Programmierblatt 3.

Bearbeiten bis: **Sonntag, 14.05.2023**

Auf dem zweiten Programmierblatt haben wir hierarchische Matrizen kennengelernt, welche für die Annäherung von Kernmatrizen benutzt werden können. Anstatt die Nebendiagonalblöcke direkt zu approximieren, wollen wir nun das Abklingverhalten der Kerne ausnutzen. Sind nämlich μ, ν Cluster und I_μ, I_ν die damit assoziierten Intervalle, garantiert das exponentielle Abklingverhalten des Kerns, dass der Matrixblock $\mathbf{K}_{\mu,\nu}$ gut angenähert werden kann sofern $\operatorname{dist}(I_\mu, I_\nu)$ hinreichend gross ist.

Blockweise Niedrigrangapproximation revisited

Wir betrachten äquidistante, dyadische Punkte auf dem Einheitsintervall $I := [0,1)$, das heisst, für $j \in \mathbb{N}$ sei $x_{j,k} = k \cdot 2^{-j}$, $k = 0, \dots, 2^j - 1$. Wir erinnern uns an die Cluster

$$\nu_{j,\ell} := \left\{ 2^{J-j}\ell, \dots, 2^{J-j}(\ell+1) - 1 \right\}, \qquad \ell = 0, \dots, 2^j - 1.$$

Das mit einem Cluster $\nu_{j,\ell}$ assoziierte Teilintervall bezeichnen wir von nun an mit $I_{j,\ell}$. Anders als im vorangehenden Blatt wollen wir nun feiner unterscheiden, welche Matrixblöcke wir approximieren. Dazu sei $\eta > 0$ ein vorgegebener Parameter. Wir approximieren den zu zwei Clustern $\nu_{j,\ell}$ und $\nu_{j,\ell'}$ gehörigen Matrixblock, falls die Zulässigkeitsbedingung

$$\operatorname{dist}(I_{j,\ell},\, I_{j,\ell'}) \geq 2^{-j}\eta \tag{1}$$

erfüllt ist. Ansonsten verfahren wir rekursiv weiter, bis wir ein maximales Level erreicht haben. Dazu sei angemerkt, dass

$$\operatorname{diam} I_{j,\ell} = \operatorname{diam} I_{j,\ell'} = 2^{-j}$$

gilt.

Bei der Approximation ist es wichtig, zwischen vollen und zulässigen Matrixblöcken zu unterscheiden. Die zulässigen Matrixblöcke wollen wir vorerst nochmals mittels *ACA* annähern. Alle Eigenschaften sollen in *logical*-Variablen gespeichert werden. Dies führt auf den folgenden Algorithmus:

Algorithmus Blockweise Niedrigrangapproximation

Input: Cluster $\nu_{j,\ell}$, $\nu_{j,\ell'}$, Level J, maximales Level L, Kernfunktion k, $\eta > 0$.
Output: Hierarchische Matrix $\mathbf{T}$.

1: **if** $\operatorname{dist}(I_{j,\ell}, I_{j,\ell'}) \geq 2^{-j}\eta$ **then**
2: setze $\mathbf{T}$.`isAdmissible := true`
3: assembliere eine Niedrigrangapproximation des Matrixblocks $\mathbf{LR} \approx \mathbf{K}_{\nu_{j,\ell},\nu_{j,\ell'}}$
4: **else if** $j = L$ **then**
5: setze $\mathbf{T}$.`isFull := true`
6: assembliere den vollen Block $\mathbf{K}_{\nu_{j,\ell},\nu_{j,\ell'}}$
7: **else**
8: berechne die vier Söhne von $\mathbf{T}$ mit diesem Algorithmus bezüglich der Clusterpaare

$$(\nu_{j+1,2\ell}, \nu_{j+1,2\ell'}),\ (\nu_{j+1,2\ell}, \nu_{j+1,2\ell'+1}),\ (\nu_{j+1,2\ell+1}, \nu_{j+1,2\ell'}),\ (\nu_{j+1,2\ell+1}, \nu_{j+1,2\ell'+1}).$$

9: **end if**

Da wir nun auf jedem Level mehrere Cluster-Cluster-Interaktionen haben, müssen wir auch den Algorithmus zur Matrix-Vektor-Multiplikation entsprechend anpassen.

