Zusammenfassung, Num Met FS26, Ruben Leuzinger

Allgemeines

Def (Fehler)

- Absoluter Fehler von $\tilde{x}$ ist $\|x-\tilde{x}\|$
- Relativer Fehler ist $\frac{\|x-\tilde{x}\|}{\|x\|}$ für $\|x\| \neq 0$

Maschinenpräzession

Computer stellen Zahlen diskret (binär) dar. D.h. nicht alle Zahlen können exakt dargestellt werden und es kommt zu Rundungsfehler.

Darstellung: Zahl $= d \cdot B^e$, d Mantisse, B basis

Auslöschung

Bei subtrahieren zweier ungefähr gleich grossen Zahlen, kann es zu drastischer vergrösserung des relativen Fehlers kommen (wegen Computerpräzession)

Ableitungen

Wegen Auslöschung ist $f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$ problematisch. Es gibt mehr alternative Methoden. Alle (wie auch DiffQuot) kommen von guter verwendung von Taylor.

I) **Imaginär** $f(x_0+ih) = f(x_0) + f'(x_0)ih - \frac{1}{2}\dots$

$\Rightarrow f'(x_0) \approx \frac{\operatorname{Im}(f(x_0+ih))}{h}$, kann mit h^2

II) Richardson kombinieren Taylor von $f(x_0+h)$ und $f(x_0-h)$

$$\Rightarrow f'(x) = \underbrace{\frac{f(x+h)-f(x-h)}{2h}}_{:= d(h)} + \frac{1}{6}f'''(x)h^2 + \dots$$

$$f'(x) = \underbrace{\frac{4d(h/2)-d(h)}{3}}_{d_1(h)} + c_2h^4 + \dots$$

etc. wir finden $f'(x) = R_{\ell,k} + C\left(\frac{h}{2^\ell}\right)^{2k+2}$

wobei $R_{\ell,k} := \dfrac{4^k R_{\ell,k-1} - R_{\ell-1,k-1}}{4^k-1}$, $R_{\ell,0} = d\left(\frac{h}{2^\ell}\right)$

(rekursiv definiert)

Quadratur

Bei der Quadratur geht es um numerische Berechnungen von Integralen. Dabei suchen wir Knoten $\{c_j\}$ und Gewichte $\{w_j\}$, so dass der Fehler

$E(n) = \left|\int_a^b f(x)\,dx - Q_n(f;a,b)\right|$ möglichst klein ist,

wobei $Q_n(f;a,b) := \sum_{j=1}^{n} f(c_j)w_j$.

Konvergenz

- algebraisch, falls $E(n) = O\left(\frac{1}{n^p}\right)$
- exponentiell, falls $E(n) = O(q^n)$, $0 \le q < 1$

Big-O Notation

$f(n) = O(g(n)) \Rightarrow \exists C \in \mathbb{R}, N \in \mathbb{N}$ s.d.
$\forall n \geq N \quad |f(n)| < C|g(n)|$

Def (Ordnung von QF)

Eine Quadraturformel besitzt Ordnung $n+1$, wenn sie Polynome vom Grad n exakt integriert.
Wir sagen die Quadraturgenauigkeit ist n.

Def (Symmetrisch)

Eine QF auf $[-1,1]$ heisst symmetrisch, falls gilt

$$w_i = w_{n+1-i} \quad \text{und} \quad c_i = -c_{n+1-i}$$

wobei w_i die Gewichte und c_i die Knoten sind

Theorem

- Die Ordnung einer symmetrischen QF ist gerade
- Für $f \in C^n[a,b]$ gilt folgende obere Schranke

$$\left|\int_a^b f(x)\,dx - Q_n(f)\right| \leq \frac{1}{n!}(b-a)^{n+1}\|f^{(n)}\|_{L^\infty([a,b])}$$

- Für eine zusammengesetzte QF Q, die auf QF der Ordnung $p \in \mathbb{N}$ basiert $\exists C > 0$ s.d.

$$\left|\int_a^b f(x)\,dx - Q(f)\right| \le C h^p \|f^{(p)}\|_{L^\infty([a,b])},$$

wobei h die Länge der Teilintervalle ist.

QF Äquidistanter Knoten

- Mittelpunktsregel

$$Q^M(f;a,b) = (b-a)\, f\left(\frac{a+b}{2}\right)$$

und dazugehörige zusammengesetzte MPR

$$Q(f;a,b) = \sum_{i=0}^{N-1} \frac{(b-a)}{N} f\left(\frac{(2i+1)\cdot(b-a)}{2N}\right), \quad h = \frac{b-a}{N}$$

Ordnung 2 und Fehler $\sim O(h^2)$

- Trapezregel

$$Q^T(f;a,b) = \frac{b-a}{2}\left(f(a)+f(b)\right)$$

$$Q(f;a,b) = \frac{b-a}{2}\left(f(a)+f(b) + 2\sum_{i=1}^{N-1} f\left(\frac{i}{N}(b-a)\right)\right)$$

Ordnung ist 2, Fehler $\sim O(h^2)$

- Simpsonregel

$$Q^S(f;a,b) = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)$$

$$= \frac{1}{3} Q^T + \frac{2}{3} Q^M$$

Ordnung ist 4 und Fehler $\sim O(h^4)$

- Newton Cotes (Äquidistant höhere Ordnung) nicht verwenden.

Romberg Schema

Für die Trapezregel ist der Fehler gegeben durch $T(h) - I = c_1 h^2 + c_2 h^4 + \dots$

Ähnlich wie bei der Richardson Konvergenzbeschleunigung, kann man also rekursiv mit nicht viel kleinen Integralen eine deutlich bessere Ordnung erreichen.

↳ Intuitiv werden Fehler ausgelöscht durch geschickte Addition.

$$R_{e,n} - \int_a^b f(x)\,dx = C\left(\frac{h}{2^e}\right)^{2n+2}$$

Quadratur in $\mathbb{R}^d$ $(\Omega \subset \mathbb{R}^d)$

$$\int_\Omega f(x)\,dx \approx \sum_{i_1=1}^{n_1} \dots \sum_{i_d=1}^{n_d} \omega_{i_1} \dots \omega_{i_d} f(c_{i_1}, \dots, c_{i_d})$$

Adaptive Quadratur

Idee: Nehme zwei QF unterschiedlicher Ordnung. Falls der Unterschied zwischen den beiden QF lokal gross ist, wählen wir lokal die Teilintervalle kleiner. Falls der Unterschied klein ist, wählen wir Teilintervalle gross.

