Die Singulärwertzerlegung (SVD) und

parallele Verfahren zur Berechnung der SVD

Udo Schendel Alke El-Fares−Ziegler

Forschungsprojekt *Parallele Numerik*
Fachbereich Mathematk und Informatik
Freie Universität Berlin 1999

Inhaltsverzeichnis

Vorwort

Jede Matrix $A \in \mathbb{K}^{m\times n}$ kann in ein Produkt $A = U\Sigma V^H$ zerlegt werden, wobei $U \in \mathbb{K}^{m\times m}$ und $V \in \mathbb{K}^{n\times n}$ unitäre Matrizen sind und $\Sigma \in \mathbb{K}^{m\times n}$ eine Matrix, die die sogenannten Singulärwerte von A enthält. Die Pseudoinverse von A ist dann $A^+ = V\Sigma^+U^H$, und man erhält daher mit Hilfe dieser Singulärwertzerlegung sehr leicht die Lösungsgesamtheit des konsistenten linearen Gleichungssystems $Ax = b$ bzw. des linearen Ausgleichsproblems $||Ax - b||_2 = \min!$ Die Anzahl der positiven Elemente der Matrix Σ ist gerade der Rang von A, und das kleinste positive Element gibt sogar den geringsten Abstand von A (in einem geeigneten Maß) zu einer Matrix vom nächst kleineren Rang an.
Die Singulärwerte sind die Quadratwurzeln der Eigenwerte von A^HA.

Die Theorie der Matrixzerlegungen ist schon lange bekannt und bekam durch die Numerische Lineare Algebra eine immer größere Bedeutung, da sie eine sichere Plattform zum Lösen vollkommen unterschiedlicher Probleme liefert.
Eine der wichtigsten Zerlegungen ist die Singulärwertzerlegung (SVD[1]), sie spielt in vielen Gebieten eine große Rolle:

- Da die SVD eine Zerlegung mit unitären Matrizen ist, ist sie zur Diskussion von n-dimensionalen Räumen gut geeignet.
- Es ist eine sehr stabile Zerlegung, kleine Veränderungen an der Matrix A erzeugen auch nur kleine Veränderungen an Σ und umgekehrt.
- Da Σ eine Diagonalmatrix ist, ist die Zerlegung auch für eine fast reguläre Matrix A leicht zu bestimmen. Bei einer regulären Matrix sorgt die Zerlegung für eine optimale Rang-Approximation.
- Dank der Pionierarbeiten von Gene Golub existieren effiziente und stabile Algorithmen.

Die Mathematiker Eugenio Beltrami (1835-1899), Camille Jordan (1838-1921), James Joseph Sylvester (1814-1897), Erhard Schmidt (1876-1959) und Hermann Weyl (1885-1955) untermauerten die Existenz und entwickelten die Theorie der SVD.

Beltrami, Jordan und Joseph Sylvester kamen über die Lineare Algebra zu ihrer Theorie, und Schmidt und Weyl näherten sich über Integralgleichungen $\varphi(s) = \lambda \int_a^b A(s,t)\varphi(t)dt$, für die die beste Approximation des Kerns $A(s,t) = A \cong \sum_{i=1}^{k} x_i y_i^T$ gesucht wurde: $\left\| A - \sum_{i=1}^{k} x_i y_i^T \right\| = min!$

Gauß kann als 'Vater' der Zerlegungen bezeichnet werden. Er transformierte die quadratische Form x^TAx in das Produkt $RD^{-1}R$, mit der Diagonalmatrix D und der oberen Dreiecksmatrix R mit denselben Diagonalelementen wie D [Gau1823]. Außerdem transformierte Gauss das Gleichungssystem $Ax = y$ durch *eliminatio indefinita* in das inverse System $x = By$. Das Geschick von Gauß im Umgang mit quadratischen Formen und Gleichungssystemen machte eine allgemeine Behandlung der least-square-Probleme[2] erst möglich.

---

[1] singular value decomposition

[2] Kleinste-Quadrate-Problem

Weiter Zerlegungen folgten:

1. Cauchy [Cau1829] festigte die Eigenschaften der Eigenwerte und Eigenvektoren symmetrischer Systeme durch in Betracht ziehen der entsprechenden homogenen Gleichungssysteme.

2. Jacobi veröffentlichte 1846 seinen berühmten Algorithmus zur Diagonalisierung einer symmetrischen Matrix [Jac1846], und nach seinem Tod folgte die Veröffentlichung der LU-Zerlegung durch Zerlegen einer bidiagonal Form im Stile von Gauß [Jac1857].

3. Weierstrass beschäftigte sich mit kanonischen Formen für Paare bilinearer Funktionen [Wei1868] - heute als das allgemeine Eigenwertprobelm bezeichnet.

Der Beginn der Singulärwertzerlegung 1873 ist also als eine lange Folge von Ergebnissen der kanonischen Formen zu betrachten.

Bezeichnungen

- $\mathbb{R}$ ist der Körper der reellen Zahlen.
- $\mathbb{C}$ ist der Körper der komplexen Zahlen.
- $\mathbb{K}$ bezeichnet einen beliebigen Körper aus $\{\mathbb{R}, \mathbb{C}\}$.
- $\mathbb{N} = \{1, 2, 3, ...\}$ sind die natürlichen Zahlen.
- $\mathbb{K}^n$ bezeichnet die Menge der Vektoren mit $n \in \mathbb{N}$ Koeffizienten aus $\mathbb{K}$.
- $\mathbb{K}^{m \times n}$ bezeichnet die Menge der Matrizen, deren $m \times n$ Koeffizienten in $m \in \mathbb{N}$ Zeilen und $n \in \mathbb{N}$ Spalten angeordnet sind.
- v_i ist je nach Zusammenhang entweder
  - die i-te Komponente des Vektors $v \in \mathbb{K}^n$ mit $v_i \in \mathbb{K}$ oder
  - ein Vektor als Element einer Folge von Vektoren $(v_i)_{i \in \mathbb{N}}$ mit $v_i \in \mathbb{K}^n$.
- $A = [a_{ij}]_{\substack{i=1,\dots,m \\ j=1,\dots,n}} \in \mathbb{K}^{m \times n}$ sei eine *Matrix* mit a_{ij} als Koeffizient in der i-ten Zeile und j-ten Spalte.
- a_{ij}^k sei der Koeffizient in der i-ten Zeile und j-ten Spalte der k-ten Iteration A_k.
- $e_i \in \mathbb{K}^n$ ist der i-te *Einheitsvektor.*
- $I = I_n \in \mathbb{K}^{n \times n}$ ist die *Einheitsmatrix.*
- $A^{-1} \in \mathbb{K}^{n \times n}$ heißt *inverse Matrix* von $A \in \mathbb{K}^{n \times n}$, und es gilt $A^{-1}A = AA^{-1} = I$.
- $A^T =: B = [b_{ij}] \in \mathbb{K}^{n \times m}$ ist die *transponierte Matrix* von $A = [a_{ij}] \in \mathbb{K}^{m \times n}$ und es gilt $b_{ij} = a_{ji}$ für alle $i = 1, ..., n$ und $j = 1, ..., m$.
- $\bar{A} =: B = [b_{ij}] \in \mathbb{C}^{m \times n}$ ist die *konjugiert-komplexe Matrix* von $A = [a_{ij}] \in \mathbb{C}^{m \times n}$ mit $b_{ij} = \bar{a}_{ij}$ für alle $i = 1, ..., n$ und $j = 1, ..., m$.
- $A^H := \bar{A}^T \in \mathbb{C}^{n \times m}$ heißt die zu $A \in \mathbb{C}^{m \times n}$ *konjugiert-transponierte Matrix.*
- $A \in \mathbb{K}^{m \times n}$ heißt *nichtnegativ* $(A \geq 0)$, wenn $a_{ij} \geq 0$ für alle $i = 1, ..., m$ und $j = 1, ..., n$.
- $A \in \mathbb{K}^{m \times n}$ heißt *positiv* $(A > 0)$, wenn $a_{ij} > 0$ für alle $i = 1, ..., m$ und $j = 1, ..., n$.
- $A \in \mathbb{R}^{n \times n}$ heißt *symmetrisch*, falls $A^T = A$.
- $A \in \mathbb{C}^{n \times n}$ heißt *hermitesch*, falls $A^H = \bar{A}^T = A$.
- $A \in \mathbb{R}^{n \times n}$ heißt *orthogonal*, falls $A^T A = I$.
- $A \in \mathbb{R}^{n \times n}$ heißt *normal*, falls $A^T A = AA^T$.
- $A \in \mathbb{C}^{n \times n}$ heißt *unitär*, falls $A^H A = I$.

1 Grundlagen

Es werden zunächst einige klassische Begriffe und Eigenschaften der numerischen Linearen Algebra für die folgenden Betrachtungen zusammengefaßt.

1.1 Normen

Definition 1.1.1 (Vektornorm) *Die Abbildung $\|\cdot\| : \mathbb{K}^n \longrightarrow \mathbb{R}$ heißt Vektornorm auf $\mathbb{K}^n$, wenn für alle $x, y \in \mathbb{K}^n$ und $\alpha \in \mathbb{K}$ gilt:*

1. $\|x\| \geq 0$ *(Definitheit)*
2. $\|x\| = 0 \Longleftrightarrow x = 0$
3. $\|\alpha x\| = |\alpha| \cdot \|x\|$ *(Homogenitätseigenschaft)*
4. $\|x + y\| \leq \|x\| + \|y\|$ *(Dreiecksungleichung)*

Bemerkung 1.1.2 $\|x\|$ kann als Abstand des Vektors $x \in \mathbb{K}^n$ zum Nullpunkt angesehen werden. ∘

Die wichtigsten **Vektornormen** für $x \in \mathbb{K}^n$:

- Betragssummennorm (l_1-Norm): $\|x\|_1 := \sum_{i=1}^{n} |x_i|$
- Euklidische Norm (l_2-Norm): $\|x\|_2 := \left(\sum_{i=1}^{n} |x_i|^2 \right)^{\frac{1}{2}}$
- Hölder-Norm: $\|x\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$ für $1 \leq p < \infty$
- Maximumnorm (l_∞-Norm): $\|x\|_\infty := \max_{i=1,\ldots,n} |x_i|$

Bemerkung 1.1.3 1. Ein Vektorraum mit einer Norm $\|\cdot\|$ heißt *normierter Raum*.

2. Für $p < 1$ erfüllt $\|x\|_p$ nicht mehr die Norm-Eigenschaften.
3. Es gilt $\lim_{p \to \infty} \|x\|_p = \|x\|_\infty$.
4. Die Euklidische Norm $\|x\|_2$ läßt sich im $\mathbb{K}^n$ mit Hilfe des Skalarproduktes darstellen: $\|x\|_2 = +\sqrt{\langle x, x \rangle}$, da $\langle x, y \rangle := \sum_{i=1}^{n} x_i \bar{y}_i,\ x, y \in \mathbb{K}^n$ gilt.

∘

Bemerkung 1.1.4 (Norm-Ungleichungen) Für alle $x, y \in \mathbb{K}^n$ gilt

1. Cauchy-Schwarz-Ungleichung: $|\langle x, y \rangle| \leq \|x\|_2 \cdot \|y\|_2$.
2. Dreiecksungleichung: $\|x + y\|_2 \leq \|x\|_2 + \|y\|_2$.
3. $|\|x\| - \|y\|| \leq \|x - y\|$.

∘

Satz 1.1.5 (Norm-Äquivalenz-Theorem für Vektornormen) *$\|\cdot\|$ sei eine beliebige Norm auf dem normierten Vektorraum $\mathbb{K}^n$ und $\|\cdot\|_2$ die euklidische Norm. Dann existieren positive Zahlen $c_1 > 0$ und $c_2 > 0$ derart, daß für alle $x \in \mathbb{K}^n$ gilt*

$$c_1\|x\|_2 \leq \|x\| \leq c_2\|x\|_2. \qquad \text{[SchZi97]}$$

Bemerkung 1.1.6 Alle Normen des $\mathbb{K}^n$ sind äquivalent.

∘

Satz 1.1.7 *Für alle $x \in \mathbb{K}^n$ und $n \in \mathbb{N}$ gilt:*

1. $\|x\|_\infty \leq \|x\|_2 \leq \|x\|_1,$
2. $\|x\|_\infty \leq \|x\|_1 \leq n \cdot \|x\|_\infty,$
3. $\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n} \cdot \|x\|_\infty,$
4. $\frac{1}{\sqrt{v}}\|x\|_1 \leq \|x\|_2 \leq \|x\|_1.$ [SchZi97]

Definition 1.1.8 (Matrixnorm) *Die Matrixnorm $\|\cdot\| : \mathbb{K}^{m\times n} \longrightarrow \mathbb{R}$ ist eine nichtnegative Funktion mit folgenden Eignschaften für alle $A, B \in \mathbb{K}^{m\times n},\ \alpha \in \mathbb{K}$*

1. $\|A\| \geq 0$ *(Definitheit)*
2. $\|A\| = 0 \Longleftrightarrow A = 0$
3. $\|\alpha A\| = |\alpha| \cdot \|A\|$ *(Homogenität)*
4. $\|A + B\| \leq \|A\| + \|B\|$ *(Dreiecksungleichung).*

Definition 1.1.9 (submultiplikativ) *Eine Matrixnorm $\|\cdot\|$ für quadratische Matrizen nennt man submultiplikativ, wenn für alle $A, B \in \mathbb{K}^{n\times n}$ gilt*

$$\|A \cdot B\| \leq \|A\| \cdot \|B\|.$$

Definition 1.1.10 *Sei $\|\cdot\|$ eine Vektornorm auf $\mathbb{K}^n$. Die Abbildung*

$$\begin{aligned} \|\cdot\|_{\text{nat}} : \mathbb{K}^{n\times n} &\longrightarrow \mathbb{R} \\ A &\longmapsto \|A\|_{\text{nat}} := \sup_{x\neq 0} \tfrac{\|Ax\|}{\|x\|} \end{aligned}$$

definiert die der Vektornorm $\|\cdot\|$ zugehörige natürliche Matrixnorm.

Bemerkung 1.1.11 Sind auf $\mathbb{K}^m$ und $\mathbb{K}^n$ je eine Vektornorm $\|\cdot\|$ gegeben, so kann entsprechend der Definition 1.1.10 durch

$$\|A\| := \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|}$$

für $A \in \mathbb{K}^{m\times n}$ eine zugehörige Matrixnorm auf $\mathbb{K}^{m\times n}$ definiert werden.

∘

Die wichtigsten **Matrixnormen** für $A = [a_{ij}] \in \mathbb{K}^{m\times n}$:

- Spaltennorm: $\|A\|_1 := \max\limits_{1\leq j\leq n} \sum\limits_{i=1}^{n} |a_{ij}|$
- Zeilennorm: $\|A\|_\infty := \max\limits_{1\leq i\leq m} |a_{ij}|$
- Spektralnorm (Euklidische Norm): $\|A\|_2 := \sqrt{\lambda_1}$, dabei ist λ_1 ist der betragsgrößte Eigenwert von $A^H A$.
- Frobenius-Norm oder Schur-Norm: $\|A\|_F := \sqrt{\sum\limits_{j=1}^{n}\sum\limits_{i=1}^{m} |a_{ij}|^2}$.

Bemerkung 1.1.12 $\|E\|_1 = \|E\|_2 = \|E\|_\infty = 1 \quad \text{und} \quad \|E\|_F = \sqrt{n}$.

∘

Satz 1.1.13 *1. Die Zeilennorm $\|\cdot\|_\infty$ ist die durch die Vektornorm $\|\cdot\|_\infty$ induzierte natürliche Matrixnorm.*

2. Die Spaltennorm $\|\cdot\|_1$ ist die durch die Vektornorm $\|\cdot\|_1$ induzierte natürliche Matrixnorm.

3. Für die natürliche Matixnorm, die von der Vektornorm $\|\cdot\|_2$ induziert wird gilt: $\|A\|_2 := \max\limits_{\|x\|_2=1} \|Ax\|_2$.

4. Die Matrixnorm $\|\cdot\|_F$ ist mit der Vektornorm $\|\cdot\|_2$ kompatibel, d.h. $\|Ax\|_2 \leq \|A\|_F \|x\|_2$.

[SchZi97]

Satz 1.1.14 (Norm-Äquivalenz-Theorem für Matrixnormen) *Sei $\|\cdot\|$ eine beliebige Matrixnorm und $A = [a_{ij}] \in \mathbb{K}^{m\times n}$, dann existieren positive Zahlen $c_1 > 0$ und $c_2 > 0$ mit*

$$c_1 \cdot \max_{\substack{1\leq i\leq m\\1\leq j\leq n}} |a_{ij}| \leq \|A\| \leq c_2 \cdot \max_{\substack{1\leq i\leq m\\1\leq j\leq n}} |a_{ij}|.$$

[SchZi97]

1.2 Betrachtungen spezieller Matrizen

Lemma 1.2.1 *Sei $A \in \mathbb{R}^{m\times n}$ und $Q \in \mathbb{R}^{m\times m}$ eine orthogonale Matrix, dann gilt $\|QA\|_2 = \|A\|_2$.* [Maess85]

Definition 1.2.2 (Äquivalenz von Matrizen)

1. *Zwei $m \times n-$Matrizen A und B sind äquivalent ($A \sim B$), wenn es zwei reguläre Matrizen $U \in \mathbb{K}^{m\times m}$ und $V \in \mathbb{K}^{n\times n}$ mit $B = U^T AV$ gibt.*
2. *Die Transformation $A \mapsto B := U^T AV$ heißt Äquivalenztransformation.*

Definition 1.2.3 (Ähnliche Matrizen)

1. *Zwei $n \times n-$Matrizen A und B, die mittels einer nichtsingulären $n \times n-$Matrix X durch $B = X^{-1}AX$ verknüpft sind, heißen ähnlich.*
2. *Die Transformation $A \mapsto X^{-1}AX$ nennt man Ähnlichkeitstransformation.*

Definition 1.2.4 (Dreiecksmatrix) *Eine Matrix* $A = [a_{ij}] \in I\!\!R^{n\times n}$ *heißt obere (untere) Dreiecksmatrix, wenn* $a_{ij} = 0$ *für alle* $i, j = 1, ..., n$ *mit* $i > j$ $(i < j)$ *gilt.*

Definition 1.2.5 (Hessenberg-Matrix) *Eine Matrix* $A = [a_{ij}] \in I\!\!R^{n\times n}$ *heißt obere (untere) Hessenberg-Matrix, wenn* $a_{ij} = 0$ *für alle* $i, j = 1, ..., n$ *mit* $i > j + 1$ $(j > i + 1)$ *gilt.*

Definition 1.2.6 (Diagonalmatrix) *Eine Matrix* $A = [a_{ij}] \in I\!\!R^{n\times n}$ *heißt Diagonalmatrix, wenn* $a_{ij} = 0$ *für alle* $i, j = 1, ..., n$ *mit* $i \neq j$ *gilt.*
Sei $d := a_{ii}$*, dann ist* $A := \mathrm{diag}(d_1 ... d_n)$.

Definition 1.2.7 (Tridiagonalmatrix) *Eine Matrix* $A = [a_{ij}] \in I\!\!R^{n\times n}$ *heißt Tridiagonalatrix, wenn* $a_{ij} = 0$ *für alle* $i, j = 1, ..., n$ *mit* $\|i - j\| \geq 1$ *gilt.*

Definition 1.2.8 (unreduziert) *Eine Hessenberg-Matrix oder Tridiagonalmatrix heißt unreduziert, wenn sämtliche Subdiagonalelemente* $a_{i+1,i}$, $i = 1, ..., n-1$*, von Null verschieden sind.*

Definition 1.2.9 (Housholder-Matrix) *Die Matrix* $H := I - \frac{2hh^T}{h^Th} \in I\!\!R^{n\times n}$, $h \in I\!\!R^n \backslash \{0\}$*, wird Housholder-Matrix genannt.*

Bemerkung 1.2.10 Eine Housholder-Matrix ist

1. symmetrisch, denn sowohl I, als auch $\frac{2hh^T}{h^Th}$ sind symmetrisch,
2. orthogonal, denn es gilt $HH^T = \left(I - \frac{2hh^T}{h^Th}\right)\left(I - \frac{2hh^T}{h^Th}\right) = I - 2\frac{2hh^T}{h^Th} + \frac{4hh^T}{h^Th} = I$.

