het oplossen van dergelijke stelsels, steunen op de Gauss eliminatiemethode (zie cursus Algebra en volgende paragraaf). Het is een rechtstreekse methode in de zin dat, wanneer afrondingsfouten geïgnoreerd worden, de exacte oplossing in een eindig aantal stappen wordt bereikt.

Een tweede belangrijk type van problemen resulteert in het oplossen van $Ax = b$ met A opnieuw een vierkante matrix met de volgende typische eigenschappen :

(a) de orde van A is groot, bv. $100\,000 \times 100\,000$ of meer
(b) de meeste coëfficiënten in A zijn nul (zgn. "sparse" of ijle matrices).

Dergelijke stelsels kunnen niet opgelost worden met een rechtstreekse methode als de Gauss eliminatiemethode. Meestal gaat men in deze gevallen steunen op iteratieve methodes.

2.1 Gauss eliminatie

Gauss eliminatie is een algemene naam gegeven aan de oplossingsmethode waarbij opeenvolgend onbekenden x_i geëlimineerd worden en het stelsel gereduceerd wordt tot stelsels van lagere orde. M.a.w. om $Ax = b$ op te lossen zullen we het reduceren tot een equivalent stelsel $Ux = g$, waarbij U een boventriangulaire matrix is. Dit gebeurt in een reeks stappen :

Stap 1 :

Noteren we het oorspronkelijk stelsel voor verdere eenvoud van notatie als $A^{(1)}x = b^{(1)}$, met

$$A^{(1)} = [a_{ij}^{(1)}] \qquad b^{(1)} = [b_1^{(1)}, \ldots, b_n^{(1)}]^T \qquad 1 \leq i\,,\, j \leq n\,.$$

Veronderstel $a_{11}^{(1)} \neq 0$; definieer dan de rij vermenigvuldigers

$$m_{i1} = a_{i1}^{(1)}/a_{11}^{(1)} \quad , \quad i = 2, 3, \ldots, n\,.$$

Deze worden vervolgens aangewend voor de eliminatie van de x_1 term uit de vergelijkingen 2 t.e.m. n, m.b.v.

$$\begin{aligned} a_{ij}^{(2)} &= a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)} && i, j = 2, \ldots, n \\ b_i^{(2)} &= b_i^{(1)} - m_{i1}b_1^{(1)} && i = 2, \ldots, n\,. \end{aligned}$$

Aldus blijven de eerste rijen van A en b onveranderd en de eerste kolom van $A^{(1)}$ beneden de diagonaal wordt nul. Het equivalente nieuwe stelsel $A^{(2)}x = b^{(2)}$ heeft

de vorm

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(2)} \end{bmatrix}.$$

We vervolgen met het elimineren van onbekenden en dit kan algemeen uitgedrukt worden als :

Stap k : $(1 < k \leq n-1)$

Neem aan dat $A^{(k)}x = b^{(k)}$ reeds geconstrueerd werd, waarbij $x_1, x_2, \ldots, x_{k-1}$ geëlimineerd werden in de verschillende stappen en dat $A^{(k)}$ de vorm bezit :

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & \cdots & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & & & & a_{2n}^{(2)} \\ \vdots & & \ddots & & & \vdots \\ 0 & \cdots & 0 & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} \end{bmatrix}.$$

Voor $a_{kk}^{(k)} \neq 0$ definieer

$$m_{ik} = a_{ik}^{(k)} / a_{kk}^{(k)} \qquad i = k+1, \ldots, n\,. \tag{2.4}$$

Gebruik deze om de onbekende x_k te verwijderen uit de vergelijkingen $k+1$ t.e.m. n door

$$\begin{aligned} a_{ij}^{(k+1)} &= a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)} \\ b_i^{(k+1)} &= b_i^{(k)} - m_{ik} b_k^{(k)} \end{aligned} \qquad i, j = k+1, \ldots, n \tag{2.5}$$

Hierdoor blijven de rijen 1 t.e.m. k onveranderd en worden nullen geïntroduceerd in kolom k onder het diagonaal element. Door dit uit te voeren tot $k = n-1$ bekomen we $A^{(n)}x = b^{(n)}$

$$\begin{bmatrix} a_{11}^{(1)} & \cdots & \cdots & a_{1n}^{(1)} \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & & & a_{nn}^{(n)} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ \vdots \\ \vdots \\ b_n^{(n)} \end{bmatrix}.$$

Bovenstaande stappen zijn in een beknopte vorm neer te schrijven in volgend algoritme:

Algoritme 2.2.1

input $n, [a_{ij}], [b_i]$
for $k = 1, 2, \ldots, n-1$ **do**
 for $i = k+1, k+2, \ldots, n$ **do**
  $m_{ik} \leftarrow a_{ik}/a_{kk}$
  $a_{ik} \leftarrow 0$
  for $j = k+1, k+2, \ldots, n$ **do**
   $a_{ij} \leftarrow a_{ij} - m_{ik}a_{kj}$
  end for
  $b_i \leftarrow b_i - m_{ik}b_k$
 end for
end for
output $[a_{ij}], [b_i]$
end

Hier is dus stilzwijgend verondersteld dat alle diagonaalelementen (dit zijn de zo genoemde *pivotelementen*) verschillend van nul zijn. De vermenigvuldigers m_{ik} zijn zo gekozen dat alle waarden onder de diagonaal in A nul worden. In plaats van die berekening uit te voeren, hebben we eenvoudig die elementen opgevuld met 0.

Voor eenvoud van notatie stel $U = A^{(n)}$ en $g = b^{(n)}$. Het systeem $Ux = g$ is boventriangulair en gemakkelijk op te lossen.
Vooreerst

$$x_n = g_n/u_{nn}$$

en dan is

$$x_k = \frac{1}{u_{kk}}(g_k - \sum_{j=k+1}^{n} u_{kj}x_j) \ , \quad k = n-1, n-2, \ldots, 1 \tag{2.6}$$

Voorbeeld 2.1.1

Los op

$$\begin{cases} x_1 + 2x_2 + x_3 = 0 \\ 2x_1 + 2x_2 + 3x_3 = 3 \\ -x_1 - 3x_2 = 2 \end{cases} \tag{2.7}$$

De gerande coëfficiëntenmatrix

$$[A \mid b] = \left[\begin{array}{ccc|c} 1 & 2 & 1 & 0 \\ 2 & 2 & 3 & 3 \\ -1 & -3 & 0 & 2 \end{array}\right] .$$

Diagrammatisch levert bovenstaande methode

$$\left[\begin{array}{ccc|c} 1 & 2 & 1 & 0 \\ 2 & 2 & 3 & 3 \\ -1 & -3 & 0 & 2 \end{array}\right] \begin{array}{c} \longrightarrow \\ m_{21}=2 \\ m_{31}=-1 \end{array} \left[\begin{array}{ccc|c} 1 & 2 & 1 & 0 \\ 0 & -2 & 1 & 3 \\ 0 & -1 & 1 & 2 \end{array}\right]$$

$$\begin{array}{c} \longrightarrow \\ m_{32}=1/2 \end{array} \left[\begin{array}{ccc|c} 1 & 2 & 1 & 0 \\ 0 & -2 & 1 & 3 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{array}\right] = \Big[U \mid g\Big]$$

waaruit $x_3 = 1 \qquad x_2 = -1 \qquad x_1 = 1\,.$ $\diamond$

2.1.1 Triangulaire factorizatie van een matrix

Het is interessant om de vermenigvuldigers m_{ij} ter beschikking te houden, vermits we dikwijls zullen wensen $Ax = b$ op te lossen met dezelfde A maar met een verschillende vector b. Bij het programmeren van het Gauss algoritme kunnen de $a_{ij}^{(k+1)}$, $j > i$ opgeborgen worden in de geheugenplaatsen voorbehouden voor $a_{ij}^{(k)}$. De elementen onder de diagonaal werden nul gemaakt, wat wil zeggen dat deze geheugenplaatsen kunnen gebezigd worden voor de opberging van de elementen m_{ij} , m.a.w. bewaar m_{ij} in de ruimte origineel gereserveerd voor $a_{ij} \;\; i > j$. Er is daarenboven nog een andere reden om de vermenigvuldigers m_{ij} te interpreteren als elementen van een matrix.

Theorema 2.1.1

Introduceren we de benedentriangulaire matrix

$$L = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ m_{21} & 1 & 0 & & 0 \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ m_{n1} & m_{n2} & \cdots & \cdots & 1 \end{bmatrix}$$

dan kan aangetoond worden dat

$$A = LU \tag{2.8}$$

met A en U de matrices geïntroduceerd in de Gauss eliminatie methode.

Bewijs

Het bewijs hiervan steunt op de definities (2.4) en (2.5). Om het matrixelement $(LU)_{ij}$ te visualiseren, kunnen we gebruik maken van de vector vermenigvuldiging

$$(LU)_{ij} = [m_{i1}, \ldots, m_{i\,i-1}, 1, 0, \ldots, 0] \begin{bmatrix} u_{1j} \\ \vdots \\ u_{jj} \\ 0 \\ \vdots \\ 0 \end{bmatrix} .$$

Voor $i \leq j$

$$\begin{aligned} (LU)_{ij} &= m_{i\,1}u_{1\,j} + m_{i\,2}u_{2\,j} + \ldots + m_{i\,i-1}u_{i-1j} + u_{i\,j} \\ &= \sum_{k=1}^{i-1} m_{ik}a_{kj}^{(k)} + a_{ij}^{(i)} \\ &= \sum_{k=1}^{i-1} \left[a_{ij}^{(k)} - a_{ij}^{(k+1)}\right] + a_{ij}^{(i)} \\ &= a_{ij}^{(1)} = a_{ij} \, . \end{aligned}$$

Voor $i > j$

$$\begin{aligned} (LU)_{ij} &= m_{i\,1}u_{1j} + \ldots + m_{i\,j}u_{j\,j} \\ &= \sum_{k=1}^{j-1} m_{ik}a_{kj}^{(k)} + m_{ij}a_{jj}^{(j)} \\ &= \sum_{k=1}^{j-1} \left[a_{ij}^{(k)} - a_{ij}^{(k+1)}\right] + a_{ij}^{(j)} = a_{ij}^{(1)} = a_{ij} \, . \end{aligned}$$

◇

Uit (2.8) volgt dan

$$\det(A) = \det(L)\det(U) \, .$$

Vermits L en U triangulair zijn, volgen hun determinanten uit het product van hun diagonaalelementen. Vermits $\det(L) = 1$ volgt hieruit

$$\begin{aligned} \det(A) = \det(U) &= u_{11}u_{22}\ldots u_{nn} \\ &= a_{11}^{(1)}a_{22}^{(2)}\ldots a_{nn}^{(n)} \, . \end{aligned}$$

Bovenstaande werkwijze herleidt dus de berekening van de waarde van de determinant van de n^{de} graad tot een vermenigvuldiging van n factoren.