Gauss-Quadratur

Idee: Knoten und Gewichte (Nicht äquidistant) so wählen, das Ordnung $p = s+m$ maximal ist für s Knoten $c_1 \dots c_s$.
D.h. Polys $\leq p-1$ werden exakt integriert.

Polynomdiv

Sei $f \in \text{Polys}_{\leq p-1}$, $M(t) = (t-c_1)\dots(t-c_s) \in \text{Polys}_s$

$f(t) = M(t)\,g(t) + r(t)$, $r \in \text{Polys}_{\leq s-1}$

D.h. falls QF Polys $\leq s+m-1$ exakt integriert gilt

$$\int_0^1 f(t)\,dt = \sum_{i=1}^{s} f(c_i) w_i = \sum_{i=1}^{s} w_i \left(M(c_i) g(c_i) + r(c_i) \right)$$
$$= \sum_{i=1}^{s} w_i\, r(c_i) = \int_0^1 r(t)\,dt$$

D.h. $\langle M, g \rangle = \int_0^1 M(t)\,g(t)\,dt = 0$

Theorem

Ordnung der QF ist $s+m \iff \langle M, g \rangle = 0$
$\forall g \in \text{Polys}_{\leq m-1}$, $M(t) = (t-c_1)\dots(t-c_s)$ $(M \perp g)$

Theorem

Ordnung einer QF mit s Knoten ist höchstens $2s$.

Legendre-Polynome

Die Legendre Polynome $P_i(x)$ sind orthogonal zu allen Polys $< i$, somit sind sie deren Nullstellen gut für Gauss-Knoten geeignet.

Radau & Lobatto

Bei Radau wird einer der Enden des Intervalls als Knoten definiert.
Bei Lobatto werden beide Ende als Knoten festgelegt.
Wir nennen QF bei den Endpunkte keine Knoten sind offen.

Fehler der Gauss-Quadratur

$$\int_a^b f(x)dx - \sum_{j=1}^{s} w_j f(c_j) = \frac{(b-a)^{2s-1}}{2s!} f^{(2s)}(z) \quad , z \in [a,b]$$

$$\left|\int_a^b f(x)dx - \sum_{n=1}^{N} G_s(f, x_{n-1}, x_n)\right| \le C \cdot h^{2s} \max_{z\in[a,b]} |f^{(2s)}(z)|$$

Bemerkung

Die Gauss-Quadratur verwendet Legendre-Polynome, es gibt noch andere gute QF, die z.B. Chebyschev I/II, Jacobi, Hermit und Laguerre.

Trigonometrische Interpolation

Def: Trigonometrisches Polynom

Das Trig-Poly vom Grad $\le m$ ist

$$p_m(t) = \sum_{j=-m}^{m} \gamma_j e^{2\pi i j t} \quad \gamma_j \in \mathbb{C}, \ t \in \mathbb{R}$$

Bem $p_m(t)$ ist 1-Periodisch und falls $\overline{\gamma_{-j}} = \gamma_j$ ist $p_m(t)$ reell und kann als Summe von sin, cos dargestellt werden.

Bem $\varphi_n(x) = e^{2\pi i n x}$ sind orthogonale Basis für L^2-fkt

Theorem (Fourier)

Jede fkt $f \in L^2(0,1)$ ist der L^2-GW (Achtung nicht PW → siehe Gibbs) ihrer Fourier Reihe $f(t) = \sum_{n=-\infty}^{\infty} \hat{f}(n) e^{2\pi i n}$

mit $\hat{f}(n) = \int_0^1 f(t) e^{-2\pi i n t} dt$, $n \in \mathbb{Z}$

Parseval

Es gilt dabei $\sum_{n=-\infty}^{\infty} |\hat{f}(n)|^2 = \|f\|^2_{L^2(0,1)}$

Bem Da $\|f\|^2_{L^2(0,1)}$ wohldefiniert ist, muss die linke Seite konvergieren, weshalb $|\hat{f}(n)|$ schneller als $\frac{1}{\sqrt{n}}$ fallen müssen.

Theorem (ableitung mit Fourier)

seien f, f' int-bar auf $(0,1)$ dann gilt $\hat{f'}(n) = 2\pi i n \hat{f}(n)$ für $n \in \mathbb{Z}$

Bem Mit Parseval folgt, dass je glatter die Fkt, desto schneller müssen die

Fourier-koeffizienten fallen.

Explizit gilt $\int_0^1 |f^{(n)}(t)|\,dt < \infty \Rightarrow \hat{f}(k) = O(k^{-n})$

Gibbs-Phänomen

Für unstetigkeiten, z.B. bei square-waves haben wir keine Punktweise konvergenz und eine typischen overshoot von ca 9%

Fourierkoeffizienten mit Trapez

Da wir FK typischerweise nicht analytisch berechnen können, verwenden wir Approximation via Trapezregel

$$\hat{f}(k) \approx \hat{f}_N(k) := \frac{1}{N}\sum_{\ell=0}^{N-1} f(t_\ell)\, e^{-2\pi i k t_\ell} \quad \left(\text{z.B. } t_\ell = \frac{\ell}{N}\right)$$

wobei wir hier Periodizität $f(0)=f(1)$ voraussetzen.