∘

Bemerkung 1.2.11 Die Housholder-Matrizen sind elementare Spiegelungen, denn man kann mit ihnen einen Vektor $x \in I\!\!R^m$ in ein Vielfaches eines Einheitsvektors e_i linear transformieren. ∘

Lemma 1.2.12 *Ist* $x \in I\!\!R^n \backslash \{0\}$ *und definiert man* $u := x + \mathrm{sign}(x_1)\|x\|_2 e_1$*, so ist durch*

$$H := I - \frac{2hh^T}{h^Th} = I - \beta uu^T \quad \text{mit } \beta := \frac{2}{u^Tu} = \frac{1}{\|x\|_2(\|x\|_2 + |x_1|)}$$

eine Housholder-Matrix mit $Px = -\mathrm{sign}(x_1)\|x\|_2 e_2$ *gegeben.* [Werner92]

Definition 1.2.13 (Givens-Rotations-Matrix) *Eine Matrix*

$$\mathrm{R(i,k)} = \begin{bmatrix} 1 & & & & & & & 0 \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & c_{ii} & \cdots & s_{ik} & & \\ & & & & 1 & & & \\ & & & & & \ddots & & \\ & & & & & 1 & & \\ & & & -s_{ki} & \cdots & c_{kk} & & \\ & & & & & & 1 & \\ & & & & & & & \ddots \\ 0 & & & & & & & 1 \end{bmatrix}$$

mit den Eigenschaften $c_{ii} = c_{kk} = c$, $s_{ik} = s_{ki} = s$ *und* $s^2 + c^2 = 1$ *heißt Givens-Rotations-Matrix.*

Bemerkung 1.2.14 Wegen $R(i,k)R^T(i,k) = I$ gehört $R(i,k)$ zu der Klasse der orthogonalen Matrizen.

$\circ$

Bemerkung 1.2.15 (Eigenschaften der Givens-Rotation) Ist $x \in \mathbb{R}^m$ und $y := R(i,k)x$, so ist

$$y_j = \begin{cases} cx_i + sx_k & \text{für } j = i, \\ -sx_i + cx_k & \text{für } j = k, \\ x_j & \text{für } j \neq i,k. \end{cases}$$

1. D.h. der Vektor x wird bei der Multiplikation mit $R(i,k)$ nur in der i-ten und k-ten Komponente verändert.
2. Die Komponenten werden um den Winkel θ gedreht, wobei $c = cos\theta$ und $s = sin\theta$.
3. Multipliziert man also eine Matrix $A \in \mathbb{R}^{n \times n}$ von links mit der Givens-Rotation $R(i,k)$, so bewirkt dies lediglich eine Veränderung der i-ten und der k-ten Zeile.

$\circ$

Bemerkung 1.2.16 (Elimination mit Givens-Rotationen) Um die Komponenten $x_k \neq 0$ mit $R(i,k)$ in $y_k := R(i,k)x_k = 0$ zu überführen setzt man

$$s := \frac{x_k}{\sqrt{x_i^2 + x_k^2}} \quad \text{und} \quad c := \frac{x_i}{\sqrt{x_i^2 + x_k^2}}$$

und erfüllt damit $s^2 + c^2 = 1$, sofern $x_i \neq 0 \wedge x_k \neq 0$.

$\circ$

1.3 Eigenwerte

Definition 1.3.1 (Eigenwerte, Eigenvektoren) $\lambda \in \mathbb{K}$ *heißt Eigenwert (EW) der Matrix* $A \in \mathbb{K}^{n \times n}$*, wenn ein Vektor* $z = [z_1, \ldots, z_n]^T \in \mathbb{K}^n \backslash \{0\}$ *existiert, für den gilt*

$$Az = \lambda z. \tag{1.1}$$

z *heißt der zu* λ *gehörige (Rechts-)Eigenvektor (EV).*

Bemerkung 1.3.2

1. Sei $\lambda \in \mathbb{K}$ der zu $z \in \mathbb{K}^n$ gehörende Eigenwert, dann ergibt sich das Gleichungssystem
$$Az - \lambda z = (A - \lambda E)z = 0.$$
2. Die Menge $\mathcal{L}(\lambda) := \{x | \, x \in \mathbb{K}^n : \ (A - \lambda E)x = 0\}$ bildet einen linearen Teilraum des $\mathbb{K}^n$ mit der Dimension $\dim(\mathcal{L}(\lambda)) = n - \text{rang}(A - \lambda E)$.
3. λ ist genau dann Eigenwert von A, wenn $\mathcal{L}(\lambda) \neq \emptyset$, d.h. wenn $\dim(\mathcal{L}(\lambda)) > 0$ und damit $A - \lambda E$ singulär ist, also $\det(A - \lambda E) = 0$.

$\circ$

Definition 1.3.3 (Charakteristische Polynom) *Die Determinante* $\det(A - \lambda E)$ *definiert das charakteristische Polynom* $\varphi_A(\lambda)$ *der Matrix* A*:*

$$\varphi_A(t) = \det(A - tE) = (-1)^n(t^n + c_{n-1}t^{n-1} + \cdots + c_1 t + c_0), \qquad c_i \in \mathbb{K},\ i = 0, ..., n-1.$$

Definition 1.3.4 (Spektrum, Spektralradius einer Matrix)

1. *Die Menge der Nullstellen* $\lambda_1, ..., \lambda_n$ *des charakteristischen Polynoms*

$$\lambda(A) := \{\lambda_1, ..., \lambda_n\},\ \lambda_i \in \mathbb{C},$$

bezeichnet man als Spektrum von A.

2. *Der Spektralradius von* A *ist definiert als*

$$\rho(A) := \max\{|\lambda| : \lambda \in \lambda(A)\}.$$

Bemerkung 1.3.5 (Shift) Ist z Eigenvektor zum Eigenwert λ von $A \in \mathbb{K}^{n\times n}$, so ist $\lambda - k$ der Eigenwert der geschifteten Matrix $A - kI$ mit dem Eigenvektor z. $k \in \mathbb{C}$ wird als Shift bezeichnet. Diesen Vorgang nennt man auch Spektralverschiebung.

$\circ$

Eine wichtige Eigenschaft des charakteristischen Polynoms φ:

Satz 1.3.6 *Sind* A *und* B *zwei ähnliche Matrizen, so haben sie das gleiche charakteristische Polynom,* $\varphi_A(\lambda) \equiv \varphi_B(\lambda)$. [HoJo85]

2 Singulärwertzerlegung

Die Singulärwertzerlegung spielt eine entscheidende Rolle u.a. bei der Behandlung von Ausgleichsproblemen, Pseudoinversen, Integralgleichungen und in der Approximationstheorie.

Definition 2.1 (Singulärwerte) *Die Singulärwerte einer Matrix $A \in \mathbb{K}^{m \times n}$ sind die positiven Wurzeln der Eigenwerte der Matrix $A^H A \in \mathbb{K}^{n \times n}$.*

Der folgende Satz über die SVD zeigt, daß auch nicht-quadratische Matrizen mittels orthogonaler bzw. unitärer Transformationen auf Diagonalgestalt umgeformt werden können. Dieser Satz gilt sowohl für reelle, als auch für komplexe Matrizen, d.h. $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$. Im weiteren werden dann nur noch reelle Matrizen betrachtet, da diese für die numerischen Berechnungen relevant sind.

Satz 2.2 (Singulärwertzerlegung) *Für jede Matrix $A \in \mathbb{K}^{m \times n}$ gibt es zwei unitäre Matrizen*

$$U = [u_1, ..., u_m] \in \mathbb{K}^{m \times m} \quad \textit{und} \quad V = [v_1, ..., v_n] \in \mathbb{K}^{n \times n},$$

mit denen eine Äquivalenztransformation von A auf Diagonalform

$$U^H A V = \Sigma := diag(\sigma_1, \sigma_2, ..., \sigma_k) \in \mathbb{K}^{m \times n} \qquad k := \min\{m, n\} \tag{2.3}$$

möglich ist, so daß die Diagonalwerte von Σ folgende Ungleichung erfüllen:

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_k = 0 \qquad \textit{mit } r = \text{rang(A)}.$$

Die Diagonalelemente σ_i sind die Singulärwerte der Matrix A.
Falls $A \in \mathbb{K}^{n \times n}$ eine reguläre Matrix ist, dann gilt speziell $\sigma_1 \geq \sigma_2 \geq \sigma_n > 0$.

Beweis: $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$ seien die Eigenwerte von $A^H A$ und
$\sigma_1, ..., \sigma_r$ die positiven Singulärwerte von A mit $\sigma_i := +\sqrt{\lambda_i}$ für alle $i = 1, 2, ...n$.
$V = [v_1 ... v_n] \in \mathbb{K}^{n \times n}$ sei eine unitäre Matrix, deren Spalten $v_1, ..., v_n$ ein Orthonormalsystem von Eigenvektoren von $A^H A$ zu den Eigenvektoren $\lambda_1, \lambda_2, ..., \lambda_n$ bilden.
Es gelte $V = [V_1 V_2]$, mit
$V_1 = [v_1 ... v_r]$ und
$V_2 = [v_{r+1} ... v_n]$, wobei $v_{r+1}, ..., v_n$ die zum Eigenwert 0 gehörenden Eigenvektoren sind.
Sei $\Sigma_r := \text{diag}(\sigma_1, ..., \sigma_r)$ und $\Sigma := \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{m \times n}$.
Dann ist $V^H A^H A V = [V_1 V_2]^H A^H A [V_1 V_2] = \Sigma^H \Sigma = \begin{bmatrix} \Sigma_r^2 & 0 \\ 0 & 0 \end{bmatrix}$,
es gilt also $V_2^H A^H A V_2 = 0$, d.h. $AV_2 = 0$ und $V_1^H A^H A V_1 = \Sigma_r^2$, d.h. $\Sigma_r^{-1} V_1^H A^H A V_1 \Sigma_r = I_r$.
$U_1 := A V_1 \Sigma_r^{-1} \Rightarrow U_1^H U_1 = I_r$.
Die Spalten von U_1 bilden also ein Orthonormalsystem. Diese werden mit $m - r$ weiteren Vektoren $[u_{r+1} ... u_m] =: U_2 \in \mathbb{K}^{m-r}$ zu einer Orthonormalbasis ergänzt.
$U := [U_1 U_2] \in \mathbb{K}^m$ ist also unitär, insbesondere ist $0 = U_2^H U_1 \Sigma_r = U_2^H A V_1$.

$$\Rightarrow U^H A V = \begin{bmatrix} U_1^H \\ U_2^H \end{bmatrix} A [V_1 V_2] = \begin{bmatrix} U_1^H A V_1 & U_1^H A V_2 \\ U_2^H A V_1 & U_2^H A V_2 \end{bmatrix} = \begin{bmatrix} \Sigma_r^{-1} V_1^H A^H A V_1 & 0 \\ U_2^H U_1 \Sigma_r & 0 \end{bmatrix} = \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix}.$$

$\square$

Definition 2.4 (Singulärvektoren)

1. *Die Matrix U besteht aus den orthonormierten Eigenvektoren $u_1, ..., u_m$ von AA^H, die man Linkssingulärvektoren von A nennt, und*

2. *die Matrix V aus den orthonormierten Eigenvektoren $v_1, ..., v_n$ von $A^H A$, den Rechtssingulärvektoren von A.*

Bemerkung 2.5 Es gelten die Relationen $Av_i = \sigma_i u_i$ und $A^H u_i = \sigma_i v_i,\ i = 1, ..., r := \min\{m, n\}$.
∘

2.1 Eigenschaften der Singulärwertzerlegung

- $A^T A$ und $\Sigma^T \Sigma = \operatorname{diag}(\sigma_1^2, \sigma_2^2, ..., \sigma_r^2, 0, ..., 0)$ sind orthogonalähnlich, denn es gilt $\Sigma^T \Sigma = (U^T AV)^T (U^T AV) = V^T A^T U U^T AV = V^{-1} A^T AV$.
- Da ähnliche Matrizen das gleiche Spektrum besitzen (vgl. Satz 1.3.6) gilt $\sigma_i = +\sqrt{\lambda_i}$ mit $\lambda_i \in \lambda(A^T A),\ i = 1, 2, ..., n$.
- Die Matrizen $A \in \mathbb{R}^{m \times n}$ und $A^T \in \mathbb{R}^{n \times m}$ haben die gleichen Singulärwerte, welche eindeutig bestimmt werden können.
- Die Singulärvektoren hingegen sind nur dann eindeutig, wenn σ_i^2 ein einfacher Eigenwert von $A^T A$ bzw. AA^T ist. Für mehrfache Singulärwerte können die zugeordneten Singulärvektoren als irgendeine orthonormale Basis des entsprechenden Eigenraums gewählt werden.
- Die Spektralnorm und Frobenius-Norm einer Matrix $A \in \mathbb{R}^{n \times n}$ stehen in engem Zusammenhang mit den Singulärwerten von A:
$$\begin{aligned} \|A\|_2 &= \sigma_1, \\ \|A\|_F &= \sqrt{\sigma_1^2 + ... + \sigma_k^2}. \end{aligned}$$ [BuGe85]
- Auch die Determinante $\det(A)$ einer quadratischen Matrix $A \in \mathbb{R}^{n \times n}$ hängt mit den Singulärwerten zusammen:
$$|\det(A)| = |\det(U) \cdot \det(S) \cdot \det(V^T)| = \prod_{i=1}^{k} \sigma_i,$$
da die Determinate einer orthogonalen Matrix entweder +1 oder -1 ist.

Lemma 2.1.1 *Sei $A \in \mathbb{R}^{m \times n}$, $m \geq n$ und $P = [p_1, ..., p_m] \in \mathbb{R}^{m \times m}$, $Q = [q_1, ..., q_n] \in \mathbb{R}^{n \times n}$ orthogonale Matrizen und*
$$P^T AQ = \begin{bmatrix} B \\ 0 \end{bmatrix} \begin{matrix} \}n \\ \}m-n \end{matrix}.$$
Dann gilt

1. *$A \in \mathbb{R}^{m \times n}$ und $B \in \mathbb{R}^{n \times n}$ haben dieselben Singulärwerte.*

2. *Ist $B = U_B \operatorname{diag}(\sigma_1, ..., \sigma_n) V_B^T$ mit den orthogonalen Matrizen $U_B \in \mathbb{R}^{n \times n}$, $V_B \in \mathbb{R}^{n \times n}$ eine Singulärwertzerlegung von B, dann ist durch*
$$A = U \operatorname{diag}(\sigma_1, ..., \sigma_n) V^T, \qquad \text{mit } U := (p_1, ..., p_n) U_B,\ V := QV_B,$$
eine Singulärwertzerlegung von A gegeben. [Werner92]

2.2 SVD und Geometrie linearer Abbildungen

Durch die Singulärwertzerlegung kann man also in der Äquivalenzklasse aller Matrizen, die einer gegebenen linearen Abbildung zugeordnet sind, durch Wahl geeigneter Basen einen besonders einfachen Repräsentanten finden.

Geht man in einem linearen Raum X durch $x = Vx'$ und im Raum Y durch $y = Uy'$ zu neuen Koordinatensytemen über, so wird die lineare Abbildung, die ursprünglich durch A repräsentiert wird, nun durch Σ dargestellt:

$$y' = U^T y = U^T AX = U^T A(Vx') = (U^T AV)x' = \Sigma x'.$$

In den neuen Koordinatensytemen hat die ursprünglich der Matrix A zugrunde liegende lineare Abbildung eine sehr einfache Darstellung. Bei einer $n \times n$-Matrix ist $y' = (\eta_1', ..., \eta_n')^T$ durch

$$\eta_1' = \sigma_1 \xi_1', \; ..., \; \eta_r' = \sigma_r \xi_r', \quad \eta_{r+1}' =, ..., = \eta_n' = 0 \tag{2.2.1}$$

mit $x' = (\xi_1', ..., \xi_n')^T$ verbunden.

Bemerkung 2.2.2 Die erste Koordinatenachse des Raums X wird auf die ersten Koordinatenachse des Raumes Y mit dem Skalierungsfaktor $\sigma_1 > 0$ abgebildet, analog bei der zweiten bis r-ten Achse. Die restlichen Koordinaten werden auf den Nullvektor $0 \in Y$ abgebildet.

○

Bemerkung 2.2.3 Aus (2.2.1) folgt, daß durch $y' = \Sigma x'$ die Oberfläche der Einheitskugel

$$K_n = \{x' : \|x'\|_2 = 1\}$$

auf das r-dimensionale Hyperellipsoid

$$E(\sigma_1, ..., \sigma_n) = \{y' = (\eta_1', ..., \eta_n')^T : \left(\frac{\eta_1'}{\sigma_1}\right)^2 + \cdots + \left(\frac{\eta_r'}{\sigma_r}\right)^2 = 1, \; \eta_{r+1}' = \cdots = \eta_n' = 0\}$$

abgebildet wird.

○

Bemerkung 2.2.4
1. Im Hyperellipsoid $E(\sigma_1, ..., \sigma_n)$ ist $(\sigma_1, 0, ..., 0)^T$ (einer) der Vektor(en) mit dem größten Abstand vom Nullvektor $0 \in I\!R^n$.
2. Ist $r < n$, so enthält $E(\sigma_1, ..., \sigma_n)$ den Nullvektor.
3. Im Fall einer regulären Matrix A enthält $E(\sigma_1, ..., \sigma_n)$ den Nullvektor nicht, und $(0, ..., 0, \sigma_n)^T$ ist (einer) der Vektor(en) mit dem kleinsten Abstand vom Nullvektor. In diesem Fall gilt
$$\Sigma^{-1} = \text{diag}(\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, ..., \frac{1}{\sigma_n}).$$
4. Daraus ergibt sich, daß $\frac{1}{\sigma_1}, ..., \frac{1}{\sigma_n}$ die Singulärwerte von A^{-1} sind.

○

Die Singulärwertzerlegung kann man im $\mathbb{R}^2$ sehr gut geometrisch veranschaulichen. Die orthogonalen Matrizen der Zerlegung $A = U\Sigma V^T$,

$$U = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \quad \text{und} \quad V^T = \begin{bmatrix} \cos\beta & \sin\beta \\ -\sin\beta & \cos\beta \end{bmatrix}$$

beschreiben Drehungen um den Winkel α bzw. β mit dem Ursprung als Mittelpunkt.

Abbildung 2.2.5: Zusammengesetzte Abbildung $U\Sigma V^T = A \in \mathbb{R}^{2\times 2}$. Ein Vektor x wird über die Zwischenstufen $V^T x$ und $SV^T x$ schließlich auf $U\Sigma V^T x = Ax$ abgebildet.

Bemerkung 2.2.6 Aus der Singulärwertzerlegung $A = U\Sigma V^T$ läßt sich sehr viel über die Struktur der von einer Matrix $A \in \mathbb{R}^{n\times n}$ dargestellten linearen Abbildung unmittelbar ablesen.

