

# Numerik Cheat Sheet

## 1 Basics

### 1.1 Sortieren

### 1.2 FFT

## 2 Lineare Gleichungssysteme

### 2.1 Allgemeine Aufgabenstellung

Geg.:  $\mathbf{A} \in \mathbb{R}^{n \times n}$ ,  $\mathbf{b} \in \mathbb{R}^n$

Ges.:  $\mathbf{x} \in \mathbb{R}^n$

$$\mathbf{Ax} = \mathbf{b}$$

### 2.2 Dreiecksmatrizen

Untere Dreiecksmatrix  $\mathbf{L} \in \mathbb{R}^{n \times n}$  und obere Dreiecksmatrix  $\mathbf{R} \in \mathbb{R}^{n \times n}$ .

REGULÄRE/INVERTIERBARE/NICHT-SINGULÄRE MATRIX  
Matrix  $\mathbf{A}$  ist regulär, wenn  $\det \mathbf{A} \neq 0$ . Determinante einer  $\Delta$ Matrix ist das Produkt ihrer Diagonalelemente.  $\mathbf{L}$  und  $\mathbf{R}$  sind regulär, wenn alle Diagonalelemente  $\neq 0$ .

VORWÄRTSEINSETZEN

$$\mathbf{Ly} = \mathbf{b}$$

Rechenaufwand:  $n^2$  AO

Um Speicher zu sparen  $b_i \leftarrow y_i$ .

forward\_subst

---

```
for j = 1 : n
    x_j ← b_j / l_jj
    for i = j + 1 : n
        b_i ← b_i - l_ij x_j
```

---

RÜCKWÄRTSEINSETZEN

$$\mathbf{Rx} = \mathbf{y}$$

Rechenaufwand:  $n^2$  AO

Um Speicher zu sparen  $b_i \leftarrow x_i$ .

backward\_subst

---

```
for j = n : 1
    x_j ← b_j / r_jj
    for i = 1 : j - 1
        b_i ← b_i - r_ij x_j
```

---

### 2.3 LR-Zerlegung

Sei  $\mathbf{A} \in \mathbb{R}^{n \times n}$  und  $\mathbf{L}, \mathbf{R} \in \mathbb{R}^{n \times n}$

$$\mathbf{A} = \mathbf{LR}$$

Ansatz:

Matrizen  $\mathbf{A}$ ,  $\mathbf{L}$ ,  $\mathbf{R}$  in Teilmatrizen  $\mathbf{A}_{**}$ ,  $\mathbf{A}_{*1}$ ,  $\mathbf{A}_{1*}$ ,  $\mathbf{L}_{**}$ ,  $\mathbf{L}_{*1}$ ,  $\mathbf{R}_{**}$ ,  $\mathbf{R}_{1*}$  zerlegen.

Es folgen 4 Gleichungen aus  $\mathbf{A} = \mathbf{LR}$ :

$$\begin{aligned} a_{11} &= l_{11} r_{11} \\ \mathbf{A}_{*1} &= \mathbf{L}_{*1} r_{11} & \Leftrightarrow \mathbf{L}_{*1} &= \mathbf{A}_{*1} / r_{11} \\ \mathbf{A}_{1*} &= l_{11} \mathbf{R}_{1*} & \Leftrightarrow \mathbf{R}_{1*} &= \mathbf{A}_{1*} \\ \mathbf{A}_{**} &= \mathbf{L}_{*1} \mathbf{R}_{1*} + \mathbf{L}_{**} \mathbf{R}_{**} \end{aligned}$$

Per Def.  $l_{11} = 1$  und damit  $r_{11} = a_{11}$ , sodass

$$\mathbf{A}_{**} - \mathbf{L}_{*1} \mathbf{R}_{1*} = \mathbf{L}_{**} \mathbf{R}_{**} \quad (1)$$

PRAKTISCHE UMSETZUNG

Elemente von  $\mathbf{A}$  überschreiben, sodass:

$$\begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ l_{21} & r_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & r_{n-1,n} \\ l_{n1} & \cdots & l_{n,n-1} & r_{nn} \end{pmatrix} \quad (2)$$

KRITERIUM

Sei  $\mathbf{A}$  regulär,  $\mathbf{A}$  besitzt eine LR-Zerlegung  $\Leftrightarrow$  Alle Hauptuntermatrizen regulär.

MODELLPROBLEM: BANDMATRIX

Irgendwas bzgl Effizienz.

LR-DECOMP

Aufwand: *kubisch*

↓ Aufwand: Nur 1x für jede Matrix betreiben. Sobald LR-Decomp vorliegt nur noch *quadratischer* Aufwand  
↓ Aufwand: Tridiagonalmatrix. Erster Schritt mit 3 AOPs. Restmatrix bleibt tridiagonal. Aufwand  $3n + 6n$  für R- und F-Einsetzen.

lr\_decomp

---

```
for k = 1 : n
    for i = k + 1 : n
        a_ik ← a_ik / a_kk
    for j = k + 1 : n
        a_ij ← a_ij - a_ik a_kj
```

---

PROBLEM DER EXISTENZ EINER LR-Z

Falls  $\mathbf{A}$  oder  $\mathbf{A}_{**}$  eine 0 auf der Diagonalen hat, existiert keine LR-Zerlegung. Lösung: Permutiere die Zeilen von  $\mathbf{A}$  so, dass das Ergebnis eine LR-Z besitzt.

PERMUTATIONSMATRIX

Sei  $\mathbf{P} \in \mathbb{R}^{n \times n}$ . Falls in jeder Zeile und Spalte von  $\mathbf{P}$  genau ein Eintrag 1 und alle anderen 0, dann ist  $\mathbf{P}$  eine Permutationsmatrix.  $\mathbf{P}$  ist orthogonal. Ein Produkt zweier Permutationsmatrizen  $\mathbf{PQ}$  ist auch eine Permutationsmatrix.

PERMUTATION

Bijektive Abbildung  $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ .

LR-Z MIT PIVOTSUCHE

$\mathbf{A}$  regulär. Es existiert  $\mathbf{P} \in \mathbb{R}^{n \times n}$  sodass  $\mathbf{PA} = \mathbf{LR}$  gilt.

Pivotisierung: Finde betragsmaximalstes Element in der aktuellen Spalte, welches unter dem aktuellen Diagonalelement von  $\mathbf{A}$  liegt und tausche die aktuelle Zeile mit der Zeile in der das betragsmaximalste Element ist, mit Hilfe von  $\mathbf{P}$ .  $a_{11} \neq 0$ , da betragsgrößtes Element.

LÖSEN EINES GLEICHUNGSSYSTEMS MIT PIVOTSUCHE

$$\mathbf{Ax} = \mathbf{b} \Leftrightarrow \mathbf{PAx} = \mathbf{Pb} \Leftrightarrow \mathbf{LRx} = \mathbf{Pb}.$$

1.)  $\mathbf{Ly} = \tilde{\mathbf{b}}$     2.)  $\mathbf{Rx} = \mathbf{y}$

lr\_pivot

---

for  $k = 1 : n$

$i_* \leftarrow k$  // Finde max. Element

for  $i = k + 1 : n$

if  $|a_{ik}| > |a_{i_*k}|$ :  $i_* \leftarrow i$

$p_k \leftarrow i_*$

for  $j = 1 : n$  // Tausche Zeilen

$\gamma \leftarrow a_{kj}$ ,  $a_{kj} \leftarrow a_{i_*j}$ ,  $a_{i_*j} \leftarrow \gamma$

for  $i = k + 1 : n$

$a_{ik} \leftarrow a_{ik} / a_{kk}$

for  $j = k + 1 : n$

$a_{ij} \leftarrow a_{ij} - a_{ik} a_{kj}$

---

$\mathbf{p}$  protokolliert, welche Vertauschungen durchgeführt wurden, um sie später auf  $\mathbf{b}$  anwenden zu können.

Aufwand:  $\frac{2}{3}n^3$ .

### 2.4 Fehlerverstärkung

NORM DES MATRIX-VEKTOR-PRODUKTS

Wie stark ändert sich die Länge eines Vektors wenn er mit  $\mathbf{A}$  multipliziert wird. Mapping von Einheitskreis auf Ellipse..