Voorbeeld 2.1.2

Voor stelsel (2.7) uit vorig voorbeeld is

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & \frac{1}{2} & 1 \end{bmatrix} \qquad U = \begin{bmatrix} 1 & 2 & 1 \\ 0 & -2 & 1 \\ 0 & 0 & \frac{1}{2} \end{bmatrix} \quad \text{en} \quad \det(A) = \det(U) = -1\,.$$

$\diamond$

DEMO 2 (Maple)

Zie Claroline

2.1.2 Aantal elementaire rekenkundige bewerkingen

Om het aantal bewerkingen te analyseren, nodig om $Ax = b$ op te lossen m.b.v. de Gauss eliminatieprocedure, beschouwen we afzonderlijk de opbouw van L en U uit A, de wijziging van b naar g en finaal de bepaling van x.

(**a**) Berekening van L en U :

Bij stap 1 werden $(n-1)$ delingen gebruikt om de m_{i1} $(2 \leq i \leq n)$ te bepalen. Daarop volgden $(n-1)^2$ vermenigvuldigingen en $(n-1)^2$ optellingen om $a_{ij}^{(2)}$ te berekenen. We kunnen op deze wijze verder redeneren voor elke stap. Het bekomen resultaat is opgesomd in onderstaande tabel.

Stap k	Optellingen	Vermenig-vuldigingen	Delingen
1	$(n-1)^2$	$(n-1)^2$	$n-1$
2	$(n-2)^2$	$(n-2)^2$	$n-2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n-1$	1	1	1
totaal	$\dfrac{n(n-1)(2n-1)}{6}$	$\dfrac{n(n-1)(2n-1)}{6}$	$\dfrac{n(n-1)}{2}$

De totale waarde voor elke kolom werd bekomen door gebruik te maken van

$$\sum_{j=1}^{p} j = \frac{p(p+1)}{2} \qquad \sum_{j=1}^{p} j^2 = \frac{p(p+1)(2p+1)}{6} \qquad p \geq 1\,.$$

Meestal beschouwde men het aantal vermenigvuldigingen en delingen tezamen als een maat voor het aantal elementaire bewerkingen. Dit was gebaseerd op het feit dat optellingen op vroegere computers in een veel snellere tijd werden uitgevoerd dan de vermenigvuldigingen en delingen. Vermits op moderne computers een vermenigvuldiging dikwijls slechts dubbel zoveel tijd vergt als een optelling, dient men alle bewerkingen expliciet te beschouwen. Als notatie gebruiken we $VD(.)$ en $AO(.)$ respectievelijk als

het aantal vermenigvuldigingen en delingen en het aantal aftrekkingen en optellingen voor de bepaling van de grootheid aangegeven tussen haakjes. D.w.z. voor de LU ontwikkeling van A hebben we voor grote n

$$\begin{aligned} VD(LU) &= \frac{n(n^2-1)}{3} \approx \frac{n^3}{3} \\ AO(LU) &= \frac{n(n-1)(2n-1)}{6} \approx \frac{n^3}{3} \end{aligned}$$

(**b**) Aanpassing van b tot $g = b^{(n)}$:

$$\begin{aligned} VD(g) &= (n-1) + (n-2) + \ldots + 1 = \frac{n(n-1)}{2} \\ AO(g) &= \frac{n(n-1)}{2}\,. \end{aligned}$$

(**c**) Oplossing van $Ux = g$:

$$VD(x) = \frac{n(n+1)}{2} \qquad\qquad AO(x) = \frac{n(n-1)}{2}\,.$$

Dus voor de oplossing van $Ax = b$ is het aantal elementaire bewerkingen voor n voldoende groot af te leiden uit (a)-(c)

$$\begin{aligned} VD(LU, g, x) &= \frac{n^3}{3} + n^2 - \frac{n}{3} \approx \frac{1}{3}n^3 \\ AO(LU, g, x) &= \frac{n(n-1)(2n+5)}{6} \approx \frac{1}{3}n^3\,. \end{aligned}$$

Dus het aantal optellingen en aftrekkingen is ongeveer hetzelfde als het aantal vermenigvuldigingen en delingen die toch nog altijd meer tijd vereisen. Daarom beschouwen we vanaf hier slechts $VD(.)$. Merk dan vooreerst op dat $Ax = b$ oplossen veel minder tijd vraagt dan bv. een eenvoudige bewerking zoals het vermenigvuldigen van twee $n \times n$ matrices. Matrix vermenigvuldiging vereist n^3 bewerkingen en het oplossen van $Ax = b$ enkel ongeveer $\frac{1}{3}n^3$ operaties. Bovendien zien we dat de meeste tijd nodig is voor de decompositie van $A = LU$. Eenmaal dit bereikt, zijn er slechts n^2 bijkomende bewerkingen nodig om $Ax = b$ op te lossen. Dus eenmaal $Ax = b$ opgelost, is het relatief weinig tijdrovend om bijkomende stelsels op te lossen met dezelfde coëfficiënten matrix wanneer de LU decompositie bewaard gebleven is. Uiteindelijk moet men zich ook realiseren dat de Gauss eliminatiemethode veel minder tijd vergt dan Cramer's regel, welke gebruik maakt van determinanten (zie cursus Algebra). Wanneer de determinanten in Cramer's regel berekend worden m.b.v. minoren ontwikkeling, is het aantal elementaire bewerkingen $(n+1)!$. Voor $n = 10$ bv. gebruikt de Gauss methode 430 bewerkingen en Cramer's regel 39916800. Dit maakt het duidelijk dat deze laatste geen praktische numerieke methode is, maar enkel een theoretische wiskundige methode.

2.1.3 De inverse van A

De inverse A^{-1} van A kan bepaald worden door gebruik te maken van de Gauss eliminatiemethode. Het vinden van $X = A^{-1}$ is equivalent met het oplossen van $AX = I$ met X een $n \times n$ onbekende matrix. Als we X en I schrijven in termen van hun kolommen, nl.

$$X = [x^{(1)}, \ldots, x^{(n)}] \qquad I = [e^{(1)}, \ldots, e^{(n)}]$$

dan is het oplossen van $AX = I$ equivalent met het oplossen van n stelsels van n vergelijkingen

$$Ax^{(1)} = e^{(1)}, \ldots, Ax^{(n)} = e^{(n)}$$

die alle dezelfde coëfficiëntenmatrix A hebben. Het aantal uit te voeren elementaire bewerkingen volgt uit (2.1.2(a)-(c)), nl.

$$VD(A^{-1}) = VD(LU) + nVD(g) + nVD(x) = \frac{4}{3}n^3 - \frac{n}{3} \approx \frac{4}{3}n^3 .$$

Dus A^{-1} berekenen vraagt viermaal meer tijd dan $Ax = b$ oplossen voor één bepaalde vector b, dus niet n maal meer tijd, wat men op het eerste gezicht zou kunnen denken.

2.1.4 Pivoteren in de Gauss methode

Het Gauss algoritme, in de zo juist beschreven eenvoudige vorm, is niet optimaal, want het werkt niet voor sommige stelsels die nochtans zeer gemakkelijk op te lossen zijn. Om dit te illustreren beschouwen we drie elementaire voorbeelden.

Voorbeeld 2.1.3

Beschouw het stelsel

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} . \tag{2.9}$$

De beschreven eenvoudige versie van het algoritme werkt niet omdat het niet mogelijk is een veelvoud van de eerste vergelijking toe te voegen aan de tweede, zodat we een 0-coëfficënt bekomen in de tweede vergelijking. Nochtans bezit dit stelsel de eenvoudige oplossing $x_1 = 1$ en $x_2 = 1$. $\diamond$

Voorbeeld 2.1.4

Het hierboven vastgestelde probleem treedt ook op voor het volgende stelsel waarin ϵ een klein getal verschillend van nul is

$$\begin{bmatrix} \epsilon & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} . \tag{2.10}$$

Toegepast op (2.10) levert het Gauss algoritme het volgende boven triangulair stelsel:

$$\begin{bmatrix} \epsilon & 1 \\ 0 & 1-\epsilon^{-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2-\epsilon^{-1} \end{bmatrix} . \tag{2.11}$$

Als ϵ voldoende klein is, vindt men als oplossing in een computer :

$$\begin{cases} x_2 = \dfrac{2-\epsilon^{-1}}{1-\epsilon^{-1}} \approx 1 \\ x_1 = (1-x_2)\epsilon^{-1} \approx 0\,. \end{cases} \tag{2.12}$$

Inderdaad zal $2-\epsilon^{-1}$, als ϵ voldoende klein is, hetzelfde zijn als $-\epsilon^{-1}$. Analoog zal de noemer $1-\epsilon^{-1}$ berekend worden als $-\epsilon^{-1}$. Derhalve zal x_2 onder deze omstandigheden de waarde 1 krijgen en x_1 de waarde 0. Vermits de correcte oplossing is

$$\begin{cases} x_1 = \dfrac{1}{1-\epsilon} \\ x_2 = \dfrac{1-2\epsilon}{1-\epsilon} \end{cases}$$

is de berekende oplossing correct voor x_2 maar volledig incorrect voor x_1.