1. Man erhält z.B. sofort orthogonale Basen des Bildraums $\mathcal{R}(A)$ und des Kerns:

$$\mathcal{R}(A) = \text{span}\{u_1, ..., u_r\} \quad \text{und} \quad \mathcal{N}(A) = \text{span}\{v_{r+1}, ..., v_n\}.$$

2. Mit den Zeilenvektoren $a_1, ..., a_n$ von A, also den Spaltenvektoren von A^T, kann man $Ax = 0$ auch in der Form

$$a_i^T x = \langle a_i, x\rangle = 0, \qquad i = 1, 2, ..., n$$

schreiben. Jeder Vektor $x \in \mathcal{N}(A)$ ist somit orthogonal zu jedem Spaltenvektor von A^T, also auch zu jedem Vektor aus

$$\mathcal{R}(A^T) := \text{span}\{a_1, a_2, ..., a_n\}.$$

$\mathcal{R}(A^T)$ ist das orthogonale Komplement des Nullraums:

$$\mathcal{R}(A^T) = \mathcal{N}(A)^\perp.$$

○

Definition 2.2.7 (Verbindungsraum, direkte Summe)

1. *Für zwei Unterräume* S_1 *und* S_2 *des* $\mathbb{R}^n$ *bezeichnet man den Unterraum* $S_1 + S_2 := \{v = v_1 + v_2 : v_1 \in S_1, v_2 \in S_2\}$ *als Verbindungsraum.*

2. *Gilt* $S_1 \cap S_2 = \{0\}$*, so wird der Verbindungsraum* $S_1 \oplus S_2 := \{v = v_1 + v_2 : v_1 \in S_1, v_2 \in S_2\}$ *als direkte Summe von* S_1 *und* S_2 *bezeichnet.*

Definition 2.2.8 (Orthogonale Projektion) *Jeder Vektor* $v \in S_1 \oplus S_2$ *kann eindeutig in der Form*

$$v = v_1 + v_2, \qquad v_1 \in S_1,\ v_2 \in S_2$$

zerlegt werden. Der Vektor v_i *heißt orthogonale Projektion von* v *auf den Unterraum* $S_i,\ i = 1, 2.$

Für eine lineare Abbildung $A : U \to W$ gilt

$$\begin{array}{rclcl} U &=& \mathcal{N}(A) \oplus \mathcal{N}(A)^\perp &=& \mathcal{N}(A) \oplus \mathcal{R}(A^T) \\ W &=& \mathcal{R}(A) \oplus \mathcal{R}(A)^\perp &=& \mathcal{R}(A) \oplus \mathcal{N}(A^T). \end{array}$$

Alle Vektoren $u \in U$ und $w \in W$ können daher durch

$$\begin{array}{lcll} u = u_A + u_\mathcal{N} & \text{mit} & u_A \in \mathcal{R}(A^T), & u_\mathcal{N} \in \mathcal{N}(A) \\ w = w_A + w_\mathcal{N} & \text{mit} & w_A \in \mathcal{R}(A), & w_\mathcal{N} \in \mathcal{N}(A^T) \end{array}$$

in eindeutiger Weise in orthogonale Komponenten zerlegt werden. u_A ist die orthogonale Projektion von u auf $\mathcal{N}(A)^\perp = \mathcal{R}(a^T)$. Die entsprechende Projektionsabbildung wird mit

$$P_{\mathcal{R}(A^T)} : U \to \mathcal{R}(A^T)$$

bezeichnet. Analog stellt

$$P_{\mathcal{R}(A)} : W \to \mathcal{R}(A)$$

die Projektion auf den Bildraum von A dar.

2.3 SVD und Pseudoinverse

Die Pseudoinverse[1] ermöglicht es, die Lösung inhomogener linearer Gleichungssysteme $Ax = b$, mit beliebigen Matrizen $A \in \mathbb{R}^{m \times n}$, im Sinne kleinster Fehlerquadrate in völlig einheitlicher Weise als Produkt gewisser Matrizen A^+ mit der rechten Seite b darzustellen[2].

Jede Abbildung A läßt sich als zusammengesetzte Abbildung

$$A = \hat{A} P_{\mathcal{R}(A^T)}$$

darstellen, wobei $\hat{A} : \mathcal{N}(A)^\perp \to \mathcal{R}(A)$ die Einschränkung von A auf $\mathcal{N}(A)^\perp$ ist:

$$\hat{A}x := Ax \qquad \text{für alle } \ x \in \mathcal{N}(A)^\perp.$$

Die Abbildung $\hat{A}$ ist injektiv und besitzt daher eine Inverse

$$\hat{A}^{-1} : \mathcal{R}(A) \to \mathcal{N}(A)^\perp,$$

mit deren Hilfe man eine neue Abbildung definieren kann, die zu einer Verallgemeinerung der Inversen einer Matrix führt.

[1] Die Pseudoinverse wird auch als verallgemeinerte Inverse bezeichnet.

[2] Wie es bei nichtsingulären quadratischen Koeffizientenmatrizen durch A^{-1} geschieht.

Definition 2.3.1 (Verallgemeinerte Inverse, Pseudoinverse) *Durch*

$$A^+w := \begin{cases} \hat{A}^{-1}w & \textit{für} \;\; w \in \mathcal{R}(A) \\ 0 & \textit{für} \;\; w \in \mathcal{R}(A)^\perp \end{cases}$$

wird die zusammengesetzte Abbildung

$$A^+ = \hat{A}^{-1}P_{\mathcal{R}}(A)$$

festgelegt, die man verallgemeinerte Inverse oder Pseudo-Inverse A^+ *von* A *nennt.*

Bemerkung 2.3.2 (Moore-Penrose-Bedingungen)

1. Die Pseudoinverse ist *eindeutig* bestimmt, und sie erfüllt die folgenden vier sogenannten Moore-Penrose-Bedingungen:

$$(AA^+)A = A \tag{2.3.3}$$
$$(A^+A)A^+ = A^+ \tag{2.3.4}$$
$$(AA^+)^T = AA^+ \tag{2.3.5}$$
$$(A^+A)^T = A^+A \tag{2.3.6}$$

2. Aus (2.3.3) und (2.3.5) folgt $P_{\mathcal{R}(A)} = AA^+$ und aus (2.3.4) und (2.3.6) folgt $P_{\mathcal{N}(A)^\perp} = A^+A$.

∘

Bemerkung 2.3.7 Ist $A \in I\!R^{n\times n}$ regulär, so sind die Projektionen $P_{\mathcal{R}(A)}$ und $P_{\mathcal{N}(A)^\perp}$ identische Abbildungen, und es gilt $A^+ = \hat{A}^{-1}P_{\mathcal{R}(A)} = \hat{A}^{-1}I = A^{-1}$.

∘

Beispiel 2.3.8 (Diagonalmatrix) Die Pseudoinverse einer Diagonalmatrix

$$A = \mathrm{diag}(\alpha_1, ..., \alpha_k, 0, ..., 0) \in I\!R^{m\times n}, \quad k \leq \min\{m, n\}$$

mit $\alpha_i \neq 0, \; i = 1, ..., k$ ist

$$A^+ = \mathrm{diag}(\frac{1}{\alpha_1}, ..., \frac{1}{\alpha_k}, 0, ..., 0) \in I\!R^{n\times m}, \tag{2.3.9}$$

denn $AA^+ = \mathrm{diag}(1, ..., 1, 0, ..., 0) \in I\!R^{m\times m}$ und $A^+A = \mathrm{diag}(1, ..., 1, 0, ..., 0) \in I\!R^{n\times n}$ erfüllen die Moore-Penrose-Bedingungen (vgl. Bemerkung 2.3.2).

∘

Bemerkung 2.3.10 Wenn zwei Matrizen $A, B \in I\!R^{m\times n}$ orthogonal-äquivalent sind, d.h. es gilt

$$A = UBV^T$$

mit orthogonalen Matrizen $U \in I\!R^{m\times m}, \; V \in I\!R^{n\times n}$, so besteht zwischen den Pseudoinversen A^+ und B^+ die Beziehung

$$A^+ = VB^+U^T. \tag{2.3.11}$$

∘

Ergebnis: Mit (2.3.9) und (2.3.11) läßt sich die Pseudoinverse einer allgemeinen Matrix $A \in I\!R^{n \times n}$ mit Hilfe ihrer Singulärwertzerlegung

$$A = U\Sigma V^T = \sum_{i=1}^{k} \sigma_i u_i v_i^T \text{ mit } \Sigma = \mathrm{diag}(\sigma_1, ..., \sigma_k, 0, ...0)$$

wie folgt darstellen:

$$A^+ = V\Sigma^+ U^T = \sum_{i=1}^{k} \frac{1}{\sigma_i} v_i u_i^T \qquad \text{mit } \Sigma^+ = \mathrm{diag}(\frac{1}{\sigma_1}, ..., \frac{1}{\sigma_k}, 0, ..., 0) \in I\!R^{n \times m}. \quad (2.3.12)$$

∘

2.4 SVD und das Ausgleichsproblem

Die Bezeichnung *Ausgleichsproblem* leitet sich von der Tatsache ab, daß mit Hilfe dieser Methode bei der mathematischen Modellbildung stochastische Schwankungen der Daten (hervorgerufen z.B. durch Meßfehler) *ausgeglichen* werden können. Zu diesem Zweck wird eine größere Anzahl an Daten erhoben, als zur Bestimmung der Modellparameter erforderlich sind, woraus folgt, daß das resultierende System überbestimmt und daher meist nicht lösbar ist. Man kann nur verlangen, daß die in den einzelnen Gleichungen auftretenden Abweichungen minimal sind, also soll gelten $\|Ax - b\|_p = \min!$. D.h. man versucht, jenen Vektor x_0 zu finden, dessen Bild Ax_0 von b die geringste Distanz hat. Diese Aufgabe ist äquivalent zur Lösung des Gleichungssystems $A^T Ax = A^T b$.
Im Jahr 1821 hat Gauss die *Methode der kleinsten Fehlerquadrate* (least-square-Lösung) entwickelt.

Bemerkung 2.4.1 (Fehlergleichung) Ein überbestimmtes lineares Gleichungssystem ist nicht eindeutig lösbar. Nach der Methode der kleinsten Quadrate können jedoch durch Minimieren der Fehlergleichung

$$Ax - b = r, \qquad A \in I\!R^{m \times n},\ x \in I\!R^n,\ b \in I\!R^m,\ r \in I\!R^m,\ m > n,$$

Lösungen gefunden werden. D.h. es wird gefordert

$$\|Ax - b\|_p = \|r\|_p = \min! \quad (2.4.2)$$

∘

Definition 2.4.3 (Residuenvektor) *Der Vektor $r^T := (r_1, ..., r_m)$ heißt Residuenvektor.*

Definition 2.4.4 (Normallösung, Pseudo-Normallösung)

1. *Ein Vektor $v \in I\!R^n$ heißt Normallösung von $Ax = b$, wenn er minimales Residuum besitzt:*

$$\|Av - b\| = \min\{\|Ay - b\| : y \in I\!R^n\}.$$

2. *Ein Vektor $x_0 \in I\!R^n$ heißt Pseudo-Normallösung von $Ax = b$, wenn er Normallösung ist und die kleinstmögliche Länge hat:*

$$\|x_0\| = \min\{\|v\| : v \textit{ ist Normallösung}\}.$$

Bemerkung 2.4.5 Welche l_p-Norm man zur Lösung von (2.4.2) verwendet, hängt stark von der Aufgabenstellung ab. Je nach Wahl des Parameters p ergibt sich ein unterschiedliches Gewicht der Abweichungen im Verhältnis zur Distanz. Kleine Abweichungen (< 1) wirken sich z. B. umso weniger aus, je größer man p wählt.
Auf Grund der Tatsache, daß die Euklidische Norm der *üblichen* Distanzmessung im Raum anschaulich entspricht und den relativ geringen Aufwand bei der Lösung linearer Ausgleichsprobleme erfordert, wird zumeist die l_2-Norm verwendet.

$\circ$

Interpretation: 2.6 Als Lösungsvektoren des Minimierungsproblems $\|Ax - b\|_2 = \min!$ treten jene $x \in I\!R^n$ auf, die unter A auf Orthogonalprojektion von b auf $\mathcal{R}(A)$ abgebildet werden. Ist die Abbildung A injektiv, d.h. gilt $\operatorname{rang}(A) = n$, so ist die Lösung eindeutig. Anderenfalls ergibt sich als Lösungsmannigfaltigkeit eine Nebenklasse des Nullraums $\mathcal{N}(A)$.

$\circ$

Die Lösung linearer Ausgleichsprobleme kann mit der Pseudoinversen, d.h. mit Hilfe der SVD einheitlich behandelt werden:

Für $A \in I\!R^{m\times n}$ existiert nach Satz 2.2 die Singulärwertzerlegung $A = U\Sigma V^T$. Setzt man nun

$$p := V^T x = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \in I\!R^n \qquad \text{und} \qquad q := U^T b = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \in I\!R^m$$

mit $p_1,\ q_1 \in I\!R^k$, so erhält man wegen der Invarianz der Euklidischen Norm gegenüber orthogonalen Transformationen (siehe Lemma 1.2.1) das Minimumsproblem

$$\|b - Ax\|_2 = \|U^T(b - U\Sigma V^T x)\|_2 = \|q - \Sigma p\|_2 = \left\| \begin{bmatrix} q_1 - \Sigma_k p_1 \\ q_2 \end{bmatrix} \right\|_2 \to \min!$$

mit $\Sigma_k := \operatorname{diag}(\sigma_1, ..., \sigma_k) \in I\!R^{k\times k}$.

Bemerkung 2.4.6 Die Norm des Residuums $\|r\|_2 = \|b - Ax\|_2$ ist für beliebiges $p_2 \in I\!R^{n-k}$ und $p_1 = \Sigma_k^{-1} q_1 \in I\!R^k$ minimal und damit ist eine Normallösung gefunden.

$\circ$

Satz 2.4.7 *Die eindeutige Pseudo-Normallösung eines linearen Ausgleichsproblems $Ax = b$ mit $A \in I\!R^{m\times n},\ x \in I\!R^n,\ b \in I\!R^m$ und $m \geq n$ ist*

$$x_0 = V \begin{bmatrix} \Sigma_k^{-1} & 0 \\ 0 & 0 \end{bmatrix} U^T b = A^+ b \qquad \textit{mit} \ \ A^+ \in I\!R^{n\times m}.$$

[Überh95]

Bemerkung 2.4.8 Hat A maximalen Rang, gilt also $\operatorname{rang}(A) = n$, so ist speziell

$$A^+ = (A^T A)^{-1} A^T.$$

Die Lösung des linearen Ausgleichsproblems ist in diesem Fall die Lösung der Normalgleichungen

$$A^T A x = A^T b. \tag{2.4.9}$$

$\circ$

2.5 Die serielle Berechnung der SVD

Die Singulärwertzerlegung einer Matrix $A \in \mathbb{R}^{m \times n}$ kann durch die Berechnung der Eigenwerte der Matrix AA^T, z.B. mit Hilfe des QR-Algorithmus, bestimmt werden. Die explizite Bildung von AA^T ist jedoch nicht empfehlenswert, da die auftretenden numerischen Störungen oft zu groß sind, um eine akzeptable Lösung zu bekommen.
Im folgenden Abschnitt wird das sehr gute serielle Verfahren von Golub und Kahn (von 1965, vgl. [GoLo84]) geschildert. Das entsprechende Programm findet man in [WiRe71].
Zu Beginn dieses Verfahrens wird die gegebene Matrix A orthogonal äquivalent in eine Bidiagonalmatrix transformiert. Die Singulärwerte werden hierbei nicht verändert (vgl. Lemma 2.1.1).

Reduktionsschritt bei der Berechnung einer Singulärwertzerlegung:

Man bestimme Housholder-Matrizen (Definition 1.2.9) $P_1, ..., P_n \in \mathbb{R}^{m \times m}$, $Q_1, ..., Q_{n-2} \in \mathbb{R}^n$ derart, daß

$$P_n \cdots P_1 A Q_1 \cdots Q_{n-2} = \left[\begin{array}{ccccc} d_1 & f_2 & 0 & \cdots & 0 \\ & d_2 & f_3 & \ddots & \vdots \\ & & \ddots & \ddots & 0 \\ & & & \ddots & f_n \\ & & & & d_n \\ \hline 0 & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & & & \vdots \\ 0 & 0 & \cdots & \cdots & 0 \end{array}\right]$$

eine obere Bidiagonalmatrix ist, mit den Diagonalelementen $d_1, ..., d_n$ und den Superdiagonalelemente $f_2, ..., f_n$.

Bemerkung 2.5.1 Grundlegend für den Reduktionsschritt ist, daß man einen vom Nullvektor verschiedenen Vektor durch Multiplikation mit einer geeigneten Housholder-Matrix in ein Vielfaches des ersten Einheitsvektors überführen kann (siehe Lemma 1.2.12).

$\circ$

Berechnung der Singulärwerte einer Bidiagonalmatrix:

Das Problem ist nun darauf reduziert, eine Singulärwertzerlegung der oberen Bidiagonalmatrix $B \in \mathbb{R}^{n \times n}$ zu berechnen.
Das folgende Lemma zeigt, daß die Singulärwertzerlung der Matrix B sich in zwei Teilprobleme aufspalten läßt. Es wird dann gezeigt, daß durch Elimination des letzten Nebendiagonalelements der Matrix B der Übergang zu einer Matrix B_+ mit $\text{rang}(B_+) = \text{rang}(B) - 1$ erreicht werden kann. Das letzte Diagonalelement ist dann ein Singulärwert.

Lemma 2.5.2 *Die Matrix $B \in I\!R^{n\times n}$ sei eine obere Bidiagonalmatrix mit Haupdiagonalelementen $d_1, ..., d_n$ sowie Superdiagonalelemente $f_2, ..., f_n$.*

1. *Sei $f_{k+1} = 0$ für ein $k \in \{1, ..., n-1\}$, so daß B zerfällt:*

$$B = \underbrace{\left[\begin{array}{c|c} B_1 & 0 \\ \hline 0 & B_2 \end{array}\right]}_{k \quad n-k} \begin{array}{l} \}k \\ \}n-k \end{array}$$

Ist dann $B_1 = U_1 \mathrm{diag}(\sigma_1, ..., \sigma_k) V_1^T$, $\quad B_2 = U_2 \mathrm{diag}(\sigma_{k+1}, ..., \sigma_n) V_2^T$

eine Singulärwertzerlegung von B_1 bzw. B_2, so ist

$$B = \left[\begin{array}{c|c} U_1 & 0 \\ \hline 0 & U_2 \end{array}\right] \mathrm{diag}(\sigma_1, ..., \sigma_n) \left[\begin{array}{c|c} V_1 & 0 \\ \hline 0 & V_2 \end{array}\right]^T$$

eine Singulärwertzerlegung von B.

2. *Ist $d_k = 0$ und $f_{k+1} \neq 0$ für ein $k \in \{1, ..., n-1\}$, so kann man $n-k$ Givens-Rotationen (Definition 1.2.13) $R_{k,k+1}, ..., R_{k,n}$ derart bestimmen, daß man eine obere Bidiagonalmatrix $B_+ := R_{k,n} \cdots R_{k,k+1} B$ mit $(b_+)_{k,k+1} = 0$ erhält.* [Werner92]

Es wird nun die obere Bidiagonalmatrix B_+ berechnet, von der man sich erhofft, daß das letzte Nebendiagonalelement wesentlich kleiner als das entsprechende in B ist. Nach wenigen Schritten ist dann eine Deflation (Übergang zu einer $(n-1)$-dimensionalen Matrix) möglich. Grundlage für diese Berechnung ist das geshiftete QR-Verfahren für symmetrische Tridiagonalmatrizen. [Werner92]

- Input: Sei $B \in I\!R^{n\times n}$ eine obere Bidiagonalmatrix mit den Haupt- bzw. Nebendiagonalelementen $d_1, ..., d_n$ bzw. $f_2, ..., f_n$. Es sei $d_1 \cdots d_n \neq 0$ und $f_2 \cdots f_n \neq 0$, d.h. $C := B^T B$ ist eine unreduzierte (Definition 1.2.8), symmetrische Tridiagonalmatrix (Definition 1.2.7). Weiter gelte $U \in I\!R^{m\times n}$, $V \in I\!R^{n\times n}$ mit $U^T U = I$, $V^T V = I$ und $A = UBV^T$.

- Bestimmen des Shift-Parameters (Bemerkung 1.3.5) $k \in I\!R$ als den Eigenwerte des unteren Blocks
$$\begin{bmatrix} d_{n-1}^2 + f_{n-1}^2 & d_{n-1} f_n \\ d_{n-1} f_n & d_n^2 + f_n^2 \end{bmatrix}$$
in C, der $d_n^2 + f_n^2$ am nächsten liegt.

