Wiederholung (Finite Differenzen)

$$\left.\frac{d^m f}{dx^n}\right|_{x=x_0} \approx \sum_{\nu=0}^{n} \delta_{n,\nu}^{m} f(x_\nu)$$

$n \leftarrow$ Grad der Approximation

$x = x_0$ auf dem Gitter

x_ν: Benachbarte Punkte auf dem Gitter

Fragestellung: Wie bestimmt man $\delta_{n,\nu}^{\nu}$ Koeffizienten.

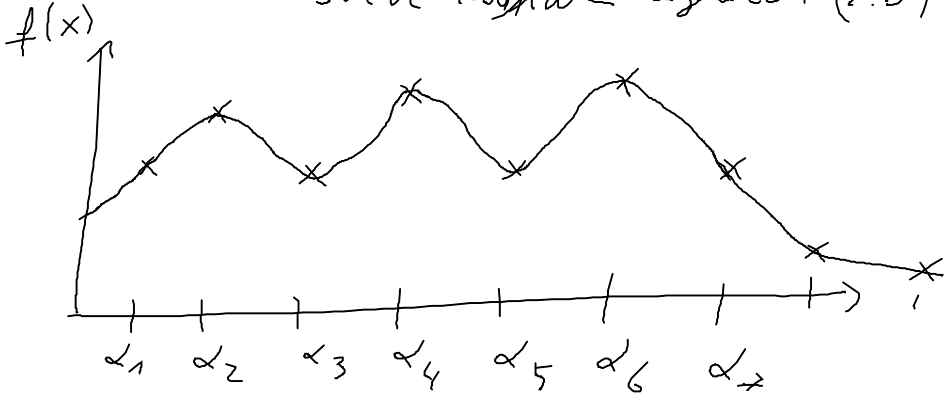
a) Tabellierte für verschiedene Situationen z.B. symmetrischer Stencil

Grad der Ableitung	Ordnung	x-Koordinaten: −4	−3	−2	−1	0	1	2	3	4	
0	∞					1					
1	2				$-\frac{1}{2}$	0	$+\frac{1}{2}$				
	4			$\frac{1}{12}$	$-\frac{2}{3}$	0	$\frac{2}{3}$	$-\frac{1}{12}$			$\cdot\frac{1}{4}$
	⋮					⋮					
	8	$\frac{1}{280}$	$-\frac{4}{105}$	$\frac{1}{5}$	$-\frac{4}{5}$	0	$\frac{4}{5}$	$-\frac{1}{5}$	$\frac{4}{105}$	$-\frac{1}{280}$	
2	2				1	−2	1				
	4			$-\frac{1}{12}$	$\frac{4}{3}$	$-\frac{5}{2}$	$\frac{4}{3}$	$-\frac{1}{12}$			$\cdot\frac{1}{4^2}$

usw

b) Es gibt auch iterative Formel zum berechnen der Koeffizienten (s. Fornberg.)

Wie werden solche Koeffizienten hergeleitet (z.B.)?



Idee: Finde ein Polynom, dass die Punkte interpoliert?

Lagrange interpolations Polynom (Waring (1779), Euler (1783), Lagrange (1795)):

$$p(x) = \sum_{\nu=0}^{n} F_{n,\nu}(x)\, f(\alpha_\nu)$$

$$F_{n,\nu}(x) := \frac{\omega_n(x)}{\omega_n'(\alpha_\nu)\,(x-\alpha_\nu)}$$

mit $\omega_n(x) := \prod_{k=0}^{n} (x-\alpha_k)$

Das $F_{n,\nu}(x)$ ist ein Polynom der Ordnung n, dass 1 bei $x=\alpha_\nu$ ist und 0 bei $x=\alpha_k$ $(k \neq \nu)$.

