Wladimir David Zakrevskyy 70351576

Marc Fiedler 70484207

Manuel Erbach 70246441

Übung: Lienare Systeme

SS2016

In Rahmen der Vorlesung ‚Numeric Algorithms‘ an der Hochschule Ostfalia.

Aufgabe 2.

1. In Rahmen dieser Aufgabe wurden zwei Java Bibliotheken eingesetzt.

Jama <http://math.nist.gov/javanumerics/jama/>

Apache Commons Math v.3.0. <http://commons.apache.org/proper/commons-math/>

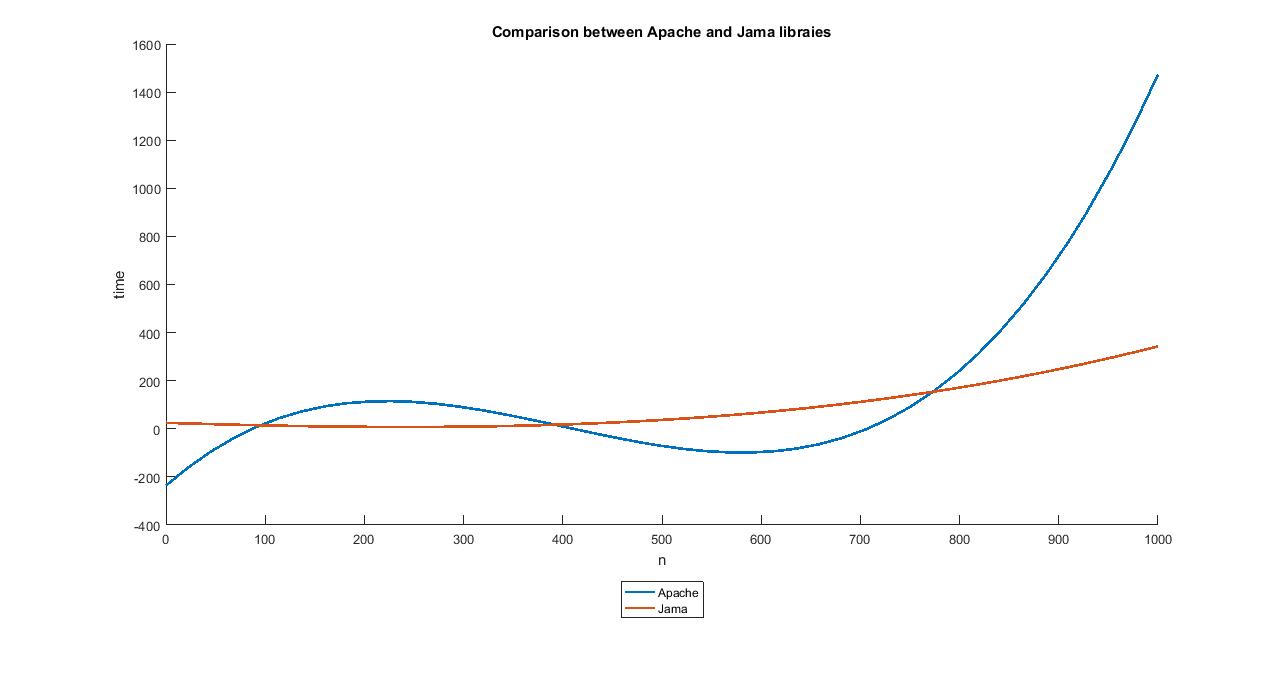
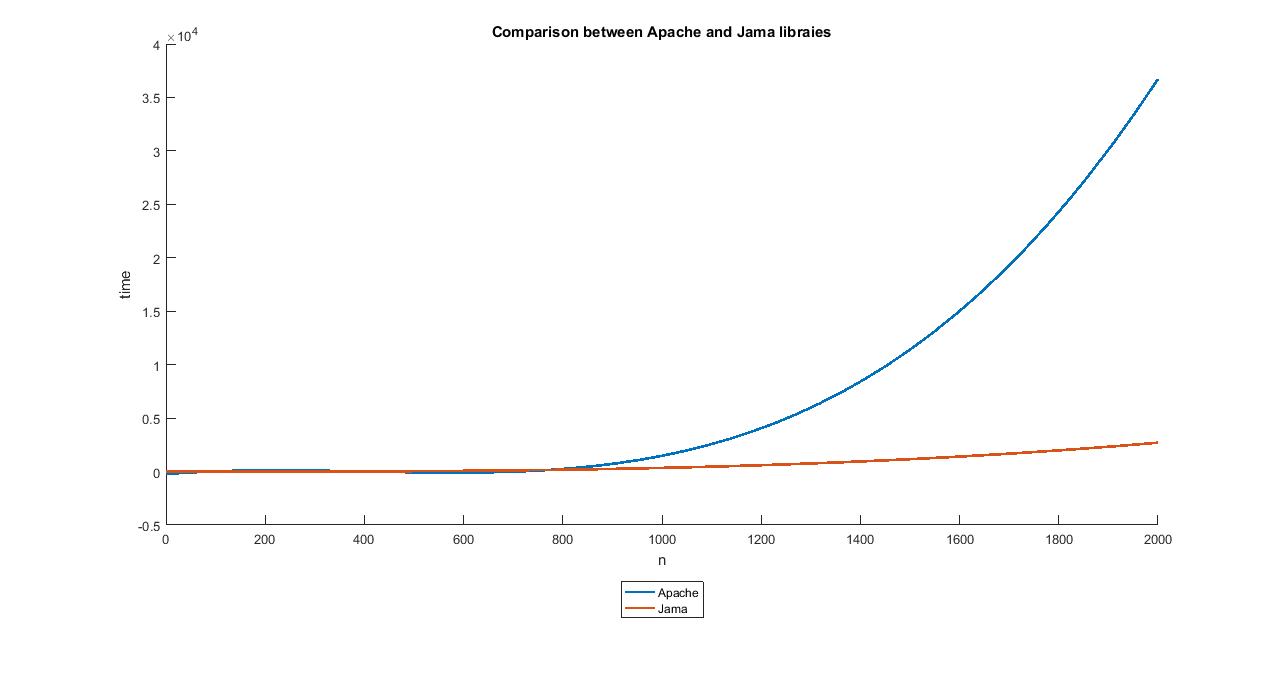
Die LU-Zerlegung einer Random Matrix wurde in Millisekunden gemessen, wobei für jede Matrix Größe *n* wurde drei Mal gemessen und wert entsprechend gemittelt. Bei Interesse Quellcode liegt auf Github[[1]](#footnote-1).

