

Projekt 6: Einführung in *Fast Multipole*

Motivation: Vollbesetzte Matrizen können “datenschwach” realisiert werden.

Wir betrachten die Matrizen $M \in \mathbb{R}^{N \times N}$

$$M_{ij} = \begin{cases} K(x_i, x_j), & i \neq j \\ 0 & i = j \end{cases} \quad \text{mit} \quad x_i = (i-1)h, \quad h = \frac{1}{N-1}. \quad (1)$$

Hier ist die Funktion K gegeben durch

$$K(x, y) = \ln |x - y|. \quad (2)$$

Eine naive Abspeicherung dieser Matrizen benötigt N^2 Einträge; insbesondere werden $O(N^2)$ Rechenoperationen für die Matrix-Vektor-Multiplikation $x \mapsto Mx$ benötigt. Ziel dieses Projektes ist es, effiziente Methoden kennenzulernen, mit denen der Speicherbedarf zur Darstellung (einer Approximation) der Matrix M nur $O(N \log N)$ ist.

Die hier vorgestellten Techniken sind eine vereinfachte Form der sog. Multipolverfahren, mit denen u.a. Vielteilchenproblem (z.B. die Newton’schen Bewegungsgleichungen von $N = 10^6$ bis 10^8 Sternen in einer Galaxie unter dem Einfluß ihrer Gravitationsfelder) gerechnet werden.

1. Sei \mathbb{P}_n der Raum der Polynome vom Grad $\leq n$. Mit $\mathbb{P}_n^{2D} = \text{span}\{x^i y^j \mid 0 \leq i, j \leq n\}$ bezeichnen wir den Raum der Polynome vom Grad $\leq n$ in jeder Variablen. Sei $I_n : C([0, 1]) \rightarrow \mathbb{P}_n$ ein Interpolationsoperator mit Lebesguekonstante Λ_n . Definieren Sie I_n^x, I_n^y, I_n^{2D} durch

$$(I_n^x f)(x, y) = (I_n f(\cdot, y))(x), \quad (I_n^y f)(x, y) = (I_n f(x, \cdot))(y), \quad I_n^{2D} := I_n^x \circ I_n^y$$

- a) Seien $x_i, i = 0, \dots, n$ die Interpolationspunkte, auf denen I_n beruht. Geben Sie I_n^x, I_n^y, I_n^{2D} explizit an. Zeigen Sie: $I_n^{2D} : C([0, 1]^2) \rightarrow \mathbb{P}_n^{2D}$, und I_n^{2D} ist ein linearer Operator.
- b) Zeigen Sie: $I_n^x \circ I_n^y = I_n^y \circ I_n^x$.
- c) Zeigen Sie:

$$\|u - I_n^{2D} u\|_{C([0,1]^2)} \leq (1 + \Lambda_n^2) \inf_{\pi \in \mathbb{P}_n^{2D}} \|u - \pi\|_{C([0,1]^2)}.$$

- d) Für ein beliebiges Intervall $I \subset \mathbb{R}$ wird der skalierte Interpolationsoperator $I_{n,I} : C(I) \rightarrow \mathbb{P}_n$ durch die “natürliche” Variablensubstitution definiert. Formal: $I_{n,I} f := (I_n(f \circ L)) \circ L^{-1}$, wobei $L : [0, 1] \rightarrow I$ die affine Bijektion ist. Analog wird der 2D-Produktinterpolationsoperator auf einem beliebigen Rechteck $I_x \times I_y$ definiert durch $I_{n,I_x \times I_y}^{2D} = I_{n,I_x}^x \circ I_{n,I_y}^y$. Zeigen Sie: Für beliebige Intervalle I, I_x, I_y gilt

$$\|u - I_{n,I} u\|_{C(I)} \leq (1 + \Lambda_n) \inf_{\pi \in \mathbb{P}_n} \|u - \pi\|_{C(I)}, \quad \|u - I_{n,I_x \times I_y}^{2D} u\|_{C(I)} \leq (1 + \Lambda_n)^2 \inf_{\pi \in \mathbb{P}_n^{2D}} \|u - \pi\|_{C(I)}$$