Für  $\mathbf{A} \in \mathbb{R}^{n \times n}$  gilt:

$$\alpha_2(\mathbf{A}) = \min\{\|\mathbf{Ay}\|_2 : \mathbf{y} \in \mathbb{R}^n, \|\mathbf{y}\|_2 = 1\}$$

$$\beta_2(\mathbf{A}) = \max\{\|\mathbf{Ay}\|_2 : \mathbf{y} \in \mathbb{R}^n, \|\mathbf{y}\|_2 = 1\}$$

und

$$\alpha_2(\mathbf{A})\|\mathbf{z}\|_2 \leq \|\mathbf{Az}\|_2 \leq \beta_2(\mathbf{A})\|\mathbf{z}\|_2$$

Eigenschaften der Norm:

$$\|\mathbf{x}\| = 0 \Leftrightarrow \mathbf{x} = \mathbf{0}$$

$$\|\lambda \mathbf{x}\| = |\lambda| \|\mathbf{x}\|$$

$$\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$$

KONDITIONSZAHL

$$\alpha_2(\mathbf{A}) = \frac{1}{\|\mathbf{A}^{-1}\|_2}, \quad \beta_2(\mathbf{A}) = \|\mathbf{A}\|_2$$

$$\kappa_2(\mathbf{A}) = \|\mathbf{A}\|_2 \|\mathbf{A}^{-1}\|_2 = \frac{\beta_2(\mathbf{A})}{\alpha_2(\mathbf{A})}$$

MATRIXPRODUKT

Mit  $\mathbf{A} \in \mathbb{R}^{n \times m}$  und  $\mathbf{B} \in \mathbb{R}^{m \times k}$  gilt:

$$\alpha_2(\mathbf{AB}) \geq \alpha_2(\mathbf{A})\alpha_2(\mathbf{B}) \quad \beta_2(\mathbf{AB}) \geq \beta_2(\mathbf{A})\beta_2(\mathbf{B})$$

STÖRUNG DER RECHTEN SEITE

Der relative Fehler der Lösung lässt sich abschätzen aus dem relativen Fehler der rechten Seite und der Konditionszahl  $\kappa_2(\mathbf{A})$ :

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|_2}{\|\mathbf{x}\|_2} \leq \kappa_2(\mathbf{A}) \frac{\|\mathbf{b} - \tilde{\mathbf{b}}\|_2}{\|\mathbf{b}\|_2}$$

STÖRUNG DER MATRIX

Der relative Fehler lässt sich beschränken. Beschränkt durch das Produkt der Konditionszahl und des relativen Fehlers der Matrix.

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|_2}{\|\mathbf{x}\|_2} \leq \frac{\kappa_2(\mathbf{A})}{1 - \kappa_2(\mathbf{A})} \frac{\|\mathbf{A} - \tilde{\mathbf{A}}\|_2}{\|\mathbf{A}\|_2} \frac{\|\mathbf{A} - \tilde{\mathbf{A}}\|_2}{\|\mathbf{A}\|_2}$$

## 2.5 QR-Zerlegung

Für jede Matrix gibt es eine QR-Z.

LR-Z: SCHLECHT KONDITIONIERTES PROBLEM

$\kappa_2(\mathbf{A}) \gg 1$ ,  $\kappa_2(\mathbf{A}) \leq \kappa_2(\mathbf{L})\kappa_2(\mathbf{R})$

Kritisch falls  $\kappa_2(\mathbf{L})\kappa_2(\mathbf{R}) \gg \kappa_2(\mathbf{A})$ .

Ziel: Suche Transformationen  $\mathbf{Q} \in \mathbb{R}^{n \times n}$ , die die Norm unverändert lassen:

$$\|\mathbf{Q}\mathbf{y}\|_2 = \|\mathbf{y}\|_2$$

Mit Hinzunahme des Skalarprodukts:

$$\langle \mathbf{y}, \mathbf{y} \rangle_2 = \|\mathbf{y}\|_2^2 = \|\mathbf{Q}\mathbf{y}\|_2^2 = \langle \mathbf{Q}\mathbf{y}, \mathbf{Q}\mathbf{y} \rangle_2 = \langle \mathbf{y}, \mathbf{Q}^* \mathbf{Q} \mathbf{y} \rangle_2$$

muss  $\mathbf{Q}^* \mathbf{Q} = \mathbf{I}$  gelten.

Gesucht:  $\mathbf{A} = \mathbf{Q}\mathbf{R}$ .

Konditionszahl bzw. Fehlerverstärkung wird nicht verschlechtert:  $\alpha_2(\mathbf{A}) = \alpha_2(\mathbf{R})$ ,  $\beta_2(\mathbf{A}) = \beta_2(\mathbf{R})$

$\kappa_2(\mathbf{A}) = \kappa_2(\mathbf{R})$ .

GIVENS-ROTATION

Mit Hilfe von Givens-Rotationen können wir beliebige

$\mathbf{A} \in \mathbb{R}^{m \times n}$  auf obere  $\Delta$ gestalt bringen.

$$\mathbf{Q} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \text{ und } \mathbf{Q}\mathbf{y} = \begin{pmatrix} cy_1 + sy_2 \\ 0 \end{pmatrix}$$

Konsekutiv Givens-Rotationen  $\mathbf{Q}_{ij}$   $i$ -te Zeile in  $j$ -te Spalte

anwenden um Eintrag  $a_{ij}$  zu beseitigen. Bsp.  $\mathbf{A} \in \mathbb{R}^{4 \times 3}$ :

$$\begin{aligned} \mathbf{R} &= \mathbf{Q}_{43} \mathbf{Q}_{32} \mathbf{Q}_{42} \mathbf{Q}_{21} \mathbf{Q}_{31} \mathbf{Q}_{41} \mathbf{A} \\ \underbrace{\mathbf{Q}_{41}^* \mathbf{Q}_{31}^* \mathbf{Q}_{21}^* \mathbf{Q}_{42}^* \mathbf{Q}_{32}^* \mathbf{Q}_{43}^*}_{\mathbf{Q}} \mathbf{R} &= \mathbf{A} \end{aligned}$$

KOMPAKTE DARSTELLUNG

Verwende Nulleinträge von  $\mathbf{A}$  bzw.  $\mathbf{R}$  um  $\mathbf{Q}_{ij}$  zu beschreiben.

Finde Givens-Rotation:

$$\rho = \begin{cases} s = \rho, c = \sqrt{1 - s^2} & \text{falls } |\rho| < 1 \\ c = 1/\rho, s = \sqrt{1 - c^2} & \text{falls } |\rho| > 1 \\ c = 1, s = 0 & \text{falls } \rho = 1 \end{cases}$$

Speichern der QR-Z in  $\mathbf{A}$ :

$$\begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ \rho_{21} & r_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & r_{n-1,n} \\ \rho_{n1} & \cdots & \rho_{n,n-1} & r_{nn} \end{pmatrix}$$

(3)

Qr Decomp von  $\mathbf{A} \in \mathbb{R}^{m \times n}$

---

```

for k = 1 : min(m, n) // Loop über Diagonale
    for i = k + 1 : m // Loop über Elemente unter Diagonalen
        if aik = 0
            ρ ← 1, c ← 1, s ← 0
        else if |akk| ≥ |aik| // Vgl. mit Diag.element
            τ ← aik/akk, ρ ← τ/√(τ² + 1), s ← ρ, c ← √(1 - s²)
        else // Vgl. mit Diag.element
            τ ← akk/aik, ρ ← √(τ² + 1)/τ, c ← 1/ρ, s ← √(1 - c²)
        // Diag.element aktual., Giv.-Rot. in aktueller It. speichern
        akk ← cakk + saik, aik ← ρ
    for j = k + 1 : n // Loop über Elemente in der k-ten Zeile
        // Giv-Rot auf Zeile anwenden
        α ← akj, akj ← cα + saij, aij ← -sα + caij

```

---

Aufwand:  $6n^2 + 2n^3$  (quadratische Matrix) 3x mehr als LR-Z.

LÖSEN GLEICHUNGSSYSTEM  $\mathbf{Ax} = \mathbf{b}$

$\mathbf{b} = \mathbf{Ax} = \mathbf{QRx} = \mathbf{Qy} \Leftrightarrow$  1.)  $\mathbf{y} = \mathbf{Q}^* \mathbf{b}$  2.)  $\mathbf{y} = \mathbf{Rx}$ .