$\diamond$

Met bovenstaand voorbeeld kan men zich de volgende vraag stellen: *Als ϵ klein is, waarom leidt de berekening in de computer van $2-\epsilon^{-1}$ naar hetzelfde machinegetal als $-\epsilon^{-1}$?* De reden is dat de exponenten van de drijvende komma voorstellingen van 2 en ϵ^{-1} in overeenstemming gebracht worden door een aangepaste verschuiving van het decimale teken alvorens de aftrekking wordt verricht. Als die verschuiving voldoende groot is zal de mantisse van 2 nul worden.

Het laatste voorbeeld dat we willen behandelen toont aan dat het niet het klein zijn is van de coëfficiënt a_{11} dat de oorzaak is van de problemen; het is de kleinheid van a_{11} *relatief* t.o.v. de andere elementen in de rij. Beschouw het volgende stelsel dat volledig equivalent is met het stelsel uit voorbeeld 2.1.4.

Voorbeeld 2.1.5

$$\begin{bmatrix} 1 & \epsilon^{-1} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \epsilon^{-1} \\ 2 \end{bmatrix} . \tag{2.13}$$

Het Gauss procédé levert:

$$\begin{bmatrix} 1 & \epsilon^{-1} \\ 0 & 1-\epsilon^{-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \epsilon^{-1} \\ 2-\epsilon^{-1} \end{bmatrix} . \tag{2.14}$$

De oplossing van (2.14) luidt:

$$\begin{cases} x_2 = \dfrac{2-\epsilon^{-1}}{1-\epsilon^{-1}} \approx 1 \\ x_1 = \epsilon^{-1} - \epsilon^{-1} x_2 \approx 0 \end{cases} .$$

Dus we krijgen voor kleine ϵ waarde opnieuw een verkeerd resultaat. $\diamond$

De moeilijkheden in deze voorbeelden zullen verdwijnen als de volgorde van de vergelijkingen wordt omgewisseld. Inderdaad, omwisseling van de twee vergelijkingen in het stelsel (2.10) leidt tot

$$\begin{bmatrix} 1 & 1 \\ \epsilon & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} .$$

Gauss eliminatie op dit stelsel levert

$$\begin{bmatrix} 1 & 1 \\ 0 & 1-\epsilon \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1-2\epsilon \end{bmatrix} .$$

En de oplossing wordt dan gegeven door:

$$\begin{cases} x_2 = \dfrac{1-2\epsilon}{1-\epsilon} \approx 1 \\ x_1 = 2 - x_2 \approx 1 \end{cases} .$$

De conclusie die kan getrokken worden uit deze elementaire voorbeelden, is dat een goed algoritme in zich moet houden dat een verwisseling van vergelijkingen in het stelsel mogelijk is, indien de omstandigheden het vereisen. In elke stap van het Gauss eliminatie proces, hebben we trouwens stilzwijgend aangenomen dat het pivotelement $a_{kk}^{(k)} \neq 0$. In feite kan men dit niet onderstellen (zie o.a. voorbeeld 2.1.3), maar dient men elke stap van het eliminatieproces te starten met het wisselen van rijen, zodat een van nul verschillend element in de pivotpositie optreedt. Als dit niet kan verwezenlijkt worden moet de matrix singulier zijn, wat in tegenstrijd is met de begin-aanname van het probleem. Nochtans is het doorgaans niet voldoende alleen te eisen dat het pivotelement niet nul is. Vaak is dergelijk element ontstaan na een aantal bewerkingen en is het verschillend van nul louter door afrondingsfouten die hebben meegespeeld

(vergelijk dit met het ϵ element uit voorbeeld 2.1.4). Dergelijk element aanwenden als pivotelement zal resulteren in grote fouten in de verdere berekeningen. Om hiertegen in te gaan en omwille van andere redenen die te maken hebben met de voortplanting van afrondingsfouten, maakt men in de praktijk veelal gebruik van zgn. partiële of rij pivotering, welke als volgt werkt :

Voor $1 \leq k \leq n-1$, bij stap k van het Gauss eliminatieproces, bepaal

$$c_k = \max_{k \leq i \leq n} \mid a_{ik}^{(k)} \mid \ . \tag{2.15}$$

Laat i de kleinste rij index zijn (met $i \geq k$) waarvoor de maximum c_k waarde bereikt wordt. Als $i > k$, verwissel in A en b rijen k en i en ga verder met stap k in het eliminatieproces. Alle vermenigvuldigers zullen nu voldoen aan $\mid m_{ik} \mid \leq 1 \quad (i = k+1, \ldots, n)$.

Om in de computer geen onnodige verschuivingen van rijen fysisch te moeten uitvoeren, kiest men doorgaans de pivotrijen op een logische manier. Veronderstel dat we i.p.v. de rijen te kiezen in de volgorde $1, 2, \ldots, n$, we de rijen in de volgorde $p_1, p_2, \ldots, p_n$ beschouwen; hierin is $p_1, p_2, \ldots, p_n$ een permutatie van $1, 2, \ldots, n$. Dit betekent dan dat in de eerste stap veelvouden van rij p_1 zullen afgetrokken worden van de overige rijen, in de tweede stap zullen veelvouden van rij p_2 afgetrokken worden van alle rijen behalve van rij p_1, etc ....

Voorbeeld 2.1.6

Beschouwen we het volgende stelsel

$$\begin{cases} 0.729\,x + & 0.81\,y + & 0.9\,z & = & 0.6867 \\ x + & y + & z & = & 0.8338 \\ 1.331\,x + & 1.21\,y + & 1.1\,z & = & 1.000 \end{cases} \tag{2.16}$$

De exacte oplossing, afgerond tot vier significante cijfers is

$$x = 0.2245 \qquad y = 0.2814 \qquad z = 0.3279 \tag{2.17}$$

Floating–point decimale bewerking, met vier cijfers in de mantisse, wordt gebruikt voor het oplossen van het lineaire stelsel.

(a) Oplossing zonder pivoteren

$$\left[\begin{array}{ccc|c} .7290 & .8100 & .9000 & .6867 \\ 1.000 & 1.000 & 1.000 & .8338 \\ 1.331 & 1.210 & 1.100 & 1.000 \end{array}\right]$$

$$\downarrow \begin{matrix} m_{21} = 1.372 \\ m_{31} = 1.826 \end{matrix}$$

$$\left[\begin{array}{ccc|c} .7290 & .8100 & .9000 & .6867 \\ 0.0 & -.1110 & -.2350 & -.1084 \\ 0.0 & -.2690 & -.5430 & -.2540 \end{array}\right]$$

$$\downarrow m_{32} = 2.423$$

$$\left[\begin{array}{ccc|c} .7290 & .8100 & .9000 & .6867 \\ 0.0 & -.1110 & -.2350 & -.1084 \\ 0.0 & 0.0 & .02640 & .008700 \end{array}\right]$$

en de oplossing is

$$x = .2252 \qquad y = .2790 \qquad z = .3295 \tag{2.18}$$

(b) Oplossing met pivoteren.

Om het wisselen van rijen i en j aan te duiden, zullen we gebruik maken van de notatie

$$r_i \longleftrightarrow r_j$$

$$\left[\begin{array}{ccc|c} .7290 & .8100 & .9000 & .6867 \\ 1.000 & 1.000 & 1.000 & .8338 \\ 1.331 & 1.210 & 1.100 & 1.000 \end{array}\right]$$

$$r_1 \longleftrightarrow r_3 \downarrow \begin{matrix} m_{21} = .7513 \\ m_{31} = .5477 \end{matrix}$$

$$\left[\begin{array}{ccc|c} 1.331 & 1.210 & 1.100 & 1.000 \\ 0.0 & .09090 & .1736 & .08250 \\ 0.0 & .1473 & .2975 & .1390 \end{array}\right]$$

$$r_2 \longleftrightarrow r_3 \downarrow m_{32} = .6171$$

$$\left[\begin{array}{ccc|c} 1.331 & 1.210 & 1.100 & 1.000 \\ 0.0 & .1473 & .2975 & .1390 \\ 0.0 & 0.0 & -.01000 & -0.003280 \end{array}\right]$$

en de oplossing luidt :

$$x = .2246 \qquad y = .2812 \qquad z = .3280 \tag{2.19}$$

De fout op (2.18) is 7 tot 16 maal groter dan de fout op (2.19), afhankelijk van de beschouwde oplossingscomponente. De resultaten in (2.19) hebben één significant cijfer meer dan die in (2.18). Dit voorbeeld illustreert het positieve effect dat het gebruik van pivoteren kan hebben op het foutengedrag bij de Gauss eliminatiemethode. ◇

In plaats van partiële of rij pivotering verkiest men soms rij en kolom pivotering, ook *volledige pivotering* genoemd. Volledige pivotering is moeilijker toe te passen dan partiële. Bij het wisselen van kolommen moet de permutatie van de elementen van de oplossingsvector bijgehouden worden. Uit studies van Wilkinson (J. Wilkinson, *The algebraic eigenvalue problem*, Oxford University Press, Oxford 1965) is gebleken dat partiële pivotering bijna in alle gevallen even nauwkeurige resultaten oplevert als volledige pivotering. Bij partiële pivotering werd voorgesteld het grootste op elk ogenblik beschikbare element als pivot te gebruiken. Dit houdt echter in dat de keuze van pivot eigenlijk afhankelijk is van de zgn. schaalverhouding van de verschillende elementen op een rij. Als we in ons voorbeeld 2.1.6 de eerste vergelijking zouden vermenigvuldigen met 1 miljoen, is het zeker dat deze vergelijking het eerste pivotelement zal leveren; nochtans is de uiteindelijke oplossing van het stelsel door die vermenigvuldiging niet veranderd. Daarom wordt dikwijls voorgesteld in bibliotheekroutines als pivot dit element te kiezen dat het grootste getal in een kolom zou zijn wanneer alle vergelijkingen herschaald zijn, zodat hun grootste coëfficiënt herleid is tot 1. Dit procédé wordt *impliciete* of *geschaalde rij pivotering* genoemd.