2. a) Sei $a > 0$ und $d \geq 2a$. Zeigen Sie für die Funktion $f : r \mapsto \ln r$, daß auf $[d, d + 2a]$ gilt: Es existieren Konstanten $C > 0, q \in (0, 1)$, welche *nicht* von a, d, n abhängen, so daß

$$\inf_{\pi \in \mathbb{P}_n} \|f - \pi\|_{C([d, d+2a])} \leq Cq^n \quad \forall n \in \mathbb{N}.$$

- b) Seien $I_x, I_y \subset \mathbb{R}$ zwei Intervalle der Länge $a > 0$ mit $d := \text{dist}(I_x, I_y) \geq 2a$. Zeigen Sie, daß die Funktion K aus (2) auf $I_x \times I_y$ gut mit Polynomen approximiert werden kann:

$$\inf_{\pi \in \mathbb{P}_n^{2D}} \|K - \pi\|_{C([I_x \times I_y])} \leq Cq^n.$$

hier hängen $C > 0, q \in (0, 1)$ wieder *nicht* von a und d ab. (*Hinweis*: Nehmen Sie an, daß I_x links von I_y liegt und betrachten Sie $(x, y) \mapsto \ln(x - y)$ und verwenden Sie Teilaufg. a)).

Folgern Sie für den skalierten Tschebyscheffinterpolanten $I_{n, I_x \times I_y}^{2D, Tsch}$ und geeignete Konstanten $C > 0, \tilde{q} \in (0, 1)$:

$$\|K - I_{n, I_x \times I_y}^{2D, Tsch} K\|_{C([I_x \times I_y])} \leq C\tilde{q}^n.$$

3. Wir zeigen nun, daß Matrizen M von der Bauart (1) mit *glatter* K gut durch Matrizen mit kleinem Rang approximiert werden können. Als Beispiel betrachten wir

$$K^{glatt}(x, y) := e^{x+y} \quad (3)$$

und die Matrix M^{glatt} ist definiert durch $M_{ij}^{glatt} = K^{glatt}(x_i, x_j)$ mit $x_i = (i - 1)h, h = \frac{1}{N-1}$.

- a) Zeigen Sie für den 2D Tschebyscheffinterpolationoperator $I_n^{2D, Tsch}$ gilt

$$\|K^{glatt} - I_n^{2D, Tsch} K^{glatt}\|_{C([0,1]^2)} \leq Cq^n$$

für geeignetes $C > 0, q \in (0, 1)$. Approximieren Sie M^{glatt} , indem Sie die Funktion K^{glatt} durch Ihre Tschebyscheffinterpolante ersetzen:

$$M_{ij}^{glatt, n} := I_n^{2D, Tsch} K^{glatt}(x_i, x_j), \quad i, j = 1, \dots, N.$$

- b) Zeigen Sie: Es gilt für den Fehler $\|M^{glatt} - M^{glatt, n}\|_F \leq Cq^n N$, wobei $C > 0$ und $q \in (0, 1)$ nicht von n und N abhängen. Hier ist $\|\cdot\|_F$ die Frobeniusnorm

$$\|B\|_F := \sqrt{\sum_{i,j} |B_{ij}|^2}.$$

- c) Zeigen Sie: $M^{glatt, n}$ ist eine Matrix mit Rang (höchstens) $n + 1$. Insbesondere läßt sie sich darstellen in der Form

$$M^{glatt, n} = W S V^T, \quad W, V \in \mathbb{R}^{N \times (n+1)}, \quad S \in \mathbb{R}^{(n+1) \times (n+1)} \quad (4)$$

Hier ist

$$S_{ij} = K^{glatt}(\xi_i^{Tsch}, \xi_j^{Tsch}), \quad i, j = 0, \dots, n,$$

wobei $\xi_i^{Tsch}, i = 0, \dots, n$ die auf $[0, 1]$ skalierten Tschebyscheffpunkte sind. Geben Sie die Matrizen W, V an. Wieviel Speicher (d.h. *double precision* Zahlen) benötigen Sie, um die Matrizen W, V, S abzuspeichern?

