Übungsblatt 2 – Cerberus

Aufgabe 1

Im Folgenden soll das 512×512 Bild aus Abbildung 1a mithilfe der Singulärwertzerlegung

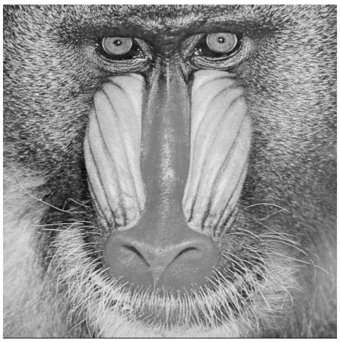
$$\mathbf{A} = \mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{V}^{\mathrm{T}}$$

und der Rang-k-Approximation komprimiert werden. Die Rekonstruktion und Approximation erfolgt mithilfe von

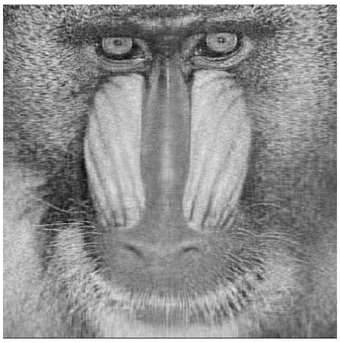
$$\tilde{\mathbf{A}} = \sum_{i=1}^{k} \sigma_i \vec{u}_i \vec{v}_i^{\mathrm{T}} \quad \text{mit } k \leq 512.$$

$\sigma_i \mathrel{\hat{=}}$ i-tes Diagonalelement von $\mathbf{\Sigma}$, $\vec{u}_i \mathrel{\hat{=}}$ i-ter Spaltenvektor von $\mathbf{U}$, $\vec{v}_i \mathrel{\hat{=}}$ i-iter Spaltenvektor von $\mathbf{V}$.

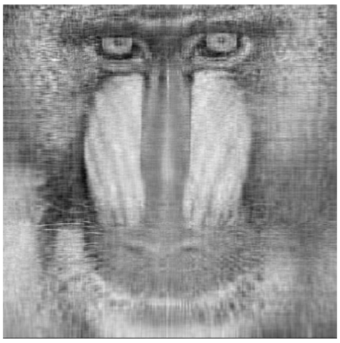
Die approximierten Bilder für $k = 10, 20, 50$ befinden sich in Abbildung 1. Die SVD



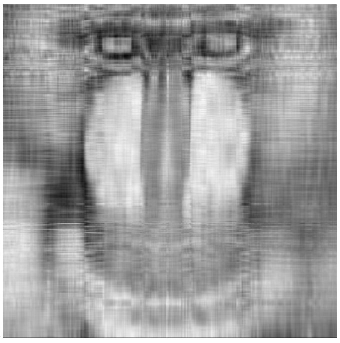
(a) Originalbild.



(b) $k = 50$.



(c) $k = 20$.



(d) $k = 10$.

Abbildung 1: Approximation von Abbildung 1a für verschiedene k.

scheint sich gut zur Kompression zu eignen, da sich das Bild, wie in Abbildung 1b zu sehen, mit nur knapp 10 % der Singulärwerte schon sehr gut rekonstruieren lässt. Selbst bei einer so extremen Kompression wie in Abbildung 1d ist das Motiv noch ganz grob zu erkennen.

Aufgabe 2

Ein Profiler wird verwendet um die Geschwindigkeit verschiedener Abschnitte einer LU-Zerlegung zu überprüfen. Ein Timer überprüft die benötigte Zeit

1. eine $N \times N$-Matrix M und einen Nd-Vektor b mit zufälligen Einträgen zu erzeugen
2. eine LU-Zerlegung durchzuführen
3. das Problem $Mx = b$ zu lösen

Die Zeit t, die für die einzelnen Schritte benötigt wird, wird in Abhängigkeit von der Matrixgröße N doppelt-logarithmisch aufgetragen. Die LU-Zerlegung mithilfe der `eigen`-Library unterstützt Multithreading und kann somit abhängig vom verwendeten Prozessor beschleunigt werden (in diesem Fall bis zu 15%). Generell zeigt sich, dass die benötigte Zeit von der verwendeten CPU abhängt, jedoch lässt sich sowohl bei logarithmisch (2) wie auch linear (3) ansteigender Matrizengröße N ein Trend erkennen.

