

I.3 Lineare Algebra für (parallele) Programmierung

Vektorisierung der phys. Problem

Wann Vektorisierung? Viele Hardwarearchitekturen, Software-Bibliotheken etc. sind optimiert um Matrixmultiplikation und Vektoroperation durchzuführen.

Beispiel für typische lineare Problem (formalisiert):
Lösung linear Gleichungssystem

$$\underline{A} \cdot \underline{x} = \underline{b}$$

Lösung von zeitabhängigen Problemen

$$\partial_t \underline{v} = \underline{M} \cdot \underline{v}$$

Eigenwertproblem

$$\underline{A} \cdot \underline{x} = \underline{\lambda} \underline{x}$$

(und weitere Probleme wie spectral transform, singular value decomposition...)

Beispiele für physikalische Probleme

Schrödinger

$$\begin{array}{c} H|\psi\rangle = E|\psi\rangle \Rightarrow \\ \uparrow \quad \uparrow \\ \text{Matrix} \quad \text{Vektor} \end{array}$$

Wichtig ist es eine geeignete Basis zu finden (s. Finite-Differenzen und Quantenmechanik)

⇓
Übersetzen in Spaltenvektoren

$$|\psi\rangle = \sum_i c_i |i\rangle$$

Liouville-von Neumann gl.

$$\partial_t \rho = -\frac{i}{\hbar} [H, \rho]$$

(Hier ist die Dichtematrix ein Vektor
 (z.B. $\langle i | \rho | j \rangle = \rho_{ij}$)
 und $-\frac{i}{\hbar} [H, \cdot]$ wird auf Matrix abgebildet.

Helmholtzgleichung:

$$\Delta \underline{A}(\underline{r}, \omega) + \frac{\omega^2}{c^2} \underline{\epsilon}(\omega) \underline{A}(\underline{r}, \omega) = 0$$

+ Randbedingungen

Matrix

Vektor

Besondere Form des Eigenwertproblems (nicht Hermitesch)
 z.B. wichtig für Quantisierung des Lichtfelds
 Entwickelt von $\underline{A}(\underline{r}, \omega)$ über Gitter oder Finite
 Elemente.

Außerdem gibt es auch vektorielle Fassungen für
 nicht lineare Probleme.

Beispiele für Vektor und Matrixformate

Sequenzielle Vektoren

Vektorformat: Vektor ist ein Array von Fließkommazahlen

Speicher $[z[0]] \quad [z[1]] \quad [z[2]] \quad [z[3]] \quad \dots \quad [z[n]]$

Abbildung in Speicher single (32 Bit) oder double (64 Bit)
 precision Fließkommazahlen.

z.B. Speicher Datenformat Hardware abhängig.

z.B. IEEE 754

$$x = s \cdot m \cdot 2^e$$

\uparrow Vorzeichen \uparrow Mantisse (23 bit single, 52 bit double) \leftarrow Exponent
 18 bit single - $126 \leq e \leq 127$
 11 bit double - $1022 \leq e \leq 1023$

Wichtig: Genauigkeit des Drucks
 (Multiplikation und Addition sehr groß und
 sehr klein Zahlen ist schwierig)

Bei komplexer Zahl besteht \mathbb{C} aus zwei Werten
 reelle und imaginäre Teil (ist z.B. C, C++ fortis implizit...)

Matrixformate

Einzelste Möglichkeit (als Block)

Matrix

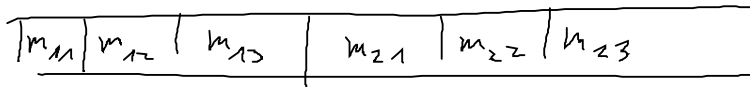
$$M = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{pmatrix}$$

Zwei Möglichkeiten das in Speicher abzulegen.

1) Spalte zuerst (Column major) (z.B. Fortran und BLAS)



2) Zeile zuerst (Row major) (z.B. Standard bei C, C++)



Low-level Routine für diese Matrixformate findet sich
 in BLAS. (Das ist meist von Admin, oder
 Hersteller des Supercomputers optimiert)

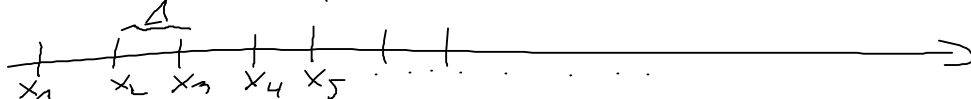
Dieses Format ist ideal für dünn besetzte Matrix
 (viele Einträge von null verschieden)

$$\begin{pmatrix} 1 & 2 & 9 \\ 6 & 7 & 8 \\ 9 & 0 & 10 \end{pmatrix} \leftarrow \text{Matrix mit wenig } 0$$

Problem: (sehr) Oft gibt es dünn besetzte Matrizen
 Ein Beispiel

$$\partial_x^2 \psi(x)$$

Wir diskretisieren ψ



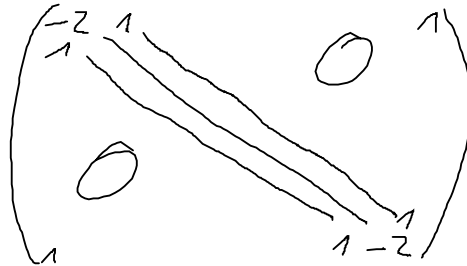
$$\varphi(x_1) \varphi(x_2) \varphi(x_3) \varphi(x_4) \varphi(x_5)$$

$$\partial_x^2 \varphi(x_n) \approx \varphi(x_{n-1}) - 2\varphi(x_n) + \varphi(x_{n+1})$$

Ans: Von N^2 Einträgen
sind nur $3N$ Einträge
besetzt!

$$\frac{3N}{N^2} = \frac{3}{N}$$

→



Bei großer Gitter wird das beliebig ineffektiv.

Das naive Format ist dann richtig ineffektiv.

(Literatur linalg.netlib.org)

Diinn besetzte Matrizen

Naivstes Format

Ein Array mit Fließkommazahlen $v[i]$

Zwei Arrays mit ganzen Zahlen

$i[i]$ $j[i]$

Beispiel

$$j = \begin{matrix} 0 & 1 & 2 & 3 \\ 0 & 2 & 6 & \\ 1 & & & 8 \\ 2 & & & \\ 3 & 7 & & \\ 4 & & & \end{matrix}$$

$$v = \{2, 6, 8, 7\}$$

$$i = \{0, 2, 3, 1\}$$

$$j = \{0, 0, 2, 3\}$$

⇒ Das benutzt
kein, da ineffektiv.

Tipps: Zusammenbau der Matrix ist aufwendig.
In der Praxis muss man bevor man Speicher reserviert,
die Anzahl der besetzten Einträge in jeder Zeile
zählen.

Blockmatrix (z.B. bei multiphysik Problem)

$$\begin{pmatrix} A_{aa} & A_{ab} & A_{ac} \\ A_{ba} & A_{bb} & A_{bc} \\ A_{ca} & A_{cb} & A_{cc} \end{pmatrix}$$

Zwei Möglichkeiten

1) monolithische Matrizen.

oder

2) verschärfte Matrizen