4. a) Für die Funktion K aus (2) ist der Approximationsfehler $K - I_{n, [0,1] \times [0,1]}^{2D, Tsch} K$ groß auf $[0, 1]^2$. Jedoch läßt sich K *stückweise* durch seine Tschebyscheffinterpolante gut approximieren. Zeigen Sie: Falls zwei Intervalle $I_x, I_y \subset [0, 1]$ die Zulässigkeitsbedingung

$$\max\{\text{diam} I_x, \text{diam} I_y\} \leq \frac{1}{2} \text{dist}(I_x, I_y) \quad (5)$$

erfüllen, dann gilt für den Interpolationsfehler auf $I_x \times I_y$:

$$\|K - I_{n, I_x \times I_y}^{2D, Tsch} K\|_{C(I_x \times I_y)} \leq Cq^n.$$

- b) Für eine Teilmenge $\mathcal{I} = \{i \in \mathbb{N} \mid I_{lower} \leq i \leq I_{upper}\}$ der Indices bezeichnet $\text{BB}(\mathcal{I})$ die “bounding box”, d.h. das kleinste Intervall, das alle Punkte x_i , $i \in \mathcal{I}$, enthält (siehe (1) für die Definition von x_i). Damit können wir den Zulässigkeitsbegriff auf Indexmengen ausdehnen: Zwei Teilmengen $\mathcal{I}, \mathcal{J} \subset \{1, \dots, N\}$ heißen zulässig, falls

$$\max\{\text{diamBB}(\mathcal{I}), \text{diamBB}(\mathcal{J})\} \leq \frac{1}{2} \text{dist}(\text{BB}(\mathcal{I}), \text{BB}(\mathcal{J})). \quad (6)$$

Sind \mathcal{I} und \mathcal{J} zulässig, dann kann der zugehörige Matrixblock $M(\mathcal{I}, \mathcal{J})$ durch eine Matrix vom Rang $n + 1$ nach obigem Schema gut approximiert werden, indem in der Definition von $M(\mathcal{I}, \mathcal{J})$ die Funktion K durch $I_{n, \text{BB}(\mathcal{I}) \times \text{BB}(\mathcal{J})}^{2D, Tsch} K$ ersetzt wird; dies führt dann wie in Aufg. 3,c) auf eine Approximation der Form

$$M^{approx}(\mathcal{I}, \mathcal{J}) := WSV^\top, \quad \text{mit } W \in \mathbb{R}^{|\mathcal{I}| \times (n+1)}, \quad V \in \mathbb{R}^{|\mathcal{J}| \times (n+1)}, \quad S \in \mathbb{R}^{(n+1) \times (n+1)}$$

5. Die Aufgabe ist deshalb, eine möglichst gute Partition der Matrix M in gut approximierbare Blöcke zu finden. Jeder Matrixblock wird beschrieben durch

```

struct block {
int I_lower , I_upper ; // Grenzen der Zeilenindizes des Blocks
int J_lower , J_upper ; // Grenzen des Spaltenindizes des Blocks
int rank ; // der Rang
double* V ; // V ist (I_upper - I_lower + 1) * rank Matrix
double* W ; // W ist (J_upper - J_lower + 1) * rank Matrix
double* S ; // S ist eine rank * rank Matrix
double* F ; // Darstellung des Blocks als volle Matrix F
// Groesse: (I_upper - I_lower + 1) * (J_upper - J_lower + 1)
int typ ; // typ=EXACT: Darstellung des Blocks als Matrix F
// typ=APPROX: Darstellung des Blocks als W S V'
}