1.) Qr-transform: Über einzelne  $G_{ij}$  (oben links angefangen) iterieren und auf  $b$  multiplizieren.

2.) Rückwärtseinsetzen.

EFFIZIENTERE QR-Z

Householder-Spiegelungen: Aufwand 2x mehr als LR-Z.

Mit Optimierungen bei Speicherzugriffen bei QR-Z ähnlich schnell wie LR-Z.

## 2.6 Ausgleichsprobleme

Wir suchen  $\mathbf{x}$  so, dass alle Gleichungen möglichst gleich gut erfüllt werden - oder - von unbekannten Parametern abhängige Kurve durch Messdaten in unterschiedlichen APs zu fitten.

GRUNDLAGEN

Wir suchen die unbekannte Funktion  $y$  aus den Messwerten  $b$ , sodass  $b_i = y(t_i) \quad \forall i \in \{1, \dots, m\}$ .

Annahme:  $y$  setzt sich zusammen aus Linearkombination bekannter Funktionen  $y_1, \dots, y_m$  mit  $n < m$  mit Faktoren

$x_1, \dots, x_m$  sodass:  $y(t) = x_1 y_1(t) + \dots + x_n y_n(t)$  bzw.

$b_i = y(t_i) = x_1 y_1(t_i) + \dots + x_n y_n(t_i)$ . Als Gleichungssystem schreiben - überbestimmt  $m$  Gleichungen für  $n$  Unbekannte -

mehr Messwerte als Unbekannte:

$$\begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix} = \begin{pmatrix} y_1(t_1) & \cdots & y_n(t_1) \\ \vdots & \ddots & \vdots \\ y_1(t_m) & \cdots & y_n(t_m) \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

AUFGABE

Suche  $\mathbf{x} \in \mathbb{R}^n$  mit  $\|\mathbf{Ax} - \mathbf{b}\|_2 \leq \|\mathbf{Az} - \mathbf{b}\|_2 \quad \forall \mathbf{z} \in \mathbb{R}^n$  - also optimale Näherung. Falls  $\mathbf{A}$  injektiv, dann ex. genau eine Lösung. Es gelte  $\mathbf{A} = \mathbf{QR}$  mit  $\mathbf{Q} \in \mathbb{R}^{m \times m}$  und  $\mathbf{R} \in \mathbb{R}^{m \times n}$ .

$$\begin{aligned} \|\mathbf{Az} - \mathbf{b}\|_2 &= \|\mathbf{QRz} - \mathbf{Q}\mathbf{Q}^* \mathbf{b}\|_2 = \|\mathbf{Q}(\mathbf{Rz} - \mathbf{Q}^* \mathbf{b})\|_2 \\ &= \|\mathbf{Rz} - \mathbf{Q}^* \mathbf{b}\|_2 \end{aligned}$$

Für  $\mathbf{R}$  gilt:  $\mathbf{R} = \begin{pmatrix} \hat{\mathbf{R}} \\ \mathbf{0} \end{pmatrix}$   $\hat{\mathbf{R}} \in \mathbb{R}^{n \times n}$  und beide injektiv und  $\hat{\mathbf{R}}$

regulär. Außerdem  $\mathbf{Q}^* \mathbf{b} = \begin{pmatrix} \hat{\mathbf{b}} \\ \mathbf{b}_0 \end{pmatrix}$   $\hat{\mathbf{b}} \in \mathbb{R}^n$ ,  $\mathbf{b}_0 \in \mathbb{R}^{m-n}$ . Norm

ausnutzen ergibt:  $\|\hat{\mathbf{R}}\mathbf{z} - \hat{\mathbf{b}}\|_2^2 + \|\mathbf{b}_0\|_2^2$ . Es folgt:

$$\begin{aligned} \|\mathbf{Ax} - \mathbf{b}\|_2^2 &= \|\mathbf{Rx} - \mathbf{Q}^* \mathbf{b}\|_2^2 = \underbrace{\|\hat{\mathbf{R}}\mathbf{x} - \hat{\mathbf{b}}\|_2^2}_{\text{löse } \hat{\mathbf{R}}\mathbf{z} = \hat{\mathbf{b}} \text{ oben}} + \|\mathbf{b}_0\|_2^2 = \|\mathbf{b}_0\|_2^2 \\ &\leq \|\hat{\mathbf{R}}\mathbf{z} - \hat{\mathbf{b}}\|_2^2 + \|\mathbf{b}_0\|_2^2 = \|\mathbf{Rz} - \mathbf{Q}^* \mathbf{b}\|_2^2 = \|\mathbf{Az} - \mathbf{b}\|_2^2 \end{aligned}$$

NORMALENGLEICHUNG

Anstatt mit QR-Zerlegung zu lösen, Normalengleichung nutzen

- bei Ausnutzung von  $\mathbf{A}^* \mathbf{A}$  symmetrisch schneller als QR

Ansatz:

$$\mathbf{A}^* \mathbf{Ax} = \mathbf{A}^* \mathbf{b}$$

Nachteil, mögliche Fehlerverstärkung durch

$$\kappa_2(\mathbf{A}^* \mathbf{A}) = \kappa_2(\mathbf{A})^2$$

## 3 Nichtlineare Gleichungssysteme

Wir untersuchen nichtlineare Gleichungssysteme der Form

Gegeben eine stetige Funktion  $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ , finde  $\mathbf{x}^* \in \mathbb{R}^n$  mit

$$f(\mathbf{x}^*) = \mathbf{0}$$

Transformieren in ein Nullstellenproblem.

### 3.1 Bisektionsverfahren

Einfache Technik, das in jedem Schritt den Fehler mindestens halbiert. Basierend auf dem Zwischenwertsatz für stetige Funktionen.

Funktioniert nur für 1D.

ZWISCHENWERTSATZ

Eine reelle Funktion  $f$ , die in  $[a, b]$  stetig ist, nimmt jeden Wert zwischen  $f(a)$  und  $f(b)$  an. Haben  $f(a)$  und  $f(b)$  verschiedene Vorzeichen, so ist eine Existenz mindestens einer Nullstelle in  $[a, b]$  garantiert.

VERFAHREN IN MATHEMATISCHER NOTATION

$$(a^{(0)}, b^{(0)}) = (a, b)$$

$$x^{(m)} = \frac{a^{(m)} + b^{(m)}}{2}$$

$$(a^{(m+1)}, b^{(m+1)}) = \begin{cases} (a^{(m)}, x^{(m)}) & \text{if } f(a^{(m)})f(x^{(m)}) < 0 \\ (x^{(m)}, b^{(m)}) & \text{sonst.} \end{cases}$$

Bisection

$f_a \leftarrow f(a)$ ,  $f_b \leftarrow f(b)$

while  $b - a > \epsilon$

$x \leftarrow (a + b)/2$

$f_x \leftarrow f(x)$

if  $f_a f_x < 0$

$b \leftarrow x$ ,  $f_b \leftarrow f_x$

else

$a \leftarrow x$ ,  $f_a \leftarrow f_x$

Aufwand:  $m + 2$  Auswertungen von  $f$  und  $2m$

Rechenoperationen, mit  $m = \lceil \log_2((b - a)/\epsilon) \rceil$  Schritten. Hohe Stabilität und jedes konstruierte Intervall muss eine Nullstelle enthalten. Nur auf reellwertige Funktionen auf geeigneten Intervallen anwendbar.

## 3.2 Allgemeine Fixpunktiterationen

### ITERATION

$U \subseteq \mathbb{R}^n$  eine abgeschlossene Teilmenge und  $\Phi : U \rightarrow U$  eine (Selbst-)Abbildung. Dann ist  $\Phi$  eine Iteration auf  $U$ . Folge der Iterierten  $\mathbf{x}^{(m+1)} = \Phi(\mathbf{x}^{(m)})$ . Konstruiere Iteration so, dass  $\Phi$  gegen gesuchte Lösung  $\mathbf{x}^*$  konvergiert. Es soll  $\Phi(\mathbf{x}^*) = \mathbf{x}^*$  gelten. Die Lösung muss ein Fixpunkt von  $\Phi$  sein.

