Übung zur Algorithmik kontinuierlicher Systeme

Übung 4 – Lineare Ausgleichsrechnung und Matrixformate

Sommersemester 2021

Florian Frank

Friedrich-Alexander-Universität Erlangen-Nürnberg

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Was machen wir heute?

Wo stehen wir?

Rekapitulation — Lösbarkeit von LGS

Rekapitulation — Lineare Ausgleichsprobleme

Aufgabe 1 — Auslgeichsgerade

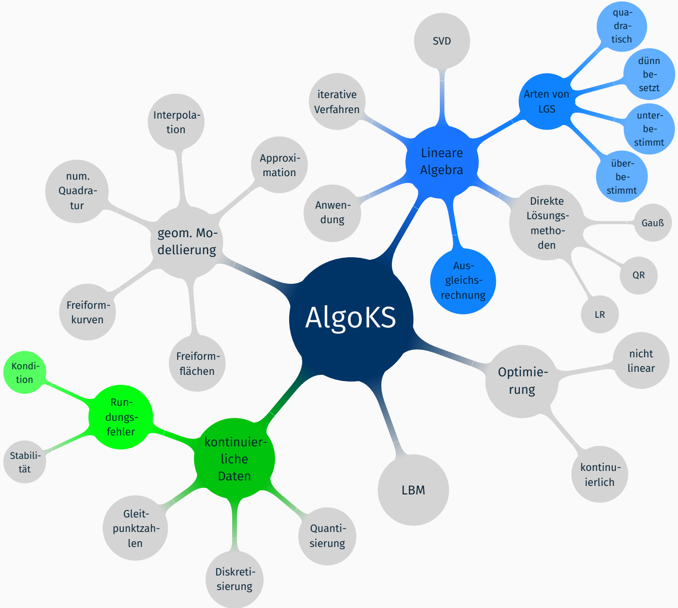
Rekaptiulation — Effizientes Speichern von Matrizen

Aufgabe 1 — Matrixformate

Hausaufgabe — Ausgleichspolynom und Matrixformate

Wo stehen wir?

Wo stehen wir?



Rekapitulation — Lösbarkeit von LGS

Bestimmtheit von Gleichungssystemen

Wir betrachten die drei linearen Gleichungssysteme

$$\underbrace{\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} x = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}}_{\text{(I)}} \qquad \underbrace{\begin{pmatrix} 1 & 2 & 0 \\ 0 & 2 & 3 \\ 2 & 4 & 0 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix}}_{\text{(II)}} \qquad \underbrace{\begin{pmatrix} 1 & 2 \\ 0 & 2 \\ 1 & 0 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix}}_{\text{(III)}}$$

und stellen uns die Frage, welche eindeutig lösbar sind.

Bestimmtheit von Gleichungssystemen

$$\underbrace{\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} x = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}}_{\text{(I)}} \qquad \underbrace{\begin{pmatrix} 1 & 2 & 0 \\ 0 & 2 & 3 \\ 2 & 4 & 0 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix}}_{\text{(II)}} \qquad \underbrace{\begin{pmatrix} 1 & 2 \\ 0 & 2 \\ 1 & 0 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix}}_{\text{(III)}}$$

Bestimmtheit von Gleichungssystemen

$$\underbrace{\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} x = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}}_{\text{(I)}} \qquad \underbrace{\begin{pmatrix} 1 & 2 & 0 \\ 0 & 2 & 3 \\ 2 & 4 & 0 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix}}_{\text{(II)}} \qquad \underbrace{\begin{pmatrix} 1 & 2 \\ 0 & 2 \\ 1 & 0 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix}}_{\text{(III)}}$$

Gleichungssystem (I) ist **eindeutig lösbar**, denn jede Gleichung bringt eine neue Information in das zu lösende System mit ein. Keine Gleichung kann ersetzt werden und *steht im Widerspruch zu anderen Gleichungen.*

Bestimmtheit von Gleichungssystemen

$$\underbrace{\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} x = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}}_{\text{(I)}} \qquad \underbrace{\begin{pmatrix} 1 & 2 & 0 \\ 0 & 2 & 3 \\ 2 & 4 & 0 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix}}_{\text{(II)}} \qquad \underbrace{\begin{pmatrix} 1 & 2 \\ 0 & 2 \\ 1 & 0 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix}}_{\text{(III)}}$$

Gleichungssystem (II) ist **nicht eindeutig lösbar**, Gauß lieferte

$$\left(\begin{array}{ccc:c} 1 & 2 & 0 & 2 \\ 0 & 2 & 3 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right),$$

was die Lösungsmenge $\left(1 + 3\lambda \,,\, {}^{1-3\lambda}/_{2} \,,\, \lambda \right)^T$ für **beliebiges** $\lambda \in \mathbb{R}$ ergibt.

Bestimmtheit von Gleichungssystemen

$$\underbrace{\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} x = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}}_{\text{(I)}} \qquad \underbrace{\begin{pmatrix} 1 & 2 & 0 \\ 0 & 2 & 3 \\ 2 & 4 & 0 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix}}_{\text{(II)}} \qquad \underbrace{\begin{pmatrix} 1 & 2 \\ 0 & 2 \\ 1 & 0 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix}}_{\text{(III)}}$$

Gleichungssystem (II) ist **nicht eindeutig lösbar**, Gauß lieferte

$$\left(\begin{array}{ccc:c} 1 & 2 & 0 & 2 \\ 0 & 2 & 3 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right),$$

was die Lösungsmenge $\left(1 + 3\lambda \, , \, {}^{1-3\lambda}/_{2} \, , \, \lambda \right)^T$ für **beliebiges** $\lambda \in \mathbb{R}$ ergibt.

Wir haben zu wenig Informationen um das LGS eindeutig zu lösen, es ist also *unter*bestimmt/*unter*spezifiziert.

Bestimmtheit von Gleichungssystemen

$$\underbrace{\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} x = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}}_{\text{(I)}} \qquad \underbrace{\begin{pmatrix} 1 & 2 & 0 \\ 0 & 2 & 3 \\ 2 & 4 & 0 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix}}_{\text{(II)}} \qquad \underbrace{\begin{pmatrix} 1 & 2 \\ 0 & 2 \\ 1 & 0 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix}}_{\text{(III)}}$$

Gleichungssystem (III) ist **nicht nur nicht eindeutig**, sondern auch **gar nicht lösbar**, Gauß lieferte das wiedersprüchliche LGS

$$\left(\begin{array}{cc|c} 1 & 2 & 2 \\ 0 & 2 & 1 \\ 0 & 0 & 3 \end{array}\right).$$

Bestimmtheit von Gleichungssystemen

$$\underbrace{\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} x = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}}_{\text{(I)}} \qquad \underbrace{\begin{pmatrix} 1 & 2 & 0 \\ 0 & 2 & 3 \\ 2 & 4 & 0 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix}}_{\text{(II)}} \qquad \underbrace{\begin{pmatrix} 1 & 2 \\ 0 & 2 \\ 1 & 0 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix}}_{\text{(III)}}$$

Gleichungssystem (III) ist **nicht nur nicht eindeutig**, sondern auch **gar nicht lösbar**, Gauß lieferte das wiedersprüchliche LGS

$$\left(\begin{array}{cc|c} 1 & 2 & 2 \\ 0 & 2 & 1 \\ 0 & 0 & 3 \end{array}\right).$$

Wir haben *zu viele* Informationen, welche sich auch noch widersprechen, um das LGS eindeutig bzw. überhaupt zu lösen, es ist also *über*bestimmt/*über*spezifiziert.

Definition 4.1

Sei durch $A \in \mathbb{R}^{m \times n}, x \in \mathbb{R}^n, b \in \mathbb{R}^m$ ein lineares Gleichungssystem $Ax = b$ bestimmt, so nennen wir das LGS …

Definition 4.1

Sei durch $A \in \mathbb{R}^{m \times n}, x \in \mathbb{R}^n, b \in \mathbb{R}^m$ ein lineares Gleichungssystem $Ax = b$ bestimmt, so nennen wir das LGS ...

... **eindeutig lösbar**, wenn die Lösungsmenge $\mathbb{L} = \{ x \in \mathbb{R}^n \mid Ax = b \}$ aus nur einem Element besteht.

Bestimmtheit von Gleichungssystemen — Definitionen

Definition 4.1

Sei durch $A \in \mathbb{R}^{m \times n}, x \in \mathbb{R}^n, b \in \mathbb{R}^m$ ein lineares Gleichungssystem $Ax = b$ bestimmt, so nennen wir das LGS …

… **eindeutig lösbar**, wenn die Lösungsmenge $\mathbb{L} = \{\, x \in \mathbb{R}^n \mid Ax = b \,\}$ aus nur einem Element besteht.

… **quadratisch**, wenn das Gleichungssystem **genauso viele** Gleichungen wie Unbekannte hat. $(n = m)$

Bestimmtheit von Gleichungssystemen — Definitionen

Definition 4.1

Sei durch $A \in \mathbb{R}^{m\times n}, x \in \mathbb{R}^n, b \in \mathbb{R}^m$ ein lineares Gleichungssystem $Ax = b$ bestimmt, so nennen wir das LGS …

… **eindeutig lösbar**, wenn die Lösungsmenge $\mathbb{L} = \{\, x \in \mathbb{R}^n \mid Ax = b \,\}$ aus nur einem Element besteht.

… **quadratisch**, wenn das Gleichungssystem **genauso viele** Gleichungen wie Unbekannte hat. $(n = m)$

… **überbestimmt**, wenn das Gleichungssystem **mehr** Gleichungen als Unbekannte hat. $(n > m)$

Bestimmtheit von Gleichungssystemen — Definitionen

Definition 4.1

Sei durch $A \in \mathbb{R}^{m \times n}, x \in \mathbb{R}^n, b \in \mathbb{R}^m$ ein lineares Gleichungssystem $Ax = b$ bestimmt, so nennen wir das LGS ...

... **eindeutig lösbar**, wenn die Lösungsmenge $\mathbb{L} = \{\, x \in \mathbb{R}^n \mid Ax = b \,\}$ aus nur einem Element besteht.

... **quadratisch**, wenn das Gleichungssystem **genauso viele** Gleichungen wie Unbekannte hat. $(n = m)$

... **überbestimmt**, wenn das Gleichungssystem **mehr** Gleichungen als Unbekannte hat. $(n > m)$

... **unterbestimmt**, wenn das Gleichungssystem **weniger** Gleichungen als Unbekannte hat. $(n < m)$

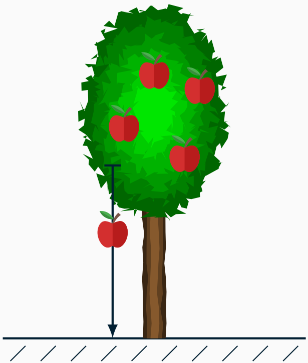
Rekapitulation — Lineare Ausgleichsprobleme

Motivation (I)

Wir betrachten einen fallenden Apfel und wundern uns, ob wir mit unseren bisherigen Kenntnissen nicht sowohl die Fallhöhe h, als auch die Erdbeschleunigung g experimentell bestimmen können.
Wir erinnern uns dunkel an den Physikunterricht und wissen, dass beide Größen den Zusammenhang

$$y = h - \frac{1}{2} \cdot g \cdot t^2$$

erfüllen. Sprich, wenn wir die aktuelle Höhe des Apfels, sowie die vergangene Zeit seit Beginn des Falls berechnen können, bekommen wir g und h.



