Ausarbeitung Übung 3

Studienarbeit von Dominik Schiller, Constanze Kramer, Simon Arnold & Tobias Lingenberg
Datum: 8. Dezember 2020

Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Ausarbeitung Übung 3

Studienarbeit von Dominik Schiller, Constanze Kramer, Simon Arnold & Tobias Lingenberg

Datum: 8. Dezember 2020

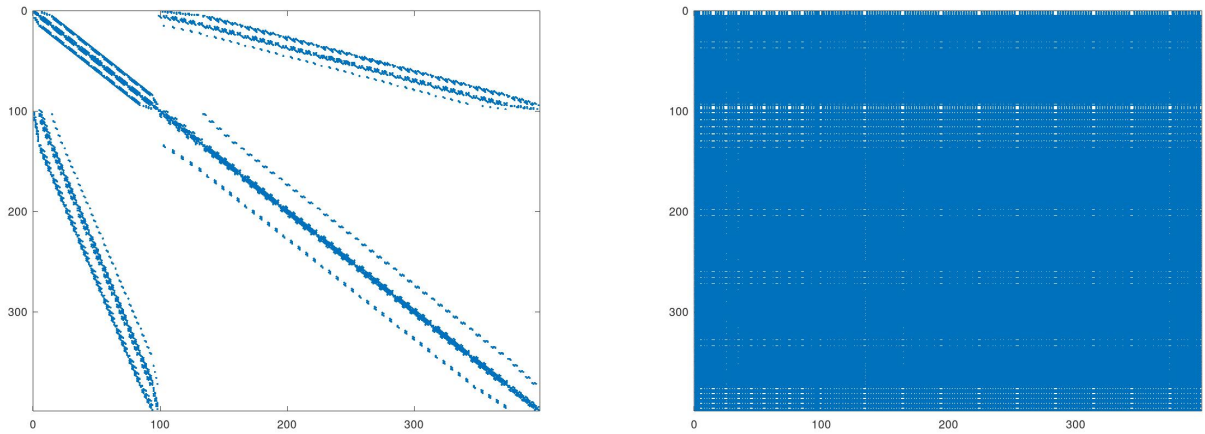
Darmstadt

Inhaltsverzeichnis

0.1 Numerische Lösung linearer Gleichungssysteme

Bei numerischen Berechnungen spielen lineare Gleichungssysteme der Form $\mathbf{Ax}=\mathbf{y}$ oft eine überaus wichtige Rolle. Daher ist es sinnvoll und von Nöten effiziente Lösungsverfahren für diese Gleichungssysteme zu entwickeln und anzuwenden. In diesem Abschnitt werden verschiedene Verfahren in Octave implementiert und an beispielhaften Gleichungssystemen getestet und gegeneinander verglichen.

Zunächst wird eine vorgegebene quadratische Matrix betrachtet. Mit Hilfe des Befehls `spy` lassen sich Matrizen in Octave graphisch darstellen. In Abb. 0.1a ist diese Graphik zu sehen, die blauen Punkte in dem Plot sind Einträge, die ungleich null sind. Zählt man diese mit dem Befehl `nnz` findet man heraus, dass 3678 der 158404 Einträge ungleich null sind. Weiterhin lässt sich aus der Abbildung und der Darstellung der Werte in Octave herausfinden, dass es sich um eine symmetrische Matrix handelt, das heißt die Einträge der Matrix sind spiegelsymmetrisch bezüglich der Hauptdiagonalen.



(a) Graphische Darstellung der mit Octave eingelesene Beispielmatrix

(b) Berechnete Inverse der eingelesenen Matrix

Abbildung 0.1: Graphische Darstellung von Matrizen, die blauen Punkte stellen Einträge dar, die ungleich null sind

Nun soll ein erstes Lösungsverfahren in Octave implementiert werden. Das GAUSS'SCHE-Eliminationsverfahren wurde in Listing 0.1 implementiert. Die Routine `gaussElim` erhält einen stehenden Vektor $\mathbf{b}$ mit Länge n und eine quadratische Matrix $\mathbf{A}$ der Größe $n \times n$. Die Rückgabe ist ein Vektor $\mathbf{x}$ mit Länge n, der die berechneten Lösungen $x_1, \ldots, x_n$ unseres linearen Gleichungssystems enthält.