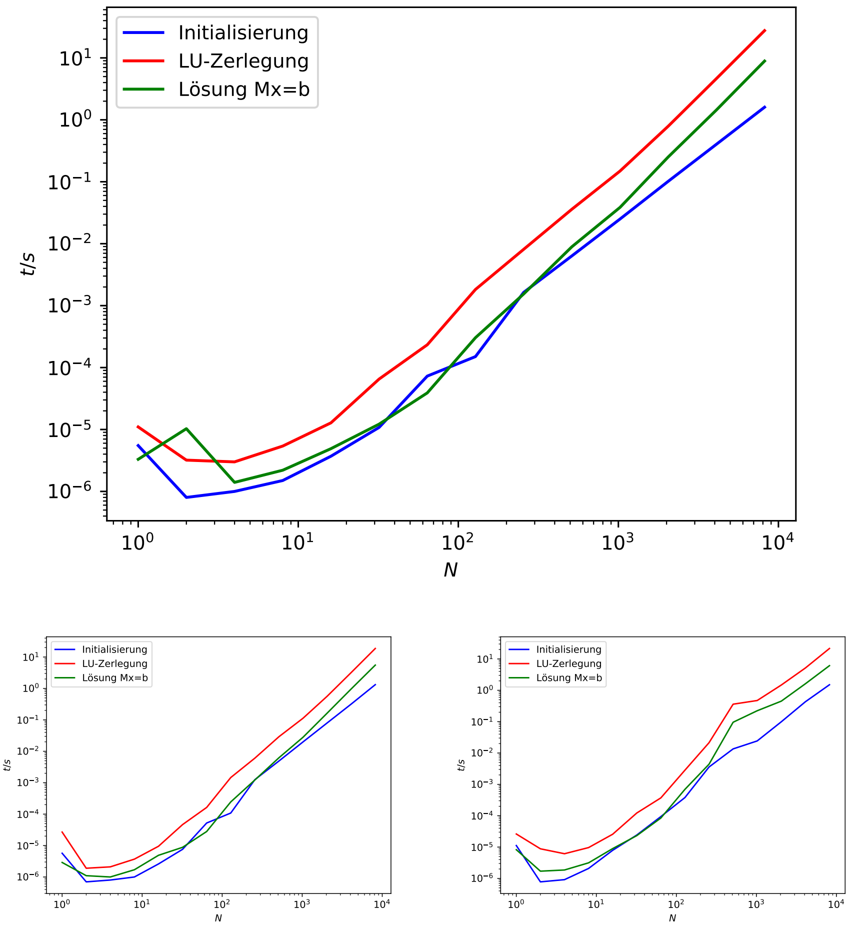


Abbildung 2: Die benötigte Zeit für Operationen bei logarithmisch ansteigendem N mit verschiedenen CPUs.

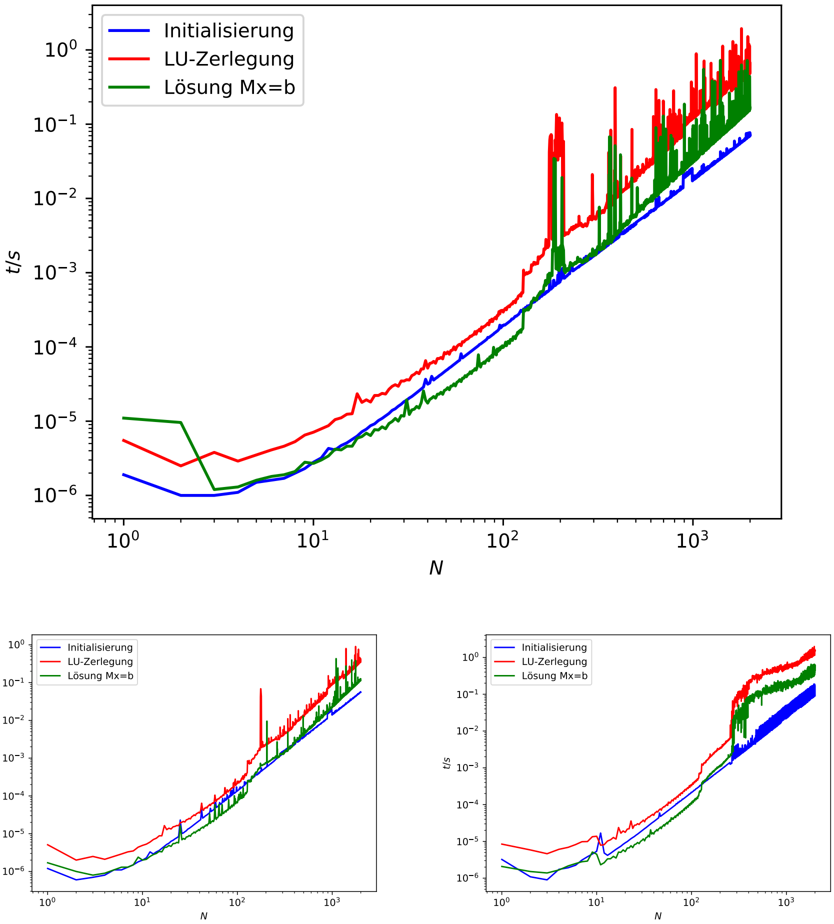


Abbildung 3: Die benötigte Zeit für Operationen bei linear ansteigendem N mit verschiedenen CPUs.