```

Der folgende *greedy algorithm* erzeugt nun eine Liste `blocklist` mit Elementen vom Typ `struct block`, so daß jeder Block entweder zulässig ist (und damit wird der Block in der Form WSV^\top dargestellt) oder er ist “klein” (und wird damit als vollbesetzte Matrix F dargestellt):

```

build_blocks ( I_lower , I_upper , J_lower , J_upper , blocklist ) {
if ( ( I_upper == I_lower ) || ( J_upper == J_lower ) ) {
    block newblock ;
    newblock . I_lower = I_lower ; newblock . I_upper = I_upper ;
    newblock . J_lower = J_lower ; newblock . J_upper = J_upper ;
    newblock . typ = EXACT ; // Block wird als Matrix M dargestellt
    append_to_blocklist ( blocklist , newblock ) ;
}
else {
    if admissible ( I_lower , I_upper , J_lower , J_upper ) {
        block newblock ;
        newblock . I_lower = I_lower ; newblock . I_upper = I_upper ;
        newblock . J_lower = J_lower ; newblock . J_upper = J_upper ;
        newblock . typ = APPROX ; // Block wird in der Form W S V' dargestellt
        append_to_blocklist ( blocklist , newblock ) ;
    } else {
        I_mid = ( I_lower + I_upper ) / 2 ;
        J_mid = ( J_lower + J_upper ) / 2 ;

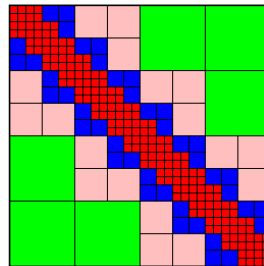
```

```

    build_blocks ( I_lower , I_mid    , J_lower , J_mid , blocklist );
    build_blocks ( I_mid+1, I_upper , J_lower , J_mid , blocklist );
    build_blocks ( I_lower , I_mid    , J_mid+1, J_upper , blocklist );
    build_blocks ( I_mid+1, I_upper , J_mid+1, J_upper , blocklist );
  }
}
}

```

Die Blockstruktur, die dieser Algorithmus erzeugt, sieht dann im Prinzip so aus:



Programmieren Sie die Erzeugung dieser Blockliste. Greifen Sie hierzu auf die Datenstrukturen zurück, die Sie auf http://www.math.tuwien.ac.at/~melenk/teach/numerik_WS0809/projekte finden. Gehen Sie in mehreren Schritten vor:

1. Schreiben Sie die oben fehlenden Funktionen `append_to_blocklist` und `admissible`.
2. In einem ersten Durchlauf wird `build_blocks (1, N, 1, N, blocklist)` aufgerufen, um die Blockliste (ohne die Matrizen F , V , W , S) zu erzeugen.
3. In einem zweiten Durchlauf, werden dann die Matrizen V , W , S oder F für jeden Block erzeugt. Schreiben Sie ein Programm, das durch `blocklist` geht und für jeden Block die entsprechende Matrix erzeugt, d.h.
 - falls `typ = EXACT`, dann wird die Matrix F mit den Werten von $M|_{\mathcal{I} \times \mathcal{J}}$ gefüllt.
 - falls `typ = APPROX`, dann werden die Matrizen W , S , V mittels Tschebysheffinterpolation erzeugt.
6. In der vorangegangenen Aufgabe haben Sie die Matrix M blockweise approximiert. Es soll nun numerisch untersucht werden, wie groß der Speicherbedarf für `blocklist` ist und wie genau die erhaltene Approximation ist.
 - a) Für festes $n = 0, \dots, 5$ und $N = 2^i$, $i = 2, \dots, 15$ berechnen Sie den Speicherbedarf Ihrer `blocklist`, indem Sie `blocklist` aufstellen und anschließend einmal durchgehen und den Speicherbedarf aller Blöcke aufsummieren. Stellen Sie das Ergebnis graphisch dar.
 - b) Für festes $N = 2^i$, $i = 6, \dots, 10$ und $n = 0, \dots, 5$ bestimmen Sie die Genauigkeit Ihrer Approximation in der Frobeniusnorm. Gehen Sie hierzu wie folgt vor: Gehen Sie durch `blocklist`, stellen Sie für jeden Block des Typs APPROX die Matrix $M|_{\mathcal{I} \times \mathcal{J}}$ exakt auf und bestimmen Sie $\|M|_{\mathcal{I} \times \mathcal{J}} - WSV'\|_F^2$. Summieren Sie diese Fehler über alle Blöcke. Stellen Sie das Ergebnis graphisch dar.
7. Die Matrix M hat spezielle Struktur¹: Überlegen Sie sich, daß, $M_{ij} = m_{i-j}$ für einen geeigneten Vektor m . Formulieren Sie einen FFT-basierten Algorithmus, mit dem man die Matrix-Vektor-Multiplikation $x \mapsto Mx$ mit Aufwand $O(N \log N)$ für Zweierpotenzen N realisieren kann.

¹ M ist eine sog. Töplitzmatrix