Zunächst wird `n`, also die Größe der Matrix bestimmt, daraufhin wird der Ergebnisvektor $\mathbf{x}$ mit entsprechender Länge erzeugt. In Zeile 4 und 5 werden zwei Laufvariablen initialisiert, `j` läuft hierbei über die Spalten, `i` über die Zeilen. Das Ziel des Algorithmus ist es schrittweise aus der übergebenen Matrix eine untere Dreiecksmatrix zu erstellen. Hierzu werden sogenannte Eliminationsfaktoren bestimmt, siehe Listing 0.1 Zeile 6. Mit Hilfe einer weiteren Laufvariable `k`, die über die Spalten von `j` bis `n` läuft. In der Schleife, Zeile 7 bis 9, werden Zeilenadditionen durchgeführt, um so Spalte für Spalte Nulleinträge zu erzeugen. Auch der Vektor $\mathbf{b}$ wird

entsprechend angepasst.
In den Zeilen 14 bis 17 wird das Gleichungssystem final gelöst, die einzelnen $x_1, \ldots, x_n$ werden von unten nach oben mit Hilfe der Matrix und dem Vektor **b** berechnet.
Durch die drei ineinander geschachtelten For-Schleifen in den Zeilen 4 bis 12 erhalten wir eine kubische Anzahl an Rechenschritten, $n^3 - n^2$, Zeile 14 bis 17 führen zu einem zusätzlichen Rechenaufwand von $n^2 - n$, daraus ergibt sich die Abschätzung $\mathcal{O}(n^3)$.

```
1  function [x] = gaussElim (A,b)
2    n = size(A)
3    x = zeros(n,1)
4    for j=1:n-1
5      for i = j+1:n
6        l = A(i,j) / A(j,j);
7        for k = j:n
8          A(i,k) = A(i,k)  - l * A(j,k);
9        endfor
10     b(i) = b(i) - l*b(j);
11     endfor
12   endfor
13
14   for i = n:-1:1
15     f = @(k) A(i,k) * x(k);
16     x(i)= (1/A(i,i))*(b(i)-sum(f([i+1:n])));
17   endfor
18
19 endfunction
```

Lineare Gleichungssysteme lassen sich auch mit Hilfe der Inversen einer Matrix bestimmen. In Octave geht dies mit dem Befehl `inv(A)`. Wie aus Abbildung 0.1b, verglichen mit Abbildung 0.1a, hervorgeht, hat die Invertierte Matrix deutlich mehr Einträge, die ungleich null sind. Die Berechnung der Inversen benötigt sehr viel Arbeitsspeicher und es werden deutlich mehr Rechenoperationen ausgeführt, um die Lösungen zu berechnen.

Eine weitere Methode mit der lineare Gleichungssysteme effizient berechnet werden können ist die LUPQ-Zerlegung. Bei der LUPQ-Zerlegung wird die Matrix **A** mit Hilfe des bereits vorgestellten Gauss'schen-Eliminationsverfahren in eine obere und eine untere Dreiecksmatrizen aufgeteilt, es gilt $\mathbf{A} = \mathbf{LU}$. Mit Hilfe von Vorwärts-/Rückwärtseinsetzen lässt sich nun das Gleichungssystem lösen. Zum Lösen wird zunächst vorwärts