- Bestimmen der Givens-Rotation R_{12} mit $\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} d_1^2 - k \\ d_1 f_2 \end{bmatrix} = \begin{bmatrix} *_1 \\ *_2 \end{bmatrix} = \begin{bmatrix} *_1 \\ 0 \end{bmatrix}$.
Berechnen von $B := BR_{12}^T$, $V := VR_{12}^T$

Bemerkung 2.5.3 In der transformierten Matrix BR_{12}^T verändert sich nur der obere 2×2-Block:

$$\begin{bmatrix} d_1 & f_2 \\ 0 & d_2 \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix} = \begin{bmatrix} d_1 c + f_2 s \\ -d_1 s + f_2 c d_2 s \\ d_2 c \end{bmatrix}.$$

In BR_{12}^T wird die obere Bidiagonal-Gestalt also nur durch das Element in der Position (2,1) gestört. ○

Das Element in der Position (2,1) soll durch Multiplikation von *links* mit einer geeigneten Givens-Rotation S_{12} annulliert werden. Die neuen ersten beiden Zeilen sind eine Linearkombination der alten, so daß nach der Transformation ein die Bidiagonal-Gestalt störendes Element in der Position (1,3) steht. Dieses Element wiederum wird durch eine Multiplikation von *rechts* mit einer Givens-Rotation R_{23}^T annulliert und auf Position (3,2) gedrängt. Durch abwechselnde Multiplikation von rechts und links mit geeigneten Givens-Rotationen wird das die Bidiagonal-Gestalt störende Element aus der Matrix gedrängt:

- Für $k = 1, ..., n-1$:

  Bestimme Givens-Rotation $S_{k,k+1}$ mit

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} b_{kk} \\ b_{k+1,k} \end{bmatrix} = \begin{bmatrix} *_1 \\ *_2 \end{bmatrix} = \begin{bmatrix} *_1 \\ 0 \end{bmatrix}.$$

  Berechne $B := S_{k,k+1}B,\ U := US_{k,k+1}^T$.
  Falls $k < n-1$, dann:

  Bestimme Givensrotaion $R_{k+1,k+2}$ mit

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} b_{k,k+1} \\ b_{k,k+2} \end{bmatrix} = \begin{bmatrix} *_1 \\ *_2 \end{bmatrix} = \begin{bmatrix} *_1 \\ 0 \end{bmatrix}.$$

  Berechne $B := BR_{k+1,k+2}^T B,\ V := VR_{k+2,k+2}^T$.

- Output: Die Ausgangsmatrix $B \in \mathbb{R}^{n \times n}$ ist überschrieben mit der oberen Bidiagonalmatrix

$$B_+ := S_{n-1,n} \cdots S_{12} B R_{12}^T \cdots R_{n-1,n}^T.$$

  Ferner sind die Matrizen $U \in \mathbb{R}^{m \times n},\ V \in \mathbb{R}^{n \times n}$ überschrieben mit

$$U_+ := US_{12}^T \cdots S_{n-1,n}^T, \qquad V_+ := VR_{12}^T \cdots R_{n-1,n}^T.$$

  Daher ist $U_+ B_+ V_+^T = UBV^T$.

Bemerkung 2.5.4 1. Durch das obige Verfahren ist ein impliziter QR-Schritt mit Shift k auf $B^T B$ durchgeführt worden. [Werner92]

2. Die im Verlauf des Algorithmus transformierte Matrix B weicht immer nur in (höchstens) einem Element von der Bidiagonal-Gestalt ab. Nach jedem Schritt $B \longrightarrow B_+$ sollte getestet werden, ob die neue Bidiagonalmatrix B_+ zerfällt, ob also z.B. das letzte Superdiagonalelement in der transformieretn Bidiagonalmatrix B_+ betragsmäßig kleiner oder gleich einer vorgegeben Toleranz ist.

$\circ$

3 Parallele Rechnerarchitekturen und Algorithmen

Die kontinuierliche Steigerung der Rechenleistung moderner Computer geht einher mit einer ebenso starken Zunahme von neuen Anwendungsgebieten, in denen sogenannte Supercomputer Verwendung finden. Gleichzeitig werden in den bereits bestehenden Einsatzgebieten Modelle weiter verbessert und verfeinert und damit auch rechenintensiver. Einige wichtige Einsatzgebiete von Parallelrechnern sind u.a. Fahrzeugkonstruktion, Wettervorhersage und Realtime-Bildverarbeitung.
Die massiv-parallelen Rechner, d.h. Computer mit bis zu mehreren tausend Prozessoren und lokalem Speicher, gewinnen zunehmend an Bedeutung, da sie den größten Leistungszuwachs versprechen. Dies stellt völlig neue Anforderungen an die Entwicklung von Algorithmen.

3.1 Parallelitätsebenen

Unter Parallelität versteht man die Möglichkeit der gleichzeitigen Ausführung mehrerer Operationen. Der Grad der Parallelität gibt an, wieviel Operationen parallel ausführbar sind. Dieser Parallelitätsgrad ist im allgemeinen keine konstante Größe, sondern er hängt vom jeweiligen Stand der Ausführung (d.h. vom Zustand des zugehörigen Programmlaufs) ab. [Giloi93]
Um eine ausreichende Übersicht über Parallelitätsstufen zu haben, genügt eine Einteilung in vier Ebenen. Von oben nach unten kann man unterscheiden:

- Programmebene: Die Parallelverarbeitung wird nur vom Betriebssystem organisiert. Darunter fallen vollständig unabhängige Programme oder solche, die nur über die vom Betriebssystem bereitgestellten Kommunikationskanäle Nachrichten austauschen.
- Prozessebene: Ein einzelnes Programm kann aus mehreren parallel ausführbaren Prozessen bestehen. Die Parallelverarbeitung kann über sprachliche Kontroll-Konstrukte (Semaphore, Monitore, Rendevous) kontrolliert werden.
- Anweisungsebene: Optimierende Compiler erkennen einzelne Maschinenbefehle oder elementare Anweisungen, die parallel zueinander ausgeführt werden.
- Ebene der Elementar-Operationen: Manche Sprachen bieten Kontroll-Konstrukte für entsprechende Datentypen und den darauf definierten Operationen an (z. B. Skalarprodukt für Vektoren in Fortran90 und High Perfomance Fortran).

3.2 Rechnermodelle

Die bekannteste Klassifikation von Rechnerarchitekturen ist nach Flynn. Er unterteilte die Architekturen bzgl. der gleichzeitigen

1. Ausführung von Instruktionen (d.h. Befehlsströme):

   SI: single-instruction-stream: ein Befehlsstrom

   MI: multiple-instruction-stream: mehrere Befehlsströme

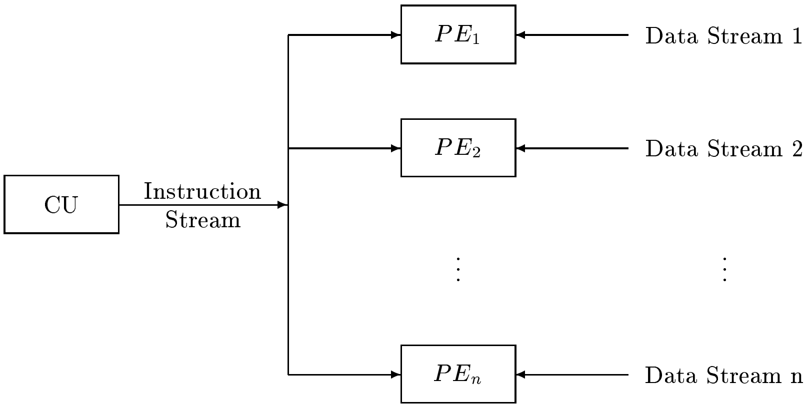
2. Bearbeitung von Daten (d.h. Operandenströme):

   SD: single-data-stream: ein Operandenstrom

   MD: multiple-data-stream: mehrere Operandenströme

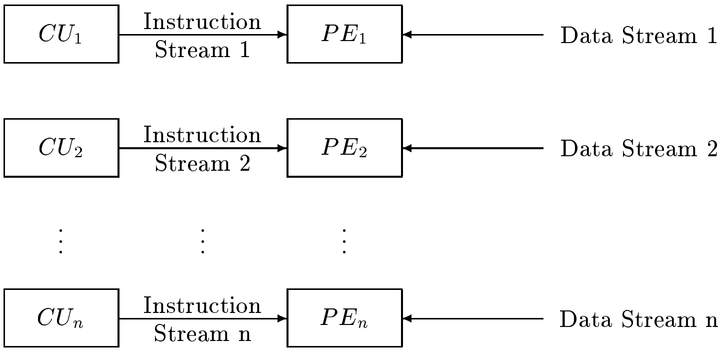
Daraus ergeben sich vier mögliche Modelle:

1. SISD-Rechner (Single Instruction- Single Datastream): Serieller J.v.Neumann-Rechner.
2. MISD-Rechner (Multiple Instruction- Single Datastream): Geringe praktische Relevanz.
3. SIMD-Rechner (Single Instruction- Multiple Datastream)

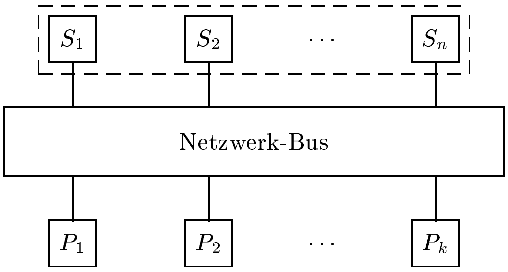


   - SIMD-Rechner bestehen aus einem Steuerwerk (Control Unit - CU) und den damit kontrollierten Prozessorelementen (PE). Das Steuerwerk bestimmt, ob ein Prozessorelement aktiv oder inaktiv ist und ist als einziges mit Peripheriegeräten (Monitor, Speichermedien, Drucker usw.) verbunden.
   - Die Prozessorelemente bestehen aus arithmetisch-logischen Einheiten (ALU) und lokalen Speichern. SIMD-Rechner können auch mit gemeinsamem Speicher ausgestattet sein.
   - SIMD-Rechner arbeiten synchron: Aktive Prozessorelemente können zu einem Zeittakt nur den gleichen Befehl ausführen. Der Datenaustausch zwischen den Prozessorelementen findet ebenfalls synchron statt.
   - Bei SIMD-Rechner unterscheidet man zwei Typen:
     - Array-Rechner (Feldrechner): besitzen mehrere Prozessoren, die simultan gleiche Operationen ausführen,
     - Vektorrechner (Pipeline-Vektorrechner): nutzen die Struktur von Vektoroperationen aus und besitzten in der Regel mehrere gepipelinte Funktionseinheiten; es wird parallel auf verschiedenen Komponenten eines Vektors operiert. [Schen95]
   - SIMD-Rechner eignen sich also besonders gut für Anwendungen, die auf Vektoren bzw. Matrizen operieren.

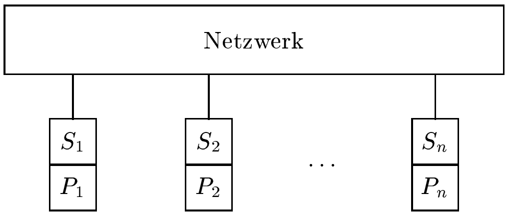
4. MIMD-Rechner (Multiple Instruction- Multiple Datastream)



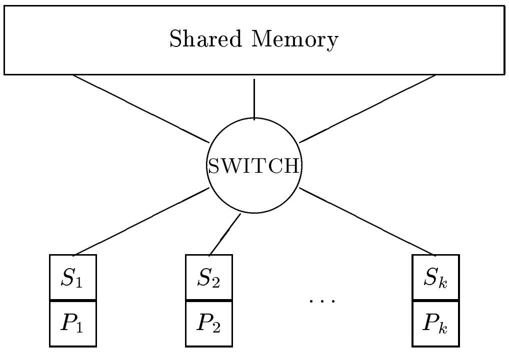
- MIMD-Rechner besitzen im allgemeinen Universalprozessoren, die unabhängig voneinander parallel arbeiten können. Daher spricht man auch von Multiprozessor-Systemen.
- Im Unterschied zu den SIMD-Rechnern können die MIMD-Prozessoren komplexe Teilaufgaben einer übergeordneten Aufgabe asynchron lösen.
- Eine wichtige Charakteristik der MIMD-Rechner ist die physikalische Organisation des Speichers. Es gibt zwei grundlegende Ansätze:
  - Shared Memory Architektur: Der Rechner besitzt einen zentralen Speicher für alle Prozessoren, in dem sich alle Programme und Daten befinden. Kommunikation und Synchronisation finden über Variablen im zentralen Speicher statt, die Prozessoren können aber noch zusätzlich über lokale Speicher verfügen.
  - Distributed Memory Architektur: Jeder Prozessor verfügt nur über lokalen Speicher. Die Kommunikation geschieht durch Austausch von Nachrichten über ein Verbindungsnetzwerk. Die Synchronisation geschieht implizit durch die Kommunikationsanweisungen.
  - Oft werden beide Organisationstypen in der Speicherhierarchie kombiniert, d.h. einen kleinen schnellen privaten Speicher für jeden Prozessor und einen im Gegensatz dazu langsamen größeren gemeinsamen Speicher.



a) Shared Memory MIMD-Rechner



b) Distributed Memory MIMD-Rechner



c) General Purpose MIMD-Rechner

Abbildung 3.2.1:

3.3 Amdahls Gesetz

Von entscheidender Bedeutung für parallele Berechnungen ist der Geschwindigkeitsgewinn bei der Parallelarbeit von p Prozessoren. Hierzu dient die Maßzahl (Speedup)

$$S := \frac{\text{Rechenzeit auf seriellem Rechner}}{\text{Rechenzeit auf parallelem Rechner}}.$$

Bemerkung 3.3.1 Um ein korrektes Ergebnis zu erhalten, muß man bei einem gegebenen Problem den *besten* seriellen Algorithmus mit dem *besten* parallelen Algorithmus vergleichen.

∘

Definition 3.3.2 (Speedup) *Der Speedup des parallelen Algorithmus A gegenüber dem korrespondierenden seriellen Algorithmus $\bar{A}$ ist definiert durch*

$$S_p(A) := \frac{T_1(\bar{A})}{T_p(A)} \geq 1.$$

T_p, $p \geq 1$, *ist die Anzahl der Zeiteinheitsschritte, die von einem parallelen Algorithmus bei p Prozessoren benötigt werden und T_1 die Zeiteinheitsschritte des entsprechenden seriellen Algorithmus.*

Definition 3.3.3 (Effizienz) *Die Effizienz ist definiert durch $E_p(A) := \frac{S_p(A)}{p} \leq 1$.*

Definition 3.3.4 (Kosten, Effektivität)

1. *Die Kosten eines gegebenen Algorithmus A werden gemessen durch $C_p(A) := p \cdot T_p(A)$.*
2. *Ein Maß für die Effektivität des Algorithmus A ist definiert durch $F_p(A) := \frac{S_p(A)}{C_p(A)}$.*

Bemerkung 3.3.5

1. Die Effizienz kann als Maß für die Auslastung eines Parallelrechners durch einen Algorithmus interpretiert werden.
2. Die Effektivität vergleicht parallele Algorithmen, die das gleiche Problem lösen.
3. Mit festem T_1 gilt dann: $F_p(A) = \frac{S_p(A)}{p \cdot T_p(A)} = \frac{E_p}{T_p} = \frac{E_p S_p}{T_1} \leq 1$ und daraus folgt $F_p \cdot T_1 = E_p \cdot S_p$.
4. Somit kann man einen gegebenen Algorithmus A als einen effektiven parallelen Algorithmus empfehlen, wenn A die Funktion $F_p(A)$ maximiert.

∘

Die eingeführten Größen sind ein Maß für die Beurteilung eines parallelen Algorithmus. Oft werden jedoch für einen Algorithmus A Zeit- und Prozessorschranken gesucht:

$N(A) :=$ Minimale Anzahl von Prozessoren, um ein maximales Speedup $S_N(A)$ zu erhalten.

$T_N(A) :=$ Anzahl der benötigten Zeiteinheiten bei $N(A)$ Prozessoren für den Algorithmus A.

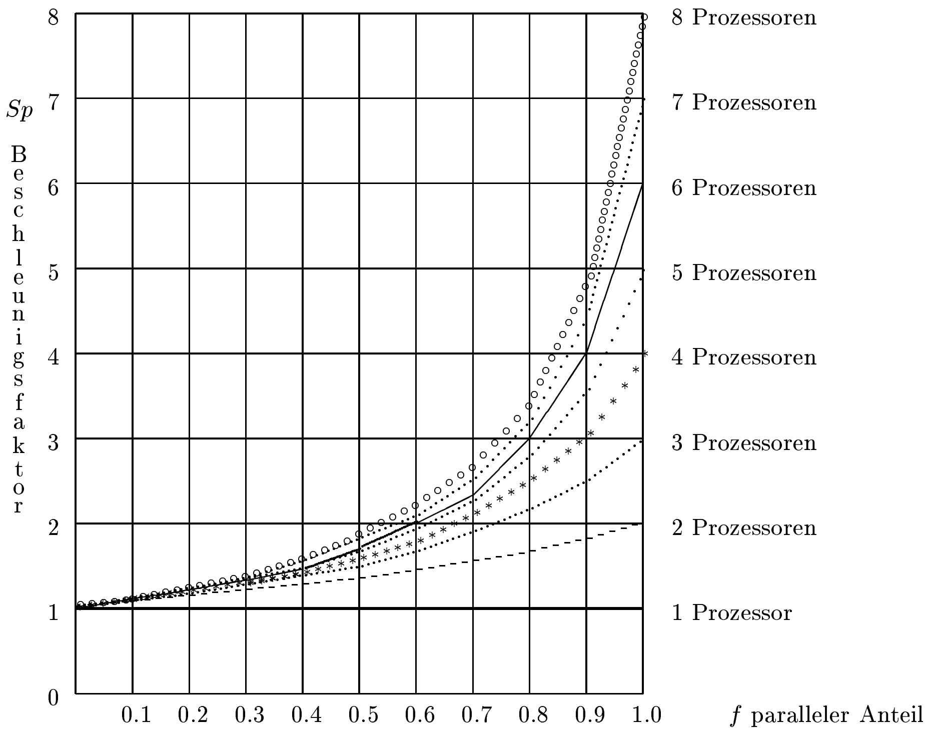
$T_p(A) :=$ Anzahl der erforderlichen Zeiteinheiten im eingeschränkten Prozessorfall ($p < N(A)$).