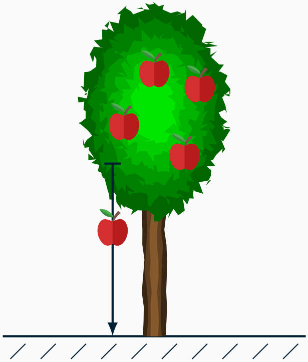
Motivation (II)

Zusammenhang:

$$y = h - \frac{1}{2} \cdot g \cdot t^2$$

Wir messen:

- aktuelle Höhe des Apfels y
- vergangene Zeit seit Beginn des Falls t



Wir stellen fest:

Theoretisch genügen uns nur **zwei** Messungen. Dennoch wird uns empfohlen deutlich öfters zu messen. Warum?

Motivation (III)

Messungenauigkeiten

Generell gilt: Messungen sind **immer** fehlerbehaftet!

Deswegen wollen wir deutlich mehr als nur zwei Messungen durchführen und danach g und h so bestimmen, dass die Messungen *bestmöglich* beschrieben werden.

Motivation (III)

Messungenauigkeiten

Generell gilt: Messungen sind **immer** fehlerbehaftet!

Deswegen wollen wir deutlich mehr als nur zwei Messungen durchführen und danach g und h so bestimmen, dass die Messungen *bestmöglich* beschrieben werden.

Ausgleichsprobleme

Solche Ausgleichsprobleme treten meistens dann auf, wenn aus Messdaten gewisse Parameter bestimmt werden sollen. Charakteristisch für diese Art von Problemen ist, dass die Anzahl n der Parameter in der Regel deutlich kleiner ist als die Anzahl m der Messpunkte.

Motivation (IV) — Beispiel

Man vermutet, dass die Messdaten

t	-1	0	1	2	3	7
y	9	42	17	84	13	57

den Zusammenhang

$$y = \alpha \cdot \frac{1}{1+t^2} - \beta \cdot t + \gamma$$

mit $\alpha, \beta, \gamma \in \mathbb{R}$ erfüllen.

Motivation (IV) — Beispiel

Man vermutet, dass die Messdaten

t	-1	0	1	2	3	7
y	9	42	17	84	13	57

den Zusammenhang

$$y = \alpha \cdot \frac{1}{1+t^2} - \beta \cdot t + \gamma$$

mit $\alpha, \beta, \gamma \in \mathbb{R}$ erfüllen.

Ziel

Bestimme die Parameter α, β, und γ, so dass die Messdaten *ungefähr* dem obigen Zusammenhang gehorchen.

Motivation (V) — Beispiel

Mit $x = \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}$ ergibt sich ein zu lösendes lineares Gleichungssystem:

$$\underbrace{\begin{pmatrix} 1/2 & 1 & 1 \\ 0 & 0 & 1 \\ 1/2 & -1 & 1 \\ 1/5 & -2 & 1 \\ 1/10 & -3 & 1 \\ 1/50 & -7 & 1 \end{pmatrix}}_{=A} \cdot x = \underbrace{\begin{pmatrix} 9 \\ 42 \\ 17 \\ 84 \\ 13 \\ 57 \end{pmatrix}}_{=b}.$$

Motivation (V) — Beispiel

Mit $x = \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}$ ergibt sich ein zu lösendes lineares Gleichungssystem:

$$\underbrace{\begin{pmatrix} 1/2 & 1 & 1 \\ 0 & 0 & 1 \\ 1/2 & -1 & 1 \\ 1/5 & -2 & 1 \\ 1/10 & -3 & 1 \\ 1/50 & -7 & 1 \end{pmatrix}}_{=A} \cdot x = \underbrace{\begin{pmatrix} 9 \\ 42 \\ 17 \\ 84 \\ 13 \\ 57 \end{pmatrix}}_{=b}.$$

Dieses Gleichungssystem ist überbestimmt und hat *keine eindeutige* Lösung. Wir suchen deswegen nach einer *besten* Näherungslösung $\widehat{x}$ in dem Sinne, dass der Rest (*Residuum*) $r(x) = Ax - b$ minimal bezüglich aller Lösungen wird.

Lineares Ausgleichsproblem

Definition 4.2 (Lineares Ausgleichsproblem)

Sei ein lineares Gleichungssystem durch $A \in \mathbb{R}^{n \times m}$ und $b \in \mathbb{R}^n$ mit $n \geqslant m$ gegeben. Diese haben im Allgemeinen keine Lösung.
Anstelle betrachtet man das *lineare Ausgleichsproblem*

$$\boxed{\text{Suche ein } x \in \mathbb{R}^m \text{, so dass das } \textit{Residuum } r(x) = Ax - b \text{ min. wird.}} \qquad (*)$$

als *Ersatzproblem*.
Die Minimalbedingung wird durch Normen implementiert, da dem $\mathbb{R}^n$ für $n > 1$ die Ordnungsstruktur fehlt.

Lineares Ausgleichsproblem

Definition 4.2 (Lineares Ausgleichsproblem)

Sei ein lineares Gleichungssystem durch $A \in \mathbb{R}^{n \times m}$ und $b \in \mathbb{R}^n$ mit $n \geqslant m$ gegeben. Diese haben im Allgemeinen keine Lösung.
Anstelle betrachtet man das *lineare Ausgleichsproblem*

$$\boxed{\text{Suche ein } x \in \mathbb{R}^m\text{, so dass das } \textit{Residuum } r(x) = Ax - b \text{ min. wird.}} \qquad (*)$$

als *Ersatzproblem*.
Die Minimalbedingung wird durch Normen implementiert, da dem $\mathbb{R}^n$ für $n > 1$ die Ordnungsstruktur fehlt.

Algorithmus 4.1 — Methode der kleinsten Quadrate: *least-squares*

Das Problem $(*)$ wird hier zu:

$$\boxed{\text{Suche ein } x \in \mathbb{R}^m\text{, so dass } \varphi(x) := \|r(x)\|_2^2 \text{ minimal wird.}} \qquad (*_{\mathfrak{ls}})$$

Lin. AGP — Lösungsvariante I: QR

Wir erinnern uns an letzte Woche:

Wiederholung: QR-Zerlegung

Eine jede Matrix $A \in \mathbb{R}^{n \times m}$ lässt sich in eine orthogonale Matrix $Q \in \mathbb{R}^{n \times n}$ und eine rechte obere Dreiecksmatrix $R \in \mathbb{R}^{n \times m}$ zerlegen, so dass

$$A = Q \cdot R.$$

Wiederholung: Orthogonale Matrizen

Orthogonale Matrizen haben viele angenehme Eigenschaften, denn …

[1] Die 2-Kondition κ_2 zumindest …

Wiederholung: Orthogonale Matrizen

Orthogonale Matrizen haben viele angenehme Eigenschaften, denn ...

... sie sind *einfach* zu invertieren, ähnlich wie bei den *Permutationsmatrizen*:

$$Q^{-1} = Q^T$$

[1]Die 2-Kondition κ_2 zumindest ...

Wiederholung: Orthogonale Matrizen

Orthogonale Matrizen haben viele angenehme Eigenschaften, denn ...

... sie sind *einfach* zu invertieren, ähnlich wie bei den *Permutationsmatrizen*:

$$Q^{-1} = Q^T$$

... die Kondition[1] einer Matrix ändert sich durch Multiplikation mit einer orhtogonalen Matrix nicht, es gilt

$$\kappa_2(A) = \kappa_2(Q \cdot A) \quad \textit{für alle invertierbaren Matrizen}.$$

[1] Die 2-Kondition κ_2 zumindest ...

Wiederholung: Orthogonale Matrizen

Orthogonale Matrizen haben viele angenehme Eigenschaften, denn ...

... sie sind *einfach* zu invertieren, ähnlich wie bei den *Permutationsmatrizen*:

$$Q^{-1} = Q^T$$

... die Kondition[1] einer Matrix ändert sich durch Multiplikation mit einer orhtogonalen Matrix nicht, es gilt

$$\kappa_2(A) = \kappa_2(Q \cdot A) \quad \textit{für alle invertierbaren Matrizen.}$$

... sie sind bezüglich der euklidschen Norm invariant; es gilt

$$\|Q\|_2 = 1 \qquad \text{und} \qquad \|Qx\|_2 = \|x\|_2 \quad \forall\, x \in \mathbb{R}^n.$$

[1]Die 2-Kondition κ_2 zumindest ...

Wir erinnern uns an letzte Woche:

Wiederholung: QR-Zerlegung

Eine jede Matrix $A \in \mathbb{R}^{n \times m}$ lässt sich in eine orthogonale Matrix $Q \in \mathbb{R}^{n \times n}$ und eine rechte obere Dreiecksmatrix $R \in \mathbb{R}^{n \times m}$ zerlegen, so dass

$$A = Q \cdot R.$$

Lin. AGP — Lösungsvariante I: QR

Wir erinnern uns an letzte Woche:

Wiederholung: QR-Zerlegung

Eine jede Matrix $A \in \mathbb{R}^{n\times m}$ lässt sich in eine orthogonale Matrix $Q \in \mathbb{R}^{n\times n}$ und eine rechte obere Dreiecksmatrix $R \in \mathbb{R}^{n\times m}$ zerlegen, so dass

$$A = Q \cdot R.$$

Da die Multiplikation mit orthogonalen Matrizen die 2-Norm nicht ändert, können wir folgern:

$$\varphi(x) = \|Ax - b\|_2^2 = \left\|Q^T \cdot (Ax - b)\right\|_2^2 = \left\|Rx - Q^T b\right\|_2^2 \longrightarrow \min$$

Lin. AGP — Lösungsvariante I: QR

Wir erinnern uns an letzte Woche:

Wiederholung: QR-Zerlegung

Eine jede Matrix $A \in \mathbb{R}^{n\times m}$ lässt sich in eine orthogonale Matrix $Q \in \mathbb{R}^{n\times n}$ und eine rechte obere Dreiecksmatrix $R \in \mathbb{R}^{n\times m}$ zerlegen, so dass

$$A = Q \cdot R.$$

Da die Multiplikation mit orthogonalen Matrizen die 2-Norm nicht ändert, können wir folgern:

$$\varphi(x) = \|Ax - b\|_2^2 = \left\|Q^T \cdot (Ax - b)\right\|_2^2 = \left\|Rx - Q^T b\right\|_2^2 \longrightarrow \min$$

Nun wissen wir aus der Problemstellung, dass $A, R \in \mathbb{R}^{n\times m}$ mit $n \geqslant m$ gilt. Damit lässt sich aber R darstellen als

$$R = \underbrace{\begin{pmatrix} \widetilde{R} \\ 0 \end{pmatrix}}_{n} \begin{matrix} \}\, m \\ \}\, n-m \end{matrix} \quad , \quad \widetilde{R} \in \mathbb{R}^{m\times m} \text{ ist obere Dreiecksmatrix.}$$

Lin. AGP — Lösungsvariante I: QR

Wir können also $\varphi(x)$ umschreiben zu

$$\varphi(x) = \left\| \begin{pmatrix} \widetilde{R}x \\ 0 \end{pmatrix} - \begin{pmatrix} \left(Q^T b\right)^{(1)} \\ \left(Q^T b\right)^{(2)} \end{pmatrix} \right\|_2^2,$$

wobei $y = Q^T b \in \mathbb{R}^n$ zerlegt wird in $y^{(1)} \in \mathbb{R}^m$ und $y^{(2)} \in \mathbb{R}^{n-m}$.

