Topdown Vorverarbeitendes Determinanten Schema

Optimierung des Laplace Schemas zur Determinanten Berechnung

Die Determinante

Jede quadratische Matrix hat eine sog. „Determinante“. Diese Determinante ist in der Linearen Algebra von Bedeutung im Zusammenhang mit der Matrizeninversion (eine Matrix, die die Determinante 0 hat ist nicht invertierbar) und besonders zur Lösung homogener linearer Gleichungssysteme, wodurch man sie z.B. bei affinen Abbildungen zur Bestimmung von Eigenwerten einsetzt (-> eine Abbildungsmatrix A bildet Objekte auf Objekte mit det(A)-fachem Flächeninhalt ab). Außerhalb der Linearen Algebra ist sie z.B. von Bedeutung für die Integrationstheorie für Funktionen mehrerer Variablen, weil sie eng mit dem Begriff des Volumens zusammenhängt. Diese Arbeit setzt sich aber nicht mit den Anwendungsgebieten der Determinante auseinander, sondern mit ihrer Berechnung.

Das Laplace`sche Entwicklungsschema

Die Determinante einer 2x2 Matrix wird folgendermaßen berechnet:

$$\det\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} = \boxed{a_{1,1} * a_{2,2}} - \boxed{a_{2,1} * a_{1,2}}$$

Hauptdiagonale - Nebendiagonale

Die Determinante einer nxn Matrix wird rekursiv berechnet: man wählt eine Zeile z, durchläuft alle Spalten s, berechnet die Unterdeterminante $\det(A_{z,s})$ (Die Determinante der Matrix ohne die Zeile z und die Spalte s, man sagt: „man streicht die Zeile/Spalte" bzw. „man entwickelt nach der Zeile z und der Spalte s") und multipliziert sie mit dem Element $a_{z,s}$ der Matrix A. Ist z eine ungerade Zahl und s eine gerade oder z gerade und s ungerade, wird das Ergebnis mit -1 multipliziert. Für alle $s \in \{1,2,..,n\}$ summiert man diese Ergebnisse auf und erhält die Determinante. Mathematisch ausgedrückt:

$$\det(A) := \sum_{s=1}^{n} (-1)^{s+z} a_{z,s} \det\left(A_{z.s}\right)$$

Pierre Simon Marquis de Laplace (1749-1827)

Das Gauß Verfahren

Carl Friedrich Gauß (1777-1855)

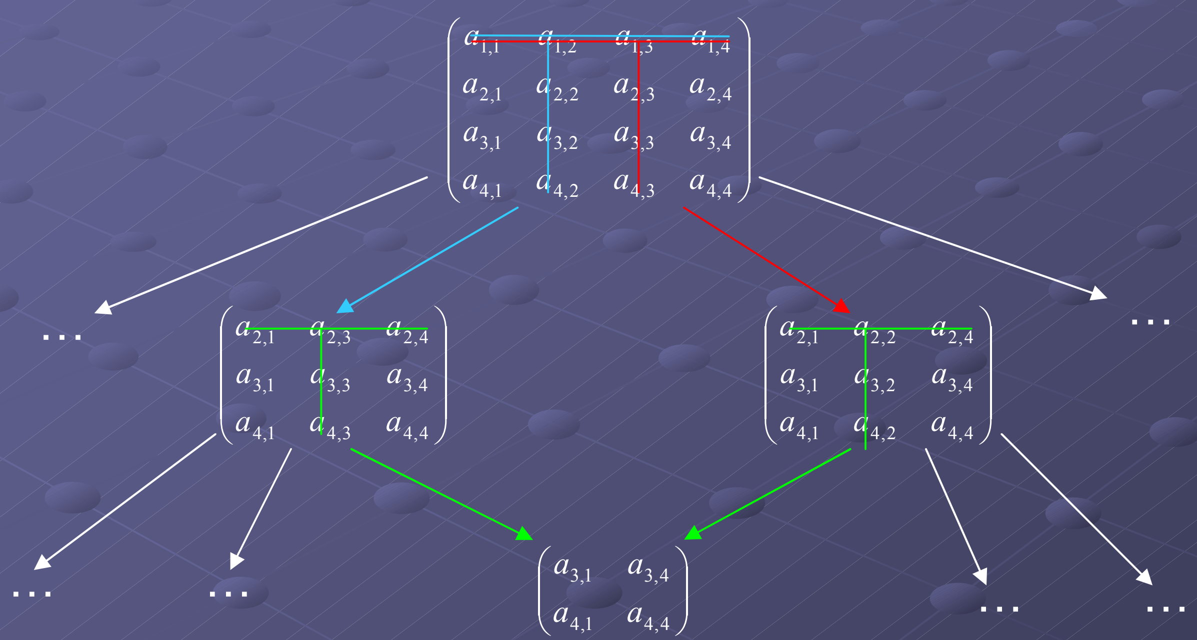
Es empfiehlt sich, z jeweils so zu wählen, dass möglichst viele $a_{z,s}=0$ sind, weil man dann die Unterdeterminanten $\det(A_{z,s})$ nicht mehr berechnen muss (weil dann $a_{z,s}*\det(A_{z,s})=0*\det(A_{z,s})=0$ ist).