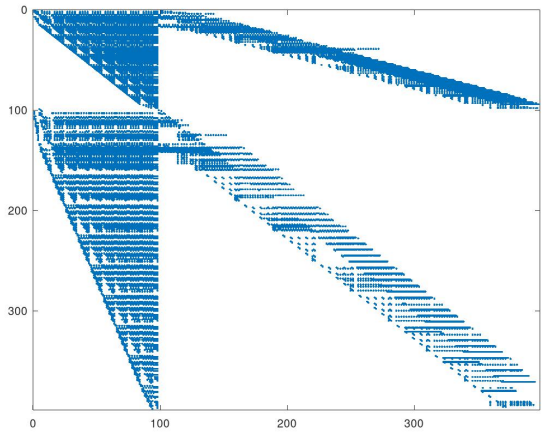
$$\mathbf{Ly} = \mathbf{b}$$

eingesetzt und anschließend rückwärts

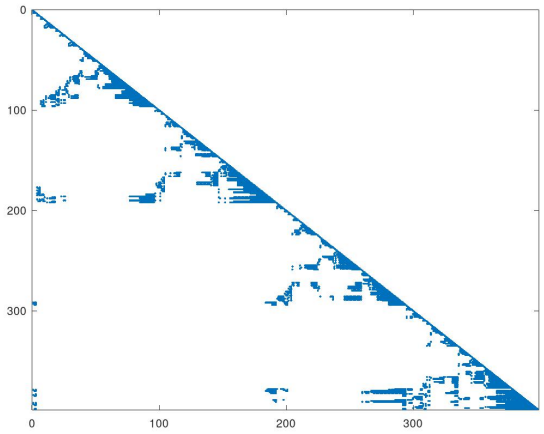
$$\mathbf{Ux} = \mathbf{y}$$

berechnet. Zum berechnen unterschiedlicher Seiten **b** muss die Matrix nicht neu zerlegt, sondern nur das Vorwärts-/Rückwärtseinsetzen betrachtet werden. Auch dieses Verfahren ist in Octave bereits vorimplementiert und lässt sich mit dem Befehl `lu(A)` aufrufen. Es ergeben sich bei dieser Methode Unterschiede, die durch die Anzahl der zur Verfügung gestellten Variable in denen die Lösung gespeichert werden soll, hervorgerufen

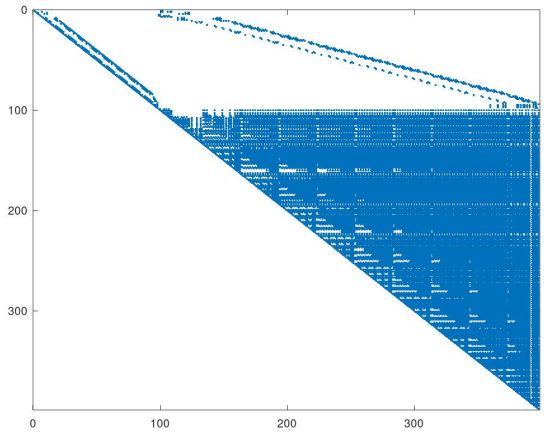
werden.
Gibt man der Methode zwei Variablen zum Speichern der Ergebnisse vor, so entsteht nicht wie erwartet eine obere und eine untere Dreiecksmatrix, sondern die in Abbildung 0.2a zu sehende Matrix **L** und die in Abb. 0.2c Matrix **U**. Stellt man der Routine vier Variablen `[L, U, P, Q]` zum Speichern zur Verfügung so entsteht wie in Abb. 0.2b und 0.2d zu sehen ist jeweils eine obere und untere Dreiecks Matrix. Zudem haben diese Matrizen deutlich weniger Einträge. **P** ist eine Permutationsmatrix. Mit dieser werden Zeilen und Spalten der Ausgangsmatrix **A** vertauscht, um zu kleine Pivot Elemente, die zu großen Rundungsfehlern führen können, zu verhindern und zu garantieren, dass die Zerlegung in eine obere und untere Dreiecksmatrix möglich ist.



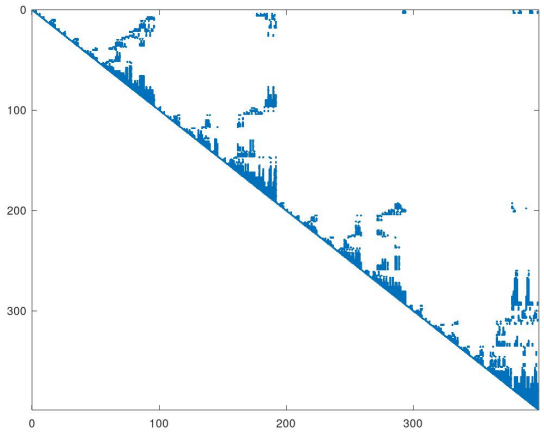
(a) Matrix **L** nach LU-Zerlegung



(b) Matrix **L** nach LUPQ-Zerlegung



(c) Matrix **U** nach LU-Zerlegung



(d) Matrix **U** nach LUPQ-Zerlegung

Abbildung 0.2: Darstellung der unterschiedlichen Matrizen, die abhängig der zur Verfügung gestellten Lösungsvariablen entstehen

Um den zeitlichen Aufwand der einzelnen Lösungsverfahren besser vergleichen zu können wird im Folgenden eine Messung der benötigten Rechenzeit in Octave durchgeführt und beschrieben. Die zu vergleichenden Lösungsverfahren zur Berechnung linearer Gleichungssysteme sind die LUPQ-Zerlegung, Lösung mit Hilfe des Backslash-Operators und die Berechnung mit der Inversen.