Damit gilt aber auch:

$$\varphi(x) = \underbrace{\left\| \widetilde{R}x - \left(Q^T b\right)^{(1)} \right\|_2^2}_{\geqslant 0} + \underbrace{\left\| \left(Q^T b\right)^{(2)} \right\|_2^2}_{\text{konst.}}$$

Lin. AGP — Lösungsvariante I: QR

$$\varphi(x) = \underbrace{\left\| \widetilde{R}x - \left(Q^T b\right)^{(1)} \right\|_2^2}_{\geqslant 0} + \underbrace{\left\| \left(Q^T b\right)^{(2)} \right\|_2^2}_{\text{konst.}}$$

Algorithmus 4.2 — Lösung eines lin. AGP mittels einer QR-Zerlegung

Eingabe : $A \in \mathbb{R}^{n \times m}$ mit $\operatorname{rang} A = m$ und $b \in \mathbb{R}^m$

Ausgabe : Vektor $x \in \mathbb{R}^n$, welcher das Problem $(*_{\text{ls}})$ löst.

(1) Bestimme eine QR-Zerlegung der Matrix A mit $Q \in \mathbb{R}^{n \times n}$ und $R \in \mathbb{R}^{n \times m}$ als rechte obere Dreiecksmatrix

(2) Löse das Problem $\widetilde{R}x = \left(Q^T b\right)^{(1)}$ durch Rückwärtseinsetzen.

Lin. AGP — Lösungsvariante I: QR

$$\varphi(x) = \underbrace{\left\| \widetilde{R}x - \left(Q^T b\right)^{(1)} \right\|_2^2}_{\geqslant 0} + \underbrace{\left\| \left(Q^T b\right)^{(2)} \right\|_2^2}_{\text{konst.}}$$

Mögliche Probleme

Wir sehen am Algorithmus klarerweise, dass wir ein Problem bekommen, sollte R eine trapezoide Form haben, also nicht von vollem Rang sein. In diesen Fällen ist auch die ursprüngliche Matrix A nicht von vollem Rang und wir müssen auf alternative Verfahren ausweichen.

Lin. AGP — Lösungsvariante II: Normalengleichung

Algorithmus 4.1 — Methode der kleinsten Quadrate: *least-squares*

Das Problem $(*)$ wird hier zu:

$$\boxed{\text{Suche ein } x \in \mathbb{R}^m\text{, so dass } \varphi(x) := \|r(x)\|_2^2 \text{ minimal wird.}} \qquad (*_{\mathfrak{ls}})$$

Lin. AGP — Lösungsvariante II: Normalengleichung

Algorithmus 4.1 — Methode der kleinsten Quadrate: *least-squares*

Das Problem $(*)$ wird hier zu:

$$\boxed{\text{Suche ein } x \in \mathbb{R}^m\text{, so dass } \varphi(x) := \|r(x)\|_2^2 \text{ minimal wird.}} \qquad (*_{\text{ls}})$$

Betrachten wir nun das Problem nicht algebraisch, sondern variationstechnisch, so ergibt sich, dass unsere Lösung $x \in \mathbb{R}^m$ ein Minimum von φ ist:

$$\varphi(x) = \|Ax - b\|_2^2 \leqslant \|Ay - b\|_2^2 \qquad \forall\, y \in \mathbb{R}^m$$

Lin. AGP — Lösungsvariante II: Normalengleichung

Algorithmus 4.1 — Methode der kleinsten Quadrate: *least-squares*

Das Problem $(*)$ wird hier zu:

$$\boxed{\text{Suche ein } x \in \mathbb{R}^m, \text{ so dass } \varphi(x) := \|r(x)\|_2^2 \text{ minimal wird.}} \qquad (*_{ls})$$

Betrachten wir nun das Problem nicht algebraisch, sondern variationstechnisch, so ergibt sich, dass unsere Lösung $x \in \mathbb{R}^m$ ein Minimum von φ ist:

$$\varphi(x) = \|Ax - b\|_2^2 \leqslant \|Ay - b\|_2^2 \qquad \forall\, y \in \mathbb{R}^m$$

Schauen wir nun speziell auf die Linien der Form $y = x + t \cdot z$ mit $\mathbb{R} \ni t > 0$ und $0 \neq z \in \mathbb{R}^m$, so wissen wir, dass die Funktion

$$\varphi_z \colon \mathbb{R} \to \mathbb{R}_0^+, \; t \mapsto \|A(x + tz) - b\|_2^2$$

ein Minimum bei $t = 0$ hat.

Lin. AGP — Lösungsvariante II: Normalengleichung

$$\varphi_z\colon \mathbb{R} \to \mathbb{R}_0^+,\ t \mapsto \|A(x+tz)-b\|_2^2$$

hat ein Minimum bei $t = 0$. Damit gilt für beliebiges $z \in \mathbb{R}^m$:

$$\begin{aligned}
\varphi_z(t) \quad &= \quad \|A(x+tz)-b\|_2^2 = (A(x+tz)-b)^T\,(A(x+tz)-b) \\
&= \quad \left((x+tz)^T A^T - b^T\right)(A(x+tz)-b) \\
&= \quad (x+tz)^T A^T A(x+tz) - 2(x+tz)^T A^T b + b^T b \\
&= \quad t^2\left(z^T A^T A z\right) + x^T A^T A x + 2tz^T A^T A x - 2x^T A^T b - 2tz^T A^T b + b^T b \\
&= \quad t^2\left(z^T A^T A z\right) + 2t\left(z^T A^T A x - z^T A^T b\right) + \underbrace{\left(x^T A^T A x - 2x^T A^T b + b^T b\right)}_{=:\ C_{x,b}}
\end{aligned}$$

Lin. AGP — Lösungsvariante II: Normalengleichung

$$\varphi_z \colon \mathbb{R} \to \mathbb{R}_0^+,\ t \mapsto \|A(x+tz) - b\|_2^2$$

hat ein Minimum bei $t = 0$. Damit gilt für beliebiges $z \in \mathbb{R}^m$:

$$\varphi_z(t) \quad = \quad t^2 \left(z^T A^T A z\right) + 2t \left(z^T A^T A x - z^T A^T b\right) + C_{x,b}$$

Aus der Analysis I wissen wir nun, dass für alle $z \in \mathbb{R}^m$ auch

$$0 = \frac{\partial \varphi_z}{\partial t}(0) = 2z^T \cdot \left(A^T A x - A^T b\right)$$

gelten muss und somit für unser $x \in \mathbb{R}^m$ der Zusammenhang

$$A^T A x = A^T b$$

erfüllt ist.

Lin. AGP — Lösungsvariante II: Normalengleichung

Man kann somit zeigen, dass $x \in \mathbb{R}^n$ eine Lösung von $(*_{\mathfrak{ls}})$ ist genau dann, wenn x auch die Normalengleichung erfüllt.

Definition 4.3 (Normalengleichung)

Sei ein Problem $(*_{\mathfrak{ls}})$ gegeben, so bezeichnet man die Gleichung

$$A^T A x = A^T b \tag{$G_{\mathfrak{N}}$}$$

auch als **Normalengleichung** des Problems. Die Matrix $A^T A$ ist dann quadratisch und für $\operatorname{rang}(A) = n$ auch invertierbar.

Lin. AGP — Lösungsvariante II: Normalengleichung

Man kann somit zeigen, dass $x \in \mathbb{R}^n$ eine Lösung von $(*_{\mathfrak{ls}})$ ist genau dann, wenn x auch die Normalengleichung erfüllt.

Warum die Normalengleichung nicht benutzt werden sollte

(Außer in wenigen ***trivialen*** Fällen)

Dummerweise hat die Normalengleichung eine **deutlich größere Fehlersensitivität** als ein lineares Gleichungssystem mit Systemmatrix A. So gilt bspw. im Fall $n = m$ mit invertierbarem A, dass

$$\kappa_2\left(A^T A\right) = \kappa(A)^2,$$

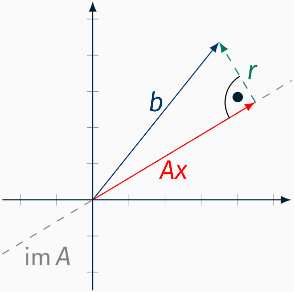
und im Allgemeinen

$$\kappa_2(A^T A) \gg \kappa(A).$$

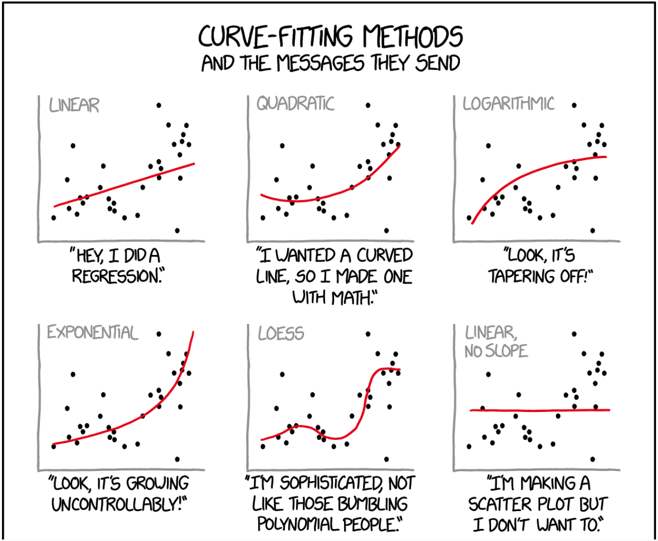
Geometrisch hat die Normalengleichung auch eine Interpretation, welche den Namen rechtfertigt: Weil

$$A^T(Ax - b) = 0$$

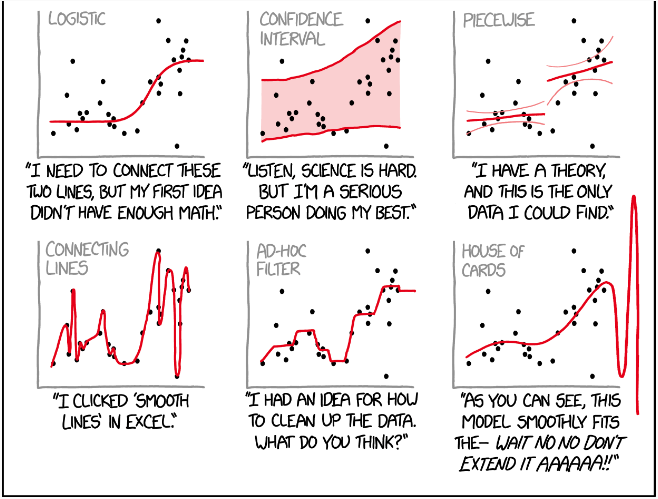
gilt, wissen wir, dass das *Residuum* $r = Ax - b$ orthogonal auf den Spalten der Matrix A steht. Damit ist es also eine Normale auf dem Bild von A ($\operatorname{im} A$).