2.2 Varianten op de Gauss eliminatiemethode

De meeste varianten op de Gauss eliminatiemethode zijn aanpassingen en vereenvoudigingen, die ontstaan door rekening te houden met bepaalde klassen van matrices (bv. symmetrisch, positief definiete matrices, enz...).

2.2.1 De Gauss–Jordan methode

Deze procedure is een echte variant op het Gauss eliminatie procédé. I.p.v. in stap k enkel de onbekende x_k te elimineren uit de $k-1$ laatste vergelijkingen, wordt in de Gauss–Jordan methode in elke stap k de x_k onbekende geëlimineerd uit alle vergelijkingen, behalve de k^{de}. Dit kan als volgt formeel weergegeven worden. Kies bij stap k een aanvaardbaar pivotelement $a_{kk}^{(k)}$. Definieer vervolgens

$$\begin{aligned} a_{kj}^{(k+1)} &= a_{kj}^{(k)}/a_{kk}^{(k)} \qquad j = k, \ldots, n \\ b_k^{(k+1)} &= b_k^{(k)}/a_{kk}^{(k)} \,. \end{aligned}$$

Elimineer de onbekende x_k, zowel in de vergelijkingen boven als onder vergelijking k, d.m.v.

$$\begin{aligned} a_{ij}^{(k+1)} &= a_{ij}^{(k)} - a_{ik}^{(k)} a_{kj}^{(k+1)} \qquad j = k, \ldots, n \\ b_i^{(k+1)} &= b_i^{(k)} - a_{ik}^{(k)} b_k^{(k+1)} \qquad i = 1, \ldots, n, i \neq k. \end{aligned}$$

De procedure converteert $Ax = b$ naar $Ix = b^{(n+1)}$, zodat na het voleindigen van alle eliminaties, $x = b^{(n+1)}$. Deze techniek vereist (trek dit zelf na als oefening)

$$\frac{n^3 + 3n^2}{2} \approx \frac{n^3}{2}$$

vermenigvuldigingen en delingen om $Ax = b$ op te lossen. Dit is 50% meer dan de klassieke Gauss eliminatiemethode.

2.2.2 De Cholesky methode

Neem aan dat A een symmetrische en positief definiete matrix is van orde n. De matrix A is positief definiet als

$$(Ax, x) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}\, x_i\, x_j > 0$$

voor alle $x \in \mathrm{R}^n$, $x \neq 0$. Voor dergelijke matrix A bestaat er een handige factorizatie die kan doorgevoerd worden zonder expliciet gebruik te maken van pivoteren. Dit is de Cholesky methode welke vooropstelt dat voor bovenstaande matrices er een beneden triangulaire matrix L bestaat zodat

$$A = LL^T\,. \tag{2.20}$$

Introduceer hiertoe $L = [l_{ij}]$, met $l_{ij} = 0$ voor $j > i$. Start de constructie van L door zijn eerste rij te vermenigvuldigen met de eerste kolom van L^T rekening houdend met (2.20), i.e.

$$l_{11}^2 = a_{11}\,.$$

Omdat A positief definiet is hebben we $a_{11} > 0$ en $l_{11} = \sqrt{a_{11}}$. Vermenigvuldig nu de tweede rij van L met de eerste twee kolommen van L^T, wat resulteert in

$$l_{21} l_{11} = a_{21} \qquad\qquad l_{21}^2 + l_{22}^2 = a_{22}$$

waaruit

$$l_{21} = a_{21}/l_{11} = a_{21}/\sqrt{a_{11}} \qquad\qquad l_{22} = \sqrt{a_{22} - a_{21}^2/a_{11}}\,.$$

In het algemeen leiden we aldus af voor $i = 1, 2, \ldots, n$

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}}{l_{jj}} \qquad\qquad j = 1, \ldots, i-1 \tag{2.21}$$

$$l_{ii} = \left[a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2 \right]^{1/2}\,. \tag{2.22}$$

Als $j = 1$ in (2.21) of $i = 1$ in (2.22) betekent $\sum_{k=1}^{0}$ dat de som niet moet inachtgenomen worden. De argumenten van de optredende vierkantswortel zijn alle positief, wat kan bewezen worden door het positief definiet zijn van A te gebruiken.

Eenmaal de triangulaire factorizatie van de vorm (2.20) doorgevoerd, herleidt het oplossen van $Ax = b$ zich tot

$$LL^T x = b \qquad \text{of} \qquad L\xi = b \qquad \text{en} \qquad L^T x = \xi$$

wat, gezien L en L^T beide triangulair zijn, relatief eenvoudig wordt.

De beide stelsels behoren tot de klasse van de zo genoemde *gemakkelijk op te lossen stelsels*. Een stelsel, zoals $L\xi = b$, met een beneden triangulaire matrix wordt als volgt behandeld. Om dit stelsel te kunnen oplossen veronderstellen we uiteraard dat $l_{ii} \neq 0$ $(\forall i)$. Dan volgt ξ_1 onmiddellijk uit de eerste vergelijking. Met de bekende waarde voor ξ_1 gesubstitueerd in de tweede vergelijking, los die tweede vergelijking op voor ξ_2. We zetten die techniek voort om aldus $\xi_1, \xi_2, \ldots, \xi_n$ te bekomen de één na de ander en in die volgorde. Een formeel algoritme voor het bepalen van de oplossing ξ in dat geval wordt *voorwaartse substitutie* genoemd.

Algoritme 2.2.2

input $n, [l_{ij}], [b_i]$
for $i = 1, 2, \ldots, n$ **do**
$\xi_i \leftarrow \Big(b_i - \sum_{j=1}^{i-1} l_{ij}\xi_j\Big)/l_{ii}$
end for
output $[\xi_i]$
end

Het andere stelsel $L^T x = \xi$ met de boven triangulaire matrix kan even eenvoudig opgelost worden als $l_{ii} \neq 0$ voor $1 \leq i \leq n$. Het formele algoritme om x op te lossen noemt men *achterwaartse substutie*. Het werd ook reeds aangewend om het Gauss equivalent stelsel $Ux = g$ op te lossen.

Algoritme 2.2.3

input $n, [l_{ij}], [\xi_i]$
for $i = n, n-1, \ldots, 1$ **do**
$x_i \leftarrow \Big(\xi_i - \sum_{j=i+1}^{n} l_{ji}x_j\Big)/l_{ii}$
end for
output $[x_i]$
end

DEMO 3 (Maple)

Zie Claroline

2.2.3 Het geval van tridiagonale matrices

De matrix $A = [a_{ij}]$ is tridiagonaal als

$$a_{ij} = 0 \qquad \text{voor} \qquad |i-j| > 1\,.$$

Dit levert de volgende gedaante voor A

$$A = \begin{bmatrix} a_1 & c_1 & 0 & 0 & \cdots & 0 \\ b_2 & a_2 & c_2 & 0 & & \vdots \\ 0 & b_3 & a_3 & c_3 & & \vdots \\ \vdots & & & \ddots & & \vdots \\ \vdots & & & b_{n-1} & a_{n-1} & c_{n-1} \\ 0 & \cdots & \cdots & 0 & b_n & a_n \end{bmatrix}. \tag{2.23}$$

Na enkele theoretische beschouwingen is het vast te stellen dat een factorizatie $A = LU$ zonder pivotering kan bekomen worden, waarbij de meeste elementen van L en U nul zullen zijn. In feite is de volgende algemene formule geldig

$$A = LU = \begin{bmatrix} \alpha_1 & 0 & \cdots & \cdots & & 0 \\ b_2 & \alpha_2 & 0 & & & \vdots \\ 0 & b_3 & \alpha_3 & 0 & & \vdots \\ \vdots & & & \ddots & & \vdots \\ \vdots & & & & \ddots & \vdots \\ 0 & & & & b_n & \alpha_n \end{bmatrix} \begin{bmatrix} 1 & \gamma_1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & \gamma_2 & \cdots & \cdots & \vdots \\ \vdots & & \ddots & & & \vdots \\ \vdots & & & \ddots & & \vdots \\ \vdots & & & & 1 & \gamma_{n-1} \\ 0 & \cdots & \cdots & \cdots & 0 & 1 \end{bmatrix}.$$

Door rechtstreekse identificatie kunnen $\{\alpha_i\}$ en $\{\gamma_i\}$ recursief berekend worden, m.a.w.

$$\begin{array}{lll} a_1 = \alpha_1 & \alpha_1\gamma_1 = c_1 & \\ a_i = \alpha_i + b_i\gamma_{i-1} & & i = 2, \ldots, n \\ \alpha_i\gamma_i = c_i & & i = 2, 3, \ldots, n-1\,. \end{array} \tag{2.24}$$

Dit kan opgelost worden

$$\begin{array}{lll} \alpha_1 = a_1 & \gamma_1 = c_1/\alpha_1 & \\ \alpha_i = a_i - b_i\gamma_{i-1} & \gamma_i = c_i/\alpha_i & i = 2, 3, \ldots, n-1\,. \\ \alpha_n = a_n - b_n\gamma_{n-1}\,. & & \end{array}$$

Om $LUx = f$ op te lossen, beschouw $Ux = z$ en $Lz = f$ dan

$$\begin{array}{lll} z_1 = f_1/\alpha_1 & z_i = (f_i - b_i z_{i-1})/\alpha_i & i = 2, 3, \ldots, n \\ x_n = z_n & x_i = z_i - \gamma_i x_{i+1} & i = n-1, n-2, \ldots, 1\,. \end{array}$$

De constanten in (2.24) kunnen gestockeerd worden voor later gebruik, wanneer andere stelsels $Ax = f$ moeten opgelost worden met een aantal verschillende rechterleden f. Het aantal vermenigvuldigingen en delingen voor de bepaling van L en U is $2n-2$ en voor het oplossen van $Ax = f$ dienen nog $3n-2$ dergelijke bewerkingen inachtgenomen te worden. Dus enkel $5n-4$ vermenigvuldigingen en delingen zijn er nodig om een dergelijk stelsel $Ax = f$ volledig op te lossen; voor elk bijkomend rechterlid f, en dezelfde A, zijn er dan verder nog slechts $3n-2$ dergelijke bewerkingen noodzakelijk.