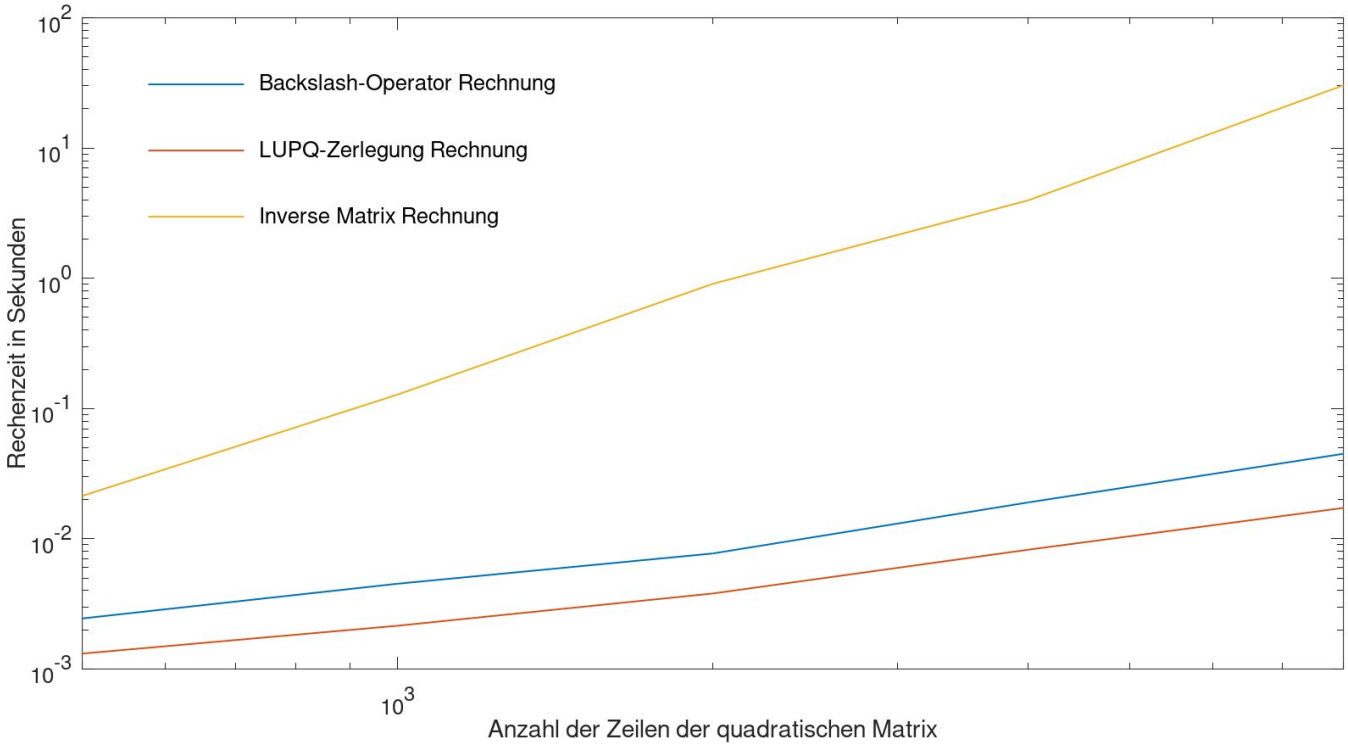


Abbildung 0.3: Graphische Darstellung der Rechenzeit, die unterschiedliche Lösungsverfahren zum Lösen linearer Gleichungen benötigen

Verschiedene tridiagonale Testmatrizen **A** der Größe $n \times n$ mit $n \in [500, 1000, 2000, 4000, 8000]$ haben jeweils 10 unterschiedliche, zufällige Seiten **b** zugewiesen bekommen. Die Zeit zur Berechnung jeder einzelnen Seite wurde gespeichert, die Berechnung der Inversen bzw. der LUPQ Zerlegung fließt nur bei der ersten Berechnung mit in die Rechenzeit ein.
Die Rechenzeit, die benötigt wurde um alle 10 Seiten zu berechnen wurde schließlich durch 10 geteilt, um eine durchschnittliche Rechenzeit zu erhalten. Die Ergebnisse wurden abschließend auf einem doppelt logarithmischen Koordinatensystem geplottet, wobei die x-Achse die Größe n der Matrix ist und die y-Achse die zur Berechnung einer Seite benötigte Rechenzeit.

Wie aus Abbildung 0.3 erkenntlich ist, benötigt die LUPQ-Zerlegung die geringste Zeit, um die Gleichungssysteme zu lösen. Der Backslash Operator benötigt mehr Zeit, ist aber, verglichen zur Berechnung mit Hilfe der Inversen, immer noch effizient.

Abbildungsverzeichnis