Lineares Ausgleichsproblem — *curve fitting*



Lineares Ausgleichsproblem — *curve fitting*



Lineares Ausgleichsproblem — *curve fitting*

Wir wissen jetzt, wie wir lineare Ausgleichsprobleme lösen können, nicht jedoch wie wir auf die Matrizen für das *curve fitting* kommen.

Lineares Ausgleichsproblem — *curve fitting*

Satz 4.3 (Curve-Fitting)

Sei eine (endliche) Indexmenge I, sowie eine durch I indizierte Menge an Datenpunkten $(t_i, y_i)_{i \in I}$ gegeben, welche einem funktionalen Zusammenhang $t \mapsto f(t)$ genügen sollen. Seien des Weiteren $\varphi_1, \ldots, \varphi_m$ $(m < |I|)$ linear unabhängige Funktionen und $f(t) = \sum_{i=1}^{m} x_i \varphi_i(t)$, so erstellt man sich das passende lineare Ausgleichsproblem mit der Matrix

$$A = \begin{pmatrix} \varphi_1(t_1) & \ldots & \varphi_m(t_1) \\ \vdots & \ddots & \vdots \\ \varphi_1(t_n) & \ldots & \varphi_m(t_n) \end{pmatrix} \in \mathbb{R}^{n \times m},$$

dem Ergebnisvektor

$$b = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^n \quad \text{, und dem Lösungsvektor} \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} \in \mathbb{R}^m.$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2,-6)$, $(-2,4)$, $(-1,-2)$, $(1,2)$, $(2,4)$, $(4,6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} \varphi_1(t_1) & \varphi_2(t_1) & \varphi_3(t_1) & \varphi_4(t_1) \\ \varphi_1(t_2) & \varphi_2(t_2) & \varphi_3(t_2) & \varphi_4(t_2) \\ \varphi_1(t_3) & \varphi_2(t_3) & \varphi_3(t_3) & \varphi_4(t_3) \\ \varphi_1(t_4) & \varphi_2(t_4) & \varphi_3(t_4) & \varphi_4(t_4) \\ \varphi_1(t_5) & \varphi_2(t_5) & \varphi_3(t_5) & \varphi_4(t_5) \\ \varphi_1(t_6) & \varphi_2(t_6) & \varphi_3(t_6) & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2, -6)$, $(-2, 4)$, $(-1, -2)$, $(1, 2)$, $(2, 4)$, $(4, 6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} \varphi_1(t_1) & \varphi_2(t_1) & \varphi_3(t_1) & \varphi_4(t_1) \\ \varphi_1(t_2) & \varphi_2(t_2) & \varphi_3(t_2) & \varphi_4(t_2) \\ \varphi_1(t_3) & \varphi_2(t_3) & \varphi_3(t_3) & \varphi_4(t_3) \\ \varphi_1(t_4) & \varphi_2(t_4) & \varphi_3(t_4) & \varphi_4(t_4) \\ \varphi_1(t_5) & \varphi_2(t_5) & \varphi_3(t_5) & \varphi_4(t_5) \\ \varphi_1(t_6) & \varphi_2(t_6) & \varphi_3(t_6) & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2,-6)$, $(-2,4)$, $(-1,-2)$, $(1,2)$, $(2,4)$, $(4,6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & \varphi_2(t_1) & \varphi_3(t_1) & \varphi_4(t_1) \\ 1 & \varphi_2(t_2) & \varphi_3(t_2) & \varphi_4(t_2) \\ 1 & \varphi_2(t_3) & \varphi_3(t_3) & \varphi_4(t_3) \\ 1 & \varphi_2(t_4) & \varphi_3(t_4) & \varphi_4(t_4) \\ 1 & \varphi_2(t_5) & \varphi_3(t_5) & \varphi_4(t_5) \\ 1 & \varphi_2(t_6) & \varphi_3(t_6) & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $-2, -6), (-2, 4), (-1, -2), (1, 2),$ $(2, 4), (4, 6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & \varphi_2(t_1) & \varphi_3(t_1) & \varphi_4(t_1) \\ 1 & \varphi_2(t_2) & \varphi_3(t_2) & \varphi_4(t_2) \\ 1 & \varphi_2(t_3) & \varphi_3(t_3) & \varphi_4(t_3) \\ 1 & \varphi_2(t_4) & \varphi_3(t_4) & \varphi_4(t_4) \\ 1 & \varphi_2(t_5) & \varphi_3(t_5) & \varphi_4(t_5) \\ 1 & \varphi_2(t_6) & \varphi_3(t_6) & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Seien beispielsweise die Punkte $(-2,-6)$, $(-2,4)$, $(-1,-2)$, $(1,2)$, $(2,4)$, $(4,6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & \varphi_3(t_1) & \varphi_4(t_1) \\ 1 & \varphi_2(t_2) & \varphi_3(t_2) & \varphi_4(t_2) \\ 1 & \varphi_2(t_3) & \varphi_3(t_3) & \varphi_4(t_3) \\ 1 & \varphi_2(t_4) & \varphi_3(t_4) & \varphi_4(t_4) \\ 1 & \varphi_2(t_5) & \varphi_3(t_5) & \varphi_4(t_5) \\ 1 & \varphi_2(t_6) & \varphi_3(t_6) & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2, -6)$, $(-2, 4)$, $(-1, -2)$, $(1, 2)$, $(2, 4)$, $(4, 6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & \varphi_3(t_1) & \varphi_4(t_1) \\ 1 & -2 & \varphi_3(t_2) & \varphi_4(t_2) \\ 1 & \varphi_2(t_3) & \varphi_3(t_3) & \varphi_4(t_3) \\ 1 & \varphi_2(t_4) & \varphi_3(t_4) & \varphi_4(t_4) \\ 1 & \varphi_2(t_5) & \varphi_3(t_5) & \varphi_4(t_5) \\ 1 & \varphi_2(t_6) & \varphi_3(t_6) & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2,-6)$, $(-2,4)$, $(-1,-2)$, $(1,2)$, $(2,4)$, $(4,6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & \varphi_3(t_1) & \varphi_4(t_1) \\ 1 & -2 & \varphi_3(t_2) & \varphi_4(t_2) \\ 1 & -1 & \varphi_3(t_3) & \varphi_4(t_3) \\ 1 & \varphi_2(t_4) & \varphi_3(t_4) & \varphi_4(t_4) \\ 1 & \varphi_2(t_5) & \varphi_3(t_5) & \varphi_4(t_5) \\ 1 & \varphi_2(t_6) & \varphi_3(t_6) & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2, -6)$, $(-2, 4)$, $(-1, -2)$, $(1, 2)$, $(2, 4)$, $(4, 6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & \varphi_3(t_1) & \varphi_4(t_1) \\ 1 & -2 & \varphi_3(t_2) & \varphi_4(t_2) \\ 1 & -1 & \varphi_3(t_3) & \varphi_4(t_3) \\ 1 & 1 & \varphi_3(t_4) & \varphi_4(t_4) \\ 1 & \varphi_2(t_5) & \varphi_3(t_5) & \varphi_4(t_5) \\ 1 & \varphi_2(t_6) & \varphi_3(t_6) & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2, -6)$, $(-2, 4)$, $(-1, -2)$, $(1, 2)$, $(2, 4)$, $(4, 6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & \varphi_3(t_1) & \varphi_4(t_1) \\ 1 & -2 & \varphi_3(t_2) & \varphi_4(t_2) \\ 1 & -1 & \varphi_3(t_3) & \varphi_4(t_3) \\ 1 & 1 & \varphi_3(t_4) & \varphi_4(t_4) \\ 1 & 2 & \varphi_3(t_5) & \varphi_4(t_5) \\ 1 & \varphi_2(t_6) & \varphi_3(t_6) & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2,-6)$, $(-2,4)$, $(-1,-2)$, $(1,2)$, $(2,4)$, $(4,6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & \varphi_3(t_1) & \varphi_4(t_1) \\ 1 & -2 & \varphi_3(t_2) & \varphi_4(t_2) \\ 1 & -1 & \varphi_3(t_3) & \varphi_4(t_3) \\ 1 & 1 & \varphi_3(t_4) & \varphi_4(t_4) \\ 1 & 2 & \varphi_3(t_5) & \varphi_4(t_5) \\ 1 & 4 & \varphi_3(t_6) & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2, -6)$, $(-2, 4)$, $(-1, -2)$, $(1, 2)$, $(2, 4)$, $(4, 6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & \varphi_3(t_1) & \varphi_4(t_1) \\ 1 & -2 & \varphi_3(t_2) & \varphi_4(t_2) \\ 1 & -1 & \varphi_3(t_3) & \varphi_4(t_3) \\ 1 & 1 & \varphi_3(t_4) & \varphi_4(t_4) \\ 1 & 2 & \varphi_3(t_5) & \varphi_4(t_5) \\ 1 & 4 & \varphi_3(t_6) & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2, -6)$, $(-2, 4)$, $(-1, -2)$, $(1, 2)$, $(2, 4)$, $(4, 6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & 4 & \varphi_4(t_1) \\ 1 & -2 & \varphi_3(t_2) & \varphi_4(t_2) \\ 1 & -1 & \varphi_3(t_3) & \varphi_4(t_3) \\ 1 & 1 & \varphi_3(t_4) & \varphi_4(t_4) \\ 1 & 2 & \varphi_3(t_5) & \varphi_4(t_5) \\ 1 & 4 & \varphi_3(t_6) & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2, -6)$, $(-2, 4)$, $-1, -2)$, $(1, 2)$, $(2, 4)$, $(4, 6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & 4 & \varphi_4(t_1) \\ 1 & -2 & 4 & \varphi_4(t_2) \\ 1 & -1 & \varphi_3(t_3) & \varphi_4(t_3) \\ 1 & 1 & \varphi_3(t_4) & \varphi_4(t_4) \\ 1 & 2 & \varphi_3(t_5) & \varphi_4(t_5) \\ 1 & 4 & \varphi_3(t_6) & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Seien beispielsweise die Punkte $(-2,-6)$, $(-2,4)$, $(-1,-2)$, $(1,2)$, $(2,4)$, $(4,6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & 4 & \varphi_4(t_1) \\ 1 & -2 & 4 & \varphi_4(t_2) \\ 1 & -1 & 1 & \varphi_4(t_3) \\ 1 & 1 & \varphi_3(t_4) & \varphi_4(t_4) \\ 1 & 2 & \varphi_3(t_5) & \varphi_4(t_5) \\ 1 & 4 & \varphi_3(t_6) & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2,-6)$, $(-2,4)$, $(-1,-2)$, $(1,2)$, $(2,4)$, $(4,6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & 4 & \varphi_4(t_1) \\ 1 & -2 & 4 & \varphi_4(t_2) \\ 1 & -1 & 1 & \varphi_4(t_3) \\ 1 & 1 & 1 & \varphi_4(t_4) \\ 1 & 2 & \varphi_3(t_5) & \varphi_4(t_5) \\ 1 & 4 & \varphi_3(t_6) & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2,-6)$, $(-2,4)$, $(-1,-2)$, $(1,2)$, $(2,4)$, $4,6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & 4 & \varphi_4(t_1) \\ 1 & -2 & 4 & \varphi_4(t_2) \\ 1 & -1 & 1 & \varphi_4(t_3) \\ 1 & 1 & 1 & \varphi_4(t_4) \\ 1 & 2 & 4 & \varphi_4(t_5) \\ 1 & 4 & \varphi_3(t_6) & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2,-6)$, $(-2,4)$, $(-1,-2)$, $(1,2)$, $(2,4)$, $(4,6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & 4 & \varphi_4(t_1) \\ 1 & -2 & 4 & \varphi_4(t_2) \\ 1 & -1 & 1 & \varphi_4(t_3) \\ 1 & 1 & 1 & \varphi_4(t_4) \\ 1 & 2 & 4 & \varphi_4(t_5) \\ 1 & 4 & 16 & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $-2, -6)$, $(-2, 4)$, $(-1, -2)$, $(1, 2)$, $(2, 4)$, $(4, 6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & 4 & \varphi_4(t_1) \\ 1 & -2 & 4 & \varphi_4(t_2) \\ 1 & -1 & 1 & \varphi_4(t_3) \\ 1 & 1 & 1 & \varphi_4(t_4) \\ 1 & 2 & 4 & \varphi_4(t_5) \\ 1 & 4 & 16 & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2,-6)$, $(-2,4)$, $(-1,-2)$, $(1,2)$, $(2,4)$, $(4,6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & 4 & -8 \\ 1 & -2 & 4 & \varphi_4(t_2) \\ 1 & -1 & 1 & \varphi_4(t_3) \\ 1 & 1 & 1 & \varphi_4(t_4) \\ 1 & 2 & 4 & \varphi_4(t_5) \\ 1 & 4 & 16 & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2,-6)$, $(-2,4)$, $(-1,-2)$, $(1,2)$, $(2,4)$, $(4,6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & 4 & -8 \\ 1 & -2 & 4 & -8 \\ 1 & -1 & 1 & \varphi_4(t_3) \\ 1 & 1 & 1 & \varphi_4(t_4) \\ 1 & 2 & 4 & \varphi_4(t_5) \\ 1 & 4 & 16 & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2, -6)$, $(-2, 4)$, $(-1, -2)$, $(1, 2)$, $(2, 4)$, $(4, 6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & 4 & -8 \\ 1 & -2 & 4 & -8 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & \varphi_4(t_4) \\ 1 & 2 & 4 & \varphi_4(t_5) \\ 1 & 4 & 16 & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Seien beispielsweise die Punkte $(-2,-6)$, $(-2,4)$, $(-1,-2)$, $(1,2)$, $(2,4)$, $(4,6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & 4 & -8 \\ 1 & -2 & 4 & -8 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & \varphi_4(t_5) \\ 1 & 4 & 16 & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2,-6)$, $(-2,4)$, $(-1,-2)$, $(1,2)$, $(2,4)$, $4,6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & 4 & -8 \\ 1 & -2 & 4 & -8 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 4 & 16 & \varphi_4(t_6) \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2, -6)$, $(-2, 4)$, $(-1, -2)$, $(1, 2)$, $(2, 4)$, $(4, 6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & 4 & -8 \\ 1 & -2 & 4 & -8 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 4 & 16 & 64 \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2, -6)$, $(-2, 4)$, $(-1, -2)$, $(1, 2)$, $(2, 4)$, $(4, 6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & 4 & -8 \\ 1 & -2 & 4 & -8 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 4 & 16 & 64 \end{pmatrix} \qquad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2, -6)$, $(-2, 4)$, $(-1, -2)$, $(1, 2)$, $(2, 4)$, $(4, 6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & 4 & -8 \\ 1 & -2 & 4 & -8 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 4 & 16 & 64 \end{pmatrix} \qquad b = \begin{pmatrix} -6 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2, -6)$, $(-2, 4)$, $(-1, -2)$, $(1, 2)$, $(2, 4)$, $(4, 6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & 4 & -8 \\ 1 & -2 & 4 & -8 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 4 & 16 & 64 \end{pmatrix} \qquad b = \begin{pmatrix} -6 \\ 4 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2,-6)$, $(-2,4)$, $(-1,-2)$, $(1,2)$, $(2,4)$, $(4,6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & 4 & -8 \\ 1 & -2 & 4 & -8 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 4 & 16 & 64 \end{pmatrix} \qquad b = \begin{pmatrix} -6 \\ 4 \\ -2 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2, -6)$, $(-2, 4)$, $(-1, -2)$, $(1, 2)$, $(2, 4)$, $(4, 6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & 4 & -8 \\ 1 & -2 & 4 & -8 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 4 & 16 & 64 \end{pmatrix} \qquad b = \begin{pmatrix} -6 \\ 4 \\ -2 \\ 4 \\ y_5 \\ y_6 \end{pmatrix}$$