### MITTELWERTSATZ DER DIFFERENTIALRECHNUNG

Zwischen  $a$  und  $b$  von  $f$  gibt es mindestens einen Kurvenpunkt, für den die Tangente an  $\eta$  parallel zur Sekante durch  $a$  und  $b$  ist:  $(b-a)f'(\eta) = f(b) - f(a)$ .

### FIXPUNKTSATZ VON BANACH

Sei  $\Phi$  eine Iteration auf einer **abgeschlossenen** Menge  $U \subseteq \mathbb{R}^n$ ,  $\Phi : U \rightarrow U$  (**Selbstabbildung**: Kontraktion bleibt bei Iteration erhalten). Sei  $L \in [0, 1)$  so gegeben, dass:  $\|\Phi(\mathbf{x}) - \Phi(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$  für alle  $\mathbf{x}, \mathbf{y} \in U$  (**Kontraktion**).  $\Phi$  besitzt **genau** einen Fixpunkt und die Folge der Iterierten konvergiert für jeden Startwert  $\mathbf{x}^{(0)} \in U$  gegen diesen Fixpunkt.

Fehlerabschätzung *a-priori* (Vorhersagen wieviele Schritte) und *a-posteriori* (Prüfen ob Näherung schon genau genug):

$$\|\mathbf{x}^{(m)} - \mathbf{x}^*\| \leq \frac{L^m}{1-L} \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|$$

$$\|\mathbf{x}^{(m)} - \mathbf{x}^*\| \leq \frac{1}{1-L} \|\mathbf{x}^{(m)} - \mathbf{x}^{(m+1)}\| \quad \forall m \in \mathbb{N}_0$$

Nachteil:  $\Phi$  konstruieren mit Fixpunkt, wo  $\mathbf{x}$  eine Nullstelle hat.

## 3.3 1D-Newton-Verfahren

$U \subseteq \mathbb{R}^n$  offene Menge und  $f : U \rightarrow \mathbb{R}$  zweimal stetig differenzierbar mit Nullstelle  $\mathbf{x}^* \in U$ , so dass  $f(\mathbf{x}^*) = 0$ . Ziel: Konstruiere Iteration  $\Phi$ , die  $\mathbf{x}^*$  als Nullstelle besitzt.

### TAYLOR

$$0 = f(x^*) = f(x) + f'(x)(x^* - x) + \frac{f''(\eta)}{2}(x^* - x)^2$$

### EINDIMENSIONALES NEWTON-VERFAHREN

$$\Phi : U \rightarrow \mathbb{R} \quad x \mapsto x - \frac{f(x)}{f'(x)} \rightarrow_{m \rightarrow \infty} x^*$$

Indem der dritte Term der Taylorreihe *wegfällt*, approximieren wir die Funktion  $f$  durch ihre Tangente im Punkt  $x$  (lineare Näherung). Die Nullstelle der Tangente ist die nächste Iterierte  $\Phi(x)$ .

### KONVERGENZ

Sei  $r \in \mathbb{R}_{>0}$  und  $U = (x^* - r, x^* + r)$  und gelte  $f \in C^2(U)$ ,  $|1/f'(x)| \leq C_1 \forall x \in U$ ,  $|f''(x)| \leq C_2 \forall x \in U$  und  $r \leq \frac{2}{C_1 C_2}$ , dann ist die Abbildung  $\Phi$  für das Newton Verfahren eine Selbstabbildung auf  $U$ , sodass gilt:

$$|\Phi(x) - x^*| \leq \frac{C_1 C_2}{2} |x - x^*|^2 \quad \forall x \in U$$

Newton-Verfahren konvergiert, falls  $x^{(0)}$  in  $U$  liegt (lokale Konvergenz). Konvergenz ist umso schneller, je näher die Iterierten an der Lösung liegen. **Quadratische Konvergenz**.

## 3.4 ND-Newton-Verfahren

Herleitung über Hilfsfunktionen  $\gamma(t) = \mathbf{x} + t(\mathbf{x}^* - \mathbf{x})$ ,  $g(t) = f(\gamma(t))$  in  $[0, 1]$  und *Hauptsatz der Integral- und Differentialrechnung*  $f(b) - f(a) = \int_a^b f'(t)dt$  um höhere Ableitungen (wie bei Taylor) zu vermeiden.

$0 = f(\mathbf{x}^*) = g(1) = g(0) + \int_0^1 g'(t)dt = g(0) + g'(0) + \int_0^1 g'(t) - g'(0)dt = f(\mathbf{x}) + Df(\mathbf{x})(\mathbf{x}^* - \mathbf{x}) + \mathfrak{C} = Df(\mathbf{x})^{-1}f(\mathbf{x}) + \mathbf{x}^* - \mathbf{x} + Df(\mathbf{x})^{-1}\mathfrak{C}$ . Umstellen nach  $\mathbf{x}^*$  und mit  $Df(\mathbf{x})^{-1}\mathfrak{C}$  vernachlässigen ergibt Newton-Verfahren.

### NEWTON-VERFAHREN

Sei  $U \subseteq \mathbb{R}^n$  und  $Df(\mathbf{x})$  für alle  $\mathbf{x} \in U$  regulär (also invertierbar). Es gilt:

$$\Phi : U \rightarrow \mathbb{R}^d, \quad \mathbf{x} \mapsto \mathbf{x} - Df(\mathbf{x})^{-1}f(\mathbf{x})$$

### KONVERGENZ

Sei  $r \in \mathbb{R}_{>0}$  und  $U = K(\mathbf{x}^*, r)$  und gelte

$f \in C^1(U, \mathbb{R}^b)$ ,  
 $\|Df(\mathbf{x})^{-1}\|_2 \leq C_1 \quad \forall \mathbf{x} \in U$ ,  
 $\|Df(\mathbf{x}) - Df(\mathbf{y})\|_2 \leq C_2\|\mathbf{x} - \mathbf{y}\|_2 \quad \forall \mathbf{x}, \mathbf{y} \in U$  (Lipschitz stetig)  
 $r \leq \frac{2}{C_1 C_2}$ .

Dann ist  $\Phi$  eine Selbstabbildung auf der Kugel  $U$  und es gilt

$$\|\mathbf{x}^{(m+1)} - \mathbf{x}^*\|_2 \leq \frac{C_1 C_2}{2} \|\mathbf{x}^{(m)} - \mathbf{x}^*\|_2^2 \quad \forall \mathbf{x} \in U$$

Quadratische Konvergenz unter schwächeren Voraussetzungen, denn  $f'$  muss nur Lipschitz-stetig sein. Geeigneter Anfangswert  $< \frac{C_1 C_2}{2}$

### UMSETZUNG

Anstatt Inverse Jacobimatrix zu berechnen, lineares Gleichungssystem nach  $\mathbf{d}$  lösen (erhöhte numerische Stabilität):

$$\mathbf{x}^{(m+1)} = \mathbf{x}^{(m)} + \mathbf{d}^{(m)} \quad \mathbf{d}^{(m)} = -Df(\mathbf{x}^{(m)})^{-1}f(\mathbf{x}^{(m)})$$

$$Df(\mathbf{x}^{(m)})\mathbf{d}^{(m)} = -f(\mathbf{x}^{(m)})$$

### GEDÄMPFTES NEWTON-VERFAHREN

Um Divergenz zu vermeiden Newton-Richtung mit Dämpfungsparameter  $\sigma^{(m)}$  multiplizieren. Sorgt dafür, dass der Fehler nicht größer als im vorangehenden Schritt werden kann.

$$\mathbf{x}^{(m+1)} = \mathbf{x}^{(m)} + \sigma^{(m)}\mathbf{d}^{(m)}$$

## 4 Eigenwertprobleme

Im Allgemeinen werden Eigenwertprobleme in der Form

$\mathbf{Ax} = \lambda \mathbf{x}$ , mit  $\mathbf{A} \in \mathbb{R}^{n \times n}$  und  $\mathbf{x} \neq 0 \in \mathbb{R}^n$  untersucht.

Eigenwertproblem in die Form eines linearen Gleichungssystems bringen:

$$(\mathbf{A} - \lambda \mathbf{I})\mathbf{x} = \mathbf{0} \quad \Leftrightarrow \quad \det(\mathbf{A} - \lambda \mathbf{I}) = 0$$

Finden der Nullstellen des *charpolys* ist ein schlecht konditioniertes Problem (nur geeignet für kleine  $n$ ). Deswegen andere Methode...