Algorithmus Matrix-Vektor-Multiplikation

Input: hierarchische Matrix $\mathbf{T}$, Vektor $\mathbf{x} \in \mathbb{R}^{2^J}$
Output: $\mathbf{b} = \mathbf{T} \cdot \mathbf{x}$

1: **if** $\mathbf{T}$.`isAdmissible` **then**
2: setze $\mathbf{b} = \mathbf{T.L} \cdot (\mathbf{T.R} \cdot \mathbf{x})$
3: **else if** $\mathbf{T}$.`isFull` **then**
4: setze $\mathbf{b} = \mathbf{T.K} \cdot \mathbf{x}$
5: **else**
6: unterteile $\mathbf{b}^\intercal = [\mathbf{b}_1^\intercal, \mathbf{b}_2^\intercal]$ und analog $\mathbf{x}^\intercal = [\mathbf{x}_1^\intercal, \mathbf{x}_2^\intercal]$
7: berechne rekursiv mit diesem Algorithmus

$$\mathbf{b}_1 = \mathbf{T}.\texttt{son1} \cdot \mathbf{x}_1 + \mathbf{T}.\texttt{son2} \cdot \mathbf{x}_2, \qquad \mathbf{b}_2 = \mathbf{T}.\texttt{son3} \cdot \mathbf{x}_1 + \mathbf{T}.\texttt{son4} \cdot \mathbf{x}_2$$

8: **end if**

Aufgabe 1. Schreiben Sie Funktionen

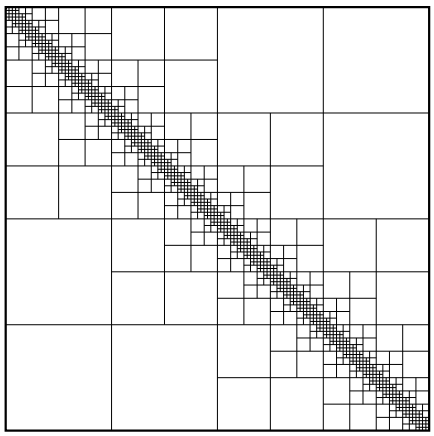
```
function T = build_Hmatrix_ACA(J, eta, k, tol, L, j, l1, l2),
function b = Hmatrix_vector_mult_ACA(T, x),
```

welche eine hierarchische Matrix aufstellen, respektive die Multiplikation mit einem Vektor implementieren.

Aufgabe 2. Testen Sie Ihre Implementierung. Benutzen Sie $J = 10$ und $L = 5$, um die hierarchische Matrix $\mathbf{T}$ aufzustellen und stellen Sie zusätzlich die volle Matrix $\mathbf{K}$ auf. Generieren Sie anschliessend $N = 10$ normierte und zentrierte Zufallsvektoren der Länge 2^J. Messen Sie jeweils die Fehler

$$\|\mathbf{Tx} - \mathbf{Kx}\|_2.$$

Verwenden Sie hierfür den Gauß-Kern mit Korrelationslänge $\sigma = 1$, $\eta = 1.5$ und $\texttt{tol} = 10^{-4}$. Ihre Fehler sollten im Bereich 10^{-7} liegen.



Polynomiale Approximation

Eine weitere Möglichkeit, zulässige Matrixblöcke zu approximieren, bieten Polynome. Ist etwa der Abstand $|x - x'|$ hinreichend gross, so gilt

$$k(x, x') \approx \sum_{m,n=1}^{p} \alpha_{m,n}(x - x_c)^m (x' - x'_c)^n, \tag{2}$$

wobei $x_c, x'_c \in I$ sinnvoll gewählt und $\alpha_{m,n}$ die Koeffizienten eines interpolierenden bivariaten Polynoms seien. Erfüllen zwei Cluster $v_{j,\ell}$ und $v_{j,\ell'}$ die Zulässigkeitsbedingung (1), so lässt sich der zugehörige Matrixblock gemäss