Zur Erinnerung $\left.\frac{d^m f}{dx^m}\right|_{x=x_0} \approx \sum_{\nu=0}^{n} \delta^m_{n,\nu} f(\alpha_\nu)$

Die entsprechenden Gewichte ergeben sich, dann ist

$$\delta^m_{n,\nu} = \left[\frac{d^m}{dx^m} F_{n,\nu}(x)\right]_{x=0}$$

Zurück zum Problem

Laplace

Der Term war

$$-\frac{\hbar^2}{2m_r}\Delta_r = -\frac{\hbar^2}{2m_r}\left(\partial_{x_r}^2 + \partial_{y_r}^2\right)$$

Also $\partial_{x_r}^2 \, \psi(x[n_1], y[n_2])$

$$= \sum_{\nu=0}^{n} \delta_{n,\nu}^2 \, \psi\left(x\left[n_1+\nu-\frac{n}{2}\right], y[n_2]\right)$$

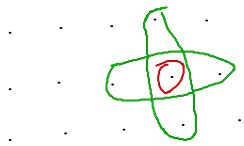
bzw.

$$\partial_y^2 \, \psi(x[n_1], y[n_2])$$

$$\Rightarrow \sum_{\nu=0}^{n} \delta_{n,\nu}^2 \, \psi\left(x[n_1], y\left[n_2+\nu-\frac{n}{2}\right]\right)$$

Gibt es eine Konsequenz aus dem Zugriffsverhalten, für die Vektorisierung?

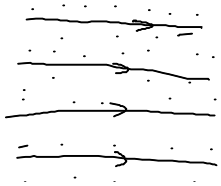
Gitter



Naive Berechnung des Vektorindex

$$i[n_1, n_2] = n_1 + n_2 \cdot M$$

M: Anzahl der Punkte

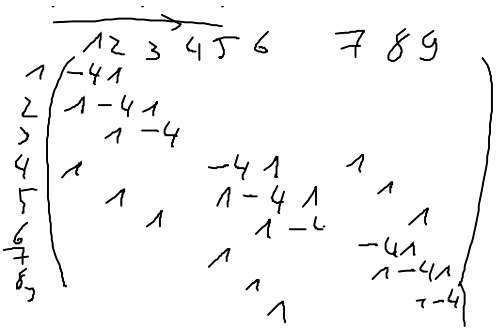


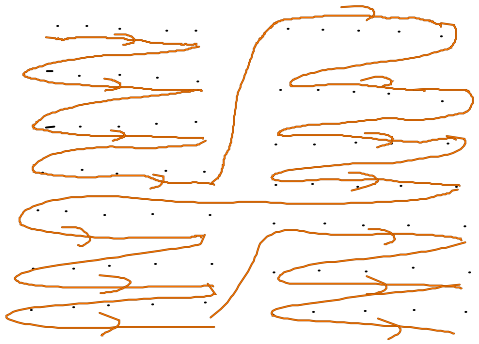
Weit entfernte off diagonale Anteile (im Speicher nicht im Cache)

Großer Transfer bei MPI

Lösung andere Indizierung

3x3 Stilu





⇒ Vorteil Die meisten Zugriffe erfolgen auf Vektorpositionen, die sehr nah beieinander sind. Weitere Zugriffe sind weit entfernt im Speicher.

Nachteil: Programm ist aufwendig ....

Vielleicht: Verteilung auf verschiedene CPUs oder Kerne entlang der Blockgrenzen, sonst wird der Vorteil nicht ausgespielt.

Es fehlt der Term (Trion Problem)

$-\frac{\hbar^2}{2m_n} \nabla_{r_1} \cdot \nabla_{r_2}$ bei Trion.

Vorgehen

$$\nabla_{r_1} \cdot \nabla_{r_2} = \partial_{x_1} \partial_{x_2} + \partial_{y_1} \partial_{y_2}$$

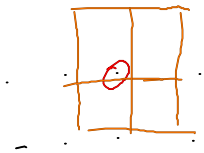
$$\partial_{x_1} \partial_{x_2} \psi(x_1[n_1], y_1[n_2], x_2[n_3], y_2[n_4])$$

$$\approx \sum_{\nu_1=0}^{n} \delta^{\wedge}_{n_1,\nu_1} \partial_{x_2} \psi(x_1[n_1+\nu_1-\tfrac{n}{2}], y_1[n_2], x_2[n_3], y_2[n_4])$$

$$\approx \sum_{\nu_1=0}^{n} \sum_{\nu_2=0}^{n} \delta^{\wedge}_{n_1,\nu_1} \delta^{\wedge}_{n_1,\nu_2} \psi(x_1[n_1+\nu-\tfrac{n}{2}], y_1[n_2], x_2[n_3+\nu_2-\tfrac{n}{2}], y_2[n_4])$$

Form des Stencils

. . . . .