Seien beispielsweise die Punkte $(-2,-6)$, $(-2,4)$, $(-1,-2)$, $(1,2)$, $(2,4)$, $(4,6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

$$A = \begin{pmatrix} 1 & -2 & 4 & -8 \\ 1 & -2 & 4 & -8 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 4 & 16 & 64 \end{pmatrix} \qquad b = \begin{pmatrix} -6 \\ 4 \\ -2 \\ 4 \\ 4 \\ y_6 \end{pmatrix}$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel

Seien beispielsweise die Punkte $(-2,-6)$, $(-2,4)$, $(-1,-2)$, $(1,2)$, $(2,4)$, $(4,6)$ gegeben, wir wollen ein Polynom vom Grad 3 durch diese Punkte legen. Wir stellen also die Systemmatrix A, sowie den Ergebnisvektor b für das Polynom

$$f(t) = a_4 \cdot \underbrace{t^3}_{\varphi_4} + a_3 \cdot \underbrace{t^2}_{\varphi_3} + a_2 \cdot \underbrace{t}_{\varphi_2} + a_1 \cdot \underbrace{1}_{\varphi_1}$$

auf:

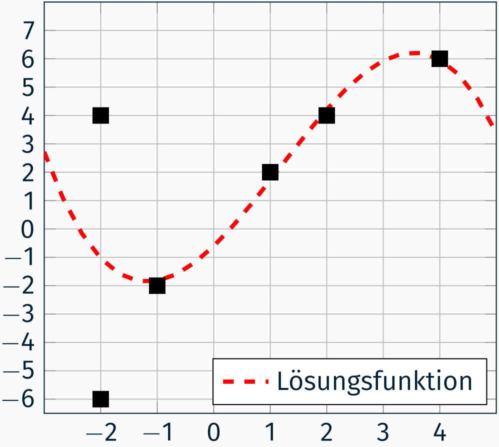
$$A = \begin{pmatrix} 1 & -2 & 4 & -8 \\ 1 & -2 & 4 & -8 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 4 & 16 & 64 \end{pmatrix} \qquad b = \begin{pmatrix} -6 \\ 4 \\ -2 \\ 4 \\ 4 \\ 6 \end{pmatrix}$$

Damit ergibt sich das zu lösende Gleichungssystem

$$Ax = b \qquad \text{mit} \qquad x = \begin{pmatrix} a_1 \,, a_2 \,, a_3 \,, a_4 \end{pmatrix}^T .$$

Lineares Ausgleichsproblem — *curve fitting*: Beispiel (II)

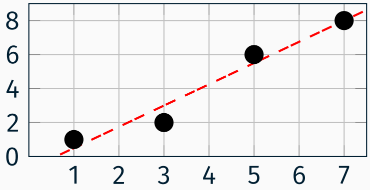
Skizziert man die Punkte in ein Koordinatensystem, so erhält man:



Aufgabe 1 — Auslgeichsgerade

Aufgabe 1 — Auslgeichsgerade

a) Bestimmen Sie die Ausgleichsgerade $g(x) = a_1 + a_2 \cdot x$ mit Hilfe der Normalengleichung für die Punkte $(1,1)$, $(3,2)$, $(5,6)$ und $(7,8)$.



Rekaptiulation — Effizientes Speichern von Matrizen

Speichern einer Matrix

Gegeben sei die *dünnbesetzte* Matrix

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix},$$

welche nun sinnvoll abgespeichert werden soll.

Speichern einer Matrix

Gegeben sei die *dünnbesetzte* Matrix

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix},$$

welche nun sinnvoll abgespeichert werden soll.

Idee I — Quadratisches Array

Wir legen ein zweidimensionales, quadratisches Array `double[][] A` an, brauchen dafür aber auch $16 \cdot$ `sizeof(double)` Byte Speicherplatz.

Speichern einer Matrix

Gegeben sei die *dünnbesetzte* Matrix

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix},$$

welche nun sinnvoll abgespeichert werden soll.

Idee I — Quadratisches Array

Wir legen ein zweidimensionales, quadratisches Array `double[][] A` an, brauchen dafür aber auch $16 \cdot$ `sizeof(double)` Byte Speicherplatz.
Bei nur vier genutzten Werten ergibt dies einen Nutzungsgrad von nur 25%!
Vor allem bei großen Matrizen $(n \gtrsim 10^6)$ ist dies nicht mehr vorzustellen!

Speichern einer Matrix

Gegeben sei die *dünnbesetzte* Matrix

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix},$$

welche nun sinnvoll abgespeichert werden soll.

Idee II — Abspeichern der Werte $\neq 0$

Wir könnten in einem Feld `val` die Nichtnullwerte der ursprünglichen Matrix abspeichern.

Speichern einer Matrix

Gegeben sei die *dünnbesetzte* Matrix

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix},$$

welche nun sinnvoll abgespeichert werden soll.