$$\begin{aligned}
\left[k(x_r, x_s)\right]_{\substack{r\in v_{j,\ell}\\ s\in v_{j,\ell'}}} &\approx \left[\sum_{m,n=1}^{p} \alpha_{m,n}(x_r - x_c)^m (x_s - x'_c)^n\right]_{\substack{r\in v_{j,\ell}\\ s\in v_{j,\ell'}}} \\
&= \left[(x_r - x_c)^m\right]_{\substack{r\in v_{j,\ell}\\ m\le p}} \left[\alpha_{m,n}\right]_{\substack{m\le p\\ n\le p}} \left[(x_s - x'_c)^n\right]^{\mathsf{T}}_{\substack{s\in v_{j,\ell'}\\ n\le p}} \\
&=: \mathbf{X}_{v_{j,\ell}} \mathbf{K}_{v_{j,\ell}, v_{j,\ell'}} \mathbf{X}^{\mathsf{T}}_{v_{j,\ell'}}
\end{aligned}$$

approximieren. Dabei wählen wir x_c und x'_c als Mittelpunkte der zugehörigen Cluster $v_{j,\ell}$ und $v_{j,\ell'}$. Ist zusätzlich $p \ll 2^{J-j}$, so haben wir eine Niedrigrangapproximation an den Matrixblock gefunden.

Ein wichtiger Punkt bei der Assemblierung der hierarchischen Matrix $\mathbf{T}$ ist die Bestimmung der Koeffizienten des interpolierenden Polynoms. Ist beispielsweise $p = 1$, so sollen die Koeffizienten $\alpha_{m,n}$ aus (2) dem Gleichungssystem

$$\begin{bmatrix}
1 & x_1 - x_c & x'_1 - x'_c & (x_1 - x_c)(x'_1 - x'_c) \\
1 & x_1 - x_c & x'_2 - x'_c & (x_1 - x_c)(x'_2 - x'_c) \\
1 & x_2 - x_c & x'_1 - x'_c & (x_2 - x_c)(x'_1 - x'_c) \\
1 & x_2 - x_c & x'_2 - x'_c & (x_2 - x_c)(x'_2 - x'_c)
\end{bmatrix}
\begin{bmatrix} \alpha_{0,0} \\ \alpha_{1,0} \\ \alpha_{0,1} \\ \alpha_{1,1} \end{bmatrix}
=
\begin{bmatrix} k(x_1, x'_1) \\ k(x_1, x'_2) \\ k(x_2, x'_1) \\ k(x_2, x'_2) \end{bmatrix} \tag{3}$$

genügen. Hierbei müssen die Interpolationspunkte x_ℓ und $x'_{\ell'}$ paarweise verschieden gewählt werden. Für eine möglichst genaue Approximation empfehlen sich *Tschebyscheff-Knoten*. Diese sind auf einem Intervall $[a, b]$ gegeben durch

$$x_\ell = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{2k+1}{2p+2}\pi\right), \qquad \ell = 0, 1, \ldots, p.$$

Die Punkte x_c und x'_c entsprechen dabei den Mittelpunkten der involvierten Cluster.

Aufgabe 3. Schreiben eine Funktion

```
function T = build_Hmatrix_MP(J, eta, k, p, L, j, l1, l2),
```

welche eine hierarchische Matrix aufstellen und dabei die zulässigen Matrixblöcke mittels Polynomen approximiert. Ein zulässiger Matrixblock soll dabei nur die Variablen `T.isFull`, `T.isAdmissible` und `T.K` enthalten, wobei `T.K` die Koeffizienten α des Interpolationspolynoms bezüglich der Tschebyscheff-Knoten enthält. Diese können dabei gemäss (3) berechnet werden.

Aufgabe 4. Schreiben Sie eine Funktion

```
function b = Hmatrix_vector_mult_MP(T, x, J, p, j, l1, l2),
```

welche die Matrix-Vektor-Multiplikation einer hierarchischen Matrix $\mathbf{T}$ mit einem Vektor $\mathbf{x}$ implementiert. Dabei sollen für zulässige Matrixblöcke die Momentmatrizen $\mathbf{X}_{v,\ell}$ direkt in der Matrix-Vektor-Multiplikation aufgestellt werden.