Es existieren weitere Determinanten Gesetze, die es erlauben, die Matrix mit Hilfe des *Gauß'schen Eliminations-verfahrens* auf Diagonalenform zu bringen (wodurch man die Determinante dann praktisch ablesen kann), was mit einem Aufwand proportional zu n³ iterativ funktioniert. Mit diesem Algorithmus beschäftige ich mich hier aber nicht, sondern mit der *Optimierung des Laplace'schen Entwicklungsschemas.*

Das TVDS

§1

Der erste Gedanke hinter dem TVDS ist der, dass es überflüssige Arbeit ist, **jede** Unterdeterminante in einer Matrix zu berechnen, denn wenn eine Unterdeterminante mehrmals vorkommt,



ist es sinnvoller, ihren Wert **abzuspeichern und bei Bedarf wieder abzurufen**

Das TVDS

§1

Wie speichert man nun eine Unterdeterminante ab, sodass man sie bei Bedarf wieder findet? In der Matrix ist jede (komplette) Zeile und jede (komplette) Spalte entweder gestrichen oder nicht gestrichen, es gibt also für jede Zeile und jede Spalte zwei mögliche Zustände, es bietet sich an, einen Boole’schen Wert für jede Zeile und Spalte anzulegen:

$$\begin{array}{cccc|c} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & 1 \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & 0 \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & 1 \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & 1 \\ \hline 1 & 1 & 0 & 1 & \end{array} = \begin{array}{ccc} a_{1,1} & a_{1,2} & a_{1,4} \\ a_{3,1} & a_{3,2} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,4} \end{array}$$

Man kann also die Unterdeterminante einer nxn Matrix (nachdem man sie einmal berechnet hat) festhalten indem man ihren Zahlenwert abspeichert, verknüpft mit einem 2n-Tupel von Boole’schen Werten (zur Identifizierung).

Das TVDS

§2

Der zweite Gedanke hinter dem TVDS ist die Überlegung, warum man ein **2**n-Tupel benötigen sollte. Schränkt man das Laplace'sche Entwicklungsschema dahingehend ein, dass man nicht nach einer beliebigen Zeile entwickelt, sondern immer nach der obersten, ist die Unterdeterminante durch ein **n**-Tupel von Boole'schen Werten eindeutig definiert, denn die Anzahl der (von oben) gestrichenen Zeilen entspricht dann der Anzahl der gestrichenen Spalten:

$$\begin{array}{cccc} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \\ \hline 1 & 0 & 0 & 1 \end{array} = \begin{array}{cccc|c} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & 0 \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & 1 \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & 1 \\ \hline 1 & 0 & 0 & 1 & \end{array} = \begin{array}{cc} a_{3,1} & a_{3,4} \\ a_{4,1} & a_{4,4} \end{array}$$

Wir können also eine hinreichende Anzahl von Unterdeterminanten durch ein **n**-Tupel von Boole'schen Zahlen repräsentieren.