Somit können wir approximieren

$$f(t) \approx \sum_{k=-N/2}^{N/2-1} \hat{f}_N(k)\, e^{2\pi i k t} := p_{N-1}(t)$$

Bem: Wir können zeigen, dass $f(\frac{\ell}{N}) = p_{N-1}(\frac{\ell}{N})$

D.h. die obige Approximation ist exakt für die Knotenpunkte. ⇝ Interpolation

Def (Fourier Matrix)

Matrix, welche Koeffizienten einer Darstellung in Standardbasis auf Fourier-koeffizienten abbildet. (bis auf Faktor 1/N)

$$F_N = V^H = \begin{pmatrix} \omega_N^0 & \cdots & \cdots & \omega_N^0 \\ \vdots & \omega_N^1 & \omega_N^2 & \vdots \\ \vdots & \omega_N^2 & \ddots & \\ \omega_N^0 & \omega_N^{N-1} & \cdots & \omega_N^{(N-1)^2} \end{pmatrix} = \left(\omega_N^{jk}\right)_{j,k=0}^{N-1} \in \mathbb{C}^{N\times N}$$

wobei $\omega_N = \exp\left(\frac{-2\pi i}{N}\right)$ die N-te Einheitswurzel ist. Es gilt $F_N^T = F_N \neq F_N^H$

Def (DFT)

Die diskrete Fouriertransformation kurz DFT ist die lin. Abb. $\mathcal{F}_N : \mathbb{C}^N \to \mathbb{C}^N$, $\mathcal{F}_N(\vec{y}) = F_N \vec{y}$

$$\vec{c} = \mathcal{F}_N(\vec{y}) \Rightarrow c_k = \sum_{j=0}^{N-1} y_j \, \omega_N^{kj}, \quad k = 0 \dots N-1$$

Bem Es gilt $\frac{1}{\sqrt{N}} F_N$ ist unitär, d.h.

$$F_N^{-1} = \frac{1}{N} F_N^H = \frac{1}{N} \overline{F_N}$$

Bem $\frac{1}{N} F_N F_N \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_{N-1} \\ \vdots \\ y_1 \end{pmatrix}$ } switched

Daraus folgt $\frac{1}{N^2} F_N^4 = \mathbb{1} \Rightarrow \sigma\left(\frac{1}{\sqrt{N}} F_N\right) = \{\pm 1, \pm i\}$

Bem. Für uns wäre $\vec{y}$ der Vektor, der die Messpunkte $f\left(\frac{\ell}{N}\right)$ enthält und c der Vektor der Approximierten Fkt.

Bem (Numpy) $c = \mathrm{fft}(y)$, $y = \mathrm{ifft}(c)$

Diese fkt geben die Koeffizienten von Null aufsteigend und dann von Maximal negativ aufsteigend an.

Theorem (zur Diag. zirkulanter Matrizen)

Die Elemente der Trig-Basis, sind Eigenvektoren der Shift-Matrix

$$S_N = \begin{pmatrix} 0 & 1 \\ \mathbb{1}_{N-1} & 0 \end{pmatrix} \quad \text{und} \quad C = \begin{pmatrix} c_0 & c_{N-1} & \cdots & c_1 \\ c_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & c \\ c_{N-1} & \cdots & c_1 & c_0 \end{pmatrix}$$

also Zirkulante Matrizen werden durch $p(S_N)=C$ gefunden, wobei $p(z)=c_0 z^0 + \dots c_{N-1} z^{N-1}$ gilt. Es gilt $S_N = UDU^H$ mit $U=\frac{1}{\sqrt{N}}V$, $D=\mathrm{diag}(\omega_N^{-k})$ ist. D.h. auch, es gilt $C = U p(D) U^H$

↳ Idk why relevant...

More in Script... FFT ~ $O(N \log N)$

Zero-Padding

Um eine höhere Auflösung im Frequenzbereich zu erzielen, können wir die Fourierkoeffizienten für grössere Frequenzen zu Null setzen. Das ändert nichts am Signal, jedoch wird die Auflösung der Fourierkoeffizienten künstlich erhöht, was eine bessere Darstellung im Frequenzraum ermöglicht.

Fehlerabschätzung

Es gilt für $f \in C^p, p \geq 2$ und 1-Periodisch, dass

$$\hat{p}_N(k) - \hat{f}(k) = O(N^{-p}) \quad \text{für } |k| \leq \frac{N}{2}$$

Falls $\sum_{k\in\mathbb{Z}} \hat{f}(k)$ absolut konvergiert, gilt zudem

$$|p_N(x) - f(x)| \leq 2 \sum_{|k| \geq N/2} |\hat{f}(k)| \quad \forall x \in \mathbb{R}$$

Abtasttheorem

Falls f eine 1-Periodischen Fkt mit maximaler Frequenz m ist, d.h. $\hat{f}(|k|>m)=0$ und $N>2m$, so gilt $p_N(x)=f(x)\ \forall x$

Fehlerabschätzung #2

Sei $k\in\mathbb{N}$ $f^{(k)}\in L^2(0,1)$ dann gilt

$$\|f-p_N(f)\|_{L^2(0,1)} \le \sqrt{1+c_k}\, N^{-k}\, \|f^{(k)}\|_{L^2(0,1)}$$

mit $c_k = 2\sum_{l=1}^{\infty}(2l-1)^{-2k}$

Spektralverfahren für DGLs

Idee: Numerisches Verfahren zum lösen von DGL, wobei Fkt als Linearkombination von Basisfkt (mit dem gesamten Bereich als Träger) dargestellt werden.

Typischerweise haben wir exponentielle Konvergenz, einfache Geometrien und Orthogonalität bzg eines vorhandenen Differentialoperators.

Für periodischen Problem → Fourier

Nicht periodischen Problem können mit sponge layers behoben werden. (absorbierenden Termen)

BSP: 1D-Transportgleichung

$u(x,t):[0,T]\times[0,1]\to\mathbb{R}$ s.d.

$$\begin{cases} \partial_t u(t,x)+c(x)\partial_x u(t,x)=0 \\ u(0,x)=e^{-100(2\pi x-1)^2} \end{cases}$$

Bem: Für allgemeine Problem können wir $[0,1]\to[0,L]$ mappen. $\partial_x u = L\,\partial_y u \quad x\in[0,1],\ y\in[0,L]\ y=Lx$

Def (Kollokation)

DGL wird an Punkten (x_n) exakt erfüllt.

Lösen der Transportgleichung

Cauchy-Kowalewskaja: $\partial_t u(x,t) = -c(x)\frac{\partial}{\partial x}u(t,x)$

→ zweite Ableitung ∂_t^2 mit Kettenregel

Lax-Wendroff: Einsetzen in Taylor um $u(t+h)$ mit $u(t,x)$ etc zu Beginn. Start bei $u(0,x)$

⇒ Iterativ finden wir $u(t,x)\ \forall t$

(Nicht immer stabil)

CFL-Bedingung: $h \leq \frac{\Delta x}{\max|c(x)|}$, typisch $h = \frac{1}{2}\frac{\Delta x}{\max|c(x)|}$

Sponge Layer

Idee: Um Aperiodizitäts-Probleme zu lösen brauchen wir einen Sponge-Term, der wie $\partial_t u(x,t) = -c(x)\partial_x u(x,t) - \sigma(x)u(x,t)$ aussieht und die Welle ausserhalb eines gewissen zu analysierenden Bereichs $\sigma \neq 0$ ist, s.d. die Amplituden dort exponentiell abfällt und damit periodizität die Lösung nicht stark beeinflusst.