2.3 Normen

In voorgaande hoofdstuk (paragraaf 1.4.2) werd het begrip norm, $\| \ . \ \|$, gebruikt. In de cursus Analyse werd voor elke $x \in V$ met V (R^n of C^n) een inwendige productruimte, de zgn. Euclidische norm, geïntroduceerd als

$$\| x \| = \sqrt{< x, x >} = \sqrt{\sum_{i=1}^{n} | x_i |^2} . \tag{2.25}$$

In veel gevallen wordt de lengte van een vector op deze wijze gedefinieerd. Nochtans zijn er veel situaties waarin het aan te raden is de grootte van een vector op andere wijzen te meten. Daartoe introduceren we hier het algemeen concept voor de norm van een vector :

Definitie : Zij V een vectorruimte en zij N een reële functie gedefinieerd op V, dan is N een norm als

$$\text{(a)} \quad N(x) \geq 0 \text{ voor alle } x \in V \,;\; N(x) = 0 \Leftrightarrow x = 0 \tag{2.26}$$

$$\text{(b)} \quad N(\alpha x) = | \alpha | \, N(x) \text{ voor alle } x \in V \text{ en alle scalairen } \alpha \tag{2.27}$$

$$\text{(c)} \quad N(x+y) \leq N(x) + N(y) \text{ voor alle } x,\ y \in V \tag{2.28}$$

(driehoeksongelijkheid) .

Het is nu mogelijk verschillende soorten normen N te introduceren, nl. voor $1 \leq p < \infty$ hebben we de p–norm

$$\| x \|_p = \left[\sum_{j=1}^{n} | x_j |^p \right]^{1/p} \qquad x \in \mathrm{C}^n , \tag{2.29}$$

en de maximum norm is gedefinieerd als

$$\| x \|_\infty = \max_{1 \leq j \leq n} | x_j | , \qquad x \in \mathrm{C}^n . \tag{2.30}$$

De Euclidische norm correspondeert dan uiteraard met $\| x \|_2$.

Voorbeeld 2.3.1

Voor de vector $x = (1, 0, -1, 2)$ is

$$\| x \|_1 = 4 \qquad \| x \|_2 = \sqrt{6} \qquad \| x \|_\infty = 2\,.$$

$\diamond$

Normen kunnen ook ingevoerd worden voor matrices. Voor een $n \times n$ matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$

hebben we o.a. de kolomnorm

$$\| A \|_1 = \max_{1 \le j \le n} \sum_{i=1}^{n} | a_{ij} | \tag{2.31}$$

en de rijnorm

$$\| A \|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} | a_{ij} | \ . \tag{2.32}$$

Bovendien gelden ook

$$\| AB \|_v \le \| A \|_v \, \| B \|_v \qquad \text{en} \tag{2.33}$$

$$\| Ax \|_v \le \| A \|_v \, \| x \|_v \qquad \text{voor alle } x \in \mathrm{R}^n \tag{2.34}$$

waarbij A en B willekeurige $n \times n$ matrices zijn en $v = 1$ of ∞.

Voorbeeld 2.3.2

Beschouw de vectorruimte R^2 en matrices van de orde 2×2, bv.

$$A = \begin{bmatrix} 1 & -1 \\ 3 & 2 \end{bmatrix} \qquad x = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad y = Ax = \begin{bmatrix} -1 \\ 7 \end{bmatrix}$$

dan $\| A \|_\infty = 5 \qquad \| x \|_\infty = 2 \qquad \| y \|_\infty = \| Ax \|_\infty = 7$
en (2.34) wordt gemakkelijk geverifieerd. $\diamond$

2.4 Foutenanalyse en stabiliteit van de oplossing van $Ax = b$ met de Gauss eliminatiemethode

De analyse van het effect van afrondingsfouten bij de Gauss eliminatiemethode kan o.m. gevonden worden in "J. Wilkinson, *The algebraic eigenvalue problem*; Oxford University Press, Oxford, 1965, pp. 209–215". Hier zullen we niet in detail op ingaan. Wij zullen hier aandacht hebben voor de stabiliteit van de oplossing x, wanneer kleine storingen op het rechterlid b kunnen optreden. Hiertoe zullen we het algemeen schema uit paragraaf 1.4.2 volgen.

Neem aan dat $Ax = b$ eenduidig oplosbaar is en beschouw de oplossing van het gestoord probleem

$$A\hat{x} = b + r\,.$$

Noem $e = \hat{x} - x$ en trek van bovenstaande betrekking $Ax = b$ af; dit resulteert in :

$$Ae = r \qquad \text{of} \qquad e = A^{-1}\,r\,. \tag{2.35}$$

Om de stabiliteit van de oplossing van $Ax = b$ te onderzoeken (cfr. 1.15) is het nodig de grenzen te kennen van de grootheid

$$\frac{\parallel e \parallel_v \;/\; \parallel x \parallel_v}{\parallel r \parallel_v \;/\; \parallel b \parallel_v} \tag{2.36}$$

wanneer r waarden aanneemt uit R^n, die klein zijn relatief t.o.v. b. De aan te wenden normen worden hier nog niet nader gespecificeerd. Beschouw de normen vertrekkend uit (2.35) en rekening houdend met (2.34), i.e.

$$\parallel r \parallel_v \leq \parallel A \parallel_v \parallel e \parallel_v \qquad\qquad \parallel e \parallel_v \leq \parallel A^{-1} \parallel_v \parallel r \parallel_v \;.$$

Deel in de eerste ongelijkheid door $\parallel A \parallel_v \parallel x \parallel_v$ en in de tweede door $\parallel x \parallel_v$ om te bekomen :

$$\frac{\parallel r \parallel_v}{\parallel A \parallel_v \parallel x \parallel_v} \leq \frac{\parallel e \parallel_v}{\parallel x \parallel_v} \leq \frac{\parallel A^{-1} \parallel_v \parallel r \parallel_v}{\parallel x \parallel_v}\,. \tag{2.37}$$

Uit $Ax = b$ en $x = A^{-1}b$ volgen, mits opnieuw rekening te houden met (2.34) :

$$\parallel b \parallel_v \leq \parallel A \parallel_v \parallel x \parallel_v \qquad \text{en} \qquad \parallel x \parallel_v \leq \parallel A^{-1} \parallel_v \parallel b \parallel_v \;.$$

Daarvan gebruik makend in (2.37) leidt tot :

$$\frac{1}{\parallel A \parallel_v \parallel A^{-1} \parallel_v} \frac{\parallel r \parallel_v}{\parallel b \parallel_v} \leq \frac{\parallel e \parallel_v}{\parallel x \parallel_v} \leq \parallel A \parallel_v \parallel A^{-1} \parallel_v \; \frac{\parallel r \parallel_v}{\parallel b \parallel_v}\,. \tag{2.38}$$

Uit (2.38) halen we dat het conditiegetal $K(x)$, zoals gedefinieerd in (1.15), voor dit specifiek probleem begrensd wordt door :

$$\frac{1}{K(A)} \leq K(x) \leq K(A)\,. \tag{2.39}$$

De grootheid

$$K(A) = \|A\|_v \|A^{-1}\|_v \tag{2.40}$$

die onafhankelijk is van de oplossing x wordt het conditiegetal van de matrix A genoemd. Vermits er soorten normen zijn naar gelang de waarde van v (zie (2.25), (2.29)–(2.30)) varieert $K(A)$ naargelang de gebruikte norm–definitie. Hoe dan ook, deze grootheid is naar beneden begrensd, vermits

$$1 = \| I \|_v = \| AA^{-1} \|_v \leq \| A \|_v \, \| A^{-1} \|_v = K(A)\,. \tag{2.41}$$

Wanneer $K(A)$ ongeveer 1 is, halen we uit (2.39) dat kleine relatieve storingen op b leiden tot vergelijkbare kleine relatieve storingen op de oplossing x. Als daarentegen $K(A) \gg 1$ is het niet uitgesloten dat ook $K(x) \gg 1$, in welk geval instabiliteiten in de oplossing optreden.