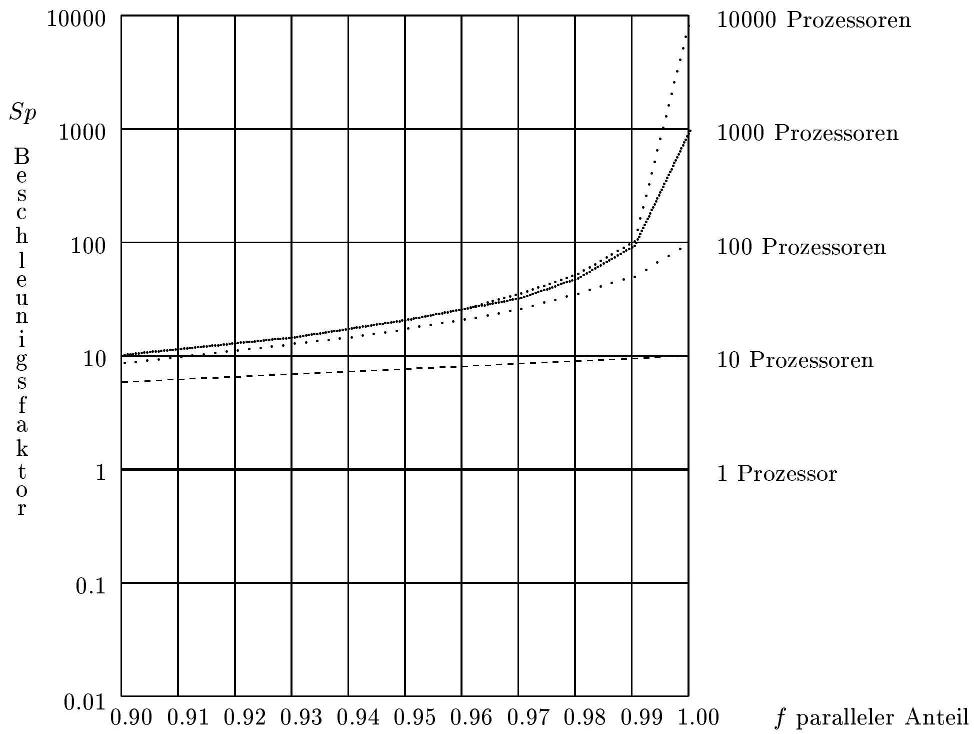
Kaum ein Algorithmus ist vollständig parallelisierbar. Mit **Amdahls Gesetz** läß sich ein maximal theoretisch erreichbarer Speedup unter Berücksichtigung der nicht parallelen Anteile eines Algorithmus bestimmen:

$$S_p = \frac{T_1}{(1-f)T_1 + f\frac{T_1}{p}} = \frac{1}{(1-f) + \frac{f}{p}} \tag{3.3.6}$$

T_1	Gesamtzeit des Algorithmus bei sequentieller Ausführung
f	parallel-/vektorisierbarer Anteil des Algorithmus
$1-f$	sequentieller Anteil des Algorithmus
p	Beschleunigungsfaktor des parallel-/vektorisierbaren Anteils bei Parallel- bzw. Vektorisierung
S_p	Beschleunigungsfaktor parallel/vektoriell zu sequentiell

Die folgenden Abbildungen demonstrieren, daß unter den obigen (unrealistischen) Annahmen bei großen Prozessorzahlen und kleinem sequentiellen Anteil nur Beschleunigungsfaktoren zu erzielen sind, die weit unterhalb von p liegen.
Beispielsweise kann man mit 1000 Prozessoren und nur 1% sequentiellem Anteil schon theoretisch keinen Beschleunigungsfaktor über 91 erwarten. Praktisch ist dieser Faktor jedoch kaum zu erreichen.





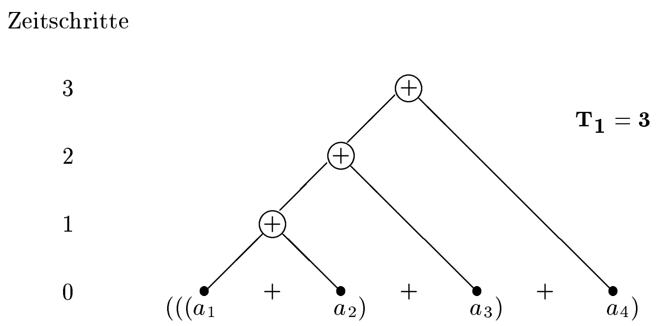
3.4 Parallele Algorithmen

Definition 3.4.1 (Arithmetische Ausdrücke) *Ein arithmetischer Ausdruck A ist ein wohldefinierter String, zusammengesetzt aus den vier arithmetischen Operationen,* $\{+, -, *, /\}$, *Links- und Rechtsklammern und aus Atomen, die Konstanten oder Variablen sind.*

Bemerkung 3.4.2 Ein arithmetischer Ausdruck A, der n verschiedene Atome enthält, wird oft mit $A < n >$ bezeichnet. ∘

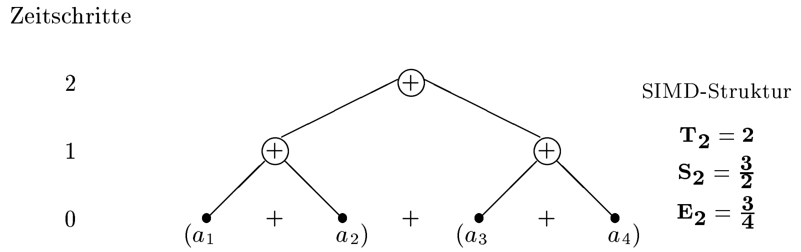
Definition 3.4.3 *Zwei arithmetische Ausdrücke A und $\tilde{A}$ heißen äquivalent, wenn A in $\tilde{A}$ überführt werden kann, durch eine endliche Anzahl von Anwendungen des Kommutativitäts-, Distributivitäts- und/oder Assoziativitätsgesetzes.*

Beispiel 3.4.4 Auflösung des einfache Ausdruck $A < 4 > := (((a_1 + a_2) + a_3) + a_4)$ anhand von Verzweigungsbäumen (parse-trees):



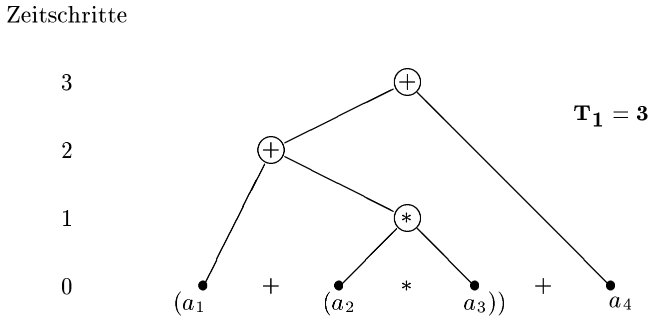
Dieser Baum hat die Höhe 3. Benutzt man jedoch das Assoziativitätsgesetz für die Addition, kann man die Klammern anders setzen und erhält den äquivalenten Ausdruck der Höhe 2: $\tilde{A} < 4 > := (a_1 + a_2) + (a_3 + a_4)$.

Baumhöhenreduktion durch Assoziativität:



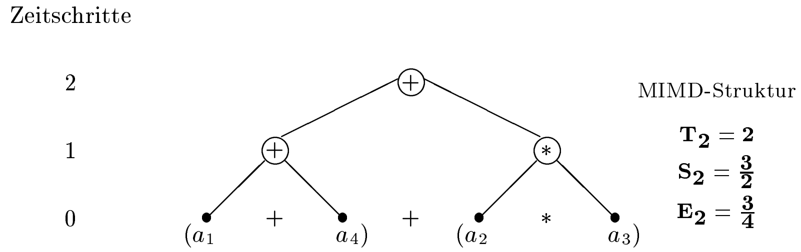
∘

Beispiel 3.4.5 Der Ausdruck $A < 4 > := a_1 + a_2 * a_3 + a_4$ hat die Baumhöhe 3 und durch Assoziativität ist kein 'niedrigerer' Baum zu erreichen:



Mit dem Kommutativitäts-Gesetz für die Addition erhält man den äquivalenten Ausdruck: $\tilde{A} < 4 > := (a_1 + a_4) + a_2 * a_3$ und einen Verzweigungsbaum der Höhe 2.

Baumhöhenreduktion durch Kommutativität:



∘

Bemerkung 3.4.6 Allgemein ergibt sich, daß die Berechnung eines Ausdrucks $A < n >$, bestehend aus n Operanden, mit einem Prozessor $(n-1)$ Zeiteinheiten erfordert. Wird eine beliebige Zahl von Prozessoren benutzt, dann kann ein Ausdruck $A < n >$ bzw. der transformierte Ausdruck $\tilde{A} < n >$ günstigenfalls in $log_2 n$ Zeiteinheiten ausgewertet werden. Damit ergeben diese Ausdrücke auf Parallelrechnern asymptotische Speedups von $O(\frac{n}{log_2 n})$ und Effizienzen von $O(\frac{1}{log_2 n})$.

∘

Lemma 3.4.7 *Für den arithmetischen Ausdruck $A < n >$ gilt: $T(A < n >) \geq \lceil log_2(n) \rceil$.*

Satz 3.4.8 *Es sei $A < n|d >$ ein arithmetischer Ausdruck mit der Tiefe d der genesteten Klammern. Durch Anwendung von Assoziativität und Kommutativität kann $A < n|d >$ so transformiert werden, daß*

$$T_p(A < n|d >) \leq \lceil \log_2 n \rceil + 2d + 1$$

mit $p \leq \lceil n/2 - d \rceil$ wird. [KuMu74]

3.5 Prinzipien zur Konstruktion paralleler Algorithmen

Die Art der Parallelität hat einen wesentlichen Einfluß auf die Struktur der Algorithmen. Die grundlegenden Prinzipien zur Konstruktion eines parallelen Algorithmus sind:

I Vektorisieren von seriellen Algorithmen (auf SIMD-Rechnern), d.h. Operationen konstruieren, die auf Vektoren arbeiten und daher parallel ausführbar sind.

Beispiel 3.5.1 Die Lösung eines $(n \times n)-$Gleichungssystems mit Dreiecksmatrix

$$Ax = b \Leftrightarrow \begin{bmatrix} a_{11} & & 0 \\ \vdots & \ddots & \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} \tag{3.5.2}$$

A nichtsingulär, erfordert auf einem seriellen Rechner n^2 Operationen. Mit

$$L^{(i)} := \begin{bmatrix} 1 & & & & 0 \\ & \ddots & & & \\ & & \frac{1}{a_{ii}} & & \\ & & \vdots & \ddots & \\ 0 & & \frac{-a_{ni}}{a_{ii}} & \dots & 1 \end{bmatrix}, \quad i = 1, 2, ..., n \tag{3.5.3}$$

gilt $x = A^{-1}b = L^{(n)}L^{(n-1)}...L^{(1)}b$.
Es müssen nun die Vektoren $y^{(1)}, ..., y^{(n+1)}$ berechnet werden, mit $y^{(1)} := b$ und

$$y^{(i+1)} := L^{(i)}y^{(i)} = \begin{bmatrix} y_1^{(i)} \\ \vdots \\ \frac{y_i^{(i)}}{a_{ii}} \\ y_{i+1}^{(i)} - \frac{a_{i+1,i}}{a_{ii}} y_i^{(i)} \\ \vdots \\ y_n^{(i)} - \frac{a_{ni}}{a_{ii}} y_i^{(i)} \end{bmatrix}, \quad i = 1, 2, ..., n. \tag{3.5.4}$$

○

Bemerkung 3.5.5 1. Stehen n Prozessoren zur Verfügung, kann die Lösung x in $O(n)$ Schritten (exakt: $3n-1$ Schritte) berechnet werden, das ergibt ein Speedup $S_n = \frac{n^2}{3n} = \frac{n}{3}$ und eine Effizienz $E_n = \frac{1}{3}$.

2. Stehen $k < n$ Prozessoren bereit, so benötigt man für n serielle $\lceil \frac{n}{k} \rceil$ parallele Schritte. Zur Berechnung der n Vektoren $y^{(2)}, ..., y^{(n+1)}$ benötigt man daher $O\left(\frac{n^2}{k}\right)$ Schritte, das ergibt ein Speedup $S_k = O(k)$ und eine Effizienz $E_k = O(1)$.

○

II Vektoriteration ersetzt einen direkten (also nichtiterativen) seriellen Algorithmus durch einen iterativen parallelen Algorithmus. Das Speedup des parallelen Algorithmus ergibt sich dann aus dem Verhältnis der seriellen Rechenschritte zur Anzahl der Iterationsschritte.

Beispiel 3.5.6 Dreieckszerlegung einer tridiagonalen Matrix A:

$$A = \begin{bmatrix} d_1 & f_1 & & 0 \\ e_2 & \ddots & \ddots & \\ & \ddots & \ddots & f_{n-1} \\ 0 & & e_n & d_n \end{bmatrix} = LU \tag{3.5.7}$$

$$\text{mit } L = \begin{bmatrix} 1 & & & 0 \\ m_2 & \ddots & & \\ & \ddots & \ddots & \\ 0 & & m_n & 1 \end{bmatrix} \quad \text{und} \quad U = \begin{bmatrix} u_1 & f_1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & f_{n-1} \\ 0 & & & u_n \end{bmatrix}.$$

Die u_i berechnen sich gemäß

$$\begin{aligned} u_1 &:= d_1 \\ u_i &:= d_i - e_i \tfrac{f_{i-1}}{u_{i-1}}, \qquad 2 \le i \le n. \end{aligned} \tag{3.5.8}$$

Die m_i können nun parallel bestimmt werden: $m_i := \frac{e_i}{u_{i-1}} \qquad i = 2, ..., n.$
Der serielle Vorgang (3.5.8) läßt sich durch Iteration parallelisieren:

$$\begin{aligned} u_i^{(0)} &:= d_i & i &= 1, ..., n \\ u_i^{(j)} &:= d_i - e_i \frac{f_{i-1}}{u_{i-1}^{(j-1)}} & i &= j+1, ..., n; \quad j > 0. \end{aligned} \tag{3.5.9}$$

∘

Bemerkung 3.5.10 Diese Parallelisierung kann sinnvoll sein, wenn der betrachtete Rechner Operationen mit Vektoren der Länge n wesentlich schneller als n skalare Operationen ausführen kann. Außerdem muß natürlich die Zahl der benötigten Iterationen (wesentlich) kleiner als n sein.

∘

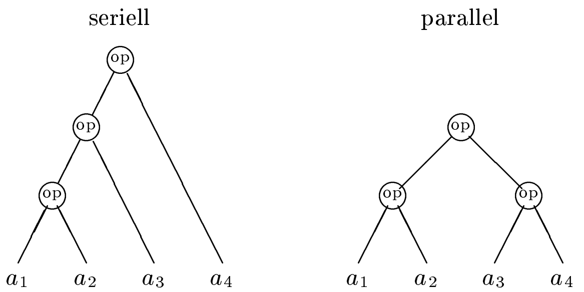
III Rekursives Doppeln ist das allgemeinste Prinzip der Baumhöhenreduktion durch Transformation in äquivalente Ausdrücke. [Kogge74]
Mit der Menge $S = \{a_1, ..., a_N | N = 2^k, k \in \mathbb{N}\} \subset \mathbb{R}$ und einer beliebigen assoziativen Operation $op \in \{+, -, \cdot, \max, ...,\}$ wird der Ausdruck

$$A < N >= a_1 op\ a_2\ op\ \cdots\ op\ a_N$$

gebildet. Die Assoziativität liefert den parallel zu berechnenden äquivalenten Ausdruck

$$\tilde{A} < N >= (a_1\ op\ a_2)\ op\ (a_3\ op\ a_4)\ \cdots\ op\ (a_{N-1}\ op\ a_N)$$

Für $N = 4$ ergeben sich folgende Bäume:



Bemerkung 3.5.11 1. Allgemein erfordert das rekursive Doppeln bei $N = 2^k$ Elementen $log_2 N$ (parallele) Schritte auf einem echten SIMD-Rechner. Zur seriellen Anwendung sind $(N-1)$ Schritte nötig.

2. Ein Algorithmus, der durch rekursives Doppeln entwickelt wird, gehört zu den sogenannten **log-sum-Algorithmen**.

○

Beispiel 3.5.12 (allgemeine Fibonacci-Folge) Die Auswertung dieser zweistelligen rekurrenten Relation (vgl. Kapitel 3.6) ist im wesentlichen ein serieller Prozeß.

$$\begin{aligned} f_0 &:= 1, \qquad f_1 := b_1, \qquad b_1 \in I\!R \\ f_i &:= b_i f_{i-1} + a_i f_{i-2}, \quad a_i, b_i \in I\!R, \quad 2 \le i \le N = 2^k. \end{aligned} \tag{3.5.13}$$

Durch diese Transformationen $F_j := \begin{bmatrix} f_j \\ f_{j-1} \end{bmatrix}$, $A_i := \begin{bmatrix} b_i & a_i \\ 1 & 0 \end{bmatrix}$, $j = 1, ..., N,\ i = 2, ..., N,$ erhält man für (3.5.13) eine äquivalente parallele Darstellung:

$$\begin{aligned} F_i &= A_i F_{i-1} = A_i A_{i-1} ... A_2 F_1 \\ F_N &= A_N A_{N-1} ... A_2 F_1. \end{aligned} \tag{3.5.14}$$

Da die Matrix-Vektor-Multiplikation assoziativ ist, kann F_N in $O(log_2 N)$ Schritten berechnet werden gegenüber $O(N)$ Schritten bei serieller Rechnung.

○

Bemerkung 3.5.15 Das rekursive Doppeln beruht auf dem Aufspalten eines Ausdruckes A_0 in zwei einfachere, kleinere Ausdrücke $A_1,\ A_2$, die einzeln von je einem Prozessor gleichzeitig berechnet werden können.

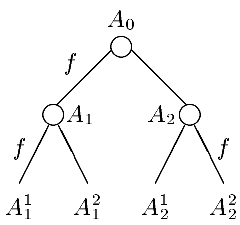
Dabei muß folgendes erfüllt sein:

a) Es muß eine Funktion f existieren, so daß $A_0 = f(A_1, A_2)$ ist.

b) Die Berechnungen von A_1 und A_2 sind unabhängig voneinander, beide sind in ihrer Komplexität gleich.

c) A_1 und A_2 erfordern die gleiche Operationsfolge zu ihrer Berechnung.

Durch weitere Aufspaltung von A_1 und A_2 gemäß a) bis c) erhält man schließlich durch rekursives Doppeln den Ausdruck A_0:



○

3.6 Rekurrente Relationen

In der Numerischen Mathematik ergibt sich die Lösung eines Problems oft als eine Folge von Elementen $x_1, x_2, ..., x_n, \ \ x_i \in I\!R$, mit gegebenen Anfangsbedingungen wobei jedes $x_i, \ i = 1, 2, ..., n$, von m anderen Elementen $x_j, \ j = 1, ..., m$, abhängt. Die Gleichungen, die diese Abhängigkeit beschreiben, werden **rekurrente Gleichungen** genannt. Zusammen mit bestimmten Anfangswerten für verschiedene x_k bilden sie ein **rekurrentes Problem**. [SchBr79]

Definition 3.6.1 (lineare rekurrente Relation) *Ein lineares rekurrentes System $R < n, m >$ der Ordnung m für $n \geq 2$ Gleichungen ist definiert für $1 \leq m \leq n-1$ durch*

$$R < n, m >: \ x_k := \begin{cases} 0 & \text{für } \ k \leq 0 \\ c_k + \sum\limits_{j=k-m}^{k-1} a_{kj} x_j, & \text{für } \ i \leq k \leq n. \end{cases} \tag{3.6.2}$$

Ist m=n-1, wird dieses System ein gewöhnliches lineares, rekurrentes System $R < n >$ genannt.

Bemerkung 3.6.3 In Matrix-Vektor-Notation erhält man für (3.6.2)

$$x = c + Ax \tag{3.6.4}$$

mit $x = [x_1, ..., x_n]^T, \ c = [c_1, ..., c_n]^T$ und $A = [a_{ik}] \ (i, k = 1, 2, ..., n)$, eine strikte Dreiecksmatrix mit $a_{ik} = 0$ für $i \leq k$ oder $i - k > m$. A ist eine Bandmatrix für $m < n - 1$.

∘

Beispiel 3.6.5 1. Für $n = 4$ und $m = 2$ erhält man das folgende lineare, rekurrente System $R < 4, 2 >$:

$$\begin{aligned} x_1 &= c_1 \\ x_2 &= c_2 + a_{21}x_1 \\ x_3 &= c_3 + a_{31}x_1 + a_{32}x_2 \\ x_4 &= c_4 \qquad\qquad\ + a_{42}x_2 + a_{43}x_3 \end{aligned}$$

2. Als eine lineare rekurrente Relation 2. Ordnung, die ein Vektorresultat erzeugt, kann die Fibonacci-Folge betrachtet werden (vgl. Beispiel 3.5.12).

$$\begin{aligned} f_0 &:= 1, \\ f_1 &:= 1, \\ f_k &:= f_{k-1} + f_{k-2}, \qquad k = 2, ...n. \end{aligned} \tag{3.6.6}$$

∘

Bemerkung 3.6.7 Das rekursive Doppeln (Seite 26) ist eine Methode, um eine rekurrente Relation auf einem SIMD-Rechner parallel berechnen zu können.