Andere Bsp

∘) mehrdimensionale Transportgleichung

∘) Plane-Wave-Expansion $c(x) = $ konst

⇒ $u(t,x) = v(x)e^{\lambda t}$ ⇝ EW-Problem

Def (ODE)

Eine gewöhnliche DGL erster Ordnung hat die allg. Form $\dot{\vec{y}} = f(t,\vec{y})$

gewöhnlich ⟺ nur Ableitung einer Variable (hier t)

falls $\vec{y}$ ein mehrdimensionaler Vektor ist, also $y \in \mathbb{R}^d$, $f : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ sprechen wir von System von ODEs.

Def (autonom)

ODE ist autonom, falls RHS zeitunabhängig ist.

Def (Ordnung)

ODE n-ter Ordnung ist $\vec{y}^{(n)} = f(t,\vec{y},\dots,\vec{y}^{(n-1)})$

Def (AWP n-ter Ordnung)

AWP n-ter Ordnung ist ODE n-ter Ordnung zusammen mit $\vec{y}_i = \vec{y}^{(i)}(t_0)$

Idee:

ODEs n-ter Ordnung als System von n ODEs erster Ordnung schreiben.

Gleiche Idee verwenden um ODE autonomisieren.

Daraus folgt, dass wir nur das Lösen von autonomen ODEs erster Ordnung beherrschen müssen.

Def (Propagator/Evolutionsoperator)

$\Phi^{s,t}$ heisst Evo-Op zur DGL $\dot{\vec{y}} = \vec{f}(\vec{y},t)$, falls

$\Phi^{s,t}\vec{y}_0 = \vec{y}(t)$ ist Lösung des AWP $\vec{y}(s) = \vec{y}_0$

Eigenschaften des Propagators

- $\Phi^{t,t} = \mathrm{Id}$
- $\Phi^{s,r} = \Phi^{t,r} \circ \Phi^{s,t}$
- Für Autonom ODEs gilt
  $\Phi^{s,t} = \Phi^{0,t-s} := \Phi^{t-s}$

Ziel: Numerisch Approx Ψ von Φ finden

Def Fixpunkt

y^* heisst Fixpunkt von f, falls $f(t,y^*) = 0 \ \forall t$

Linearisieren von AWP

Idee: Taylor um Fixpunkt anwenden.
wir finden $\dot{\vec{y}} = A\vec{y} + \vec{g}(t)$ falls $A = S\,\mathrm{diag}(\lambda_i)S^{-1}$ diagonalisierbar ist und $g = 0$ (autonom), finden wir $\vec{y}(t) = S\,\mathrm{diag}(e^{\lambda_i t})S^{-1}\vec{y}_0 = e^{At}\vec{y}_0$

Def (erstes Integral)

Ein erstes Integral ist eine Funkt, s.d.
$I(\vec{y}(t)) = \text{const}$ $\forall$ Lösungen $\vec{y}(t)$ der ODE

Satz

diff'bares erstes Integral $\iff \operatorname{grad} I(\vec{y})\vec{f}(t,\vec{y}) = 0$

Satz

Hamilton Funkt ist erstes Integral des autonomen Hamilton Systems

$$\begin{cases} \dot{p}(t) = -\dfrac{\partial H}{\partial q}(p(t),q(t)) \\ \dot{q}(t) = \dfrac{\partial H}{\partial p}(p(t),q(t)) \end{cases}$$

Expliziter Eulerverfahren

$y_{n+1} = y_n + hf(t_n, y_n)$

Implizites Eulerverfahren

$$y_{n+1} = y_n + hf(t_n+h, y_{n+1})$$

Implizite Mittelpunktsregel

$$y_{n+1} = y_n + hf\left(t_n + \frac{h}{2}, \frac{1}{2}(y_n + y_{n+1})\right)$$

Implizite Trapetzregel

$$y_{n+1} = y_n + \frac{h}{2}\left(f(t_n, y_n) + f(t_n+h, y_{n+1})\right)$$

Strömer Verlet verfahren

$$\ddot{y} = f(t,y), \quad v_0 = \dot{y}(t_0)$$

$$y_{n+1} = -y_{n-1} + 2y_n + h^2 f(t_n, y_n)$$

$$y_1 = y_0 + hv_0 + \frac{h^2}{2} f(t_0, y_0)$$

explizites Zweischrittverfahren, da man zwei vergangene Werte braucht

Leap-Frog

$$v_{n+\frac{1}{2}} = v_{n-\frac{1}{2}} + hf(t_n, y_n)$$

$$y_{n+1} = y_n + hv_{n+\frac{1}{2}}$$

Verlet-Velocity

$$y_{n+1} = y_n + hv_n + \frac{h^2}{2} f(t_n, y_n)$$

$$v_{n+1} = v_n + \frac{h}{2}\left(f(t_n, y_n) + f(t_n+h, y_{n+1})\right)$$

Splitting Verfahren

Idee: ODEs, welche als ganzes schwierig sind, aber als Summe einfacher (analytisch, oder sich gut verhaltende (numerische) Lösung besitzen z.B. $\dot{y} = f_a(y) + f_b(y)$, mit

$\dot{y} = f_a(y) \to \Phi_a^t$ und $\dot{y} = f_b(y) \to \Phi_b^t$

die Propagatoren der jeweiligen Systeme.

Problem: im Allgemeinen kommutieren $\Phi_a^h \Phi_b^h$ nicht, weshalb Fehler entstehen.

<u>Def</u>: Splitting Verfahren

Propagator $\Psi_s^h = \prod_{i=1}^{s} \Phi_b^{b_i h} \Phi_a^{a_i h}$, s.d. $\sum_{i=1}^{s} a_i = \sum_{i=1}^{s} b_i = 1$

<u>Bsp</u>

- Lie-Trotter Splitting: $\Psi_1^h = \Phi_b^h \circ \Phi_a^h$
- Strang-Splitting $\Psi_2^h = \Phi_a^{h/2} \circ \Phi_b^h \circ \Phi_a^{h/2}$

<u>Bemerkung</u>: Strang splitting angewandt auf Newtonsche BWGL ($\dot{r} = v$, $\dot{v} = a(r)$) erhält man einschritt Störmer-Verlet-Verfahren.

