# Numerische Methoden

## Woche 5

David Nadlinger

nadavid@ethz.ch

19. März 2013

# Lineare Ausgleichsrechnung

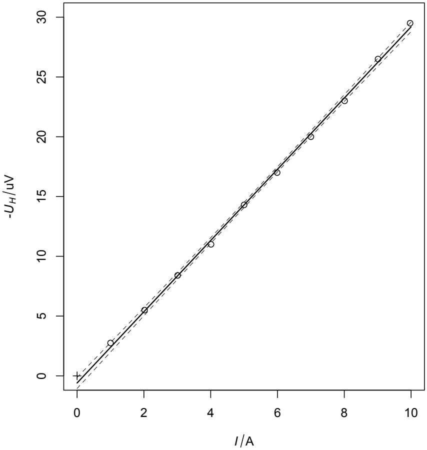
- Sei $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, möchte $\mathbf{x} \in \mathbb{R}^n$ mit $\mathbf{Ax} = \mathbf{b}$. Problem?
- Falls $m > n$ existiert ein solches $\mathbf{x}$ im Allgemeinen nicht!

- Idee: Suche $\mathbf{x} \in \mathbb{R}^n$ mit $||\mathbf{Ax} - \mathbf{b}|| = \min$.
- $\mathbf{r} := \mathbf{Ax} - \mathbf{b} \ldots$ Residuum

# Beispiel: Hall-Effekt

Experiment zum Hall-Effekt

- Messung: Hall-Spannung $V_H$ bei verschiedenen Strömen $I$ durch einen Silberleiter im $1\ \mathrm{T}$-Magnetfeld.
- Offensichtlich linearer Zusammenhang, aber fehlerbehaftet
- Wie «beste» Gerade finden?



# Geradengleichung mit lin. Ausgleichsrechnung

$$\underbrace{\begin{pmatrix} I_1 & 1 \\ I_2 & 1 \\ \vdots & \vdots \\ I_n & 1 \end{pmatrix}}_{=\mathbf{A}} \underbrace{\begin{pmatrix} k \\ d \end{pmatrix}}_{=\mathbf{x}} = \underbrace{\begin{pmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{pmatrix}}_{=\mathbf{b}}$$

# Normalengleichungen

- $\mathbf{A}^\dagger \mathbf{A}\mathbf{x} = \mathbf{A}^\dagger \mathbf{b}$
- $\mathrm{cond}(\mathbf{A}^\dagger \mathbf{A}) = \mathrm{cond}(\mathbf{A})^2$
- Auch wenn $\mathbf{A}$ sparse, $\mathbf{A}^\dagger \mathbf{A}$ i.A. nicht – es gibt aber angepasste Versionen (siehe Skript)

# QR-Zerlegung

Sei $\mathbf{Q} \cdot \mathbf{R} = \mathbf{A}$ die QR-Zerlegung von $\mathbf{A}$. Dann $\mathbf{x} = \mathbf{R}^{-1}\mathbf{Q}^{\dagger}\mathbf{b}$.

- $\mathbf{Q}$ muss nicht gespeichert werden, kann direkt auf $\mathbf{b}$ angewendet werden.

# SVD

Sei $\mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{V}^{\dagger} = \mathbf{A}$ die QR-Zerlegung von $\mathbf{A}$ mit Singulärwerten $\sigma_k$, $r = \max\{k | \sigma_k > 0\}$ der numerische Rang von A.
Seien $V_1$ die ersten $r$ Zeilen von $V$, $U_1$ die ersten $r$ Spalten von $U$, $\mathbf{\Sigma}^{+} = \operatorname{diag}(\sigma_0^{-1}, \ldots, \sigma_r^{-1})$.