Das TVDS

§3

Der dritte Gedanke hinter dem TVDS ist die Frage, warum man immer genau nach der obersten Zeile entwickeln sollte. Es genügt, eine festgelegte Reihenfolge zu haben. Die gestrichenen Zeilen ergeben sich dann aus der Anzahl der gestrichenen Spalten und der Entwicklungs-Reihenfolge. Ist z.B. eine Spalte gestrichen und ist die erste Zahl der Reihenfolge eine 3, bedeutet das, dass die dritte Zeile gestrichen ist:

$$\begin{array}{cccc} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \\ \hline 1 & 0 & 0 & 1 \\ \hline 3 & 2 & 1 & 4 \end{array} = \begin{array}{cccc|c} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & 1 \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & 0 \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & 0 \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & 1 \\ \hline 1 & 0 & 0 & 1 & \end{array} = \begin{array}{cc} a_{1,1} & a_{1,4} \\ a_{4,1} & a_{4,4} \end{array}$$

Den Vorteil des Laplace Schemas, dass man sich immer die günstigste Zeile zum Entwickeln aussuchen kann, kann man sich also zumindest teilweise zunutze machen, indem man eine sinnvolle Reihenfolge aufstellt, nach der man entwickelt. Hier fängt man mit der Zeile, die die meisten Nullen enthält an, fährt fort mit der Zeile, die die zweit meisten Nullen enthält etc.

Das TVDS

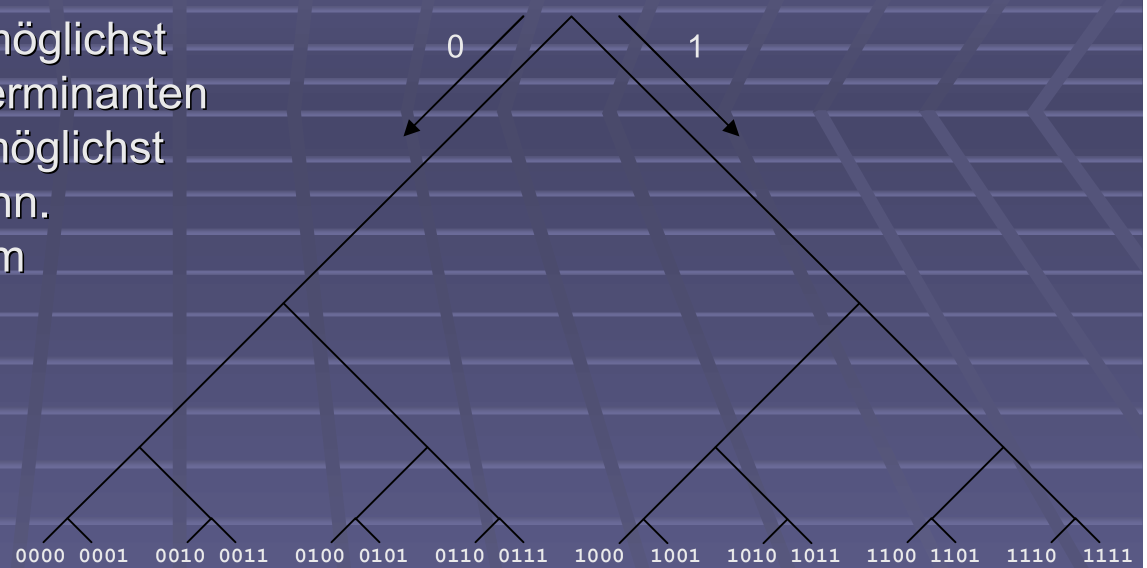
§3

Im Gegensatz zum Laplace Schema darf beim TVDS die Reihenfolge während der Entwicklung nicht verändert werden, denn dadurch wäre die Darstellung der Untermatrix durch ein Boole’sches n-Tupel nicht mehr eindeutig

$$\begin{array}{cccc} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \\ \hline 1 & 0 & 0 & 1 \\ \hline 3 & 2 & 1 & 4 \end{array} = \begin{array}{cccc|c} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & 1 \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & 0 \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & 0 \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & 1 \\ \hline 1 & 0 & 0 & 1 & \end{array} = \begin{array}{cc} a_{1,1} & a_{1,4} \\ a_{4,1} & a_{4,4} \end{array}$$

$$\neq \begin{array}{cccc} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \\ \hline 1 & 0 & 0 & 1 \\ \hline 3 & 1 & 2 & 4 \end{array} = \begin{array}{cccc|c} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & 1 \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & 0 \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & 1 \\ \hline 1 & 0 & 0 & 1 & \end{array} = \begin{array}{cc} a_{2,1} & a_{2,4} \\ a_{4,1} & a_{4,4} \end{array}$$