Idee II — Abspeichern der Werte $\neq 0$

Wir könnten in einem Feld `val` die Nichtnullwerte der ursprünglichen Matrix abspeichern.
Problem: Wie wäre eine Rekonstruktion möglich? In anderen Worten, woher wissen wir an welchen Stellen die Werte gestanden sind?

Speichern einer Matrix — RaCS

Algorithmus 4.3 — RaCS

Sei die Matrix A gegeben, so speichert man bei dieser Methode **drei** Felder:

- `val` — Das Wertefeld, welches die Nichtnullwerte der Matrix enthält.
- `col_ind` — Das Spaltenindexfeld, welches die Spaltenindizes der Werte enthält.
- `row_ind` — Das Zeilenindexfeld, welches die Zeilenindizes der Werte enthält.

Dabei gilt dann

$$\texttt{val}_k = A_{i,j} :\Leftrightarrow \texttt{col\_ind}_k = j \wedge \texttt{row\_ind}_k = i.$$

Speichern einer Matrix — RaCS

Algorithmus 4.3 — RaCS

Sei die Matrix A gegeben, so speichert man bei dieser Methode **drei** Felder:

- `val` — Das Wertefeld, welches die Nichtnullwerte der Matrix enthält.
- `col_ind` — Das Spaltenindexfeld, welches die Spaltenindizes der Werte enthält.
- `row_ind` — Das Zeilenindexfeld, welches die Zeilenindizes der Werte enthält.

Dabei gilt dann

$$\texttt{val}_k = A_{i,j} :\Leftrightarrow \texttt{col\_ind}_k = j \wedge \texttt{row\_ind}_k = i.$$

Ein kleines Beispiel:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \texttt{val} &= [\,?\,,\,?\,,\,?\,,\,?\,] \\ \texttt{col\_ind} &= [\,?\,,\,?\,,\,?\,,\,?\,] \\ \texttt{row\_ind} &= [\,?\,,\,?\,,\,?\,,\,?\,] \end{aligned}$$

Speichern einer Matrix — RaCS

Algorithmus 4.3 — RaCS

Sei die Matrix A gegeben, so speichert man bei dieser Methode **drei** Felder:

- `val` — Das Wertefeld, welches die Nichtnullwerte der Matrix enthält.
- `col_ind` — Das Spaltenindexfeld, welches die Spaltenindizes der Werte enthält.
- `row_ind` — Das Zeilenindexfeld, welches die Zeilenindizes der Werte enthält.

Dabei gilt dann

$$\texttt{val}_k = A_{i,j} :\Leftrightarrow \texttt{col\_ind}_k = j \land \texttt{row\_ind}_k = i.$$

Ein kleines Beispiel:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ \boxed{5} & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \texttt{val} &= [\,\boxed{5}, ?, ?, ?\,] \\ \texttt{col\_ind} &= [\,\boxed{1}, ?, ?, ?\,] \\ \texttt{row\_ind} &= [\,\boxed{2}, ?, ?, ?\,] \end{aligned}$$

Speichern einer Matrix — RaCS

Algorithmus 4.3 — RaCS

Sei die Matrix A gegeben, so speichert man bei dieser Methode **drei** Felder:

- `val` — Das Wertefeld, welches die Nichtnullwerte der Matrix enthält.
- `col_ind` — Das Spaltenindexfeld, welches die Spaltenindizes der Werte enthält.
- `row_ind` — Das Zeilenindexfeld, welches die Zeilenindizes der Werte enthält.

Dabei gilt dann

$$\texttt{val}_k = A_{i,j} :\Leftrightarrow \texttt{col\_ind}_k = j \wedge \texttt{row\_ind}_k = i.$$

Ein kleines Beispiel:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \texttt{val} &= [\,5, 8, ?, ?\,] \\ \texttt{col\_ind} &= [\,1, 2, ?, ?\,] \\ \texttt{row\_ind} &= [\,2, 2, ?, ?\,] \end{aligned}$$

Speichern einer Matrix — RaCS

Algorithmus 4.3 — RaCS

Sei die Matrix A gegeben, so speichert man bei dieser Methode **drei** Felder:

- `val` — Das Wertefeld, welches die Nichtnullwerte der Matrix enthält.
- `col_ind` — Das Spaltenindexfeld, welches die Spaltenindizes der Werte enthält.
- `row_ind` — Das Zeilenindexfeld, welches die Zeilenindizes der Werte enthält.

Dabei gilt dann

$$\texttt{val}_k = A_{i,j} :\Leftrightarrow \texttt{col\_ind}_k = j \wedge \texttt{row\_ind}_k = i.$$

Ein kleines Beispiel:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \texttt{val} &= [\,5, 8, 3, ?\,] \\ \texttt{col\_ind} &= [\,1, 2, 3, ?\,] \\ \texttt{row\_ind} &= [\,2, 2, 3, ?\,] \end{aligned}$$

Speichern einer Matrix — RaCS

Algorithmus 4.3 — RaCS

Sei die Matrix A gegeben, so speichert man bei dieser Methode **drei** Felder:

- `val` — Das Wertefeld, welches die Nichtnullwerte der Matrix enthält.
- `col_ind` — Das Spaltenindexfeld, welches die Spaltenindizes der Werte enthält.
- `row_ind` — Das Zeilenindexfeld, welches die Zeilenindizes der Werte enthält.

Dabei gilt dann

$$\texttt{val}_k = A_{i,j} :\Leftrightarrow \texttt{col\_ind}_k = j \land \texttt{row\_ind}_k = i.$$

Ein kleines Beispiel:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \texttt{val} &= [\,5, 8, 3, 6\,] \\ \texttt{col\_ind} &= [\,1, 2, 3, 2\,] \\ \texttt{row\_ind} &= [\,2, 2, 3, 4\,] \end{aligned}$$

Speichern einer Matrix — CRS

Algorithmus 4.4 — CRS

Sei die Matrix A gegeben, so speichert man bei dieser Methode **drei** Felder:

- `val` — Das Wertefeld, welches die Nichtnullwerte der Matrix enthält.
- `col_ind` — Das Spaltenindexfeld, welches die Spaltenindizes der Werte enthält.
- `row_ptr` — Dieses Feld enthält Zeiger auf die anderen Felder, ab welchen sich der Zeilenindex erhöht. Dieses Feld enthält (#Zeilen $+ 1$) Einträge, der letzte ist (#NZV $+ 1$).

Dabei gilt dann

$$\texttt{val}_k = A_{i,j} :\Leftrightarrow \texttt{col\_ind}_k = j \wedge \texttt{row\_ptr}_i \leqslant k < \texttt{row\_ptr}_{i+1}.$$

Speichern einer Matrix — CRS

Ein kleines Beispiel:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \texttt{val} &= \left[\ 5\,,8\,,3\,,6\ \right] \\ \texttt{col\_ind} &= \left[\ 1\,,2\,,3\,,2\ \right] \\ \texttt{row\_ptr} &= \left[\ ?\,,?\,,?\,,?\,,?\ \right] \end{aligned}$$

Speichern einer Matrix — CRS

Ein kleines Beispiel:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \texttt{val} &= \left[\ 5\,,8\,,3\,,6\ \right] \\ \texttt{col\_ind} &= \left[\ 1\,,2\,,3\,,2\ \right] \\ \texttt{row\_ptr} &= \left[\ 1\,,?\,,?\,,?\,,5\ \right] \end{aligned}$$

Speichern einer Matrix — CRS

Ein kleines Beispiel:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \texttt{val} &= \left[\,5\,,8\,,3\,,6\,\right] \\ \texttt{col\_ind} &= \left[\,1\,,2\,,3\,,2\,\right] \\ \texttt{row\_ptr} &= \left[\,1\,,1\,,?\,,?\,,5\,\right] \end{aligned}$$

Speichern einer Matrix — CRS

Ein kleines Beispiel:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \texttt{val} &= \left[\,5\,,8\,,3\,,6\,\right] \\ \texttt{col\_ind} &= \left[\,1\,,2\,,3\,,2\,\right] \\ \texttt{row\_ptr} &= \left[\,1\,,1\,,3\,,?\,,5\,\right] \end{aligned}$$

Speichern einer Matrix — CRS

Ein kleines Beispiel:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \texttt{val} &= \left[\, 5\,, 8\,, 3\,, 6 \,\right] \\ \texttt{col\_ind} &= \left[\, 1\,, 2\,, 3\,, 2 \,\right] \\ \texttt{row\_ptr} &= \left[\, 1\,, 1\,, 3\,, 4\,, 5 \,\right] \end{aligned}$$

Speichern einer Matrix — CCS

Algorithmus 4.5 — CCS

Sei die Matrix A gegeben, so speichert man bei dieser Methode **drei** Felder:

- `val` — Das Wertefeld, welches die Nichtnullwerte der Matrix enthält.
- `row_ind` — Das Zeilenindexfeld, welches die Zeilenindizes der Werte enthält.
- `col_ptr` — Dieses Feld enthält Zeiger auf die anderen Felder, ab welchen sich der Spaltenindex erhöht. Dieses Feld enthält $(\#\text{Spalten} + 1)$ Einträge, der letzte ist $(\#\text{NZV} + 1)$.

Dabei gilt dann

$$\mathtt{val}_k = A_{i,j} :\Leftrightarrow \mathtt{row\_ind}_k = i \wedge \mathtt{col\_ptr}_j \leqslant k < \mathtt{col\_ptr}_{j+1}.$$

Speichern einer Matrix — CCS

Ein kleines Beispiel:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \texttt{val} &= \left[\ 5\,,8\,,{\color{red}6}\,,{\color{red}3}\ \right] \\ \texttt{row\_ind} &= \left[\ 2\,,2\,,{\color{red}4}\,,{\color{red}3}\ \right] \\ \texttt{col\_ptr} &= \left[\ ?\,,?\,,?\,,?\,,?\ \right] \end{aligned}$$

Speichern einer Matrix — CCS

Ein kleines Beispiel:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \texttt{val} &= \left[\, 5\,,8\,,6\,,3 \,\right] \\ \texttt{row\_ind} &= \left[\, 2\,,2\,,4\,,3 \,\right] \\ \texttt{col\_ptr} &= \left[\, 1\,,?\,,?\,,?\,,5 \,\right] \end{aligned}$$

Speichern einer Matrix — CCS

Ein kleines Beispiel:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \texttt{val} &= \left[\, 5\,,8\,,6\,,3 \,\right] \\ \texttt{row\_ind} &= \left[\, 2\,,2\,,4\,,3 \,\right] \\ \texttt{col\_ptr} &= \left[\, 1\,,2\,,?\,,?\,,5 \,\right] \end{aligned}$$

Speichern einer Matrix — CCS

Ein kleines Beispiel:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \texttt{val} &= \left[\, 5\,,8\,,6\,,3 \,\right] \\ \texttt{row\_ind} &= \left[\, 2\,,2\,,4\,,3 \,\right] \\ \texttt{col\_ptr} &= \left[\, 1\,,2\,,4\,,?\,,5 \,\right] \end{aligned}$$

Speichern einer Matrix — CCS

Ein kleines Beispiel:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \texttt{val} &= \left[\ 5\,,8\,,6\,,3\ \right] \\ \texttt{row\_ind} &= \left[\ 2\,,2\,,4\,,3\ \right] \\ \texttt{col\_ptr} &= \left[\ 1\,,2\,,4\,,5\,,5\ \right] \end{aligned}$$

Speichern einer Matrix — BCRS

Algorithmus 4.6 — BCRS

Sei die Matrix A und eine Blockgröße n gegeben, so speichert man bei dieser Methode **drei** Felder:

- `val` — Das Wertefeld, welches die Nichtnullblöcke der Größe n der Matrix enthält.
- `col_ind` — Das Spaltenindexfeld, welches die Spaltenindizes der Matrixblöcke enthält.
- `row_ptr` — Dieses Feld enthält Zeiger auf die anderen Felder, ab welchen sich der Zeilenindex erhöht. Dieses Feld enthält $(\#\text{Zeilen} + 1)$ Einträge, der letzte ist $(\#\text{NZV} + 1)$.