<u>Wann brauchen?</u>

- Hamiltonsche Systeme (Symplektizität ⟹ Energieerhaltung)
- Rechenzeit sparen.
- gestörte Systeme, d.h. $\dot{y} = f_b(y) + \varepsilon f_a(y)$, $\varepsilon \ll 1$

<u>Bemerkung</u>

Splitting Verfahren funktionieren nur für autonome DGL d.h. rechte Seite darf nicht explizit zeitabhängig sein. Falls t auftritt ⟹ zuerst autonomisieren

Runge Kutta Verfahren

Voraussetzungen: AWP der Form $\dot{\vec{y}} = \vec{f}(t, \vec{y})$, $\vec{y}(t_0) = \vec{y}_0$

Idee: Durch integration über dem Zeitraum $[t_0, t_1]$

$y(t_1) = y_0 + \int_{t_0}^{t_1} f(\tau, y(\tau))\,d\tau$ mit Quadraturformel finden

Dafür werden mehrere Zwischenwerte ausgewertet und iterativ verbesserte approximationen gemacht.

Def (Runge Kutta Verfahren)

Für $b_j, c_j, a_{ij} \in \mathbb{R}$ mit $i,j \in \{1, \dots s\}$, $s \in \mathbb{N}$ definiert

$$\vec{k}_i := \vec{f}\left(t_0 + c_i h, \vec{y}_0 + h \sum_{j=1}^{s} a_{ij} \vec{k}_j\right), \quad i = 1 \dots s$$

$$\Psi^{t_0, t_0+h} \vec{y}_0 := \vec{y}_0 + h \sum_{i=1}^{s} b_i \vec{k}_i$$

ein s-Stufiges RK-ESV mit inneren $\vec{k}_i \in \mathbb{R}^d$

Butcher Bedingung: $c_i = \sum_{j=1}^{s} a_{ij}$

Beachte: $\vec{k}_i$ kommt selbst in der Def. vor. D.h. wir erhalten ein Gleichungssystem.

Butcher Tabelle

Alle Informationen eines RK-Verfahrens, d.h. Gewichte und Knoten und die Matrix für die Zwischenschritte sind in dem Butcher-Tabelle ersichtlich

$$\text{knoten} \leftarrow \begin{array}{c|c} \vec{c} & A \\ \hline & \vec{b}^T \end{array} \leftarrow \text{gewichte}$$

wobei $A_{ij} = a_{ij}$ ist

Falls A echte untere Dreiecksmatrix ⇒ explizites RKV (kein GS)

A untere Dreiecksmatrix ⇒ diagonal-implizite RKV (DIRK) (GS)

Bsp

- Expliziter Euler $\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array} \to O(h)$

- Explizit Trapez

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1/2 & 1/2 \end{array} \rightarrow O(h^2)$$

- Explizite MPR

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ \hline & 0 & 1 \end{array} \rightarrow O(h^2)$$

- Klassisches RK4

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 2/6 & 2/6 & 1/6 \end{array} \rightarrow O(h^4)$$

Bemerkung

Die sogenannte Butcher-Barriere zeigt das Verhältnis zwischen Ordnung und Anzahl benötigter Stufen. Nach Ordnung 4 steigt die Anzahl benötigter Stufen schnell, weshalb wir oft RK 4 verwenden.

Def (Konsistenzordnung)

Ein s-stufiges Runge-Kutta-Verfahren hat Konsistenzordnung q, wenn der lokale Fehler $\|\vec{y}(t_0+h)-\vec{y}_1\| \le ch^{q+1}$ beträgt.

Bemerkung: Eine QF mit s Stufen hat maximale Konvergenzordnung (mit Gauss) von 2s.
D.h. die Konsistenzordnung eines RKV ist maximal 2s.

Bemerkung: Falls ein s-stufiges RKV Konsistenzordnung q hat, so hat die QF Ordnung q.

Theorem

- Ein explizites s-stufiges RKV hat max Kons.Ord $q \leq s$
- Ein impizites " " " " " $q \leq 2s$

Schrittweitenanpassung (Adaptivität)

Idee: bei grösseren vermuteten Fehlern (vergl. zwei Methoden) verwenden wir lokal kleinere Zeitschritte. Dafür braucht man für den Vergleich zwei Methoden mit unterschiedlicher Ordnung.

Partitionierte RK Verfahren

Voraussetzung: $\dot{y} = f(y,z)$, $\dot{z} = g(y,z)$

Idee: unterschiedliche RK Verfahren auf unterschiedliche Teile des Systems anwenden.

Runge-Kutta-Nyström Verfahren

Spezialfall von PRK mit $\ddot{y} = f(t,y,\dot{y})$

$\Rightarrow \dot{y} = z$, $\dot{z} = g$ (oft ist $g(t,y,\dot{y}) = g(t,y)$)

RKN hat Ordnung n, falls beide lokalen Teile Ordnung p+1 haben.

Steife ODEs

Problem: Wir haben gesehen, dass explizite RK-Verfahren zsm mit adaptiver Zeitschrittweite sehr genaue approximation von ODEs liefen. Bei einer gewissen Klasse von ODEs, den sogenannten steifen ODEs, haben wir mit adaptiver Zeitschrittweite und expliziten RKV jedoch eine unnötig langsam konvergierendes Verhalten. → Wir verwenden in diesen Fällen implizite Verfahren.

Def (Steife ODE)

Eine ODE heisst steif, falls ein explizites Verfahren einen *kleinen Zeitschritt* braucht um die Lösung gut zu approximieren, selbst wenn die Lösung selbst glatt & langsam ist.

Bem: Es geht hier nicht um die Form der Lösung, sondern um die Struktur der ODEs.

z.B. grosse Eigenwerte → extreme Schätzung mit EE.

Bsp $x(t)=e^{-1000t}$ → EE $x(h)=(1-1000h)x_0$

Das erzeugt offensichtlich Probleme in glatten Bereichen → kleine Zeitschritte trotz annähernder Konstanz

Lösung: Verwende implizite Verfahren, welche die Entwicklung besser vorhersagen können.