Der TVDS Baum



Es stellt sich die Frage, wie man möglichst effektiv die berechneten Unterdeterminanten abspeichert, sodass man sie mit möglichst geringem Zeitaufwand abrufen kann. Aufgrund der binären Struktur (dem Boole'schen n-Tupel) durch die die Untermatrizen Identifiziert werden, bietet sich ein Binärer Suchbaum an. Dieser wird nun folgendermaßen aufgebaut: von dem Aktuell gewählten Knoten auf der Ebene i geht man zum rechten Nachfolger, falls das i-te Element des Tupels eine 1 ist,ansonsten zum linken Nachfolger. In n Schritten hat man dann eine n stellige Binärzahl aufgebaut. So entspricht jedes Blatt des Baumes genau einer eindeutig definierten Binärzahl. Man kann also die berechneten Unterdeterminanten an den Blättern des Baumes abspeichern und mit einem Aufwand proportional zu n (weil der Baum genau die Tiefe n hat) abrufen.

Aufwandsabschätzung

Die Anzahl der möglichen Boole'schen n-Tupel ist gleich 2^n, es müssen also maximal 2^n verschiedene Unterdeterminanten berechnet werden, aber n-Tupel, die nur eine 1 beinhalten werden nicht berücksichtigt, weil man schon bei einer 2x2 Matrix die Determinante explizit berechnet und ihre 1x1 Unterdeterminanten nicht mehr zu „berechnen" sind. Es wird daher die Anzahl der n-Tupel mit nur einer 1 (n Stück) vom Aufwand abgezogen: $2^n - n$. Das n-Tupel, das nur aus 0en besteht wird ebenfalls nicht berechnet: $2^n - n - 1$

Aufwandsabschätzung

Ohne Speicherung der Ergebnisse sind zu berechnen:

- eine nxn Matrix $\left(1 = \frac{n!}{n!}\right)$
- für jede der n gestrichenen Spalten eine (n-1)x(n-1) Untermatrix $\left(n = \frac{n!}{(n-1)!}\right)$
- Für jede der (n-1) gestrichenen Spalten der n Untermatrizen eine (n-2)x(n-2) Unter-Untermatrix $\left(n*(n-1) = \frac{n!}{(n-2)!}\right)$
- Für jede der (n-2) gestrichenen Spalten der n*(n-1) Unter-Untermatrizen eine (n-3)x(n-3) Unter-Unter-Untermatrix usw. $\left(n*(n-1)*(n-2) = \frac{n!}{(n-3)!}\right)$

Es fällt auf, dass die Anzahl der zu berechnenden ixi Matrizen immer $\frac{n!}{i!}$ beträgt, die Anzahl aller Determinanten, die zur Berechnung der Determinante einer nxn Matrix nötig sind ergibt sich nun aus der Summe der Anzahlen von nötigen ixi Matrizen für alle $i \in \{2,3,...,n\}$ (1x1 Matrizen gehen nicht in die Rechnung ein, weil 2x2 Determinanten explizit berechnet werden und damit ihre 1x1 „Unterdeterminanten“ überflüssig sind)

Die Formel lautet also

$$\sum_{i=2}^{n} \frac{n!}{i!}$$

Beweis: siehe Abhandlung S.6f

Aufwandsabschätzung

Diese Formel hat den Nachteil, dass sie iterativ ist.
Besser wäre eine explizite Formel:

$$\sum_{i=2}^{n} \frac{n!}{i!} \Leftrightarrow n! * \sum_{i=2}^{n} \frac{1}{i!}$$

$$\lim_{n->\infty} \sum_{i=0}^{n} \frac{1}{i!} = e$$ (Taylor Polynom)

$$\Leftrightarrow \lim_{n->\infty} \sum_{i=2}^{n} \frac{1}{i!} = e - \sum_{i=0}^{1} \frac{1}{i!}$$

$$= e - 2$$

$$\Leftrightarrow n! * (e - 2)$$

(diese Formel ist auch für n = 2 hinreichend genau)