∘

Bemerkung 3.6.8 (Column-Sweep-Algorithmus) Auch der sogenannte Column-Sweep-Algorithmus löst ein $R < n >$-System schnell und effizient, wobei mit $O(n)$ Prozessoren das System in $O(n)$-Zeiteinheitsschritten gelöst wird ($O(n^2)$ im seriellen Fall):

Schritt 1: x_1 ist bekannt. Dann werden die Ausdrücke

$$c_i^{(1)} := a_{i1} \cdot x_1 + c_i, \qquad i = 2, ..., n$$

parallel berechnet. $x_2 = c_2^{(1)}$ ist dann bestimmt, und ein $R < n-1 >$-System zu lösen.

Schritt 2: Mit dem berechneten x_2 können die Ausdrücke

$$c_i^{(2)} := a_{i2} \cdot x_2 + c_i^{(1)}, \quad i = 3, ..., n$$

parallel ausgewertet werden. x_3 ist somit bestimmt.

Schritt k: x_k und $c_i^{(k-1)} := a_{i,k-1} \cdot x_{k-1} + c_i^{(k-2)}$ sind bekannt. Damit werden parallel die Ausdrücke

$$c_i^{(k)} := a_{ik} \cdot x_k + c_i^{(k-1)}, \quad i = k+1, ..., n$$

berechnet und man erhält x_{k+1}. In dieser Vorschrift fährt man fort bis

Schritt n-1: Berechnung von x_n. Damit ist der Lösungsvektor $x = [x_1, ..., x_n]$ bestimmt.

Bei Anwendung des Column-Sweep-Algorithmus auf ein $R < n, m >$-System (Definition 3.6.9) benötigt man zur parallelen Auswertung m Prozessoren. Damit erhält man ein Speedup von $S_m = O(m)$. In der Praxis sind $R < n, m >$-Systeme mit $m << n$ von Interesse, oft ist m sogar ≤ 2. Bei einer großen Anzahl von verfügbaren Prozessoren, beispielsweise bei massiv parallelen Rechnern, ist dies unvorteilhaft.

∘

Definition 3.6.9 (allgemeine rekurrente Relation)

1. *Ein allgemeines rekurrentes System $R < n, m >$ m-ter Ordnung ist definiert durch*

$$R < n, m >: x_k := H[\bar{a}_k; x_{k-1}, x_{k-2}, ..., x_{k-m}], \qquad 1 \leq k \leq n \tag{3.6.10}$$

   mit $m \geq 1$ Anfangswerten $x_{-m+1}, ..., x_0$.

2. *$H : \mathbb{R}^p \times \mathbb{R}^m \longrightarrow \mathbb{R}$ heißt Rekursionsfunktion. Dabei ist $\bar{a}_k \in \mathbb{R}^p$ ein Parametervektor, der unabhängig von den x_i ist.*

Bemerkung 3.6.11 Diese Relationen sind prädestiniert, um SIMD-Rechner zu nutzen. Jeder Prozessor P führt dieselbe Instruktion, die zur Berechnung von H benötigt wird, mit verschiedenen Parametern und Daten aus.

∘

Beispiel 3.6.12 1. Ein einfaches Beispiel eines linearen rekurrenten Systems 1. Ordnung ist

$$\begin{aligned} x_1 &= c_1 && \text{Anfangswert} \\ x_k &= a_k x_{k-1} + c_k, && 2 \leq k \leq n \\ &= H[\bar{a}_k; x_{k-1}], \end{aligned} \tag{3.6.13}$$

$\bar{a}_k = [a_k, c_k]$ und die Funktion H ist definiert durch eine Multiplikation und eine Addition.

Bemerkung 3.6.14 (a) Oft wird (3.6.13) in der Form $x_k := a_k(1)x_{k-1}+a_k(2)$ geschrieben.

(b) Die Länge von $\bar{a}_k$ ist problemabhängig, jedoch konstant innerhalb eines Problems für jedes k.

∘

2. Eine nichtlineares rekurrentes System 1. Ordnung ist $x_k := H[\bar{a}_k; x_{k-1}] = \frac{a_k(1)+a_k(2)x_{k-1}}{a_k(3)+a_k(4)x_{k-1}}$ mit $\bar{a}_k = [a_k(1), a_k(2), a_k(3), a_k(4)] \in I\!R^4$.

∘

Bemerkung 3.6.15 Viele rekurrente Relationen sind nicht assoziativ und können deshalb nicht direkt mit dem log-sum-Algorithmus gelöst werden.

∘

Definition 3.6.16 (Begleitfunktion) *Eine Funktion G heißt Begleitfunktion (companion function) zur allgemeinen Rekursionsfunktion H der Ordnung 1 genau dann, wenn für alle $x \in I\!R$ und für alle Parametervektoren $\bar{a}, \bar{b} \in I\!R^p$ gilt:*

$$H[\bar{a}; H[\bar{b}; x]] = H[G(\bar{a}, \bar{b}); x] \tag{3.6.17}$$

und $G : I\!R^p \times I\!R^p \to I\!R^p$. H wird dann semi-assoziativ genannt.

Beispiel 3.6.18 Für die lineare rekurrente Relation 1. Ordnung (3.6.13) gilt

$$\begin{aligned} x_{k+1} &= H[\bar{a}_{k+1}; x_k] \\ x_{k+2} &= H[\bar{a}_{k+2}; x_{k+1}] = H[\bar{a}_{k+2}; H[\bar{a}_{k+1}; x_k]] \\ &= (a_{k+2}a_{k+1})x_k + (a_{k+2}c_{k+1} + c_{k+2}). \\ x_{k+2} &= H[\bar{a}(k+2, k+1); x_k] \end{aligned} \tag{3.6.19}$$

mit $\bar{a}(k+2, k+1) = [a_{k+2}a_{k+1}, a_{k+2}c_{k+1} + c_{k+2}] \in I\!R^2$. Das heißt, man hat einen neuen Parametervektor $\bar{a}(k+2, k+1)$ erhalten, der unabhängig von irgendeinem x_j ist. x_{k+1} und x_{k+2} können parallel berechnet werden.

Man erhält also $x_{k+2} = H[\bar{a}_{k+2}; H[\bar{a}_{k+1}; x_k]] = H[G(\bar{a}_{k+2}, \bar{a}_{k+1}); x_k]$

mit der Begleitfunktion $G(\bar{a}_{k+2}, \bar{a}_{k+1}) = [a_{k+2}a_{k+1}, a_{k+2}c_{k+1} + c_{k+2}] = \bar{a}(k+2, k+1)$.

∘

Satz 3.6.20 *Jede Begleitfunktion G ist semi-assoziativ bezüglich ihrer Rekursionsfunktion H, das heißt für alle $x \in I\!R$ und alle $\bar{a}, \bar{b}, \bar{c} \in I\!R^p$ gilt: $H[G(\bar{a}, G(\bar{b}, \bar{c})); x] = H[G(G(\bar{a}, \bar{b}), \bar{c}); x]$.* [Schen95]

Satz 3.6.21 *Für jedes rekurrente Problem 1. Ordnung der Form $x_k := H[\bar{a}_k; x_{k-1}]$, $1 \le k \le n$ mit gegebenem x_0 und der Begleitfunktion G gilt: $x_k = H[\bar{a}(k, j+1); x_j]$, $0 \le j < k \le n$.*

$$\bar{a}(k,l) := \begin{cases} \bar{a}_k & \text{für } \; k = l \ge 1 \\ G(\bar{a}_k, \bar{a}(k-1, l)) & \text{für } \; k > l \ge 1 \end{cases} \quad \textit{und}$$

$\bar{a}(k,l) = G(\bar{a}_k, G(\bar{a}_{k-1}, ..., G(\bar{a}_{l+1}, \bar{a}_l)...)$ *für* $k > l \ge 1$. [Schen95]

Bemerkung 3.6.22 Aus Satz (3.6.21) folgt insbesondere $x_k := H[\bar{a}(k,1); x_0]$.

○

Bemerkung 3.6.23 1. Die Parallelisierung einer rekurrenten Relation kann man also auf die Konstruktion einer Begleitfunktion G reduzieren.

2. Die Existenz von G und die Eigenschaft der Semi-Assoziativität erlauben nun die Konstruktion eines parallelen Algorithmus mittels des log-sum-Algorithmus für das ursprüngliche Problem.

○

3.7 Das Divide&Conquer-Prinzip

Einen allgemeinen Ansatz zur Entwicklung von Lösungsalgorithmen für verschiedene Probleme stellt das Divide&Conquer($\mathcal{D}\&\mathcal{C}$)-Prinzip dar. Es ergeben sich folgende Schritte innerhalb eines $\mathcal{D}\&\mathcal{C}$-Algorithus $\mathcal{D}\&\mathcal{C}(P, L)$:

1. Verfeinerungstest: Feststellung, ob zur Berechnung einer Lösung L das Problem P weiter zerlegt werden kann.
2. Divide-Schritt:
   WENN P teilbar *DANN* Teile P in $k \in \mathbb{N}$ kleinere, unabhängige Teilprobleme $P_1 \cdots P_k$.
   SONST bestimme direkt eine Lösung L; $\mathcal{D}\&\mathcal{C}(P, L)$ terminiert.
3. Conquer-Schritt:
   Anwenden des Algorithmus auf die Teilprobleme $P_1 \cdots P_k$, es entstehen die Teilaufgaben $\mathcal{D}\&\mathcal{C}(P_1, L_1) \cdots \mathcal{D}\&\mathcal{C}(P_k, L_k)$
4. Merge-Schritt:
   Setze die erhaltenen k Teillösungen $L_1 \cdots L_k$ zusammen und gebe die Gesamtlösung L aus.

Bemerkung 3.7.1 Setzt man (häufig stillschweigend) voraus, daß alle bei einer Zerlegungen entstandenen Teilprobleme gleichartig sind, erhält man einen rekursiven Algorithmus.

○

Algorithmus 3.7.2 Allgemeiner Divide&Conquer-Algorithmus

```
PROCEDURE D&C(Eingabe: Problem; Ausgabe: Lösung)

IF Problem weiter zerteilbar
   THEN Zerteile das Problem auf in k Teilprobleme
           D&C(Teilproblem 1, Teillösung 1)
           ⋮
           D&C(Teilproblem k, Teillösung k)
           Konbiniere(Teilösung 1, ..., Teillösung k)
   ELSE Bestimme direkte Lösung (Problem, Lösung)
```

Der Ablauf des $\mathcal{D}\&\mathcal{C}$-Algorithmus veranschaulicht die folgende Abbildung [Gottw96]:

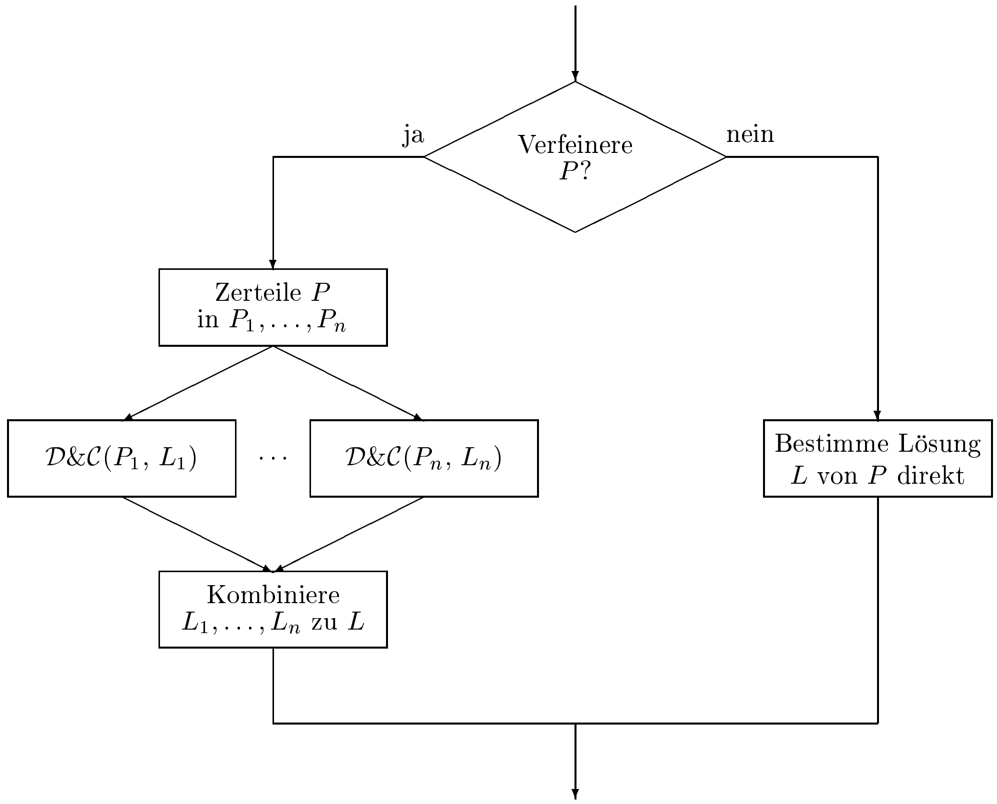


Abbildung 3.7.3: Aufbauschema von $\mathcal{D}\&\mathcal{C}(P, L)$

Bemerkung 3.7.4 Die Interaktionen zwischen den einzelnen Prozeduren sind recht einfach, da es durch die Reduktion eine feste Hierarchie gibt.

$\circ$

4 Singulärwertzerlegung mit parallelen Jacobi-Verfahren

Das Jacobi-Verfahren ist eines der ältesten numerischen Verfahren zur Lösung des vollständigen Eigenwertproblems bei symmetrischen Matrizen. Es war bis zur Einführung des QR-Algorithmus in seiner Anwendung weit verbreitet. Wegen der Einfachheit und Zuverlässigkeit ist das Jacobi-Verfahren nach wie vor von grundsätzlicher Bedeutung, insbesondere wenn auch alle Eigenvektoren gesucht sind. Das Jacobi-Verfahren bildet auch eine Grundlage für die parallele Berechnung der Singulärwerte.

4.1 Das klassische Jacobi-Verfahren

Das klassische Jacobi-Verfahren transformiert eine symmetrische Matrix mit einer unendlichen Folge von orthogonalen Ähnlichkeitstransformationen in eine Diagonalmatrix. Die Idee besteht also darin, durch ebene Drehungen die Nicht-Diagonalelemente systematisch zum Verschwinden zu bringen:

$$N(A) := \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij}^2 \to 0. \tag{4.1.1}$$

Bemerkung 4.1.2 Die Anzahl der dafür benötigten Drehungen ist unendlich, da die Eigenwerte als Nullstellen des charakteristischen Polynoms von A nicht durch einen endlichen Prozeß betimmt werden können. Bei der praktischen Durchführung des Jacobi-Verfahrens wird die Iteration abgebrochen, sobald der Wert von $N(A)$ unter einer vorgegebenen Schranke liegt.

$\circ$

In der klassischen Form des Jacobi-Algorithmus wird das momentan betragsgrößte Nicht-Diagonalelement durch ebene Drehung zum Verschwinden gebracht. Bei der nachfolgenden Drehung wird allerdings das an dieser Stelle erzeugte Nullelement wieder zerstört.

Satz 4.1.3 *Sie* $A = [a_{ij}] \in I\!R^{n \times n}$ *symmetrisch und* (p, q) *ein Indexpaar mit* $1 \leq p < q \leq n$ *und* $a_{pq} \neq 0$. *Der Drehwinkel* $\varphi \in [-\frac{\pi}{4}, \frac{\pi}{4}]$ *sei durch*

$$\begin{array}{rcll} cot\ 2\varphi & = & \frac{a_{qq}-a_{pp}}{2a_{pq}} & \textit{falls}\ \ a_{pp} \neq a_{qq}, \\ \varphi & = & \frac{\pi}{4} & \textit{falls}\ \ a_{pp} = a_{qq} \end{array}$$

gegeben. Und mit den Givens-Rotationen (Seite 4) R_{pq} *sei*

$$R_{pq}^T A R_{pq} =: B = [b_{ij}],$$

d.h.
$$\begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix}^T \begin{bmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{bmatrix} \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix} = \begin{bmatrix} b_{pp} & b_{pq} \\ b_{qp} & b_{qq} \end{bmatrix}.$$

Dann gilt:

1. $b_{pp} = a_{pp} - \tan\varphi\ a_{qp}$,
   $b_{qq} = a_{qq} + \tan\varphi\ a_{qp}$,
   $b_{pq} = b_{qp} = 0$.
2. $N(B) = N(A) - 2a_{pq}^2 < N(A)$.
3. *Ist* $|a_{pq}| = \max_{1 \leq i < j \leq n} |a_{ij}|$, *so ist* $N(B) \leq \left(1 - \frac{2}{n^2-n}\right) N(A)$. [Werner92]

Bemerkung 4.1.4 Nach k Schritten gilt also die Abschätzung $N(A^{(k)}) \leq \left(1 - \frac{2}{n^2-n}\right)^k N(A^{(0)})$. Daraus folgt die lineare Konvergenz des Jacobi-Verfahrens. Für spezielle Matrixtypen und ab einer bestimmten Anzahl von Iterationen konvergiert das Verfahren quadratisch. [GoLo84]

∘

4.2 Das zyklische Jacobi-Verfahren

Im zyklischen Jacobi-Verfahren ist die Reihenfolge der Rotationsindexpaare, zur systematischen Elimination der Nebendiagonalelemente, so festgelegt, daß die Nebendiagonalelemente der unteren Hälfte kolonnenweise pro Zyklus genau einmal Null werden

$$(1,2),(1,3),...,(1,n),(2,3),(2,4),...,(2,n),(3,4),...,(n-1,n). \tag{4.2.1}$$

Bemerkung 4.2.2

1. Eine $(p,q)-$Rotation wird nicht ausgeführt, falls das aktuelle $a_{pq} = 0$.
2. Die Werte für $\cos\varphi$ und $\sin\varphi$ werden wie im klassischen Jacobi-Verfahren bestimmt.

∘

Satz 4.2.3 *Die Matrixfolge A_k, gebildet nach dem speziellen zyklischen Jacobi-Verfahren (4.2.1) und mit dem Drehwinkel φ (Satz 4.1.3), konvergiert gegen eine Diagonalmatrix.* [FoHe60]

Bemerkung 4.2.4 Die quadratische Konvergenz des zyklischen Jacobi-Verfahrens setzt erfahrungsgemäß nach wenigen Zyklen ein, so daß in der Regel sechs bis acht Zyklen notwendig sind, um die Matrix A mit hinreichender Genauigkeit zu diagonalisieren.

∘

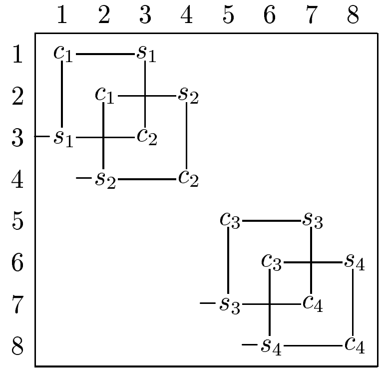
4.3 Das parallele Jacobi-Verfahren

Da die Jacobi-Methode bei jeder Multiplikation mit Rotationsmatrizen $R(i,j)$ nur je zwei Spalten und Zeilen verändert, läßt sich ein paralleler Algorithmus entwickeln.

Bemerkung 4.3.1 Für disjunkte Indexpaare $(i,j) \neq (k,l)$ können die Werte a_{ij} und a_{kl} gleichzeitig eliminiert werden, d.h. bei einer symmetrischen Matrix $A \in \mathbb{R}^{n\times n}$ können $\frac{n}{2}$ Nebendiagonalelemente gleichzeitig parallel bearbeitet werden, siehe Beispiel 4.3.2.