Def (Stabilitätsgebiet)

Für ein ESV mit diskreter Evolution

$\Psi_\lambda^h y =: S(z=\lambda h)\, y$ ist das Stabilitätsgebiet gegeben als $S_\Psi := \{ z \in \text{Definitionsbereich}, \text{s.d. } |S(z)| < 1 \}$

Bem: Explizite Verfahren haben sehr beschränktes S_Ψ, weshalb h klein sein muss.

Bem: Wir nennen $S(\cdot)$ die Stabilitätsfkt.

Theorem (Stabilitätsfkt von RKV)

Der diskrete Ev.Op zu s-stufigen RK-ESV mit Butcher-Schema $\begin{array}{c|c} c & A \\ \hline & b^T \end{array}$ (für $\dot{y} = \lambda y$) ist Multiplikation mit folgender Form

$$\Psi_\lambda^h = S(z) = 1 + z b^T (\underline{1} - zA)^{-1} \underline{1} = \frac{\det(\mathbb{1} - zA + z \underline{1} b^T)}{\det(\mathbb{1} - zA)}$$

wobei $\underline{1} := (1 \dots 1)^T \in \mathbb{R}^s$

Bem: Für explizite RKV ist $S(z) \in \mathbb{C}[z]_{\leq s}$. Für allg. RKV ist es eine Rationale Fkt.

z.B

- EE: $S(z) = 1 + z$
- IE: $S(z) = \frac{1}{1 - z}$
- ETR: $S(z) = 1 + z + \frac{1}{2} z^2$

◦) IMPR: $S(z) = \dfrac{1+\frac{1}{2}z}{1-\frac{1}{2}z}$

◦) RK4: $S(z) = 1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3 + \frac{1}{24}z^4$

Def (A-/L-Stabilität)

Ein Verfahren heisst A-stabil, falls linke Halbebene $\mathbb{C}^- \subset S_\Psi$ ⇒ beliebig grosse Zeitschritte ohne explodierende numerische Lösung

L-Stabil, falls zusätzlich $S(-\infty) = 0$

D.h. keine sinnlose Oszilationen. ($S(z) \approx e^z$)

(Genau dann der Fall, falls $\begin{array}{c|c} c & A \\ \hline & b^T \end{array}$ in A eine Zeile b^T vorhanden)

Bem: Da implizite Methoden das lösen von Gleichungssystemen voraussetzen sind sie oft teurer als explizite. Falls ein System so aufgeteilt werden kann, dass ein Teil explizit gut gelöst wird und ein anderer implizit, ist dies billiger als rein implizit. Solche Methoden heissen IMEX-RK-Verfahren.

Dafür schreiben wir Butcher Tableau

$$y' = f(t,y) + g(t,y) \implies \begin{array}{c|c|c} c & A_f & A_g \\ \hline & b_f^T & b_g^T \end{array}$$

wobei o.B.d.A A_f explizit ⟺ echte UDM ist. und A_g implizit ⟺ keine echte UDM.

Nullstellen suchen

Gesucht ist eine Nullstelle $x^* \in \Omega$ von $F: \Omega \subset \mathbb{R}^d \to \mathbb{R}^d$. Ein iteratives Verfahren sorgt für eine Folge $(x_i)_{i\in\mathbb{N}_0} \to x^*$

Das Ziel ist Konvergenz, schnelle Konvergenz, und möglichst wenige berechnete Ableitungen, da diese teuer sind.

Oft ist ein Startwert nah der Nullstelle essentiell für Konvergenz.

Def

Wir definieren ein iteratives Verfahren als $x_{n+1} = \phi(x_n)$. Dieses konvergiert, falls $x_n \xrightarrow[n\to\infty]{} x^*$ und $\phi(x^*) = x^*$

Def (Linear konvergent)

$(x_i)_{i\in\mathbb{N}_0}$ heisst linear konvergent, falls ϕ Lipschitz-stetig ist, d.h. $\exists L<1$ s.d.

$$\|x_{n+1}-x_n\| \le L\|x_n - x^*\| \quad \forall n \ge n_0$$

Def (Konvergenzordnung)

Eine Folge hat Konvergenzordnung $p \ge 1$, falls $\exists C>0$ s.d. $\|x_{n+1}-x_n\| \le C\|x_n-x^*\|^p$ und mit $\varepsilon_n := \|x_n - x^*\|$ gilt

$$p \approx \frac{\log\varepsilon_{n+1} - \log\varepsilon_n}{\log\varepsilon_n - \log\varepsilon_{n-1}}$$

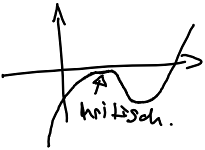
Abbruchkriterien

o) A priori : nach N Schritten wird abgebrochen.

o) A posteriori: Abbrechen sobald Toleranz unterschritten

Beachte: Beide Kriterien sind unzuverlässig, falls die Nullstelle nicht bekannt ist.
z.B. kann trotz des kleinen Residuums die Argumente grosse Unterschiede machen.

Bem: Oft wird auch für $x_{n+1} \approx x_n$ ein Abbruchkriterium eingeführt



Banachscher Fixpunktsatz

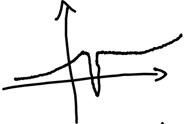
Für Kontraktion $\phi: D \to D$ existiert ein eindeutiger Fixpunkt. dieser ist Nullstelle

Bem: Es gibt weitere Abschätzungen zu Fehler und Ordnung der Konvergenz (für lineare) Kontraktion. Diese scheinen unwichtig.

Bisektion / Intervallhalbierungsverfahren

Der erste Ansatz verwendet den Zwischenwertsatz und verkleinert iterativ das Intervall auf dem ein Vorzeichenwechsel beobachtet wird.

Bem: Dafür brauchen wir zwei Argumente a, b, s.d. $\operatorname{sgn}(f(a)) \neq \operatorname{sgn}(f(b))$.
D.h. insbesondere für fkt der Form



ist dieses Verfahren oft nicht konvergent.

Bem: mehrdimensionale Verallgemeinerung ist ?anscheinend? nicht trivial. $\Rightarrow$ wieso?

Newton Verfahren

$x_{n+1} = x_n - \frac{F(x_n)}{F'(x_n)}$ braucht Ableitung.