- $\mathbf{A}^{+} = \mathbf{V_1} \mathbf{\Sigma}^{+} \mathbf{U_1}^{\dagger}$ ist die *Pseudoinverse* von A
- `np.linalg.pinv`
- $\mathbf{x} = \mathbf{A}^{+} \mathbf{b}$ löst das Ausgleichsproblem

# NumPy

`numpy.linalg.lstsq` (verwendet Hybrid-Verfahren)
Lösung im ersten Rückgabewert: `np.linalg.lstsq(A)[0]`

# Nichtlineare Ausgleichsrechnung

- Sei $F : \mathbb{R}^n \to \mathbb{R}^m$. Finde $\mathbf{x} \in \mathbb{R}^n$ mit $\|F(\mathbf{x})\|_2$ minimal.
- $\Phi(\mathbf{x}) := \frac{1}{2} \left(\|F(\mathbf{x})\|_2\right)^2$
- $\mathbf{x}$ soll regulärer Punkt von $F$ sein

# Newton-Verfahren

- Idee: $\Phi(\mathbf{x}) = \min$. impliziert $\operatorname{grad} \Phi(\mathbf{x}) = 0$, löse mit Newton-Verfahren
- $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - (\mathrm{D} \operatorname{grad} \Phi(\mathbf{x}^{(k)}))^{-1} \operatorname{grad} \Phi(\mathbf{x}^{(k)})$
- $H_\Phi(\mathbf{x}) = D(\operatorname{grad} \Phi)(\mathbf{x})$
- (lokal) quadratische Konvergenz, aber benötigt Hesse-Matrix

# Gauss-Newton-Verfahren

- Linearisiere $F$: $F(\mathbf{x}) \approx F(\mathbf{y}) + \mathrm{D}\, F(\mathbf{y})(\mathbf{x} - \mathbf{y})$
- $\mathrm{D}\, F$ ist Jacobi-Matrix, wird auch $\mathbf{J}_F$ geschrieben
- $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{s}$ mit $\mathbf{s} = \operatorname{argmin}_{\mathbf{y} \in \mathbb{R}^n} \|F(\mathbf{x}^{(k)}) - \mathbf{J}_F(\mathbf{x}^{(k)})\, \mathbf{y}\|_2^2$
- Bestimmung von $\mathbf{s}$ ist lineares Ausgleichsproblem!

# Einfache Gauss-Newton-Implementierung

```
def gauss_newton(x, F, DF, tol=1e-10, maxit=100):
  for i in xrange(maxit):
    step = linear_lstsq(DF(x), F(x))
    x = x - s
    if norm(s) < tol * norm(x):
      return x
  return None
```

- `linear_lstsq` ist eines der vorher behandelten Verfahren
- Benötigt Hesse-Matrix nicht, aber konvergiert auch nicht (immer) quadratisch
- `scipy.optimize.leastsq`

# Direkte Potenzmethode

- Sei $A \in \mathbb{K}^{n \times n}$ mit Eigenwerten $|\lambda_n| \overset{!}{>} |\lambda_{n-1}| \geq \ldots \geq 0$.
- Findet betragsgrößten Eigenwert $\lambda_n$ und zugehörigen Eigenvektor
- $\mathbf{z}^{(k+1)} = \frac{\mathbf{A}\ \mathbf{z}^{(\mathbf{k})}}{\|\mathbf{A}\ \mathbf{z}^{(\mathbf{k})}\|}$
- Konvergiert linear mit Rate $\frac{|\lambda_{n-1}|}{|\lambda_n|}$
- Schätzwert für $\lambda_n$ basierend auf aktuellem Vektor $\mathbf{z}$: Raleigh-Koeffizient $\rho_{\mathbf{A}}(\mathbf{z}) := \frac{\mathbf{z}^\dagger \mathbf{A}\ \mathbf{z}}{\mathbf{z}^\dagger \mathbf{z}}$.

# Implementierung Direkte Potenzmethode

- Zum Vergleich: `np.linalg.eigs`
- Abbruchkriterium: ohne weitere Informationen ähnlich Newton-Verfahren (für Serie 5 nicht relevant!)
- Startwert: `np.random.rand(n)`