∘

Beispiel 4.3.2 Sei $A_k \in \mathbb{R}^{8\times 8}$ dann kann man die orthogonale Matrix P_k wie folgt in vier unabhängige Rotationen unterteilen.



Berechnet man $A_{k+1} = P_k^T A_k P_k \in \mathbb{R}^{8\times 8}$, so ist die Rotationsmatrix P_k die direkte Summe der vier Rotationen

$$
R_k(i,j) = \begin{array}{c}
\begin{bmatrix}
1 & & & & & & 0 \\
& \ddots & & & & & \\
& & \cos\varphi & & \sin\varphi & & \\
& & & \ddots & & & \\
& & -\sin\varphi & & \cos\varphi & & \\
& & & & & \ddots & \\
0 & & & & & & 1
\end{bmatrix} \\
\begin{array}{ccccccc}
 & & \uparrow & & \uparrow & & \\
 & & i & & j & &
\end{array}
\end{array}
$$

$$
P_k = R_k(1,3) \oplus R_k(2,4) \oplus R_k(5,7) \oplus R_k(6,8). \tag{4.3.3}
$$

Die Elemente der Matrix A_k werden in den Positionen (1,3), (2,4), (5,7) und (6,8) Null.

∘

Bemerkung 4.3.4 Besteht ein Schritt des Jacobi-Verfahrens aus einer Zusammenfassung ähnlicher Transformationen, so wird jedes der $\frac{n(n-1)}{2}$ Nebendiagonalelemente einmal eliminiert.
Bei einer Matrix der Ordnung 8 besteht der erste Eliminationsschritt aus acht Matrizen, die jeweils vier Elemente gleichzeitig Null setzen, jede nachfolgende Transformation P_k mit $k > 8$ entspricht der Transformation P_j mit $j = 1 + (k-1) \bmod 8$.

∘

Algorithmus 4.3.5 (paralleler Jacobi-Algorithmus)

Schritt 1: Berechnung der Rotationsmatrizen $R_k(i,j)$ (vgl. (4.3.3)):

1.a) FÜR $k = 1, 2, 3, ..., n-1$

Parallele Elimination der Elemente in Positionen (i,j) für

$$\begin{cases} i = 1, 2, 3, ..., \lceil \frac{n-k}{2} \rceil \\ j = (n-k+2) - i \end{cases}$$

und ab $k > 2$ auch für

$$\begin{cases} i = (n-k+2), (n-k+3), ..., n - \lfloor \frac{k}{2} \rfloor \\ j = (2n-k+2) - i \end{cases}$$

1.b) FÜR k=n:

Parallele Elimination der Elemente in Positionen (i,j) für

$$\begin{cases} i = 2, 3, ..., \lceil \frac{n}{2} \rceil \\ j = (n+2) - i \end{cases}$$

Schritt 2: Parallele Berechnung der P_k als direkte Summe der $R_k(i,j)$:

$$P_k = R_k(i_1, j_1) \oplus \cdots \oplus R_k(i_m, j_m), \;\; m \le \lfloor \tfrac{n}{2} \rfloor.$$

Schritt 3: Parallele Berechnung der Matrix $A_k = P_{k-1}^T A_{k-1} P_{k-1}$

Schritt 4: Konvergenztest:

4.a) Die Matrix A_k wird aufgespalten in $A_k = D_k + E_k + E_k^T$.
D_k ist eine Diagonalmatrix und E_k eine obere Dreiecksmatrix.

4.b) Bilde die Frobeniusnormen $\|E_k\|_F$ und $\|D_k\|_F$.

4.c) WENN $\frac{\|E_k\|_F}{\|D_k\|_F} <$ Toleranzschranke, *DANN* wird das Verfahren abgebrochen, *SONST* gehe zu Schritt 1.

- Eine effektive Eliminationsmethode für eine vollbesetzte symmetrische Matrix der Ordnung n besteht also aus $(2r-1)$ gleichzeitigen Transformationen, wobei $r = \lfloor \frac{n+1}{2} \rfloor$ ist. Jede dieser Transformationen eliminiert $\lfloor \frac{n}{2} \rfloor$ verschiedene Nebendiagonalelemente.

- Während der Elimination eines Elementes (i,j) in Schritt 1 des Algorithmus 4.3.5 werden Nebendiagonalelemente in den Zeilen i und j und den Spalten i und j nach Satz 4.1.3 verändert.

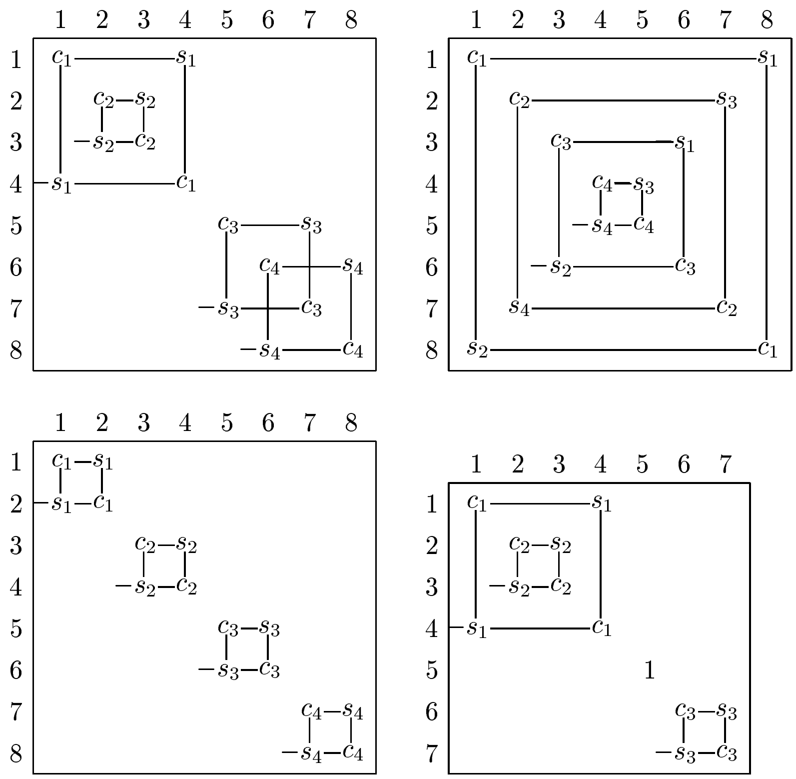
- Um Speicherplatz zu sparen nutzt man die Symmetrie der Matrix aus und betrachtet nur die Elemente oberhalb der Diagonalen.

- Schritt 2 und Schritt 3 können z.B. mit Hilfe des rekursiven Doppelns (vgl. Kapitel 3.5) parallel berechnet werden.

- In Schritt 4 wird durch das Verhältnis der sich verkleinernden Nebendiagonalelemente die Konvergenz die Algorithmus bestimmt. Eine sinnvolle Toleranzschranke liegt bei 10^{-16}.

- Bei entsprechender Aufteilung der Prozessoren kann eine weitere Parallelität im Algorithmus für die Schritte 1, 2 und 3 erzeugt werden: z.B. während in Schritt 1 die Rotationsmatrizen $R_k(i_l, j_l)_{l=1,2,\ldots,m}$ für $k = 2$ berechnet werden, kann man gleichzeitig $P_{k=1}$ in Schritt 2 berechnen.

Bemerkung 4.3.6 Es sind unterschiedliche Anordnungen der orthogonalen Transformationen $R_k(i,j)$ in einer Rotationsmatrix P_k möglich, z.B. für $n = 8$ oder $n = 7$:



○

4.4 Das einseitige Jacobi-Verfahren zur Berechnung der SVD

Das Ziel der einseitigen Transformation ist die Bestimmung einer orthogonalen Matrix V als Produkt von Rotationsmatrizen, so daß gilt:

$$AV = AV_1V_2V_3 \cdots = B = [b_1, b_2, ..., b_n] \tag{4.4.1}$$

Jede Matrix V_k ist die direkte Summe aus $\lfloor \frac{n}{2} \rfloor$ bzw. $\lfloor \frac{n-1}{2} \rfloor$ Rotationen $R_k(i,j)$, d.h. V_k wird wie P_k in (4.3.3) gebildet.
Die $b_i,\ i = 1, ..., n$, sind die orthogonalen Spaltenvektoren der Matrix B, für die gelten sollen

$$b_i^T b_j = \sigma_i^2 \delta_{ij} = \begin{cases} \sigma_i^2 & \text{falls} \quad i = j \\ 0 & \text{falls} \quad i \neq j \end{cases} . \tag{4.4.2}$$

Die σ_i bilden eine Diagonalmatrix, so daß gilt

$$B = U\Sigma \qquad \text{mit} \quad U^TU = I_j. \tag{4.4.3}$$

Bemerkung 4.4.4 Ist $j < n$ so ist der Rang der Matrix A gleich j.

∘

Bemerkung 4.4.5 Die einseitige Transformation ist ein iterativer Prozeß, da die bereits berechnete Orthogonalität der Spalten in darauffolgenden Rotationen wieder zerstört werden kann.

∘

Lemma 4.4.6 *Läßt sich die Matrix V berechnen, so bilden U, V und Σ eine Singulärwertzerlegung von $A = [a_{ij}]_{i,j=1,\cdots,n} = [a_1, a_2, \cdots, a_n]$ (vgl. Satz 2.2):*

$$A = U\Sigma V^T. \tag{4.4.7}$$

Die Matrix V wird mit Hilfe von Givens-Rotationen $R(i,j)$ bestimmt:

$$[a_i, a_j] \begin{bmatrix} c & -s \\ s & c \end{bmatrix} = [\tilde{a}_i, \tilde{a}_j] \qquad i < j, \tag{4.4.8}$$

$$\text{so daß gilt} \quad \tilde{a}_i^T \tilde{a}_j = 0 \quad \text{und} \quad \|\tilde{a}_i\| > \|\tilde{a}_j\|. \tag{4.4.9}$$

$$\text{Es gilt:} \quad \begin{array}{ll} c = \sqrt{\frac{\beta+\varphi}{2\varphi}} \ \text{und} \ s = \frac{\alpha}{2c\varphi} & \text{wenn} \ \beta > 0, \\ s = \sqrt{\frac{\varphi-\beta}{2\varphi}} \ \text{und} \ c = \frac{\alpha}{2s\varphi} & \text{wenn} \ \beta < 0. \end{array} \tag{4.4.10}$$

Mit $\alpha = 2a_i^T a_j,\ \beta = \|a_i\|^2 - \|a_j\|^2$ und $\varphi = \sqrt{\alpha^2 - \beta^2}$.

- In (4.4.9) wird gefordert, daß die Norm der Spalten der Matrix B von links nach rechts kleiner wird. Daraus ergibt sich, daß die Singulärwerte σ_i monoton fallend angeordnet sind, d.h. $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n$.
- Für die Orthogonalisierung der Spalten der Matrix A wird die Anzahl der benötigten Rotationen möglichst klein gehalten.

- In Anlehnung an Kapitel 4.2 lassen sich die Spalten von A in der gleichen zyklischen Form berechnen.

Wendet man das Eliminationsschema für den parallelen Jacobi-Algorithmus 4.3.5 auf das einseitige Jacobi-Verfahren an, so erhält man einen parallelen Algorithmus zur Berechnung der Singulärwerte einer Matrix $A = [a_{ij}]_{i,j=1,\cdots,n} = [a_1, a_2, \cdots, a_n]$:

Algorithmus 4.4.11 (paralleler einseitiger Jacobi-Algorithmus)

<u>*Schritt 1:*</u> Berechnung der Rotationsmatrizen $R_k(i,j)$:

1.a) istop = 0

1.b) WENN für (i,j) gilt: $\frac{(a_i^T a_j)^2}{(a_i^T a_i)(a_j^T a_j)} >$ Toleranzschranke, $DANN$

FÜR k=1,2,3,...,n-1

Paralleles Orthogonalisieren der Spalten a_i und a_j für

$$\begin{cases} i = 1,2,3,...,\lceil\frac{n-k}{2}\rceil \\ j = (n-k+2)-i \end{cases}$$

und ab $k > 2$ auch für

$$\begin{cases} i = (n-k+2),(n-k+3),...,n-\lfloor\frac{k}{2}\rfloor \\ j = (2n-k+2)-i \end{cases}$$

SONST istop = istop +1

1.c) FÜR k=n

Paralleles Orthogonalisieren der Spalten a_i und a_j für

$$\begin{cases} i = 2,3,...,\lceil\frac{n}{2}\rceil \\ j = (n+2)-i \end{cases}$$

<u>*Schritt 2:*</u> Parallele Berechnung der V_k als direkte Summe der $R_k(i,j)$:

$$V_k = R_k(i_1,j_1) \oplus \cdots \oplus R_k(i_m,j_m), \;\; m \leq \lfloor\tfrac{n}{2}\rfloor.$$

<u>*Schritt 3:*</u> Parallele Multiplikation der Matrix A mit V_k von rechts, d.h. parallele Berechnung der Matrixfolge $A_{k+1} = A_k V_k$.

<u>*Schritt 4:*</u> Berechnung der σ_i und Abbruchkriterium:

WENN $istop = \frac{n(n-1)}{2}$ *DANN* berechne parallel $\sigma_i = (A^T A)_{ii}^{\frac{1}{2}}$ und Ende des Verfahrens, *SONST* gehe zu Schritt 1.

- Die Orthogonalisierung der Spalten in Schritt 1 wird implementiert mit Rotationsmatrizen, in denen sin und cos die Bedingungen (4.4.10) erfüllen.
- Alle parallele Berechnungen können z.B. mit Hilfe des rekursiven Doppelns oder mit rekurrenten Realationen (vgl. Kapitel 3.5) berechnet werden.
- Der einseitige Algorithmus verändert bei jeder Transformation nur Elemente in den Spalten i und j der Matrix A :

$$\begin{aligned} a_i^{k+1} &= ca_i^k + sa_j^k, \\ a_j^{k+1} &= -sa_i^k + ca_j^k. \end{aligned} \tag{4.4.12}$$

- Das Verfahren wird beendet, sobald $istop = \frac{n(n-1)}{2}$ die genaue Anzahl der Spaltenpaare der $(n \times n)-$Matrix erreicht hat. Dann sind alle Spalten der Matrix A orthogonalisiert, d.h. die Matrix A ist in die orthogonale Matrix B (4.4.1) transformiert.
- Die Singulärwerte der Matrix A erhält man durch die positiven Wurzeln der n Diagonalelemente der transformierten Matrix $A^T A$ (die ursprüngliche Matrix A ist durch die Matrix $B = U\Sigma$ (4.4.3) ersetzt).

4.5 Das Block-Jacobi-Verfahren zur Berechnung der SVD

In diesem Abschnitt wird ein Block-Jacobi-Verfahren skizziert, das Singulärwerte einer Matrix durch wiederholtes Lösen von Problemen kleinerer Ordnung errechnet.

Definition 4.5.1 (Blockmatrix) *Die Matrix $A = [a_{ij}] \in I\!\!K^{m \times n}$ ist eine Blockmatrix, wenn sich Koeffizienten a_{ij} so zu s^2, $s \in I\!\!N$, Blöcken A_{lk} zusammenfassen lassen, so daß die Matrix eine einfacher beschreibbare Gestalt erhält:*

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \cdots & A_{1k} \\ A_{21} & A_{22} & A_{23} & & A_{2k} \\ A_{31} & A_{32} & A_{33} & & A_{3k} \\ \vdots & & & & \vdots \\ A_{lr} & & & \cdots & A_{lk} \end{bmatrix} \qquad \begin{array}{l} 1 \le k \le n \\ 1 \le l \le m \end{array}$$

Beispiel 4.5.2 1. Blockdreiecksmatrix:

$$\begin{bmatrix} x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x \\ 0 & 0 & x & x & x & x & x & x & x \\ 0 & 0 & x & x & x & x & x & x & x \\ 0 & 0 & x & x & x & x & x & x & x \\ 0 & 0 & x & x & x & x & x & x & x \\ 0 & 0 & 0 & 0 & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & 0 & 0 & x & x & x \end{bmatrix}$$

2. Blockdiagonalmatrix:

$$\begin{bmatrix} x & x & & & & & & & & \\ x & x & & & & & & & & \\ & & x & x & x & & & & & \\ & & x & x & x & & & & & \\ & & x & x & x & & & & & \\ & & & & & x & & & & \\ & & & & & & x & x & x & x \\ & & & & & & x & x & x & x \\ & & & & & & x & x & x & x \\ & & & & & & x & x & x & x \end{bmatrix}$$

$\circ$

Bemerkung 4.5.3 1. Die Idee des Block-Jacobi-Verfahren ist die Diagonalisiertung der Matrix A durch Lösen von Unterproblemen.

2. Um berechenen zu können, ob die gewünschte Diagonalform erreicht ist, wird, wie für das klassische Jacobi-Verfahren (4.1.1), die Frobenius-Norm der Nebendiagonal-Blöcke definiert:

$$N(A) := (\sum_{i \neq j} \|A_{ij}\|_F^2)^{\frac{1}{2}}.$$

$\circ$

- Ist A eine Blockmatrix, so errechnen sich die Singulärwerte dieser Matrix, indem man die Singulärwertzerlegung jedes einzelnen Blocks findet.
- Um $N(A)$ reduzieren zu können, betrachte man das 2×2-Block-Unterproblem S_{ij} der Form

$$S_{ij} = \begin{bmatrix} A_{ii} & A_{ij} \\ A_{ji} & A_{jj} \end{bmatrix}. \tag{4.5.4}$$

Man wähle orthogonale Matrizen W_{ij} und Z_{ij}, so daß gilt:

$$\begin{bmatrix} B_{ii} & B_{ij} \\ B_{ji} & B_{jj} \end{bmatrix} = W_{ij}^T S_{ij} Z_{ij} \qquad \text{und} \tag{4.5.5}$$

$$\|B_{ij}\|_F^2 + \|B_{ji}\|_F^2 \leq \Theta^2(\|A_{ij}\|_F^2 + \|A_{ji}\|_F^2), \quad \text{für ein festes } 0 \leq \Theta < 1. \tag{4.5.6}$$

B ist die durch Multiplikation von W_{ij}^T und Z_{ij} aktualisierte Matrix A. Es gilt also:

$$N(B)^2 \leq N(A)^2 - (1 - \Theta^2)(\|A_{ij}\|_F^2 + \|A_{ji}\|_F^2). \tag{4.5.7}$$

- Im allgemeinen werden bei jedem Schritt des Block-Jacobi-Verfahrens die Blöcke so gewählt, daß eine optimale Eliminierung der Nebendiagonalelemente möglich ist. Für $k > 2$ ist dies ein aufwendiges kombinatorisches Problem.
- Um die Konvergenz des Verfahrens zu sichern reicht jedoch die Forderung, daß sich in jedem Schritt das größte Nebendiagonalelement der Matrix in dem Block befindet, der bearbeitet wird. Das bedeutet wenigstens einen Suchdurchgang durch die gesamte Matrix.
- Es gibt auch Algorithmen, die nach dem normgrößten Block der Matrix suchen und dann entsprechende Vertauschungen vornehmen. [ScWa85]
- Es gibt Probleme, bei denen die Suche nach Pivotelementen zu kostspielig und zeitaufwendig sind. Der zyklische Block-Algorithmus verzichtet darauf und bearbeitet die Blöcke nach einer vorgegeben Reihenfolge.
- Durch sinnvolles Lösen der Unterprobleme läßt sich A, z.B. mit Hilfe des zyklischen Jacobi-Verfahrens (Kapitel 4.2), zu einer Block-Diagonalmatrix reduzieren.