Für mehrdimensional äquivalent

$x_{n+1} = x_n - DF(x_n)^{-1} F(x_n)$, wobei DF die Jacobi-Matrix ist.

Falls ableiten zu teuer ist verwenden wir alternativ das Sekantenverfahren

$$x_{n+1} = x_n - \frac{F(x_n)}{q_n}, \quad q_n := \frac{F(x_n)-F(x_{n-1})}{x_n - x_{n-1}}$$

Bem: Das vereinfachte Newtonverfahren verwendet $DF(x_n)^{-1}$ mehrmals um Rechenschritte zu sparen. Dafür Quad. konv $\Rightarrow$ lin. konv.

Bem: Wir können konvergenz dämpfen mit schrittabhängigen Koeffizienten λ_n. Dann

$$x_{n+1} = x_n - \lambda_n DF(x_n)^{-1} F(x_n)$$

Quasi-Newton Verfahren

Als mehrdimensionale Form des Sekantenverfahrens (kein berechnen von DF nötig), definieren wir $J_n = \frac{F(x_n)-F(x_{n-1})}{x_n - x_{n-1}}$ $\swarrow$ Version implizit als

$$J_n(x_n - x_{n-1}) = F(x_n) - F(x_{n-1})$$

Bem Broyden-Bedingung ist, dass alle z die orthogonal zu $x_n - x_{n-1}$ stehen gilt $J_n z = J_{n-1} z$

somit

$$J_n := J_{n-1} + \frac{F(x_n)(x_n - x_{n-1})^T}{\| x_n - x_{n-1} \|^2}$$

somit gilt für Broyden-Quasi-Newton:

$$x_{n+1} := x_n - J_n^{-1} F(x_n)$$

$$J_{n+1} := J_n + \frac{F(x_{n+1})(-J_n^{-1}F(x_n))^T}{\| -J_n^{-1}F(x_n) \|^2}$$

mit $J_0 := DF(x_0)$ oder ähnlich

Bem: Für die Implementierung gehen wir direkt über die Inverse von J_n mit

$$J_{n+1}^{-1} := \left(\mathbb{1} - \frac{J_n^{-1}F(x_{n+1})(-J_n^{-1}F(x_n))^T}{\| -J_n^{-1}F(x_n) \|^2 + (-J_n^{-1}F(x_n))^T J_n^{-1} F(x_{n+1})} \right) J_n^{-1}$$

Bem: Für grosse Nichtlineare Systeme gibt es weitere Methoden

Bem: Broyden lohnt sich für grosse Dimensionen. ⇝ besser als root löser.

Lineare Algebra

Idee: Es geht darum $Ax = b$ so effizient wie möglich zu berechnen. Dabei ist np.linalg.inv(A) teuer.

LU-Zerlegung

$A=LU$, wobei L untere Dreiecksmatrix mit einsen auf der Diagonalen und U obere Dreiecksmatrix ist.

Wir finden $Ax=b$ durch

① $Lz=b$ → Vorwärtssubstitution

② $Ux=z$ → Rückwärtssubstitution

`P,L,U = scipy.linalg.lu(A), x=scipy.linalg.lu_solve(P,LU`

Bem: Jede Vektornorm induziert Matrixnorm mit $\|A\| = \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|} = \sup_{x=1} \frac{\|Ax\|}{\|x\|}$

Def: Konditionszahl einer Matrix bez. einer Norm ist $\mathrm{cond}(A) := \|A^{-1}\|\cdot\|A\|$

Bem: Es gilt $\mathrm{cond}(\mathbb{1}) = 1$

Bem: Falls $\mathrm{cond}(A) >> 1$ sagen wir das LGS ist schlecht konditioniert. Dann propagieren Rundungsfehler schnell, was die Gausselimination fehleranfällig macht.

Bem: Für hermit-symmetrischen Matrizen ist die LU zerlegen effizienter

$$A=LU=LDL^H=R^HR \text{ mit } R=\sqrt{D}L^H$$

verwenden in solchen Fällen `L=scipy.linalg.cholesky(A)` und `scipy.linalg.cho_solve(..)`

QR-Zerlegung

$A = QR$, wobei $A \in K^{m \times n}$, $Q \in K^{m \times n}$, $R \in K^{n \times n}$ s.d. $Q^H Q = \mathbb{1}$ (wir erhalten Spalten von Q aus GSOV von Spalten von A)

R ist dann Dreiecksmatrix mit

$A = [a_1 \dots a_n]$, $R_{ii} = \|a_i\|_2^2$, $R_{ik} = q_i^H a_k$ $k > i$ sonst $= 0$

Alternativ können wir A in eine Rechtecksform R bringen, indem wir Spiegelungen und Drehungen. $\leadsto$ Details siehe Skript (Householder)

Verwenden Q,R = scipy.linalg.qr(A)

oder Q,R = scipy.linalg.qr(A, mode='economic') falls $A \in K^{m \times n}$ und $m > n$.

Singulärwertzerlegung

Sei $A \in \mathbb{C}^{m \times n}$, $\exists\, U \in U(m)$, $V \in U(n)$ und $\Sigma \in \mathbb{C}^{m \times n}$ mit $\Sigma = \mathrm{diag}(\sigma_1 \dots \sigma_p)$ s.d. $p = \min\{m, n\}$ und $\sigma_1 \geq \sigma_2 \geq \dots \sigma_p \geq 0$ s.d.

$$A = U \Sigma V^H$$

Bem: Wird zur Datenkompression verwendet, indem kleinere Singulärwerte zu Null gesetzt werden.

Bem σ_i^2 sind EW von AA^H, $A^H A$ und $\begin{bmatrix} 0 & A^H \\ A & 0 \end{bmatrix}$ mit EVec $(v_1 \dots v_n)$, $(u_1 \dots u_m)$, $\begin{bmatrix} u_1 \\ v_1 \end{bmatrix}, \dots \begin{bmatrix} u_p \\ v_p \end{bmatrix}$

Nutze scipy.linalg.svd(A) $\rightsquigarrow$ plus Variation

Bem: Es gilt $\|A\|_2 = \sigma_1(A)$, Falls A invertierbar ist, dann $\text{cond}_2(A) = \frac{\sigma_1}{\sigma_m}$

Bem: Norm und cond verwenden svd (Python)

Bem: Frobenius-Norm ist $\|A\|_F^2 = Tr(AA^H) = \sum_{i=1}^{p} \sigma_i^2$

Bem: Die Information in einer m×n Matrix mit Rang k kann mit k(m+n) Zahlen dargestellt werden.
D.h. mit Matrizen niedrigen Ranges können Datenkompression approximativ realisiert werden.