Aufgabe 5. Testen Sie Ihre Implementierung. Benutzen Sie dafür dieselbe Methode wie in Aufgabe 2, wählen Sie jedoch den Kern $k(x, y) := (1 - x)(1 - y)$. Da k ein bilineares Polynom ist, sollten Sie bereits für $p = 1$ Fehler im Bereich der Maschinengenauigkeit erhalten.

Wir haben bisher vernachlässigt, dass die Systemmatrix in Gleichung (3) für einen wachsenden Polynomgrad schlecht konditioniert ist. Um dies zu umgehen, könnte man beispielsweise die Newton-Interpolation verwenden, um die Polynome zu bestimmen. Etwas weniger stabil, für unsere Anwendung jedoch ausreichend, ist auch eine angemessene Skalierung der Systemmatrix.

Da für jedes Cluster diam $v_{j,\ell} = 2^{-j}$ gilt und der Polynomgrad $q \leq 2p$ innerhalb jeder Spalte der Matrix konstant ist, können wir jene Spalte (und damit den zugehörigen Koeffizienten α) mit 2^{qj} skalieren. Konkret löst man stattdessen für $p = 1$

$$\begin{bmatrix} 1 & 2^j(x_1 - x_c) & 2^j(x_1' - x_c') & 2^{2j}(x_1 - x_c)(x_1' - x_c') \\ 1 & 2^j(x_1 - x_c) & 2^j(x_2' - x_c') & 2^{2j}(x_1 - x_c)(x_2' - x_c') \\ 1 & 2^j(x_2 - x_c) & 2^j(x_1' - x_c') & 2^{2j}(x_2 - x_c)(x_1' - x_c') \\ 1 & 2^j(x_2 - x_c) & 2^j(x_2' - x_c') & 2^{2j}(x_2 - x_c)(x_2' - x_c') \end{bmatrix} \begin{bmatrix} \beta_{0,0} \\ \beta_{1,0} \\ \beta_{0,1} \\ \beta_{1,1} \end{bmatrix} = \begin{bmatrix} k(x_1, x_1') \\ k(x_1, x_2') \\ k(x_2, x_1') \\ k(x_2, x_2') \end{bmatrix} \tag{4}$$

und setzt anschliessend

$$\left(\alpha_{0,0},\ \alpha_{1,0},\ \alpha_{0,1},\ \alpha_{1,1}\right)^\mathsf{T} = \left(\beta_{0,0},\ 2^j\alpha_{1,0},\ 2^j\alpha_{0,1},\ 2^{2j}\alpha_{1,1}\right)^\mathsf{T}. \tag{5}$$

Für $p > 1$ kann diese Methode entsprechend verallgemeinert werden, wobei dann die Koeffizienten lexikographisch angeordnet werden sollen. Dies bedeutet, wir ordnen

$$\boldsymbol{\beta} = \left(\beta_{0,0},\ \beta_{1,0},\ \ldots,\ \beta_{p,0},\ \beta_{0,1},\ \beta_{1,1},\ \ldots,\ \beta_{p,1},\ \beta_{0,2},\ \ldots,\ \beta_{p,p}\right)^\mathsf{T},$$

und lösen das entsprechende $\left((p + 1)^2 \times (p + 1)^2\right)$-Gleichungssystem.

Aufgabe 6. Ändern Sie Ihre Funktion `build_Hmatrix_MP`, so dass die Koeffizienten der Niedrigrangapproximation durch (4) und (5) berechnet werden. Wiederholen Sie den Test aus Aufgabe 5.

Aufgabe 7. Wählen Sie nun verschiedene Kerne und Polynomgrade, um die Kernmatrix $\mathbf{K}$ zu approximieren. Messen Sie analog zu den vorherigen Aufgaben für $J = 10$, $L = 5$, $\eta = 1.5$ und $\sigma = 0.5$ den durchschnittlichen Fehler in der Matrix-Vektor-Multiplikation für die Polynomgrade $p = 1, 2, \ldots, 9$. Benutzen Sie dafür jeweils $N = 50$ Zufallsvektoren. Stellen Sie Ihre Fehler für den Exponential-, den Matérn-$\frac{3}{2}$- und den Gauß-Kern in einem *semilogy*-Plot dar. Sie sollten eine exponentielle Konvergenz beobachten.