Im Prinzip identisch zu CRS, man betrachtet nun Matrixblöcke.

Speichern einer Matrix — BCRS

Ein kleines Beispiel:

$$\begin{pmatrix} 4 & 3 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 6 \end{pmatrix}$$

$$\begin{aligned} \texttt{val} &= \left[\ ?\ ,\ ?\ \right] \\ \texttt{col\_ind} &= \left[\ ?\ ,\ ?\ \right] \\ \texttt{row\_ptr} &= \left[\ ?\ ,\ ?\ ,\ ?\ \right] \end{aligned}$$

Speichern einer Matrix — BCRS

Ein kleines Beispiel:

$$\begin{pmatrix} 4 & 3 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 6 \end{pmatrix}$$

$$\begin{aligned} \texttt{val} &= \left[\begin{pmatrix} 4 & 3 \\ 5 & 8 \end{pmatrix}, ? \right] \\ \texttt{col\_ind} &= \left[1, ? \right] \\ \texttt{row\_ptr} &= \left[1, ?, 3 \right] \end{aligned}$$

Speichern einer Matrix — BCRS

Ein kleines Beispiel:

$$\begin{pmatrix} 4 & 3 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 6 \end{pmatrix}$$

$$\begin{aligned} \texttt{val} &= \left[\begin{pmatrix} 4 & 3 \\ 5 & 8 \end{pmatrix}, \begin{pmatrix} 3 & 0 \\ 0 & 6 \end{pmatrix} \right] \\ \texttt{col\_ind} &= \left[1, 2 \right] \\ \texttt{row\_ptr} &= \left[1, 2, 3 \right] \end{aligned}$$

Aufgabe 1 — Matrixformate

Aufgabe 1 — Matrixformate

b) Speichern Sie die folgende Matrix im CCS-Format ab. Die Indizierung beginnt bei 1:

$$A = \begin{pmatrix} 1 & 0 & 3 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 5 & 0 & 0 & 8 \\ 0 & 1 & 0 & 0 & 0 & 4 \\ 1 & 2 & 7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Hausaufgabe — Ausgleichspolynom und Matrixformate

Hausaufgabe — Ausgleichspolynom und Matrixformate

Diese Hausaufgabe findet sich **komplett** auf StudOn!

Zusätzliche Literatur

Zusätzliche Literatur zur 4. Übung (1)

[1] V. Eijkhout. „LAPACK working note 50 Distributed Sparse Data Structures for Linear Algebra Operations“. In: (Okt. 1994).

[2] W. Ford. *Numerical Linear Algebra with Applications*. Elsevier, 2015. DOI: 10.1016/c2011-0-07533-6. URL: https://doi.org/10.1016/c2011-0-07533-6.

[3] C. F. Gauss. *Theoria Motus Corporum Coelestium*. Springer Berlin Heidelberg, 1981. DOI: 10.1007/978-3-642-92478-1. URL: https://doi.org/10.1007/978-3-642-92478-1.

[4] C. Gauss. *Disquisitio de elementis ellipticis palladis ex oppositionibus annorum 1803, 1804, 1805, 1807, 1808, 1809.* 1810. URL: https://books.google.de/books?id=j21FGwAACAAJ.

[5] G. H. Golub und C. F. van Loan. *Matrix Computations*. Fourth. JHU Press, 2013. ISBN: 9781421407944. URL: http://www.cs.cornell.edu/cv/GVL4/golubandvanloan.htm.

Zusätzliche Literatur zur 4. Übung (2)

[6] P. G. Guest. *Numerical methods of curve fitting / by P.G. Guest*. English. First paperback edition. Cambridge University Press Cambridge, England, 2013 1961, xiv, 422 pages ; ISBN: 9781107646957 1107646952.

[7] A. M. Legendre. *Nouvelles methodes pour la determination des orbites des cometes [microform] / par A.M. Legendre*. French. F. Didot Paris, 1805, viii, 80 p., [1] leaf of plates :

[8] S. Roman. *Advanced Linear Algebra*. Springer New York, 2008. DOI: 10.1007/978-0-387-72831-5. URL: https://doi.org/10.1007/978-0-387-72831-5.

[9] Y. Saad. *SPARSKIT: a basic tool kit for sparse matrix computations - Version 2*. 1994. URL: https://www-users.cs.umn.edu/~saad/software/SPARSKIT/.

Zusätzliche Literatur zur 4. Übung (3)

[10] S. M. Stigler. „Gauss and the Invention of Least Squares“. In: *The Annals of Statistics* 9.3 (1981), S. 465 –474. DOI: 10.1214/aos/1176345451. URL: https://doi.org/10.1214/aos/1176345451.

[11] G. Strang. *Linear algebra and learning from data*. Wellesley, MA: Wellesley-Cambridge Press, 2019. ISBN: 978-06921963-8-0.

[12] R. Tewarson. *Sparse Matrices*. Mathematics in science and engineering : a series of monographs and textbooks. Academic Press, 1973. ISBN: 9780126856507. URL: https://www.sciencedirect.com/bookseries/mathematics-in-science-and-engineering/vol/99.

[13] H. Wolf. *Ausgleichungsrechnung: Formeln zur praktischen Anwendung*. Ausgleichungsrechnung / Helmut Wolf. Dümmler, 1962. ISBN: 9783427783510. URL: https://books.google.de/books?id=9b1PAQAAIAAJ.

Anhang zur Übung 4

Einschub zur Übung 4: Die Cramer'sche Regel

Satz 4.A (Cramer'sche Regel)

Sei ein lineares Gleichungssystem $Ax = b$ mit $A \in \mathbb{R}^{n \times n}$, $x, b \in \mathbb{R}^n$ gegeben. Sei außerdem A nichtsingulär, also $\det(A) \neq 0$. Dann können wir die Lösung angeben durch:

$$x_i = \frac{\det(A_i)}{\det(A)} \qquad \text{für } 1 \leqslant i \leqslant n$$

A_i beschreibt dabei die Matrix, welche durch Ersetzen der i-ten Spalte mit der rechten Seite b entsteht:

$$A_i = \begin{pmatrix} a_{1,1} & \dots & a_{1,i-1} & b_1 & a_{1,i+1} & \dots & a_{1,n} \\ a_{2,1} & \dots & a_{2,i-1} & b_2 & a_{2,i+1} & \dots & a_{2,n} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & \dots & a_{n,i-1} & b_n & a_{n,i+1} & \dots & a_{n,n} \end{pmatrix}$$

Einschub zur Übung 4: Die Cramer'sche Regel

Satz 4.A (Cramer'sche Regel)

Sei ein lineares Gleichungssystem $Ax = b$ mit $A \in \mathbb{R}^{n \times n}$, $x, b \in \mathbb{R}^n$ gegeben. Sei außerdem A nichtsingulär, also $\det(A) \neq 0$. Dann können wir die Lösung angeben durch:

$$x_i = \frac{\det(A_i)}{\det(A)} \qquad \text{für } 1 \leqslant i \leqslant n$$

A_i beschreibt dabei die Matrix, welche durch Ersetzen der i-ten Spalte mit der rechten Seite b entsteht:

$$A_i = \begin{pmatrix} a_{1,1} & \dots & a_{1,i-1} & b_1 & a_{1,i+1} & \dots & a_{1,n} \\ a_{2,1} & \dots & a_{2,i-1} & b_2 & a_{2,i+1} & \dots & a_{2,n} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & \dots & a_{n,i-1} & b_n & a_{n,i+1} & \dots & a_{n,n} \end{pmatrix}$$

Erste Reaktion: Super! Die Formel sieht ja einfach aus, warum müssen wir überhaupt unsere Systemmatrix *sinnvoll* zerlegen?

Einschub zur Übung 4: Die Cramer'sche Regel

Das Problem liegt in den Determinanten:

Einschub zur Übung 4: Die Cramer'sche Regel

Das Problem liegt in den Determinanten:

Determinanten nach Leibnitz

Aus der linearen Algebra wissen wir, dass sich die Determinante einer $n \times n$ Matrix durch

$$\det(A) = \sum_{\sigma \in S_n} \left(\operatorname{sgn}(\sigma) \cdot \prod_{i=1}^{n} a_{i,\sigma_i} \right)$$

berechnet werden kann. Auf diese Weise müssen aber

$$\begin{aligned} (n-1) \cdot n! & \quad \text{Multiplikationen und} \\ n! - 1 & \quad \text{Additionen} \end{aligned}$$

durchgeführt werden. (*Man kann zeigen, dass sich der Aufwand auch mit intelligenteren Formeln nicht lohnt.*)

Einschub zur Übung 4: Die Cramer'sche Regel

Das Problem liegt in den Determinanten:

Determinanten nach Leibnitz

$$\det(A) = \sum_{\sigma \in S_n} \left(\operatorname{sgn}(\sigma) \cdot \prod_{i=1}^{n} a_{i,\sigma_i} \right)$$

Für die Determinante müssen aber

$$\begin{aligned} (n-1) \cdot n! \quad & \text{Multiplikationen und} \\ n! - 1 \quad & \text{Additionen} \end{aligned}$$

durchgeführt werden.

Resultat

Die Cramer'sche Regel ist für Matrizen der Größe 2 und 3 wegen den einfachen Berechnungsvorschriften ($ad - bc$ und die Regel von Sarrus) durchführbar, **für größeres n ganz sicher nicht mehr. Hier lohnt sich dann nur eine Zerlegung in einfachere Teilprobleme.**

Einschub zur Übung 4: Lösen von lin. AGP mit QRP

Problem: Bislang funktionieren die Verfahren $\mathbf{I}$ und $\mathbf{II}$ nur für Matrizen A mit vollem Rang. Dies ist jedoch nicht immer gegeben.