Beispiel 4.5.8 Die Matrix $S \in \mathbb{R}^{r \times c}$ ist ein Teilbroblem mit 4×2-Blockmatrizen. Für S werden orthogonale Matrizen W und Z gesucht, so daß eine Diagonalmatrix $\Sigma = W^T S Z$ entsteht:

Schritt 1: Mit Hilfe von Rotationsmatrizen berechnet man die orthogonale Matrix Q, so daß

$$Q^T S = \begin{bmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} =: R.$$

Schritt 2: Die Zeilen, deren Elemente Null sind, werden aus der Matrix R herausgenommem, und man berechnet die Singulärwertzerlegung

$$U^T \begin{bmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \end{bmatrix} Z = \Sigma.$$

Wie im klassischen Algorithmus zur Singulärwertzerlegung (Kapitel 2.5) wird auch hier die Matrix R zunächst in Bidiagonalgestalt transformiert und dann der QR-Algorithmus angewandt. [Bisch87]

∘

Das **parallele Block-Jacobi-Verfahren** wird nach dem Divide&Conquer-Prinzip (Kapitel 3.7) in einzelne Durchgänge, *sweeps*, unterteilt. In einem sweep wird jedes Unterproblem S_{ij} genau einmal behandelt.

Für die Reihenfolge $(i_1, j_1), \cdots, (i_r, j_r)$, $r = \frac{1}{2}k(k-1)$, der Indexpaare der Nebendiagonalen einer $k \times k-$Blockmatrix wird der Block-Jacobi-Algorithmus beschrieben.
τ ist eine festgelegte positive Schranke mit $\tau \leq \epsilon \frac{\|A\|_F}{k}$.

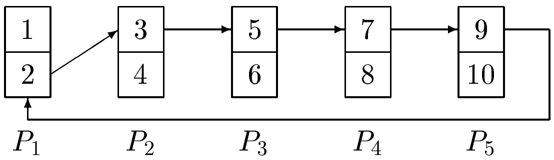
Algorithmus 4.5.9 (Block-Jacobi-Algorithmus)

$U \leftarrow I_m, \quad V \leftarrow I_n$
WHILE $N(A) > \epsilon \|A\|_F$ *DO*
  FOR $l = 1$ *TO* $\frac{k(k-1)}{2}$ *DO*
    $(i, j) \leftarrow (i_l, j_l)$
    IF $\mu_{ij} := \sqrt{\|A_{ij}\|_F^2 + \|A_{ij}\|_F^2} \geq \tau$ *THEN*
      Löse das Unterproblem S_{ij}, so daß sich Z_{ij} und W_{ij} ergeben:
      $$[A_i, A_j] \leftarrow [A_i, A_j] Z_{ij} \tag{4.5.10}$$
      $$[U_i, U_j] \leftarrow [U_i, U_j] W_{ij} \tag{4.5.11}$$
      $$[V_i, V_j] \leftarrow [V_i, V_j] Z_{ij} \tag{4.5.12}$$
      FOREACH $q \in \{1, \cdots, k\}$ *DO*
        $$\begin{bmatrix} A_{iq} \\ A_{jq} \end{bmatrix} \leftarrow W_{ij}^T \begin{bmatrix} A_{iq} \\ A_{jq} \end{bmatrix} \tag{4.5.13}$$
      END FOREACH
    END IF
  END FOR
END WHILE

- Hat man $p = \frac{1}{2}k$ (k sei gerade) Prozessoren zur Verfügung, werden die Blockmatrizen A, U und V so aufgeteilt, daß jeder Prozessor je zwei (Block-) Spalten zugeteilt bekommt. Die Prozessoren haben damit genügend Informationen, um die Unterprobleme lösen zu können.
- Das Unterproblem S_{ij} ist gelöst, wenn $\mu_{ij} \leq \tau$ ist.
- In dem Algorithmus wird davon ausgegangen, daß die Ordnung der einzelnen Unterprobleme in jedem sweep gleich ist.
- In jedem sweep werden $p(k-1)$ Teilprobleme einmal bearbeitet.
- Der Algorithmus berechnet die orthogonalen Matrizen U und V, mit $N(U^TAV) \leq \epsilon \|A\|_F$.
- Die Matrix A wird von U^TAV überschrieben.
- Das Block-Jacobi-Verfahren ist quadratisch konvergent [Bisch87].
- Das Verfahren berechnet die Singulärenwerte nicht nach ihrer Größe sortiert.
- Die Schritte (4.5.10) - (4.5.12) können ebenfalls parallel ausgeführt werden, wenn die einzelnen Prozessoren die geforderten Spalten der Matrizen A, U und V zugeteilt bekommen.
- Nur für (4.5.13) benötigt jeder Prozessor die W_{ij}'s der anderen Prozessoren.
- Sind die einzelnen Teilprobleme gelöst, weden die Spalten neu aufgeteilt.

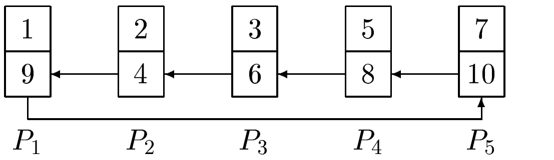
Beispiel 4.5.14 für ein sweep einer $(10 \times 10)-$Matrix mit Hilfe von 5 Prozessoren:
Prozessor P_i bekommt die Spalten $(2i-1)$ und $2i$, $i = 1, 2, 3, 4, 5$.
Die Spaltenpaare in den 5 Prozessoren sind zu Beginn $(1,2), (3,4), (5,6), (7,8), (9,10)$:

a)



Nach der ersten Verschiebung a) ergibt sich $(1,9), (3,4), (5,6), (7,8), (9,10)$:

b)



Weitere Verschiebungen:	(1,4),(2,6),(3,8),(5,10),(7,9)	b)
	(1,7),(4,6),(2,8),(3,10),(5,9)	a)
	(1,6),(4,8),(2,10),(3,9),(5,7)	b)
	(1,5),(6,4),(4,10),(2,9),(3,7)	a)
	(1,8),(6,10),(4,9),(2,7),(3,5)	b)
	(1,3),(8,10),(6,7),(4,7),(2,5)	a)
	(1,10),(8,9),(6,7),(4,5),(2,3)	b)

An diesem Punkt endet ein sweep, und man erzeugt wie in a) die neue Spaltenkombinationen $(1,10), (8,9), (6,7), (4,5), (2,3)$ für das nächste Teilproblem.

$\circ$

Bemerkung 4.5.15 1. Das Beispiel beschreibt eine optimale Anordnung zur Erzeugung neuer Teilprobleme in einem sweep: Es werden (k-1) Spaltenpaare erzeugt und jeder Pozessor vertauscht jeweils nur eine Spalte pro Rotation. Auf Grund der Ringordnung hat jeder Prozessor zu seinem Nachbarn nur einen Verbindungspunkt.

2. Die Wahl der Teilprobleme (4.5.4) hat großen Einfluß auf die Konvergenzgeschwindigkeit des Algorithmus. [Bisch87]

∘

4.6 Einseitiges Block-Jacobi-Verfahren zur Berechnung der SVD auf Distributed Memory Rechner

Singulärwertzerlegungen treten u.a. bei linearen Approximationsproblemen auf. Die folgenden Betrachtungen wurden für Gramsche Matrizen[1] vorgenommen.

Das einseitige Jacobi-Verfahren berechnet die Singulärwerte und die Linkssingulärvektoren U nach dem Prinzip der Rekurrenten Relationen (Kapitel 3.6) mit einer Folge von Matrizen

$$\begin{aligned} A_0 &= A \in I\!R^{m\times n}, \\ A_{k+1} &= A_k R_k \end{aligned} \tag{4.6.1}$$

(vgl. Kapitel 4.4). R_k eleminiert als ebene Rotation das Element $(i,j),\ i<j$, der implizit definierten Gramschen Matrix

$$G_{k+1} \equiv A_{k+1}^T A_{k+1} = R_k^T A_k^T A_k R_k. \tag{4.6.2}$$

Bemerkung 4.6.3 1. Um R_k betimmen zu können, benötigt man die Koeffizienten $g_{ii}^{(k)}, g_{jj}^{(k)}$ und $g_{ij}^{(k)} = (a_i^{(k)})^T a_j^{(k)}$.

2. Die Folge $\{A_k\}_{k\in I\!N}$ konvergiert gegen die Matrix $\bar{A}$ mit $\sigma_{ii} = \|\bar{a}_i\|_2$ und $u_i = \sigma_{ii}\bar{a}_i$ [ArSl96].

∘

Berechnet werden kann das Problem mit einer Kombination aus dem parallelen Block-Jacobi-Algorithmus (Kapitel 4.5) und dem parallelen einseitigen Jacobi-Verfahren. Wegen der vielen Matrix-Multiplikationen kann zusätzlich die Parallelität von Pipeline-Prozessoren z.B. durch den log-sum-Algorithmus (vgl. Kapitel 3.5) optimal genutzt werden.

- Zur Berechnung der Singulärwerte wird die Matrix A in $2p$ Spaltenblöcke unterteilt: $A = [A^{(1)}A^{(2)}\cdots A^{(2p)}]$.
- Zwei aufeinanderfolgende Blöcke werden jeweils einem Prozessor $i,\ i = 0,1,\cdots,p-1$, zugeteilt und als A_L und A_R bezeichnet.
- Jeder Prozessor berechnet einen Teil der Gramschen Matrix mit den entsprechenden Spalten-Blöcken: $G \equiv \begin{bmatrix} G_{11} & G_{12} \\ G_{12}^T & G_{22} \end{bmatrix} = \begin{bmatrix} A_L^T A_L & A_L^T A_R \\ A_R^T A_L & A_R^T A_R \end{bmatrix}$, vgl. (4.5.4).
- Nach einem Durchgang des klassischen Jacobi-Verfahrens (Kapitel 4.1) ist die Rotationsmatrix R berechnet, und die Iteration (4.6.1) des einseitigen Jacobi-Verfahren $[A_L A_R]R$ wird durchgeführt.

[1] Matrizen zur Berechnung der Entwicklungskoeffizienten beim allgemeinen Approximationsproblem. [Überh95]

- Der letzte Schritt und die Berechnung des Blocks G_{12} wird von der BLAS-3 Routine `dgemm` durchgeführt.
- Nach diesen Berechnungen werden die Blöcke A_R an den nächsten Prozessor weitergegeben.

Algorithmus 4.6.4 (einseitiger Block-Jacobi-Algorithmus)
Prozessor i führt folgendes Programm aus:

$G_{11} \leftarrow A_L^T A_L, \quad G_{22} \leftarrow A_R^T A_R, \quad G_{12} \leftarrow A_L^T A_R$
REPEAT bis zur Konvergenz
  FOR $j = 1, \cdots, 4p$
    IF $(j-1) \bmod 2p+1 \neq 2p-2(i-1)$ *THEN*
      berechne mit dem Jacobi-Verfahren G und R.
      $[A_L A_R] \leftarrow [A_L A_R] R$.
      gebe an den nächsten und erhalte vom vorhergehenden Prozessor A_R und G_{22}.
      $G_{12} \leftarrow A_L^T A_R$.
      ELSE
      tausche A_L mit A_R und G_{11} mit G_{22}.
      gebe an den nächsten und erhalte vom vorhergehenden Prozessor A_R und G_{22}.
      $G_{12} \leftarrow A_L^T A_R$.
    END IF
  END FOR
END REPEAT

- Nur die Spalten von U, die zu Singulärenwerten $\neq 0$ gehören, werden berechnet.
- Die Ausgangsmatrix A wird von den Singulärvektoren überschrieben.
- Der Algorithmus konvergiert bei dem Abbruchkriterium $|g_{ij}^k| < n \cdot \epsilon \cdot \sqrt{g_{ii}^k}\sqrt{g_{jj}^k}$.
- Bei großer Dimension (n=1000, 2000) konvergiert der Algorithmus ungefähr nach 8 Doppel-Durchgängen.
- Die Genauigkeit der berechneten Singulärwerte entspricht denen von Nicht-Blockverfahren [DeVe92].
- Der berechnete overhead (25%) gegenüber den Nicht-Blockverfahren wird ausgeglichen durch das Nutzen von `dgmm` für große Matrizen.
- Die Kosten (Definition 3.3.4) von Algorithmus 4.6.4 in μs:

$$\begin{aligned} C_p = \; & (2m^2+m)(2n-1)\phi(n,m) + 8 \cdot 4p[6(2m)(4m-1)\phi(n,2m) \\ & + m^2(2n-1)\phi(n,m) + 4\eta + 2\tau m(m+1)/2 + 2\tau nm]. \end{aligned}$$

  - p Anzahl der Prozessoren
  - $\phi(n,m)\mu s$ Ausführungszeit eines Flop
  - $m = \frac{n}{2p}$
  - $(\eta + \tau k)\mu s$ Übertragungszeit von $8k$ bytes
  - η Vorlaufzeit
  - $\frac{1}{\tau}$ Bandbreite

Bemerkung 4.6.5 Bei paralleler Berechnung und Kommunikation (asynchrone Kommunikation), kann die Kommunikation, in Abwandlung von Algoritmus 4.6.4, überlappend durchgeführt werden (mit Ausnahme des startup):

- G_{22} wird weitergeleitet, während die obere Hälfte von $[A_L A_R]$ mit R multipliziert wird,
- die obere Hälfte von A_R wird weitergeleitet, während die untere Hälfte von $[A_L A_R]$ mit R multipliziert wird,
- die untere Hälfte von A_R wird weitergeleitet, während die Berechnung von G_{12} mit der oberen Hälfte von $[A_L A_R]$ begonnen wird usw.

$\circ$

Theoretisch berechneter und gemessener (auf Intel Paragon mit 96 Knoten und message passing System) Speedup (Definition 3.3.2) für Algorithnus 4.6.4 und die asynchrone Version aus Bemerkung 4.6.5:

Anzahl der Prozessoren	4	8	16	32	64	96
theoretischer Speedup Algorithmus 4.6.4	4	8	16	32	63.8	95.3
theoretische Speedup asynchrone Version	3.99	7.97	15.8	31.1	60.2	87.4
gemessener Speedup Algorithmus 4.6.4				29.5	49.1	54.7
gemessener Speedup asynchrone Version				28.4	47.5	52.3

- $\eta = 65$
- $\tau = \frac{1}{8}$
- $\pi(n, m)$ ist nach der Tabelle aus Bemerkung 4.6.6 berechnet.
- $n = 2048$

Bemerkung 4.6.6 Leistung der **Blas-3 Routine** `dgemm` für eine $(2048 \times m) \times (m \times m)$ Matrix-Multiplikation auf dem Intel i860 RISC[1] Prozessor mit einer Höchstleistung von 50 MFLOPS[2]:

m	1	2	4	8	16	32	64	128	256	512
Mflop/s	11.7	14.8	26.6	34.5	40.5	43.0	44.7	45.3	45.8	45.7

$\circ$

Bemerkung 4.6.7 Die asynchrone Version benötigte 630, 342 und 300 Sekunden für $p = 32,\ 64,\ 96$. $\circ$

[1] superskalare Prozessoren mit gemeinsamem Speicher [Giloi93]

[2] million floating-point operations per second

Literatur

[ArSl96] P. Arbenz, I. Slapničar, *On an Implementation of a One-sided Block Jacobi Method on a Distributed Memory Computer,* Z. angew. Math. Mech. 76 S1, 1996.

[BeSa86] M.W. Berry, A. Sameh, *Multiprocessor Jacobi Algorithms for Dense Symetric Eigenalue and Singular Value Decomposition,* CSRD Report No.546, University of Illinois at Urbana-Champaign, 1986.

[Bisch87] Ch. Bischof, *Comuting the Singular Value Decomposition on a Distributes System of Vector Processors* TR 87-869 (Revision of TR 86-798) Department of Computer Science, Cornell University, 1987.

[BuGe85] W. Bunse, A. Bunse-Gerstner, *Numerische lineare Algebra,* Teubner Studienbücher Mathematik, Stuttgart 1985.

[Cau1829] A.L. Cauchy, *Sur l'équation á l'aide laquelle on détermine les inégalités séculaires des mouvements des planètes,* in Oeuvres Complétes (II^e Série), Vol. 9, 1829.

[DeVe92] J. Demmel, K. Veselić, *Jacobi's Method is more accurate than QR,* SIAM J. Matrix Anal. Appl. Vol. 13, No. 4, pp. 1204-1245, 1992.

[FoHe60] G.E. Forsythe, P. Henrici, *The Cyclic Jacobi Method for Computing the Principal Values of a Complex Matrix,* Trans. Amer. Math. Soc. 94, 1-23, 1960.

[Gau1823] C.F. Gauss, *Theoria combinationis observationus erroribus minimis abnoxiae, pars posterior,* in Werk, IV, Königlichen Gesellschaft der Wissenschaften zu Göttingen (1880) pp.27-53, 1823.

[Giloi93] W.K. Giloi, *Rechnerarchitektur,* ISBN 3-540-56355-5, Springer-Verlag Berlin Heidelberg, 1995.

[GoLo84] G.H. Golub, C.F. van Loan, *Matrix Computations,* The Johns Hopkins University Press, Baltimore 1989.

[Gottw96] S. Gottwald, *Das Divide&Conquer-Prinzip für massiv-parallele numerische Algorithmen,* Dissertation, ISBN 3-8265-1737-7, Shaker Verlag Aachen 1996.

[HoJo85] R.A. Horn, C.R. Johnson, *Matrix Analysis,* Cambridge University Press, Cambrdge 1985.

[Jac1846] C.G.J. Jacobi, *Über ein leichtes Verfahren die in der Theorie der Säculärströrungen vorkommenden Gleichungen numerisch aufzulösen,* J. Reine Angew. Math.,30, pp.51-94, 1846.

[Jac1857] C.G.J. Jacobi, *Über eine elementare Transformation eines in Bezug jedes von zwei Variablen-Systemen linearen und homogenen Ausdrucks,* J. Reine Angew. Math.,53 (posthumous), pp.265-270, 1857.

[KuMu74] D.J. Kuck, Y. Muraoka, *Bounds on the Parallel Evaluation of Arithmetic Expressions Using Associativity and Commutativity,* Acta Informatica, Vol.3, S.203-216, 1974.

[Kogge74] P.M. Kogge, *Parallel Solutions of Recurrence Problems* IBM J. Res. Develop. 18, S.138-148, März 1974.

[Maess85] G. Maess, *Lineare Algebra, Vorlesungen über numerische Mathematik* Birkhäuser Verlag, Basel 1985.

[SchBr79] U. Schendel, J. Brandenburger, *Algorithmen zur Lösung Rekurrenter Relationen* Preprint Nr.101/79, Freie Universität Berlin, 1979.

[Schen84] U. Schendel, *Introduction to Numerical Methods for Parallel Computers,* John Wiley & Sons, 1984.

[Schen95] U. Schendel, *Einführung in die Parallele Numerik,* Vorlesungsskript, Fachbereich Mathematik und Informatik, Freie Universität Berlin, 1995.

[SchZi97] U. Schendel, A. Ziegler *Numerische Algorithmen der Linearen Algebra,* Vorlesungsskript, Fachbereich Mathematik und Informatik, Freie Universität Berlin, 1997.

[ScWa85] D. Scott, R.C. Ward, *Parallel Block Jacobi Eigenvalue Algorithms using Systolic Arrays,* University of Texas at Austin, Dept. of Computer Science, Report TR 85-01, 1985.

[Überh95] C. Überhuber, *Computer-Numerik 2,* Springer-Verlag, Heidelberg 1995.

[Wei1868] K. Weierstrass, *Zur Theorie der bilinearen und quadratischen Formen,* Monatshefte Akademie der Wissenschaften Berlin, pp.310-338, 1868.

[Werner92] J. Werner, *Numerische Mathematik 1 und 2,* Vieweg & Sohn, Braunschweig/Wiesbaden 1992.

[Wilkin92] J.H. Wilkinson, *The Algebraic Eigenvalue Problem,* Clarendon Press, Oxford, 1992.

[WiRe71] J.H. Wilkinson, C. Reinisch, *Lineare Algebra,* Die Grundlehren der mathematischen Wissenschaften Band 186, Springer-Verlag, Berlin 1971.

[Ziegl92] A. Ziegler, *Singulärwertzerlegung einer Matrix und Anwendungen,* Diplomarbeit, Fachbereich Mathematik, Freie Universität Berlin, 1992.