## 4.1 Vektoriteration

*Eignet sich für die Berechnung des größten Eigenwerts.*

**Annahmen:**  $\mathbf{A}$  diagonalisierbar, d.h. es existiert eine Basis aus EV von  $\mathbf{A}$ :  $\mathbf{v}_1, \dots, \mathbf{v}_n$ . Es gilt  $\|\mathbf{v}_i\|_2 = 1$  und es existiert ein einfacher dominanter EW  $\lambda_1$ . Beliebigen Startvektor  $\mathbf{x}^{(0)} = c_1 \mathbf{v}_1 + \dots + c_n \mathbf{v}_n$  wählen mit  $c_1 \neq 0$  (!). Oder anders:  $\mathbf{A} = \mathbf{C}\hat{\mathbf{A}}\mathbf{C}^{-1}$  mit  $\hat{\mathbf{A}} = \text{diag}(\lambda_1, \dots, \lambda_n)$ . Außerdem gilt  $\mathbf{A}^k = (\mathbf{C}\hat{\mathbf{A}}\mathbf{C}^{-1}) \dots (\mathbf{C}\hat{\mathbf{A}}\mathbf{C}^{-1}) = \mathbf{C}\hat{\mathbf{A}}^k \mathbf{C}^{-1}$ , bzw.  $\mathbf{A}^k \mathbf{C} = \mathbf{C}\hat{\mathbf{A}}^k$ . Wendet man die  $k$ -te Potenz von  $\mathbf{A}$  auf  $\mathbf{x}^{(0)}$  an, ergibt sich:

$$\mathbf{A}^k \mathbf{x}^{(0)} = \mathbf{A}^k \mathbf{C} \mathbf{v}^{(0)} = \mathbf{C} \hat{\mathbf{A}}^k \mathbf{v}^{(0)} = \sum_{j=1}^n c_j \hat{\mathbf{A}}^k \mathbf{v}_j = \sum_{j=1}^n c_j \lambda_j^k \mathbf{v}_j$$

Also folgt:

$$\mathbf{x}^{(k)} = \mathbf{A}^k \mathbf{x}^{(0)} = \lambda_1^k \left\{ c_1 \mathbf{v}_1 + \sum_{j=2}^n \left( \frac{\lambda_j}{\lambda_1} \right)^k c_j \mathbf{v}_j \right\} = \lambda_1^k (c_1 \mathbf{v}_1 + \mathbf{r}^k)$$

wobei  $\mathbf{r}^k \rightarrow 0$  für  $k \rightarrow \infty$ , da  $\|\mathbf{r}^k\|_2 = \mathcal{O}\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)$ . Also strebt

$\mathbf{x}^{(k)} = \mathbf{A}^k \mathbf{x}^{(0)}$  gegen den ersten Eigenvektor  $\mathbf{v}_1$ .

### KONVERGENZ

$\mathbf{e}_1$  ist Eigenvektor zu Eigenwert  $\lambda_1$ . Es gilt  $\tan(\mathbf{x}^{(0)}, \mathbf{e}_1) < \infty$ , also  $\mathbf{x}^{(0)}$  soll nicht senkrecht auf  $\mathbf{e}_1$  stehen. Dann gilt

$$\tan(\mathbf{x}^{(m)}, \mathbf{e}_1) \leq \left( \left| \frac{\lambda_2}{\lambda_1} \right| \right)^m \tan(\mathbf{x}^{(0)}, \mathbf{e}_1) \quad \forall m \in \mathbb{N}_0$$

Oder Argumentation über EW:  $|\lambda^{(k)} - \lambda_1| = \mathcal{O}\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)$ .

Sind  $|\lambda_2|$  und  $|\lambda_1|$  ähnlich groß, gibt es langsame Konvergenz. Falls  $\mathbf{A}$  symmetrisch ist, sind die EV orthogonal und dann gilt  $|\lambda^{(k)} - \lambda_1| = \mathcal{O}\left(\left|\frac{\lambda_2}{\lambda_1}\right|^{2k}\right)$ .

**Numerische Probleme:** Iterationsfolge führt zu Vektoren mit sehr großen ( $|\lambda_1| > 1$ ) oder sehr kleinen Einträgen ( $|\lambda_1| < 1$ ).

**Lösung:** Normalisieren mit der Norm (also nur Skalierung):

$$\mathbf{y}^{(m)} = \mathbf{A} \mathbf{x}^{(m-1)} \quad \gamma^{(m)} = \|\mathbf{y}^{(m)}\|_2 \quad \mathbf{x}^{(m)} = \mathbf{y}^{(m)} / \gamma^{(m)} \quad \forall m \in \mathbb{N}$$

### RAYLEIGH-QUOTIENT

Rayleigh-Quotient zu  $\mathbf{A}$  ist gegeben durch

$$\Lambda_A : \mathbb{R}^n \setminus \{\mathbf{0}\} \rightarrow \mathbb{R} \quad \mathbf{x} \mapsto \frac{\langle \mathbf{A} \mathbf{x}, \mathbf{x} \rangle_2}{\langle \mathbf{x}, \mathbf{x} \rangle_2}$$

Falls  $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$  ein Eigenvektor von  $\mathbf{A}$  zu  $\lambda \in \mathbb{R}$  ist gilt

$\Lambda_A = \lambda$ . Mit der Cauchy-Schwarz-Ungleichung

$|\langle \mathbf{x}, \mathbf{y} \rangle_2| \leq \|\mathbf{x}\|_2 \|\mathbf{y}\|_2 \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$  lässt sich die Genauigkeit der Näherung des Eigenwerts abschätzen über:

$$|\Lambda_A(\mathbf{x}) - \lambda| \leq \|\mathbf{A} - \lambda \mathbf{I}\|_2 \sin(\mathbf{x}, \mathbf{e}) \leq \|\mathbf{A} - \lambda \mathbf{I}\|_2 \tan(\mathbf{x}, \mathbf{e})$$

mit  $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$  als Näherung des Eigenvektors  $\mathbf{e} \in \mathbb{R}^n$  von  $\mathbf{A}$  zu  $\lambda \in \mathbb{R}$ .

**Quadratische Konvergenz:** Falls  $\mathbf{A} = \mathbf{A}^*$ , ergibt sich die Abschätzung

$$|\Lambda_A(\mathbf{x}) - \lambda| \leq \|\mathbf{A} - \lambda \mathbf{I}\|_2 \sin^2(\mathbf{x}, \mathbf{e})$$

Näherung des Eigenwerts kann wesentlich schneller als die des Eigenvektors konvergieren.

power\_adaptive

---

```

 $\gamma \leftarrow \|\mathbf{x}\|_2, \mathbf{x} \leftarrow \mathbf{x}/\gamma$ 
 $\mathbf{y} \leftarrow \mathbf{A}\mathbf{x}$ 
 $\lambda \leftarrow \langle \mathbf{y}, \mathbf{x} \rangle_2$ 
while  $\|\lambda\mathbf{x} - \mathbf{y}\|_2 > \epsilon\|\mathbf{y}\|_2$ 
   $\gamma \leftarrow \|\mathbf{y}\|_2, \mathbf{x} \leftarrow \mathbf{y}/\gamma$ 
   $\mathbf{y} \leftarrow \mathbf{A}\mathbf{x}$ 
   $\lambda \leftarrow \langle \mathbf{y}, \mathbf{x} \rangle_2$ 

```

---

$\mathbf{y}$  wird für die Berechnung von  $\Lambda_A$ , die Prüfung auf Konvergenz und für die Bestimmung der nächsten Iterierten genutzt.

## 4.2 Inverse Iteration

*Eignet sich für die Berechnung des kleinsten Eigenwerts (geben die niedrigsten Frequenzen für Resonanzeffekte an).*  
Wenn  $\mathbf{A}$  regulär, gilt:

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x} \quad \Leftrightarrow \quad \mathbf{x} = \lambda\mathbf{A}^{-1}\mathbf{x} \quad \Leftrightarrow \quad \frac{1}{\lambda}\mathbf{x} = \mathbf{A}^{-1}\mathbf{x}$$

Also ist ein Eigenvektor von  $\mathbf{A}$  zu  $\lambda$  auch ein Eigenvektor von  $\mathbf{A}^{-1}$  zu  $1/\lambda$ . Damit ist der betragskleinste EW von  $\mathbf{A}$  der Kehrwert des betragsgrößten EWs von  $\mathbf{A}^{-1}$ .