Damit ist alles klar für Exziton und Trion, aber ich kann es jetzt Matrixform approximieren:

$$\underline{\underline{H}} \cdot \underline{v} = E \underline{v}$$

Matrix

II.5 Iterative Lösungsverfahren für das Eigenwertproblem

Ziel ist Grundlage für iterativer Lösung für:

$$\underline{\underline{H}} \cdot \underline{v} = E \underline{v}$$

wobei H z.B. aus einer partiellen DGL kommt.

Das ist ein Überblick mit Beispielen über wichtige Methoden (unvollständig).

Literatur: z.B. Technical Report zu Slepc
http://slepc.upv.es

Erstmal Grundlagen der numerischen linearen Algebra.

QR-Faktorisierung und Gram-Schmidt Verfahren

$\underline{\underline{X}} = \underline{\underline{Q}} \cdot \underline{\underline{R}}$

$\underline{\underline{Q}}$: Spalten orthogonale Matrix

$\underline{\underline{R}}$: obere Dreiecksmatrix $\begin{pmatrix} r_{11} & \cdots & r_{1n} \\ & r_{22} & \vdots \\ 0 & & r_{nn} \end{pmatrix}$

Dreiecksmatrizen haben wichtige Vorteile:

- Determinante ist das Produkt der Diagonalelemente

$\Rightarrow (\underline{\underline{R}} - \lambda \underline{\underline{I}}) = 0$ Eigenwerte stehen auf der Diagonale

Wie berechnet man $\underline{\underline{Q}} \cdot \underline{\underline{R}}$ numerisch (3 typische Verfahren):

- Haushalder Reflektion
- Givens Drehungen
- Gram-Schmidt Verfahren (in der einfachen Form numerisch instabil)

Hier Diskussion Gram-Schmidt Verfahren

+ Modifikation für bessere Stabilität!

Zur Erinnerung, wie funktioniert Gram-Schmidt?

Wir haben N Vektoren $\underline{x}_1, \ldots, \underline{x}_N$, diese wollen wir orthonormieren.

Wir erhalten die orthogonalen Vektoren $q_1 \ldots q_N$ mit der Norm $r_{jj} = \|q_j\|_2$ mit

$$q_j = \underline{x}_j - \underbrace{\sum_{i=1}^{j-1} q_i^T \underline{x}_j \, q_i}_{\text{Anteil von } \underline{x}_j \text{ in Richtung von } \operatorname{span}\{q_1, \ldots, q_{j-1}\}}$$

In Form eines Algorithmus (mit Skizze Rapat)

$q_1 = \underline{x}_1$, $q_j = \underline{x}_j$

for $i = 1, \ldots, j-1$

$$r_{ij} = q_i^T \cdot x_j$$

$$q_j = q_j - r_{ij}\, q_i$$

und

$$r_{jj} = \|q_j\|^2$$

man kann auch in Matrix

$$q_j = \left(I - Q_{j-1} \cdot Q_{j-1}^T\right) q_j$$

Q_{j-1}: Matrix mit $j-1$ Spalten mit q

Projektion auf den $\mathrm{span}\{q_1 \dots q_{j-1}\}$

⇒ was hat das mit der QR Faktorisierung zu tun?

Algorithmus

Eingabe Matrix X

$$r_{11} = \|x_1\|_2$$

$$q_1 = \frac{x_1}{r_{11}}$$

for $j = 2, \dots, n$

Erhalte die Vektoren $[q_j, r_j]$ aus dem Gram-Schmidt Verfahren für $q_1 \dots q_{j-1}$ und x_j

$$q_j = q_j / r_{jj}$$

und

Ausgabe die Matrizen Q und R