Lösung: Wir pivotisieren die QR-Zerlegung der Spalte nach. Ein kurzer Anriss des Verfahrens, welches grundsätzlich auf Householder-Spiegelungen basiert:
Im i-ten Schritt sieht die Matrix A wie folgt aus:

$$A = \left(\begin{array}{ccc|c} \times & \times & \times & \\ & \times & \times & V^{(i)} \\ & & \times & \\ \hline & \mathbf{0} & & W^{(i)} \end{array}\right)$$

Die Einträge überhalb dem horizontalen Strich (sprich die Dreiecksmatrix und $V^{(i)}$) werden nun im nächsten Householderschritt selbstverständlich nicht mehr angepasst.

Einschub zur Übung 4: Lösen von lin. AGP mit QRP

Lösung (fort.): Man ermittelt dann die Spalte von $W^{(i)}$ mit der größten Norm und vertauscht diese Spalte von

$$\begin{pmatrix} V^{(i)} \\ W^{(i)} \end{pmatrix}$$

mit der ersten Spalte dieses Vektors.
Man kann zeigen, dass dann der Eintrag der rechten oberen Dreiecksmatrix im nächsten Schritt (r_{kk}) betragsmäßig größer ist als die Normen der Spalten von $W^{(i+1)}$.
Merkt man sich die Permutationsschritte wieder in einer Permutationsmatrix P, so erhält man am Schluss eine Zerlegung

$$AP = QR.$$

Hiermit ergibt sich dann eine ähnliche Lösungsmöglichkeit wie bei Verfahren $\mathbf{I}$, welche aber numerische Vorteile mit sich führt.

Einschub zur Übung 4: Mult. im CRS-Format

Sei ein Vektor $b = \begin{pmatrix} 1 & 1 & -1 & 1 \end{pmatrix}^T$ und eine Matrix B im CRS-Format mit

```
    val  =  [1,  4,  7,  -2,  5]
col_ind  =  [2,  3,  4,   2,  3]
row_ptr  =  [1,  1,  2,   4,  6]
```

gegeben. Bestimmen Sie den Vektor Bb **ohne** die Matrix zu rekonstruieren.

Einschub zur Übung 4: Mult. im CRS-Format

```
    val  = [1, 4, 7, -2, 5]
col_ind  = [2, 3, 4,  2, 3]
row_ptr  = [1, 1, 2,  4, 6]
```

$$b = (1 \quad 1 \quad -1 \quad 1)^T$$

Einschub zur Übung 4: Mult. im CRS-Format

```
    val  =  [1,  4,  7,  -2,  5]
col_ind  =  [2,  3,  4,   2,  3]
row_ptr  =  [1,  1,  2,   4,  6]
```

$b = (1 \quad 1 \quad -1 \quad 1)^T$

Algorithmus 4.1 — Multiplikation mit CRS-Matrizen

Eingabe : Eine Matrix A im CRS-Format, ein Vektor $b \in \mathbb{R}^n$

Ausgabe : $y = Ab$

(1) $\mathrm{m} \leftarrow \max_i \texttt{col\_ind}[i]$

(2) $y \leftarrow \vec{0} \in \mathbb{R}^{\mathrm{m}}$

(3) **for** $i \leftarrow 1$ **to** $\min\{n, m\}$ **do**

(4) $\quad y_i \leftarrow \displaystyle\sum_{k=\mathrm{row\_ptr}[i]}^{\mathrm{row\_ptr}[i+1]-1} b_{\mathrm{col\_ind}[k]} \cdot \mathrm{val}[k]$

(5) **end for**

Einschub zur Übung 4: Mult. im CRS-Format

$$
\begin{aligned}
\texttt{val} &= [1, \quad 4, \quad 7, \quad \text{-}2, \quad 5]\\
\texttt{col\_ind} &= [2, \quad 3, \quad 4, \quad 2, \quad 3]\\
\texttt{row\_ptr} &= [1, \quad 1, \quad 2, \quad 4, \quad 6]\\
b &= (1 \quad 1 \quad \text{-}1 \quad 1)^T
\end{aligned}
$$

$$
\begin{aligned}
y_1 &= \sum_{i=1}^{1-1} b_{\texttt{col\_ind}[i]} \cdot \texttt{val}[i]\\
&= 0
\end{aligned}
$$

Algorithmus 4.1 — Multiplikation mit CRS-Matrizen

Eingabe : Eine Matrix A im CRS-Format, ein Vektor $b \in \mathbb{R}^n$
Ausgabe : $y = Ab$

(1) $\mathrm{m} \leftarrow \max_i \texttt{col\_ind}[i]$
(2) $y \leftarrow \vec{0} \in \mathbb{R}^{\mathrm{m}}$
(3) **for** $i \leftarrow 1$ **to** $\min\{n, m\}$ **do**
(4) $\quad y_i \leftarrow \displaystyle\sum_{k=\mathrm{row\_ptr}[i]}^{\mathrm{row\_ptr}[i+1]-1} b_{\mathrm{col\_ind}[k]} \cdot \mathrm{val}[k]$
(5) **end for**

Einschub zur Übung 4: Mult. im CRS-Format

$$
\begin{array}{rcccccc}
\texttt{val} & = & [1, & 4, & 7, & \texttt{-}2, & 5] \\
\texttt{col\_ind} & = & [2, & 3, & 4, & 2, & 3] \\
\texttt{row\_ptr} & = & [1, & 1, & 2, & 4, & 6] \\
b & = & (1 & 1 & \texttt{-}1 & 1)^T &
\end{array}
$$

$$
\begin{aligned}
y_2 &= \sum_{i=1}^{2-1} b_{\texttt{col\_ind}[i]} \cdot \texttt{val}[i] \\
&= 1 \cdot 1 = 1
\end{aligned}
$$

Algorithmus 4.1 — Multiplikation mit CRS-Matrizen

Eingabe : Eine Matrix A im CRS-Format, ein Vektor $b \in \mathbb{R}^n$

Ausgabe : $y = Ab$

(1) $\mathrm{m} \leftarrow \max_i \texttt{col\_ind}[i]$

(2) $y \leftarrow \vec{0} \in \mathbb{R}^{\mathrm{m}}$

(3) **for** $i \leftarrow 1$ **to** $\min\{n, m\}$ **do**

(4) $\quad y_i \leftarrow \displaystyle\sum_{k=\mathrm{row\_ptr}[i]}^{\mathrm{row\_ptr}[i+1]-1} b_{\mathrm{col\_ind}[k]} \cdot \mathrm{val}[k]$

(5) **end for**

Einschub zur Übung 4: Mult. im CRS-Format

$$
\begin{array}{rcccccc}
\texttt{val} & = & [\texttt{1}, & \texttt{4}, & \texttt{7}, & \texttt{-2}, & \texttt{5}] \\
\texttt{col\_ind} & = & [\texttt{2}, & \texttt{3}, & \texttt{4}, & \texttt{2}, & \texttt{3}] \\
\texttt{row\_ptr} & = & [\texttt{1}, & \texttt{1}, & \texttt{2}, & \texttt{4}, & \texttt{6}] \\
b & = & (\texttt{1} & \texttt{1} & \texttt{-1} & \texttt{1})^T &
\end{array}
$$

$$
\begin{aligned}
y_3 &= \sum_{i=2}^{3-1} b_{\texttt{row\_ind}[i]} \cdot \texttt{val}[i] \\
&= -1 \cdot 4 + 1 \cdot 7 = 3
\end{aligned}
$$

Algorithmus 4.1 — Multiplikation mit CRS-Matrizen

Eingabe : Eine Matrix A im CRS-Format, ein Vektor $b \in \mathbb{R}^n$
Ausgabe : $y = Ab$

(1) $\mathrm{m} \leftarrow \max_i \texttt{col\_ind}[i]$
(2) $y \leftarrow \vec{0} \in \mathbb{R}^{\mathrm{m}}$
(3) **for** $i \leftarrow 1$ **to** $\min\{n, m\}$ **do**
(4) $\quad y_i \leftarrow \sum_{k=\mathrm{row\_ptr}[i]}^{\mathrm{row\_ptr}[i+1]-1} b_{\mathrm{col\_ind}[k]} \cdot \mathrm{val}[k]$
(5) **end for**

Einschub zur Übung 4: Mult. im CRS-Format

$$
\begin{aligned}
\texttt{val} &= [1, \quad 4, \quad 7, \quad \texttt{-}2, \quad 5] \\
\texttt{col\_ind} &= [2, \quad 3, \quad 4, \quad 2, \quad 3] \\
\texttt{row\_ptr} &= [1, \quad 1, \quad 2, \quad 4, \quad 6] \\
b &= (1 \quad 1 \quad \texttt{-}1 \quad 1)^T
\end{aligned}
$$

$$
\begin{aligned}
y_4 &= \sum_{i=4}^{6-1} b_{\texttt{row\_ind}[i]} \cdot \texttt{val}[i] \\
&= 1 \cdot (-2) - 1 \cdot 5 = -7.
\end{aligned}
$$

Algorithmus 4.1 — Multiplikation mit CRS-Matrizen

Eingabe : Eine Matrix A im CRS-Format, ein Vektor $b \in \mathbb{R}^n$

Ausgabe : $y = Ab$

(1) $\mathrm{m} \leftarrow \max_i \texttt{col\_ind}[i]$

(2) $y \leftarrow \vec{0} \in \mathbb{R}^{\mathrm{m}}$

(3) **for** $i \leftarrow 1$ **to** $\min\{n, m\}$ **do**

(4) $\quad y_i \leftarrow \displaystyle\sum_{k=\mathrm{row\_ptr}[i]}^{\mathrm{row\_ptr}[i+1]-1} b_{\mathrm{col\_ind}[k]} \cdot \mathrm{val}[k]$

(5) **end for**

Einschub zur Übung 4: Mult. im CRS-Format

$$
\begin{array}{rcl}
\texttt{val} & = & [1,\ 4,\ 7,\ -2,\ 5] \\
\texttt{col\_ind} & = & [2,\ 3,\ 4,\ 2,\ 3] \\
\texttt{row\_ptr} & = & [1,\ 1,\ 2,\ 4,\ 6] \\
b & = & (1\ \ 1\ \ -1\ \ 1)^T
\end{array}
\qquad y = \begin{pmatrix} 0 & 1 & 3 & -7 \end{pmatrix}^T
$$

Algorithmus 4.1 — Multiplikation mit CRS-Matrizen

Eingabe : Eine Matrix A im CRS-Format, ein Vektor $b \in \mathbb{R}^n$
Ausgabe : $y = Ab$

(1) $\mathrm{m} \leftarrow \max_i \texttt{col\_ind}[i]$
(2) $y \leftarrow \vec{0} \in \mathbb{R}^{\mathrm{m}}$
(3) **for** $i \leftarrow 1$ **to** $\min\{n, m\}$ **do**
(4) $\quad y_i \leftarrow \displaystyle\sum_{k=\mathrm{row\_ptr}[i]}^{\mathrm{row\_ptr}[i+1]-1} b_{\mathrm{col\_ind}[k]} \cdot \mathrm{val}[k]$
(5) **end for**