**Inverse Iteration:**

$$\mathbf{y}^{(m)} = \mathbf{A}^{-1}\mathbf{x}^{(m-1)} \quad \gamma^{(m)} = \|\mathbf{y}^{(m)}\|_2 \quad \mathbf{x}^{(m)} = \mathbf{y}^{(m)}/\gamma^{(m)} \quad \forall m \in \mathbb{N}$$

Inverse von  $\mathbf{A}$  umgehen mit Lösung von  $\mathbf{A}\mathbf{y}^{(m)} = \mathbf{x}^{(m-1)}$ .  
MIT SHIFT

Falls  $\mu \in \mathbb{R}$  kein Eigenwert von  $\mathbf{A}$ , dann ist  $\mathbf{B} = (\mathbf{A} - \mu\mathbf{I})$  regulär. Mit EW  $\lambda \in \mathbb{R}$  und EV  $\mathbf{x} \in \mathbb{R}^n$ . Dann ist  $\frac{1}{\lambda - \mu}$  ein EW von  $\mathbf{B}^{-1}$ . Der betragsgrößte EW von  $\mathbf{B}^{-1}$  korrespondiert mit dem EW von  $\mathbf{A}$ , der  $\mu$  am nächsten liegt. **Inverse Iteration mit Shift**

$$\mathbf{y}^{(m)} = (\mathbf{A} - \mu\mathbf{I})^{-1}\mathbf{x}^{(m-1)} \quad \gamma^{(m)} = \|\mathbf{y}^{(m)}\|_2$$

$$\mathbf{x}^{(m)} = \mathbf{y}^{(m)}/\gamma^{(m)} \quad \forall m \in \mathbb{N}$$

invit\_adaptive

---

Faktorisierung von  $\mathbf{B}$  berechnen.

```

 $\gamma \leftarrow \|\mathbf{x}\|_2, \mathbf{x} \leftarrow \mathbf{x}/\gamma$ 
Löse  $(\mathbf{A} - \mu\mathbf{I})\mathbf{y} = \mathbf{x}$ 
 $\lambda \leftarrow \langle \mathbf{y}, \mathbf{x} \rangle_2$ 
while  $\|\lambda\mathbf{x} - \mathbf{y}\|_2 > \epsilon\|\mathbf{y}\|_2$ 
   $\gamma \leftarrow \|\mathbf{y}\|_2, \mathbf{x} \leftarrow \mathbf{y}/\gamma$ 
  Löse  $(\mathbf{A} - \mu\mathbf{I})\mathbf{y} = \mathbf{x}$ 
   $\lambda \leftarrow \langle \mathbf{y}, \mathbf{x} \rangle_2$ 

```

---

QR/LR-Z muss nur einmal berechnet werden, danach relativ geringer Aufwand. Rayleigh-Quotient  $\lambda$  wird gegen EW von  $\mathbf{B}^{-1}$  konvergieren - rekonstruieren des originalen EW von  $\mathbf{A}$  über  $1/\lambda + \mu$ .

KONVERGENZ

$$\tan(\mathbf{x}^{(m)}, \mathbf{e}^{(1)}) \leq \left( \frac{|\lambda_1 - \mu|}{|\lambda_2 - \mu|} \right)^m \tan(\mathbf{x}^{(0)}, \mathbf{e}^{(1)})$$

RAYLEIGH-ITERATION

Je näher  $\mu$  an  $\lambda_1$ , desto schnellere Konvergenz gegen den EV. Wenn  $\mathbf{x}^{(m)}$  eine gute Näherung eines EVs ist, wird  $\Lambda_A(\mathbf{x}^{(m)})$  eine gute Näherung des entsprechenden EWs sein (a.k.a **guter Shift-Parameter**).

**Rayleigh-Iteration:**

$$\mu^{(m)} = \Lambda_A(\mathbf{x}^{(m-1)})$$

$$\mathbf{y}^{(m)} = (\mathbf{A} - \mu^{(m)}\mathbf{I})^{-1}\mathbf{x}^{(m-1)}, \quad \mathbf{x}^{(m)} = \frac{\mathbf{y}^{(m)}}{\|\mathbf{y}^{(m)}\|_2}, \quad \forall m \in \mathbb{N}$$

invit\_rayleigh

---

```

 $\gamma \leftarrow \|\mathbf{x}\|_2, \mathbf{x} \leftarrow \mathbf{x}/\gamma$ 
Löse  $(\mathbf{A} - \mu\mathbf{I})\mathbf{y} = \mathbf{x}$ 
 $\lambda \leftarrow \langle \mathbf{y}, \mathbf{x} \rangle_2$ 
 $\mu \leftarrow 1/\lambda + \mu$ 
while  $\|\lambda\mathbf{x} - \mathbf{y}\|_2 > \epsilon\|\mathbf{y}\|_2$ 
   $\gamma \leftarrow \|\mathbf{y}\|_2, \mathbf{x} \leftarrow \mathbf{y}/\gamma$ 
  Löse  $(\mathbf{A} - \mu\mathbf{I})\mathbf{y} = \mathbf{x}$ 
   $\lambda \leftarrow \langle \mathbf{y}, \mathbf{x} \rangle_2$ 
   $\mu \leftarrow 1/\lambda + \mu$ 

```

---

$\mu$  abhängig von  $m$ , daher in jeder Schleife QR/LR-Z berechnen - wesentlich aufwendiger. Allerdings sehr schnelle (quadratische) Konvergenz bei guter Näherung - jeder Schritt verdoppelt Anzahl korrekt berechneter Stellen.

## 4.3 Orthogonale Iteration

*Berücksichtigung k-facher Eigenwerte. Keine Probleme bei mehrfachen oder eng beieinanderliegenden Eigenwerten.*

*Praktisch identisch zu Vektoriteration.*

Konvergenz gegen von  $\mathbf{e}^{(1)}, \dots, \mathbf{e}^{(k)} \in \mathbb{R}^n$  (zu  $\lambda_1, \dots, \lambda_k$ ) aufgespannten Teilraum. Zusammenfassen von  $k$  Iterierten in Matrix  $\mathbf{X}^{(m)} \in \mathbb{R}^{n \times k}$

$$\mathbf{X}^{(m)} = \mathbf{A}^m \mathbf{X}^{(0)} \quad \text{bzw.} \quad \mathbf{X}^{(m+1)} = \mathbf{A} \mathbf{X}^{(m)} \quad \forall m \in \mathbb{N}_0$$

Spalten orthogonaler Matrix bilden orthonormale Basis und konvergieren nicht gegen denselben Raum (Einhaltung der LU).  $\mathbf{X}^{(m)}$  durch  $\mathbf{Q}^{(m)} \in \mathbb{R}^{n \times k}$  und  $\mathbf{R}^{(m)} \in \mathbb{R}^{k \times k}$  ersetzen:

$$\mathbf{X}^{(m)} = \mathbf{Q}^{(m)} \mathbf{R}^{(m)} \quad \forall m \in \mathbb{N}_0$$

Vermeiden der QR-Z von instabilen Matrizen  $\mathbf{X}^{(m)}$ . Daher

$$\mathbf{Y}^{(m+1)} = \mathbf{A} \mathbf{Q}^{(m)} \quad \mathbf{Y}^{(m+1)} = \mathbf{Q}^{(m+1)} \widehat{\mathbf{R}}^{(m+1)}$$

und

$$\mathbf{X}^{(m+1)} = \mathbf{Q}^{(m+1)} \mathbf{R}^{(m+1)} \quad \mathbf{R}^{(m+1)} = \widehat{\mathbf{R}}^{(m+1)} \mathbf{R}^{(m)}$$

Die Folge  $(\mathbf{Q}^{(m)})_{m=0}^\infty$  heißt orthogonale Iteration.  
**orthoit\_rayleigh**

---

Berechne  $\mathbf{Q}\widehat{\mathbf{R}} = \mathbf{X}$

$\mathbf{Y} \leftarrow \mathbf{A}\mathbf{Q}$

$\Lambda \leftarrow \mathbf{Q}^* \mathbf{Y}$

while  $\|\mathbf{Q}\Lambda - \mathbf{Y}\|_2 > \epsilon$

  Berechne  $\mathbf{Q}\widehat{\mathbf{R}} = \mathbf{Y}$

$\mathbf{Y} \leftarrow \mathbf{A}\mathbf{Q}$

$\Lambda \leftarrow \mathbf{Q}^* \mathbf{Y}$

$\mathbf{X} \leftarrow \mathbf{Q}$

---

Verallgemeinerung der Vektoriteration. Statt einzelнем Vektor -  $k$ -spaltige Matrix. Statt Iterierte zu EV zu machen - Orthonormalbasen verwenden.