Voorbeeld 2.4.1

Beschouw het lineaire stelsel

$$\left\{ \begin{array}{rcl} 7x_1 + 10x_2 & = & b_1 \\ 5x_1 + 7x_2 & = & b_2 \end{array} \right. . \tag{2.42}$$

We hebben als matrix A en zijn inverse A^{-1}

$$A = \left[\begin{array}{cc} 7 & 10 \\ 5 & 7 \end{array} \right] \qquad\qquad A^{-1} = \left[\begin{array}{cc} -7 & 10 \\ 5 & -7 \end{array} \right] . \tag{2.43}$$

Als we in (2.40) het conditiegetal noteren als $K(A)_v$ omdat dit bepaald wordt door de matrix norm $\| \cdot \|_v$ te bezigen, dan is bv. volgens (2.31)–(2.32)

$$K(A)_1 = K(A)_\infty = (17)(17) = 289\,.$$

Deze grote waarde laat dus vermoeden dat de oplossing van (2.42) gevoelig kan zijn voor kleine veranderingen in b_1 en b_2. Om dit te illustreren beschouwen we het bijzondere geval

$$\left\{ \begin{array}{rcl} 7x_1 + 10x_2 & = & 1 \\ 5x_1 + 7x_2 & = & .7 \end{array} \right. \tag{2.44}$$

wat een oplossing $x_1 = 0 \quad x_2 = .1$ bezit. Storen we de b_1 en b_2 een weinig en beschouwen we het stelsel

$$\begin{cases} 7\tilde{x}_1 + 10\tilde{x}_2 &= 1.01 \\ 5\tilde{x}_1 + 7\tilde{x}_2 &= .69\,. \end{cases}$$

Dit bezit een oplossing $\tilde{x}_1 = -.17$ en $\tilde{x}_2 = 0.22\,.$

Deze relatieve veranderingen in x zijn groot in vergelijking met de grootte van de relatieve veranderingen in b. Dergelijk stelsel, wier oplossingen x instabiel zijn voor kleine relatieve veranderingen in b worden *slecht geconditioneerd* genoemd. Uit voorgaand voorbeeld blijkt dat $K(A)$ een goede indicator is voor dit slecht geconditioneerd zijn. ◇

Voorbeeld 2.4.2

Een ander voorbeeld van een matrix, die aanleiding geeft tot een slecht geconditioneerd stelsel, wordt gegeven door:

$$A = \begin{bmatrix} 1 & 1+\epsilon \\ 1-\epsilon & 1 \end{bmatrix}. \tag{2.45}$$

Zijn inverse wordt gegeven door

$$A^{-1} = \epsilon^{-2} \begin{bmatrix} 1 & -1-\epsilon \\ -1+\epsilon & 1 \end{bmatrix}. \tag{2.46}$$

Als de rij norm gebruikt wordt, vinden we dat $\| A \|_\infty = 2 + \epsilon$ en $\| A^{-1} \|_\infty = \epsilon^{-2}(2+\epsilon)$. Derhalve is $K(A)_\infty = [(2+\epsilon)/\epsilon]^2 > 4/\epsilon^2$. Als $\epsilon \leq 0.01$ dan is $K(A)_\infty \geq 40000$. In zulk een geval kan een kleine storing in b een relatieve storing die tot *40000 maal groter* is, induceren in de oplossing van het stelsel $Ax = b$. ◇

DEMO 4 (Maple

Zie Claroline

2.5 Iteratiemethodes

Zoals reeds aangestipt in de inleiding van dit hoofdstuk, zijn vele stelsels lineaire vergelijkingen te groot om opgelost te kunnen worden door rechtstreekse methodes die steunen op Gauss eliminatie. Voor deze stelsels zijn de iteratiemethodes dikwijls de enige oplossingsmethode.

2.5.1 De Gauss–Jacobi methode

Herschrijf $Ax = b$ als

$$x_i = \frac{1}{a_{ii}}\Big(b_i - \sum_{\substack{j=1\\ j\neq i}}^{n} a_{ij}x_j\Big) \qquad\qquad i = 1, 2, \ldots, n \tag{2.47}$$

in de veronderstelling uiteraard dat $a_{ii} \neq 0$. Definieer hieruit de iteratie

$$x_i^{(m+1)} = \frac{1}{a_{ii}}\Big(b_i - \sum_{\substack{j=1\\ j\neq i}}^{n} a_{ij}x_j^{(m)}\Big) \qquad i = 1, \ldots, n \qquad m \geq 0 \tag{2.48}$$

en neem aan dat initiale schattingen $x_i^{(0)},\ i = 1, \ldots, n$ gegeven zijn. Om een idee te hebben over de convergentie van bovenstaande iteratie definiëren we $e^{(m)} = x - x^{(m)}$, $m \geq 0$. Door lid aan lid (2.48) af te trekken van (2.47) krijgen we

$$e_i^{(m+1)} = -\sum_{\substack{j=1\\ j\neq i}}^{n} \frac{a_{ij}}{a_{ii}} e_j^{(m)} \qquad\qquad i = 1, \ldots, n \qquad m \geq 0 \tag{2.49}$$

waaruit

$$\left|e_i^{(m+1)}\right| \leq \sum_{\substack{j=1\\ j\neq i}}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| \parallel e^{(m)} \parallel_\infty \ .$$

Definiëren we

$$\mu = \max_{1\leq i\leq n} \sum_{\substack{j=1\\ j\neq i}}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| \qquad\qquad \text{dan is ook}$$

$$\left|e_i^{(m+1)}\right| \leq \mu \parallel e^{(m)} \parallel_\infty$$

en vermits het rechterlid onafhankelijk is van i geldt ook

$$\parallel e^{(m+1)} \parallel_\infty \leq \mu \parallel e^{(m)} \parallel_\infty \ . \tag{2.50}$$

Volgens (1.16) hebben we hier te maken met een lineaire convergentie. De maat van de convergentie is hier μ en dient in dit geval kleiner dan 1 te zijn (zie paragraaf 1.5). Als $\mu < 1$ dan $e^{(m)} \to 0$ als $m \to \infty$ en

$$\parallel e^{(m)} \parallel_\infty \leq \mu^m \parallel e^{(0)} \parallel_\infty \ . \tag{2.51}$$

Opdat $\mu < 1$ zou zijn moet de matrix A diagonaal dominant zijn, d.i. deze moet voldoen aan

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} \mid a_{ij} \mid < \mid a_{ii} \mid \qquad\qquad i = 1, 2, \ldots, n\,.$$

Dergelijke matrices komen tamelijk veel voor.

Voorbeeld 2.5.1

Los $Ax = b$ op met de Gauss–Jacobi methode waarbij

$$A = \begin{bmatrix} 10 & 3 & 1 \\ 2 & -10 & 3 \\ 1 & 3 & 10 \end{bmatrix} \quad b = \begin{bmatrix} 14 \\ -5 \\ 14 \end{bmatrix} \quad \text{en} \quad x^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.52}$$

De volgende $x_i^{(m)}$ $(i = 1, 2, 3)$ waarden zijn op iteratieve wijze afleidbaar uit (2.52) want voor dit stelsel is $\mu = 0.5 < 1$.

m	$x_1^{(m)}$	$x_2^{(m)}$	$x_3^{(m)}$	$\parallel e^{(m)} \parallel_\infty$	verhouding
0	0	0	0	1	
1	1.4	0.5	1.4	0.5	0.5
2	1.11	1.20	1.11	0.2	0.4
3	0.929	1.055	0.929	0.071	0.36
4	0.9906	0.9645	0.9906	0.0355	0.50
5	1.01159	0.9953	1.01159	0.01159	0.33
6	1.000251	1.005795	1.000251	0.005795	0.50

Zoals voorspeld in (2.51) zullen de fouten met tenminste de helft afnemen bij elke iteratie. Als we vooropstellen dat de ware oplossing $x = [1, 1, 1]^T$ is kunnen we bij elke stap $\parallel e^{(m)} \parallel_\infty$ expliciet bepalen, en tevens de verhoudingen $\parallel e^{(m+1)} \parallel_\infty \; / \; \parallel e^{(m)} \parallel_\infty$ vaststellen. Deze zijn ook in bovenstaande tabel aangegeven en bevestigen inderdaad (2.51). $\diamond$

2.5.2 De Gauss-Seidel methode

Vertrekkend uit (2.47) kan ook de volgende iteratie gedefinieerd worden

$$x_i^{(m+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(m+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(m)} \right) \qquad i = 1, 2, \ldots, n\,, \tag{2.53}$$

m.a.w. elke nieuwe componente $x_i^{(m+1)}$ kan onmiddellijk gebruikt worden in de berekening van de volgende componente. Om opnieuw een idee te hebben over de convergentie trekken we (2.53) af van (2.47), i.e.

$$e_i^{(m+1)} = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}}\, e_j^{(m+1)} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}}\, e_j^{(m)} \qquad i = 1, 2, \ldots, n\,. \tag{2.54}$$

Definiëren we

$$\alpha_i = \sum_{j=1}^{i-1} \left|\frac{a_{ij}}{a_{ii}}\right| \qquad\qquad \beta_i = \sum_{j=i+1}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| \qquad\qquad i = 1, \ldots, n$$

met $\alpha_1 = \beta_n = 0$, dan is μ zoals gedefinieerd in de Gauss–Jacobi methode te schrijven als

$$\mu = \max_{1\le i\le n} (\alpha_i + \beta_i)\,.$$

We vertrekken met de aanname dat $\mu < 1$. Uit (2.54) volgt tevens

$$\mid e_i^{(m+1)} \mid \le \alpha_i \parallel e^{(m+1)} \parallel_\infty + \beta_i \parallel e^{(m)} \parallel_\infty \qquad i = 1, \ldots, n\,. \tag{2.55}$$

Als dan k de index is waarvoor

$$\parallel e^{(m+1)} \parallel_\infty = \mid e_k^{(m+1)} \mid$$

dan, met $i = k$ in (2.55)

$$\begin{aligned} \parallel e^{(m+1)} \parallel_\infty &\le \alpha_k \parallel e^{(m+1)} \parallel_\infty + \beta_k \parallel e^{(m)} \parallel_\infty \qquad \text{of} \\ \parallel e^{(m+1)} \parallel_\infty &\le \frac{\beta_k}{1-\alpha_k} \parallel e^{(m)} \parallel_\infty\ . \end{aligned}$$

Wanneer we nu nog $\eta = \max\limits_{1\le i\le n} \dfrac{\beta_i}{1-\alpha_i}$ stellen, volgt uit voorgaande :

$$\parallel e^{(m+1)} \parallel_\infty \le \eta \parallel e^{(m)} \parallel_\infty\ . \tag{2.56}$$

Vermits nu voor elke i

$$(\alpha_i + \beta_i) - \frac{\beta_i}{1-\alpha_i} = \frac{\alpha_i[1 - (\alpha_i + \beta_i)]}{1-\alpha_i} \ge \frac{\alpha_i}{1-\alpha_i}\,(1-\mu) \ge 0$$

hebben we $\eta \le \mu < 1$. Hieruit leiden we af dat we opnieuw te maken hebben met lineaire convergentie. Nochtans zal de convergentie naar de te bepalen waarde hier meestal sneller gebeuren dan in de Gauss–Jacobi methode, gezien $\eta \le \mu$. Dit, gecombineerd met (2.56), toont opnieuw de convergentie van $e^{(m)} \to 0$ als $m \to \infty$ aan.