Aufgabe 3

In dieser Aufgabe wurde ein Profiler genutzt, um die Laufzeiten verschiedener Algorithmen zum Lösen eines linearen Gleichungssystems der Form

$$Mx = b \tag{1}$$

($M\hat{=}$Zufällige N x N Matrix, $x, b\hat{=}$Vektor mit Dimension N)

zu bestimmen. Dabei wird N für alle Methoden linear in dem Intervall $[1, 1000]$ variiert. Die Laufzeiten werden für die folgenden Methoden verglichen:

1. Multiplikation von M^{-1} auf der linken Seite
2. Partielle LU-Zerlegung
3. Vollständige LU-Zerlegung.

Wie in Abbildung 4 zu sehen ist, wird für die partielle LU-Zerlegung am wenigsten Zeit benötigt. Es sei zusätzlich erwähnt, dass die partielle LU-Zerlegung zusätzlich vom Multithreading profitiert [1], was ebenfalls Laufzeit sparen kann.

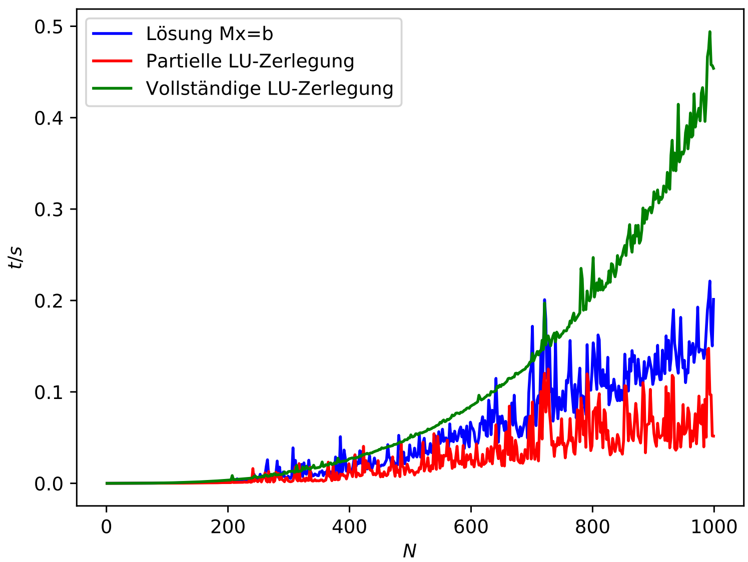


Abbildung 4: Laufzeit der verschiedenen Methoden.

Anschließend wurde verglichen, ob auch alle Algorithmen das gleiche Ergebnis liefern. Um diesen Vergleich für beliebige N zu quantisieren und vergleichbar zu machen, wurde für jede Methode der Betrag des resultierenden Vektors x gebildet. Anschließend wurde

[1] https://eigen.tuxfamily.org/dox/TopicMultiThreading.html

die Abweichung der partiellen und vollständigen LU-Zerlegung zu Methode 1 berechnet. Wie in Abbildung 5 zu sehen ist, ist die Abweichung der partiellen LU-Zerlegung am geringsten.

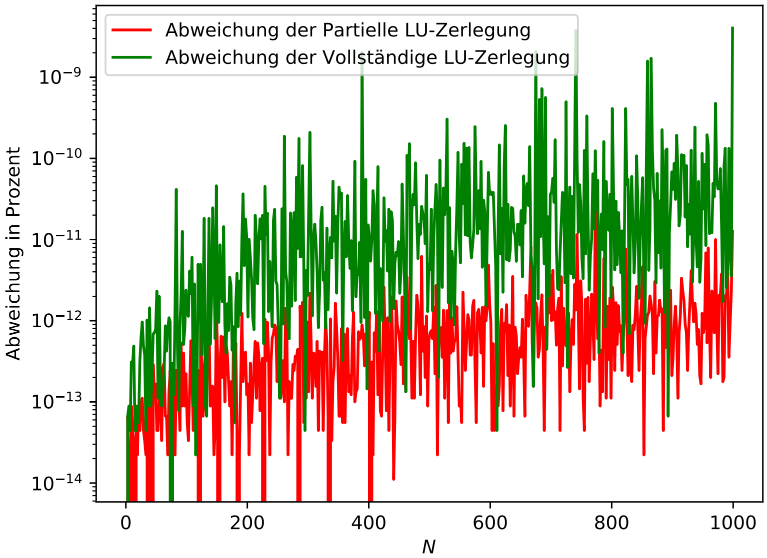


Abbildung 5: Abweichung der Partiellen und vollständigen LU-Zerlegung zu Methode 1.

Insgesamt ist also die partielle LU-Zerlegung zu bevorzugen, da die Laufzeit – bis auf wenige Ausnahmen – für alle N im untersuchten Intervall am geringsten ist. Zusätzlich liefert zu relative geringe Abweichungen im Vergleich zu Methode 1.