Ausgleichsrechnung

Idee: Wir haben ein Modell $f(t,x)$ mit unbekannten Parametern $x \in \mathbb{R}^n$ und m Messpunkte (t_i, y_i).
Gesucht sind die Parameter die am besten (im Sinne von least squares) zu den Messpunkten passen.

Def (kleinste Quadrate)

$$\sum |f(t_i,x) - y_i|^2 \to \min$$

Bem In Python verwenden wir np.linalg.lstsq

Def (lineare Ausgleichsrechnung)
Die Ausgleichsrechnung ist linear, falls f linear von $x \in \mathbb{R}^n$ abhängt.

① Normalengleichung

Ziel ist es $f(x) = \|b - Ax\|_2^2$ zu minimieren. d.h. $Df(x) = 0$, woraus die Normalengleichung folgt: $A^H A x = A^H b$

Bem: $A^H A$ ist Hermitsymmetrisch, und falls sie vollen Rang hat, mit LU bzw Cholesky Zerlegung bestimmen

Bem: $A^H A$ ist schlecht konditioniert mit $\mathrm{cond}(A^H A) = \mathrm{cond}(A)^2 = \frac{\sigma_1^2}{\sigma_n^2}$

Bem: Für Sparse Matrices gibt es effiziente Methoden (allg. ist mit A sparse $A^H A$ nicht mehr sparse)

② QR-Zerlegung

Sei $A = QR$. Dann finden wir $\arg\min_x \|Ax - b\|_2^2$ bei $x = \tilde{R}^{-1} \tilde{b}_1$ wobei $R = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}$ und $\tilde{b} = Q^H b = (\tilde{b}_1, \tilde{b}_2)^T$

Bem: Wir müssen hier $\tilde{R}^{-1}$ nicht explizit berechnen, sondern können rückwärtssubstituieren x = scipy.linalg.solve_triangular($\tilde{R}, \tilde{b}_1$, lower = False)

③ SVD mit Pseudoinverse

Falls A nicht vollen Rang r hat, so ist für $\|Ax - b\|_2^2$ minimal für $x = U_1 \Sigma_r^{-1} U_1^H b$, wobei die SVD von

$$A = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}$$

Def (Pseudoinverse

Die Pseudoinverse von A ist def. als

$$A^+ := (A^H A)^{-1} A^H = U_1 \Sigma_r^{-1} U_1^H$$

Bem: Numerisch haben wir oft

$r = \max\{i \text{ s.d. } \sigma_i/\sigma_1 > \text{toleranz}\}$

Bem: Für voll besetzte $m >> n$ nehmen wir QR oder SVD. Für dünn besetzt nehmen wir Normalengleichung mit cholesky.

Bem: Im Skript steht noch etwas dazu, was zu tun ist, wenn es eine Unsicherheit in A gilt

Nichtlineare Ausgleichsrechnung

Ziel ist wieder für ein Modell $f(t,x)$ die Parameter $x \in \mathbb{R}^n$ zu bestimmen, s.d. die residuen bzgl. squares minimiert sind.

Wir schreiben die Residuenfkt als $\Phi(x) = \frac{1}{2}\|F(x)\|_2^2$

Bem: Existenz & Eindeutigkeit der Lsg. von x ist nicht klar. Gemäss impliziten Fkt Theorem nur lokal, falls $\operatorname{rang}(D\Phi(x)) = n$.

Newton-Verfahren

Wir haben das Problem auf Nullstellensuche $\operatorname{grad} \Phi(x) \overset{!}{=} 0$ reduziert. Diese lösen wir mit Newton-Verfahren, wobei $H\Phi(x) = D(\operatorname{grad}\Phi)(x)$ ist. und

$$x^{k+1} = x^k - (H_\Phi(x))^{-1} \operatorname{grad} \Phi(x)$$

bzw äquivalent

$$x^{k+1} = x^k - s \quad \text{, wobei } D^2\Phi(x^k)\, s = (DF(x^k))^T F(x^k)$$

Bem: schnelle Konvergenz, aber Hesse Matrix muss nicht zwangsläufig existieren und falls sie existiert ist sie teuer. Ausserdem muss in jedem Schritt ein LGS gelöst werden ~~P~~teuer.

Gauss-Newton-Verfahren

Wir linearisieren $F(x) = F(y) + DF(y)(x-y)$ lokal

Für $A = DF(y)$, $b = DF(y)y - F(y)$ erhalten wir ein lineares Ausgleichsproblem

$$\underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2} \| Ax - b \|_2^2 \quad \text{mit } x = x^k,\ y = x^{k+1}$$

Finden wir also

$$x^{k+1} = x^k - \underset{z \in \mathbb{R}^n}{\operatorname{argmin}} \| F(x^k) - DF(x^k)\, z \|_2^2$$

Äquivalent mit Normalengleichung

$$x^{k+1} = x^k - s \quad \text{mit } [(DF(x^k))^T DF(x^k)]\, s = DF(x^k)^T F(x^k)$$

Bem: deutlich billiger als Newton, schlechtere Konvergenz, braucht wie Newton guten Startwert x^0 und Python implementation ist schlecht.

Weitere Methoden

Alle weiteren Methoden beruhen auf $x^{k+1} = x^k - s$ und unterschiedlichen Wahlen von s.

o) BFGS → Variante des Broyden-Verfahrens die Hesse-Matrix approximiert

o) Gradient Descent (GD) → $s = \lambda_k D\Phi(x^k)$ wobei learning rate λ_k entweder konstant, heuristisch oder algorithmisch bestimmt werden kann.

o) Levenberg-Marquardt (LM) → Kombination von Gauss-Newton und GD mit Dämpfung. (GD sucht Startpunkt für GN)

o) Stochastischer GD (SGD) → bei grossen Dimensionen in ML werden Teile von $D\Phi$ pseudozufällig gewählt.

o) Mehr: CD, ADAM, etc.