Aufwand:  $nk^2$  AO/Schritt.

VERFEINERUNGEN

Inverse Iteration, Inverse Iteration mit Shift,

Rayleigh-Iteration

Deflation - entferne bereits konvergierte EV.

## 4.4 QR-Iteration

*Alle EW und EV einer Matrix berechnen. Vorüberlegung:*  
Multiplikation mit orthonormaler (orthog. +  $\|\cdot\|$  aller Vektoren = 1) Matrix  $\mathbf{Q}$  heißt Ähnlichkeitstransformation - EW von  $\mathbf{A}$  bleiben erhalten.  $\mathbf{A}\mathbf{x} = \lambda\mathbf{x} \Leftrightarrow \mathbf{Q}^* \mathbf{A} \mathbf{Q} \mathbf{Q}^* \mathbf{x} = \lambda \mathbf{Q}^* \mathbf{x}$ . **Idee:** Durch MP mit  $\mathbf{Q}$ , z.B.  $\mathbf{A}$  auf oberer  $\Delta$ Matrix bringen, um einfacher die EW zu berechnen.

ZUSAMMENFASSUNG UNTERRAUMITERATION

$(\mathbf{Q}^{(m)})^* \mathbf{A} \mathbf{Q}^{(m)} = \mathbf{A}^{(m)} \rightarrow \mathbf{R}$  für  $m \rightarrow \infty$ . Diagonaleinträge in  $\mathbf{A}^{(m)}$  sind Approximationen für die EW von  $\mathbf{A}$ . Für  $k \rightarrow \infty$  streben die Fehler der Approximationen gegen 0.

Konvergenzgeschwindigkeit durch  $\left| \frac{\lambda_{l+1}}{\lambda_l} \right|, l = \{1, \dots, m\}$  gegeben. Die EWs stehen der Größe nach sortiert auf der Diagonalen von  $\mathbf{R}$ .

QR-ITERATION

$\mathbf{A}^{(m)}$  aus Unterraumiteration rekursiv (also  $\mathbf{A}^{(m)}$  aus  $\mathbf{A}^{(m-1)}$  berechnen.

Sei  $\tilde{\mathbf{A}}^{(m-1)} = \mathbf{Q}\mathbf{R}$  und  $\tilde{\mathbf{A}}^{(m)} = \mathbf{R}\mathbf{Q}$ , dann gilt

$$\tilde{\mathbf{A}} = \mathbf{A}^{(m)} \quad \forall m = 0, 1, 2, \dots,$$

Falls für irgendein  $k \in \{1, \dots, n-1\}$   $|\lambda_{k+1}| < |\lambda_k|$  gilt, werden die unteren  $n-k$  Zeilen der ersten  $k$  Spalten gegen Null konvergieren.

Ziel:  $\tilde{\mathbf{A}}^{(0)}$  tridiagonal, um schnell zu rechnen. (Aufwand  $\approx n$  pro Schritt)

Aufwand: 1 QR-Z pro Schritt. Aufwand proportional zu  $n^3$ . Bei sinnvollem Shift-Parameter kubische Konvergenz. Konvergierte Teilmatrizen werden nicht ausgenutzt und langsame Konvergenz bei nah beieinander liegenden EW.

## 5 Approximation von Funktionen

CAD, Bestimmung von Formeln zur numerischen Integration, numerische Differentiation, numerisches Lösen von DGL...

## 5.1 Polynominterpolation

Polynome höchstens  $m$ -ten Grades  $\Pi_m = \text{span}\{1, x, x^2, \dots, x^m\}$   
INTERPOLATIONS-AUFGABE  
Für gegebene Werte  $f_0, \dots, f_m$  und paarweise verschiedene Stützstellen  $x_0, \dots, x_m$  finde ein Polynom  $p \in \Pi_m$ , das erfüllt:

$$p(x_i) = f_i \quad \forall i \in \{0, \dots, m\}$$

LAGRANGE-POLYNOME

Für jedes  $i \in \{0, \dots, m\}$  ist  $l_i : \mathbb{R} \rightarrow \mathbb{R}$

$$x \mapsto \prod_{\substack{k=0 \\ j \neq i}}^m \frac{x - x_k}{x_i - x_k}$$

ein Polynom höchstens  $m$ -ten Grades.

$$l_i(x_j) = \begin{cases} 1 & \text{falls } i = j \\ 0 & \text{sonst} \end{cases} \quad l_i(x_i) = \prod_{\substack{k=0 \\ k \neq i}}^m \frac{x_i - x_k}{x_i - x_k} = 1$$

Für beliebige  $f_0, \dots, f_m$  löst  $p = \sum_{k=0}^m f_k l_k$  das Interpolationsproblem - eindeutig lösbar. Dieser Ansatz ist jedoch ineffizient. Stattdessen... N-A-Verf.

## 5.2 Neville-Aitken-Verfahren

Idee: Von konstanten Polynomen ausgehend Polynome höheren Grades zu konstruieren. Gut geeignet für Bestimmung an wenigen Stellen.

Für alle  $i, j \in \{0, \dots, m\}$  mit  $i \leq j$  existiert genau ein Polynom  $p_{i,j} \in \Pi_{j-i}$ , das  $p_{i,j}(x_k) = f_k \quad \forall k \in \{i, \dots, j\}$  erfüllt.

**Idee:** Interpolation höheren Grades lässt sich durch Konvexkombination von Polynomen niedrigeren Grades schreiben.

AITKEN-REKURRENZ

Sei  $i, j \in \{0, \dots, m\}$  mit  $i < j$ :

$$\begin{aligned} p_{i,j}(x) &= \frac{x - x_i}{x_j - x_i} p_{i+1,j}(x) + \frac{x_j - x}{x_j - x_i} p_{i,j-1}(x) \\ &= p_{i+1,j}(x) + \frac{x_j - x}{x_j - x_i} (p_{i,j-1} - p_{i+1,j}(x)) \end{aligned}$$

1. Konstante Polynome  $p_{i,i} = f_i$  bestimmen
2. Mit Aitken-Rekurrenz lineare, quadratische, etc., Polynome konstruieren
3. Bei Grad  $m$  ergibt sich  $p_{0,m}(x) = p(x)$ .

$$\begin{aligned} f_0 &= p_{0,0}(x) \\ f_1 &= p_{1,1}(x) & p_{0,1}(x) \\ f_2 &= p_{2,2}(x) & p_{1,2}(x) & p_{0,2}(x) \\ f_3 &= p_{3,3}(x) & p_{2,3}(x) & p_{1,3}(x) & p_{0,3}(x) = p(x) \end{aligned}$$

Algorithmus effizient: Spaltenweise von unten nach oben in-place.

neville

```
for n = 1 : m // Loope über Grad der Polynome
    for j = m : n // von unten nach oben in einer Spalte
        i ← j - n // "Obere Ecke" des Δ
        f_j ← ((x - x_i)f_j + (x_j - x)f_{j-1}) / (x_j - x_i)
    return f_m
```

Aufwand:  $\frac{7}{2}m(m+1)$  AO (quadratisch)

## 5.3 Newtons dividierte Differenzen

Auswertung in einigen Punkten ok bei quadratischem Aufwand. Bei Rendering bspw. jedoch zu aufwendig. Reduzierung des Aufwands durch Berechnung von Hilfsgrößen im Voraus.

**Idee:** Hat man  $p_{i,j-1}$  bereits bestimmt, so sucht man nach einem Korrekturterm, durch dessen Ergänzung man  $p_{i,j}$  erhält - Newton-Darstellung:

$$p_{i,j}(x) = p_{i,j-1}(x) + d_{i,j}(x - x_i) \dots (x - x_{j-1})$$

$d_{i,j}$  ist abhängig von den Stützstellen  $x_i$ .