Voorbeeld 2.5.2

Beschouw opnieuw het stelsel (2.52). Hiervoor is $\eta = 0.4$, wat laat vermoeden dat voor dit stelsel de Gauss–Seidel methode sneller naar de correcte oplossing zal convergeren dan de Gauss–Jacobi methode, omdat $\eta < \mu$. In onderstaande tabel leveren we opnieuw enkele resultaten $x_i^{(m)} \quad (i = 1, 2, 3)$, de $\| e^{(m)} \|_\infty$ waarde en de verhouding $\| e^{(m+1)} \|_\infty \; / \; \| e^{(m)} \|_\infty$.

m	$x_1^{(m)}$	$x_2^{(m)}$	$x_3^{(m)}$	$\| e^{(m)} \|_\infty$	verhouding
0	0	0	0	1	
1	1.400000000	0.780000000	1.026000000	0.4	0.4
2	1.063400000	1.020480000	0.987516000	0.0634	0.16
3	0.995104400	0.995275680	1.001906856	0.0049	0.08
4	1.001226610	1.000817379	0.999632125	0.00123	0.25

$\diamond$

Een mogelijk stop criterium voor beide iteratieschema's kan als volgt ingebouwd worden : als

$$\frac{\| x^{(k)} - x^{(k-1)} \|}{\| x^{(k)} \|}$$

kleiner is dan een vooropgegeven tolerantie ε kan het proces afgebroken worden. Voor dit doel kan elke norm definitie gebruikt worden. Nochtans is de maximum norm hier voor de hand liggend.

DEMO 6 (Matlab)

Het stelsel $Ax = b$ met

$$\begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 \\ -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix}$$

en $b = [2, 1, 2, 2, 1, 2]^T$ bezit als exacte oplossing $x = [1, 1, 1, 1, 1, 1]^T$. Implementeer dit vraagstuk. Gebruik zowel de Gauss-Jacobi als de Gauss-Seidel methode. Gebruik als initiale waarde $x^{(0)} = [0, 0, 0, 0, 0, 0]^T$. Bepaal de resultaten tot

$$\frac{||x^{(k)} - x^{(k-1)}||}{||x^{(k)}||}$$

kleiner of gelijk $\epsilon = 0.00001$ wordt.

2.5.3 De SOR methode

Om iteratieve methoden te versnellen vertrekt men doorgaans van de matrixvoorstelling van het iteratieschema. Laat ons deze techniek illustreren aan de hand van een stelsel van 4 vergelijkingen in 4 onbekenden, i.e.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} . \tag{2.57}$$

Voor de verdere bespreking is het handig de matrix A uit te drukken als een som van drie matrices, i.e.

$$A = D - L - U \,,$$

waarbij $D = \mathrm{diag}(a_{11},\, a_{22},\, a_{33},\, a_{44})\,,$

$$-L = \begin{bmatrix} 0 & 0 & 0 & 0 \\ a_{21} & 0 & 0 & 0 \\ a_{31} & a_{32} & 0 & 0 \\ a_{41} & a_{42} & a_{43} & 0 \end{bmatrix} , \qquad -U = \begin{bmatrix} 0 & a_{12} & a_{13} & a_{14} \\ 0 & 0 & a_{23} & a_{24} \\ 0 & 0 & 0 & a_{34} \\ 0 & 0 & 0 & 0 \end{bmatrix} .$$

Vergelijking (2.57) kan dan geschreven worden als :

$$Dx = (L + U)x + b \,.$$

De Gauss–Jacobi methode kan dan uitgedrukt worden als

$$Dx^{(n+1)} = (L + U)x^{(n)} + b \,, \tag{2.58}$$

of ook

$$x^{(n+1)} = D^{-1}(L + U)x^{(n)} + D^{-1}b \,. \tag{2.59}$$

De matrix $D^{-1}(L + U)$ wordt de Gauss–Jacobi iteratiematrix genoemd. Op analoge wijze wordt de Gauss–Seidel iteratie gedefinieerd als

$$Dx^{(n+1)} = Lx^{(n+1)} + Ux^{(n)} + b \,. \tag{2.60}$$

Hieruit volgt :

$$(D - L)x^{(n+1)} = Ux^{(n)} + b \,, \tag{2.61}$$

of

$$x^{(n+1)} = (D - L)^{-1}Ux^{(n)} + (D - L)^{-1}b \,.$$

Hierin is $(D-L)^{-1}U$ de Gauss–Seidel iteratiematrix.
De *correctie–* of *verplaatsingsvector* volgt uit vergelijking (2.60), i.e. :

$$\begin{aligned} d^{(n)} = x^{(n+1)} - x^{(n)} &= D^{-1}(Lx^{(n+1)} + Ux^{(n)} + b) - x^{(n)} \\ &= D^{-1}(Lx^{(n+1)} + Ux^{(n)} + b - Dx^{(n)})\,. \end{aligned}$$

Het rechterlid uit deze relatie kan beschouwd worden als de correctievector die bij $x^{(n)}$ moet gevoegd worden om een betere benadering $x^{(n+1)}$ te verkrijgen van de te zoeken oplossing. Wanneer de overeenkomstige componenten van $d^{(n)}$ en $d^{(n+1)}$ hetzelfde teken hebben kan het iteratieproces versneld worden door i.p.v. $d^{(n)}$ toe te voegen aan $x^{(n)}$, $\omega d^{(n)}$ als correctie te zien voor $x^{(n)}$. De factor ω, de versnellingsparameter of relaxatiefactor ligt over het algemeen in het gebied $1 < \omega < 2$. De bepaling van een optimale waarde voor ω, die zou aanleiding geven tot een maximum convergentie is een zeer moeilijk vraagstuk en niet algemeen oplosbaar. Een methode die gebruik maakt van een dergelijke versnellingsfactor heet de SOR– of "successive over–relaxation" methode. Voor die SOR methode is de correctievector

$$d_\omega^{(n)} = \omega d^{(n)}\,,$$

wat uitgeschreven kan worden als

$$x^{(n+1)} - x^{(n)} = \omega D^{-1}(Lx^{(n+1)} + Ux^{(n)} + b - Dx^{(n)})\,,$$

waaruit

$$(I - \omega D^{-1}L)x^{(n+1)} = \Big[(1-\omega)I + \omega D^{-1}U\Big]x^{(n)} + \omega D^{-1}b\,,$$

of

$$x^{(n+1)} = (I - \omega D^{-1}L)^{-1}\Big[(1-\omega)I + \omega D^{-1}U\Big]x^{(n)} + (I - \omega D^{-1}L)^{-1}\omega D^{-1}b\,. \quad (2.62)$$

Hierin is dan

$$H(\omega) = (I - \omega D^{-1}L)^{-1}[(1-\omega)I + \omega D^{-1}U]$$

de SOR iteratiematrix.
Merk op dat elk van de iteratieve methoden, hierboven beschreven, kan genoteerd worden als

$$x^{(n+1)} = Gx^{(n)} + c \quad (2.63)$$

met G de iteratiematrix welke voor elke methode aangegeven werd. Daarenboven is c een kolomvector van gekende waarden.

Voorbeeld 2.5.3

Beschouw het stelsel $Ax = b$ met

$$A = \begin{bmatrix} 4 & 3 & 0 \\ 3 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix}, \qquad b = \begin{bmatrix} 24 \\ 30 \\ -24 \end{bmatrix} \quad \text{en} \quad x^{(0)} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix},$$

dat de exacte oplossing $x = [3, 4, -5]^T$ bezit. We zullen de Gauss–Seidel en de SOR methode met $\omega = 1.25$ aanwenden om dit stelsel op te lossen. De vergelijkingen voor de Gauss–Seidel methode luiden :

$$\begin{aligned} x_1^{(k)} &= -0.75x_2^{(k-1)} + 6\,, \\ x_2^{(k)} &= -0.75x_1^{(k)} + 0.25x_3^{(k-1)} + 7.5\,, \\ x_3^{(k)} &= 0.25x_2^{(k)} - 6 \end{aligned}$$

voor elke $k = 1, 2, \ldots$
en de vergelijkingen voor de SOR methode met $\omega = 1.25$ zijn :

$$\begin{aligned} x_1^{(k)} &= -0.25x_1^{(k-1)} - 0.9375x_2^{(k-1)} + 7.5\,, \\ x_2^{(k)} &= -0.9375x_1^{(k)} - 0.25x_2^{(k-1)} + 0.3125x_2^{(k-1)} + 9.375\,, \\ x_3^{(k)} &= 0.3125x_2^{(k)} - 0.25x_3^{(k-1)} - 7.5\,. \end{aligned}$$

De eerste zeven iteratieresultaten voor beide methoden zijn weergegeven in de volgende tabel.
Men kan gemakkelijk verifiëren dat, om een nauwkeurigheid van zeven decimale cijfers te bereiken, er met de Gauss–Seidel methode 34 iteraties vereist zijn, daar waar met de SOR methode met $\omega = 1.25$ enkel 14 iteraties volstaan om hetzelfde effect te bereiken. ◇

m	Gauss–Seidel			SOR		
	$x_1^{(m)}$	$x_2^{(m)}$	$x_3^{(m)}$	$x_1^{(m)}$	$x_2^{(m)}$	$x_3^{(m)}$
0	1	1	1	1	1	1
1	5.250000	3.812500	–5.046875	6.312500	3.5195313	–6.6501465
2	3.1406250	3.8828125	–5.0292969	2.6223145	3.9585266	–4.6004238
3	3.0878906	3.9267578	–5.0183105	3.1333027	4.0102646	–5.0966863
4	3.0549316	3.9542236	–5.0114441	2.9570512	4.0074838	–4.9734897
5	3.0343323	3.9713898	–5.0071526	3.0037211	4.0029250	–5.0047135
6	3.0214577	3.9821186	–5.0044703	2.9963276	4.0009262	–4.9982822
7	3.0134110	3.9888241	–5.0027940	3.0000498	4.0002586	–5.0003486

DEMO 7 (Matlab

Implementeer het stelsel uit voorbeeld 2.5.3. Laat de SOR methode lopen voor verschillende waarden van ω, i.e. $\omega = 1.10, 1.15, 1.20, \ldots, 1.75$. Streef steeds een nauwkeurigheid na van de orde 1×10^{-7}. Welke ω waarde blijkt het minst aantal iteraties op te leveren ?