Folgende Tabellen stellen den gemessenen Ergebnissen in Millisekunden dar.

|  |  |
| --- | --- |
| Jama | Apache |
| |  |  | | --- | --- | | 100 | 2.33333333 | | 200 | 6.33333333 | | 300 | 9 | | 400 | 20.6666667 | | 500 | 39 | | 600 | 68.6666667 | | 700 | 109 | | 800 | 167 | | 900 | 249.666667 | | 1000 | 341.333333 | | 1100 | 461.666667 | | 1200 | 597.333333 | | 1300 | 748.666667 | | 1400 | 964.333333 | | 1500 | 1137 | | 1600 | 1417.33333 | | 1700 | 1667.66667 | | 1800 | 1984.66667 | | 1900 | 2309 | | |  |  | | --- | --- | | 100 | 3 | | 200 | 6.33333333 | | 300 | 11.6666667 | | 400 | 32 | | 500 | 66 | | 600 | 116 | | 700 | 195.333333 | | 800 | 294.333333 | | 900 | 539.666667 | | 1000 | 1106.66667 | | 1100 | 2566.33333 | | 1200 | 3923.33333 | | 1300 | 5712.33333 | | 1400 | 8342.66667 | | 1500 | 12288.6667 | | 1600 | 14205 | | 1700 | 20494.3333 | | 1800 | 23565.3333 | | 1900 | 27519.3333 | |

Gemessen wurde auf einem MacBook Pro Retina 15‘‘, mit einem 2,3 GHz Intel Core i7. Die Zahlen zeigen schon, dass die Bibliothek von Jama deutlich schneller ist. Vermutlich liegt es an den internen Implementierungen.

Mit eigens Implementierten Least-Square-Methode lassen sich Polynome des dritten Grades erzeugen. Die Implementierungen liegen ebenfalls auf Github.



|  |
| --- |
| Jama |
| *f(x)*= 2.873523678476959E-7 *x*3 + 1.572970416237519E-4 *x*2 - 0.12664244352719234 *x* + 24.95588235295793 |

In Graphen sind beide folgende Funktionen von Polynomgrad drei zu erkennen.

|  |
| --- |
| APACHE COMMONS MATH |
| *f(x)*= 9.341546267629971E-6 *x*3 - 0.011253745270038168 *x*2 + 3.621485208118603 *x* - 236.30065359482387 |

Vorhersagen für *n* = 10 000 und *n* = 100 000. Auf rechter Seite steht Wert in Millisekunden.

|  |  |
| --- | --- |
| Apache |  |
| n = 10 000 | 8.2522e+06 (2,6 Jahre) |
| n = 100 000 | 9.2294e+09 (2926,6 Jahre) |

|  |  |
| --- | --- |
| Jama |  |
| n = 10 000 | 3.0184e+05 (1,14 Monate) |
| n = 100 000 | 2.8891e+08 (91,61 Jahre) |

b)

Bestimmung von *flops* pro Sekunde.

Anzahl der Operationen zweier gleichen von Größer Matrizen lasst sich mit folgender Formeln ausrechnen: *n2 (n + (n - 1))*

Da eigene Implementierungen für Matrix Multiplikation zu langsam erwiesen wurde die Bibliothek Jama eingesetzt.

Eine Matrix was ≈1010 die Größe hat wurde ca. in einer Sekunde multipliziert. Da diese Matrix 2.0596e+09 Fließkomaoperationen hat bekommen wir ein Wert für *flops*: 2.05Gflops (Auf MacBook Pro mit i7 Prozessor).

Da Jama Framework perfomanter ist, wird nur diese in Betracht gezogen.

|  |
| --- |
| Jama |
| *f(x)*= 2.873523678476959E-7 *x*3 + 1.572970416237519E-4 *x*2 - 0.12664244352719234 *x* + 24.95588235295793 |

Um die Zeitkomplexität zu bestimmen müssen wird gegebenes Polynom mit Wert für *flops* multiplizieren.

*g(x) = f(x) \* 2.05Gflops =*

***g(x)*= 591.8309*x*3 + 3.2397e+05*x*2 - 2.6083e+08 *x* + 5.1399e+10**

1. https://github.com/davidwowa/NumericJava/blob/master/src/de/wdz/numeric/Exc2.java [↑](#footnote-ref-1)