Per Induktion folgt als Netwonsche Interpolationsformel:

$$p_{i,j}(x) = \sum_{k=i}^j d_{i,k} n_{i,k}(x) \text{ mit } n_{i,j}(x) = \begin{cases} 1 & i = j \\ \prod_{k=1}^{j-1} (x - x_k) & i < j \end{cases}$$

EFFIZIENTE GESTALTUNG

1.  $s_m(x) = d_{0,m}$  bestimmen
2.  $s_{m-1}(x)$  berechnen mit  $s_i(x) = d_{0,i} + (x - x_i)s_{i+1}(x)$
3. Stoppe, wenn  $s_0(x) = p(x)$  berechnet

eval\_newton

```
s ← d_m
for i = m - 1 : 0
    s ← d_i + (x - x_i)s
return s // returns s_0(x)
```

mit  $\mathbf{d} = (d_{0,0}, \dots, d_{0,m})$ .

Aufwand:  $3m$  AO.

NEWTONS DIVIDIERT E DIFFERENZEN

Newton'sche Interpolationsformel in Aitken-Rekurrenz einsetzen ergibt mit Koeffizientenvergleich der führenden Koeffizienten:

$$d_{i,j} = \frac{d_{i+1,j} - d_{i,j-1}}{x_j - x_i} \left( = f_j = \frac{f_j - f_{j-1}}{x_j - x_i} \right)$$

newton\_diff

```
for n = 1 : m
    for j = m : n // von unten nach oben in der Spalte
        i ← j - n
        f_j ← (f_j - f_{j-1}) / (x_j - x_i)
```

Überschreibt  $f_0, \dots, f_m$  mit  $d_{0,0}, \dots, d_{0,m}$ .

Aufwand:  $\frac{3}{2}m(m+1)$  - also quadratisch. Allerdings nur einmalige Ausführung.

## 5.4 Approximation von Funktionen

INTERPOLATIONSFEHLER

Sei  $f \in C^{m+1}[a, b]$ , sei  $p$  die Lösung der Interpolationsaufgabe und sei  $x \in [a, b]$ . Dann ex.  $\eta \in [a, b]$  mit

$$f(x) - p(x) = (x - x_0) \dots (x - x_m) \frac{f^{(m+1)}(\eta)}{(m+1)!}$$

UNABHÄNGIGE FEHLERSCHRANKE

$$\|f - p\|_{\infty, [a, b]} \leq \underbrace{\|(x - x_0) \dots (x - x_m)\|_{\infty, [a, b]}}_{\omega(x)} \frac{\|f^{(m+1)}\|_{\infty, [a, b]}}{(m+1)!}$$

TSCHEBYSCHEFF-INTERPOLATION

$$\hat{x}_i = \frac{b+a}{2} + \frac{b-a}{2} \cos\left(\pi \frac{2i+1}{2m+2}\right)$$

Bestmögliche Wahl für Interpolationspunkte, sodass

$\|\hat{\omega}\|_{\infty, [a, b]}$ .

STABILITÄTSKONSTANTE / BESTAPPROXIMATION

$$\|f - p\|_{\infty, [a, b]} \leq (\Lambda_m + 1) \|f - q\|_{\infty, [a, b]}$$

## 6 Numerische Integration

Anwendung in Berechnung von Integralen von Funktionen (Normalverteilung).

**Aufgabe:** Gegeben sind ein nicht-leeres Intervall  $[a, b]$  und eine stetige Funktion  $f \in C[a, b]$ , zu approximieren ist das Integral

$$\int_a^b f(x) dx$$

### 6.1 Quadraturformeln

QUADRATURFORMEL

Seien  $m \in \mathbb{N}_0$ , Punkte  $x_0, \dots, x_m \in [a, b]$  und  $w_0, \dots, w_m \in \mathbb{R}$  gegeben, dann definiert

$$\mathcal{Q}_{[a, b]} : C[a, b] \rightarrow \mathbb{R}, \quad \int_c^d f(x) dx = \sum_{i=0}^m w_i f(x_i)$$

eine Quadraturformel, die jeder Funktion  $f \in C[a, b]$  eine Approximation des Integrals durch Funktionswerte in den Quadraturpunkten  $x_0, \dots, x_m$  und mit den Quadraturgewichten  $w_0, \dots, w_m > 0$  zuordnet.

MITTELPUNKTREGEL

Gute Näherung des Integrals, falls  $f''$  und Intervall  $[a, b]$  klein.

$$f \mapsto \mathcal{M}_{[a, b]}(f) = (b - a) f\left(\frac{b+a}{2}\right) = \int_a^b f\left(\frac{b+a}{2}\right) dx$$

$$\mathcal{I}_{[a, b]}(f) - \mathcal{M}_{[a, b]}(f) = \frac{(b-a)^3}{24} f''(\eta)$$

Exakt für lineare Polynome.

INTERPOLATORISCHE QUADRATUR

$$\mathcal{I}_{[a,b]}(f) \approx \int_a^b p(x)dx = \int_a^b \sum_{i=0}^m f(x_i)l_i(x)dx = \sum_{i=0}^m f(x_i) \underbrace{\int_a^b l_i(x)dx}_{w_i}$$

NEWTON-COTES-QUADRATUR / TRAPEZREGEL

$$x_i = \frac{m-i}{m}a + \frac{i}{m}b, \forall i \in \{0, \dots, m\}, \quad \mathcal{Q}_{[a,b]}(f) = \frac{b-a}{2}(f(a) + f(b))$$

SUMMIERTE QUADRATURFORMEL

$$h = \frac{b-a}{l} \quad y_i = a + ih \quad \forall i \in \{0, \dots, l\}$$

$$\begin{aligned} \mathcal{I}_{[a,b]}(f) &= \int_a^b f(x)dx = \sum_{i=1}^l \int_{y_{i-1}}^{y_i} f(x)dx \\ &= \sum_{i=1}^l \mathcal{I}_{[y_{i-1}, y_i]}(f) \approx \sum_{i=1}^l \mathcal{Q}_{[y_{i-1}, y_i]}(f) = \mathcal{Q}_{[a,b],l}(f) \end{aligned}$$

Es kann jede beliebige Genauigkeit erreicht werden bei hinreichend großem  $l$ .

SUMMIERTE MITTELPUNKTREGEL

Jede beliebige Genauigkeit, falls  $f''$  beschränkt ist.

Verdoppelung der Anzahl der Teilintervalle viertelt den Fehler.

$$\begin{aligned} \mathcal{M}_{[a,b],l}(f) &= \sum_{i=1}^l \mathcal{M}_{[y_{i-1}, y_i]}(f) = h \sum_{i=1}^l f\left(\frac{y_i + y_{i-1}}{2}\right) \\ \mathcal{I}_{[a,b]}(f) - \mathcal{M}_{[a,b],l}(f) &= \frac{(b-a)^3}{24l^2} f''(\eta) \end{aligned}$$

**Beste Quadraturformel:** Gauß-Quadratur - exakt für

$n = 2m + 1$ .Q

## 6.2 Fehleranalyse

Definiere Quadraturformeln auf Referenzintervall:

$$\mathcal{I} : C[-1, 1] \rightarrow \mathbb{R} \quad f \mapsto \mathcal{I}(f) = \int_{-1}^1 f(x)dx$$

$$\mathcal{Q} : C[-1, 1] \rightarrow \mathbb{R} \quad f \mapsto \mathcal{Q}(f) = \sum_{i=0}^m w_i f(x_i)$$

Die **Quadraturformel** ist **exakt** von Grad  $n$ , wenn  $\mathcal{Q}(p) = \mathcal{I}(p) \quad \forall p \in \Pi_n$  - falls alle Polynome bis Grad  $n$  exakt integriert werden. Exaktheit ausnutzen: Approximiere den Integranden  $f$  durch  $p$  ( $p$  wird exakt integriert) - also nur Approximationsfehler beschränken über Maximumsnorm

$\|g\|_{\infty, [a,b]}$ .

Stabilitätskonstante  $C_Q = \sum_{i=0}^m |w_i|$ .