2.5.4 Voorwaarde voor convergentie van iteratieve methoden

Vergelijking (2.63) kan afgeleid worden uit de originele vergelijkingen door deze te herschikken in de vorm

$$x = Gx + c\,. \tag{2.64}$$

De fout $e^{(n)}$ in de n^{de} benadering wordt gedefinieerd als $e^{(n)} = x - x^{(n)}$. Door nu (2.64) lid aan lid af te trekken van (2.63) krijgen we

$$e^{(n+1)} = Ge^{(n)}\,,$$

zodat

$$e^{(n)} = Ge^{(n-1)} = G^2e^{(n-2)} = \ldots = G^ne^{(0)}\,.$$

De opeenvolgende iteratiewaarden $x^{(1)}$, $x^{(2)}, \ldots, x^{(n)}$ zullen convergeren naar x als

$$\lim_{n\to\infty} e^{(n)} = 0\,.$$

Vermits $x^{(0)}$ en dus ook $e^{(0)}$ arbitrair zijn, volgt er dat de iteratie zal convergeren als en slechts als

$$\lim_{n\to\infty} G^n = 0\,.$$

Als we nu aannemen dat de matrix G van orde m is en niet–singulier, dan kan deze gediagonalizeerd worden, d.i.

$$Gv_s = \lambda_s v_s \qquad (s = 1, 2, \ldots, m)\,.$$

Vermits v_1, $v_2, \ldots,$ v_m lineair onafhankelijke eigenvectoren zijn, kunnen ze gebruikt worden als een basis voor een m–dimensionale vectorruimte. Dan kan elke m–dimensionale vector en dus ook $e^{(0)}$ uitgedrukt worden als een lineaire combinatie van de eigenvectoren v_s $(s = 1, \ldots, m)$, d.i.

$$e^{(0)} = \sum_{s=1}^{m} c_s v_s \qquad (c_s \text{ scalairen})$$

zodat

$$e^{(n)} = G^ne^{(0)} = \sum_{s=1}^{m} c_s G^n v_s = \sum_{s=1}^{m} c_s \lambda_s^n v_s\,. \tag{2.65}$$

Daarom zal $e^{(n)} \to 0$ evolueren, voor $n \to \infty$, voor arbitraire $e^{(0)}$ als en slechts als $|\lambda_s| < 1$ voor alle s, m.a.w. de iteratie zal convergeren voor arbitraire $x^{(0)}$ als en slechts als de spectraalradius $\rho(G)$ kleiner is dan 1, d.i.

$$\rho(G) = \max_{1 \le s \le m} |\lambda_s| < 1 \,.$$

Als een bijproduct van dit resultaat volgt dat $\| G \| < 1$ een voldoende voorwaarde voor convergentie is. Dit is als volgt te bewijzen :
Vermits $Gv_s = \lambda_s v_s$ is ook

$$\| Gv_s \| = \| \lambda_s v_s \| = |\lambda_s| \, \| v_s \| \le \| G \| \| v_s \|$$

of

$$|\lambda_s| \le \| G \| \qquad \text{voor} \qquad s = 1, 2, \ldots, m \,.$$

Als $\| G \| < 1$ dan is $|\lambda_s| < 1$ $(s = 1, 2, \ldots, m)$, waaruit het gestelde.

Opmerking 2.5.1

Als de iteratiematrix G m lineair onafhankelijke eigenvectoren v_s bezit, corresponderend met de m eigenwaarden λ_s , waarbij

$$|\lambda_1| > |\lambda_2| \ge |\lambda_3| \ge \ldots \ge |\lambda_m|$$

dan kan (2.65) herschreven worden als :

$$e^{(n)} = \lambda_1^n \left[c_1 v_1 + \sum_{k=2}^{m} \left(\frac{\lambda_k}{\lambda_1} \right)^n c_k v_k \right] ,$$

waaruit volgt dat voor grote waarden van n

$$e^{(n)} \approx \lambda_1^n c_1 v_1 \,.$$

Analoog is $e^{(n+1)} \approx \lambda_1^{n+1} c_1 v_1$ zodat

$$e^{(n+1)} \approx \lambda_1 e^{(n)} \,.$$

Daarom is voor n voldoende groot

$$e^{(n+1)} - e^{(n)} \approx \lambda_1 (e^{(n)} - e^{(n-1)}) \,,$$

of $\quad [(x - x^{(n+1)}) - (x - x^{(n)})] \approx \lambda_1 [(x - x^{(n)}) - (x - x^{(n-1)})] \,,$
of $\quad x^{(n+1)} - x^{(n)} \approx \lambda_1 (x^{(n)} - x^{(n-1)}) \,,$
of $\quad d^{(n)} \approx \lambda_1 d^{(n-1)} \,.$

Deze laatste betrekking justifieert de basis voor de SOR methode omdat ze aantoont dat wanneer λ_1 positief is, de corresponderende componenten van de opeenvolgende correctievectoren, alle dezelfde tekens bezitten.

Algemene nota

Vele rekencentra bezitten speciale bibliotheekpakketten voor het oplossen van stelsels lineaire vergelijkingen. Vermelden we hier ter informatie enkele van de voornaamste: het pakket *LINPACK*, "A linear equations computer subroutine package"en LAPACK, "A linear algebra package", beiden ontwikkeld onder de auspiciën van Argonne National Laboratory en the National Science Foundation; het pakket *NAG*, "Numerical Algorithms Group". De iteratieve methoden worden doorgaans aangewend voor het oplossen van stelsels, die optreden bij het numeriek oplossen van partiële differentiaalvergelijkingen. In die context is het ook meestal relatief gemakkelijk de waarde voor de versnellingsparameter ω zo te bepalen, dat de convergentie van de SOR methode optimaal wordt.

Veel van de methoden besproken in dit hoofdstuk zijn verbonden met het werk van Gauss. *Carl Friedrich Gauss* (1777-1855) beïnvloedde de wiskundigen van de eerste helft van de 19^{de} eeuw meer dan elke andere wiskundige. Hij is bekend voor zijn werk in vele domeinen van de wiskunde, inclusief de numerieke analyse. Zijn studies in de aardmeetkunde, de sterrenkunde en de fysica, waarin zijn meest belangrijke werk samen met W. Weber ongetwijfeld over elektromagnetisme handelt, leverden Gauss een constante stroom van wiskundige onderzoeksproblemen. In deze cursus zullen we vooral zijn werk in de lineaire algebra en in het bijzonder zijn bijdrage tot het oplossen van stelsels lineaire vergelijkingen aan bod laten komen. Tevens zullen we in hoofdstuk 5 zijn voorstellen voor interpolatieformules bespreken.

De Franse majoor *André -Louis Cholesky* (1875-1918) was betrokken in de aardmeetkunde en aardevaluatie van 1906 tot 1909 gedurende de internationale bezetting van Griekenland, en later in Noord-Afrika. Hij ontwikkelde de methode, die nu onder zijn naam bekend is, om oplossingen te berekenen van kleinste kwadraten aanpassingsmethoden (zie hiervoor cursus Numerieke Analyse, tweede kandidatuur Informatica). De factorizatie van een symmetrische positief definiete matrix A in de vorm $A = LL^T$, die deel uitmaakt van die methode, kan nochtans ook afgeleid worden uit een vroeger gepubliceerd theorema van C.G.Jacobi.

Carl Gustav Jacobi (1804-1851) wiens naam verschillende keren voorkomt in deze cursus werkte in Königsberg en Berlijn. Zijn groot aantal publicaties handelt over elk aspect van de reële en complexe analyse, waarbij hij ook aandacht had voor getallentheorie en mechanica. In numerieke analyse zijn zijn bijdragen over het oplossen van stelsels lineaire vergelijkingen en over numerieke integratie zeer belangrijk. Jacobi's interesse voor stelsels lineaire vergelijkingen werd gewekt door zijn studie van de werken van Gauss over de methode van de kleinste kwadraten. Zijn naam is tevens verbonden aan een methode voor het berekenen van eigenwaarden van een symmetrische vierkante matrix (zie hoofstuk 4).

Gabriel Cramer (1704-1752) is een Zwitsers wiskundige, die heden ten dage nog steeds beroemd is voor zijn oplossing van stelsels van n lineaire vergelijkingen met n onbekenden m.b.v. determinanten. Eveneens van zijn hand is een *Introduction à l'analyse des courbes algébriques* verschenen in 1750 en beschouwd als één van de eerste handboeken over analytische meetkunde.

Ook nog vandaag wordt nog zeer veel wetenschappelijk onderzoek verricht over algoritmen voor vraagstukken uit de lineaire algebra. Eén van de toonaangevende personen in dit

gebied is *Gene Golub*. Golub is geboren in 1932 en bekwam zijn PhD aan de University of Illinois (Urbana, VSA). Hij droeg bij tot vele gebieden in het numeriek rekenen, zoals o.m. tot iteratieve en rechtstreekse oplossingsmethoden voor grote stelsels. Hij schreef verschillende boeken, waarbij het boek *Matrix Computation*, welke vermeld is in de bibliografie van deze cursus, als een standaard in het gebied van numerieke lineaire algebra wordt beschouwd

---