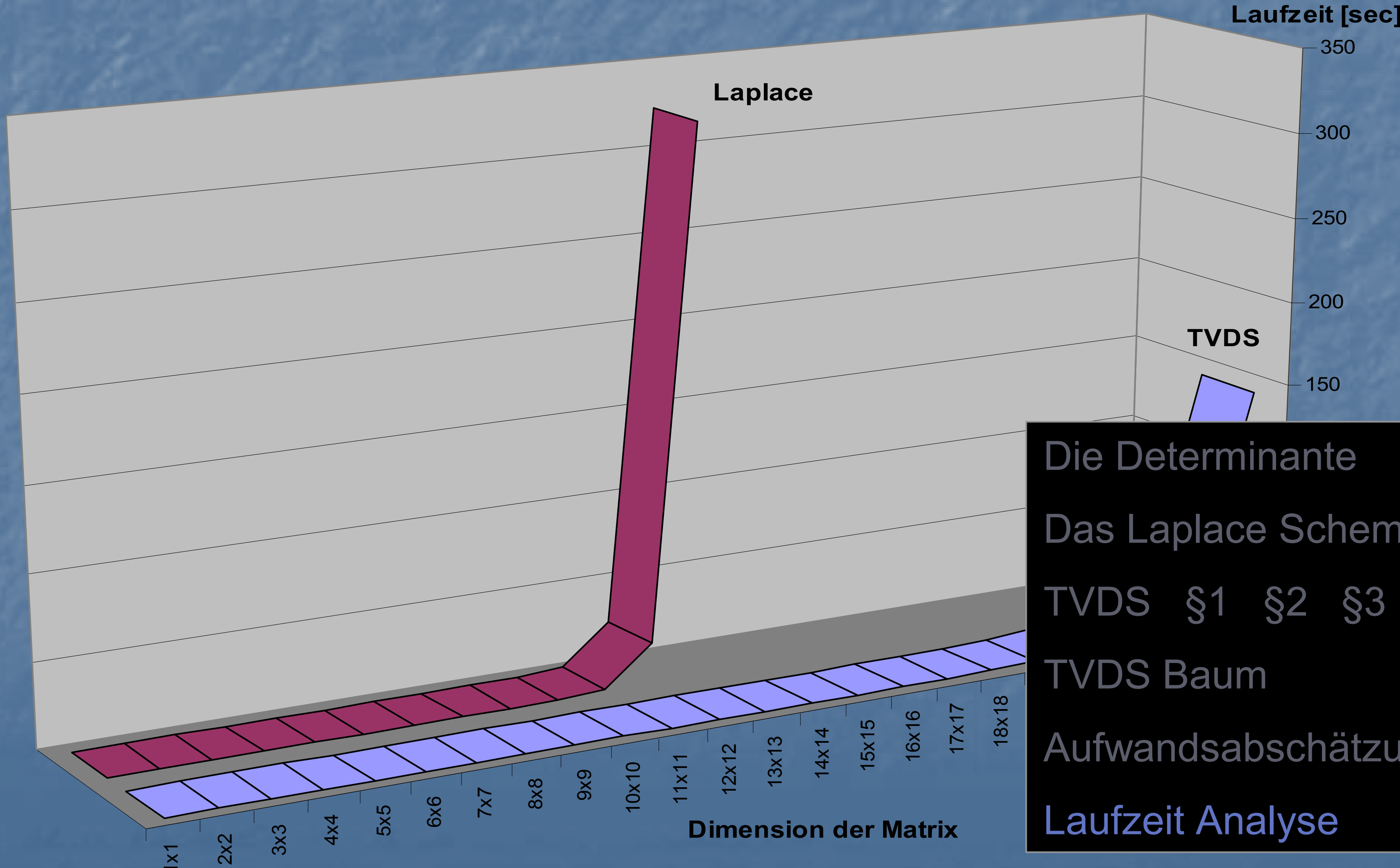
Aufwandsabschätzung

Betrachten und vergleichen wir nun die Anzahlen der Rekursionsaufrufe für verschiedene n

n	Laplace	TVDS	TVDS : Laplace	Laplace : TVDS
3	4	4	100,00 %	100,00 %
4	17	11	64,71 %	154,55 %
5	86	26	30,23 %	330,77 %
6	517	57	11,03 %	907,02 %
7	3.620	120	3,31 %	3.016,67 %
8	28.961	247	0,85 %	11.725,10 %
9	260.650	502	0,19 %	51.922,31 %
10	2.606.501	1.013	0,04 %	257.305,13 %
11	28.671.512	2.036	0,01 %	1.408.227,50 %

Laufzeit Messungen

(auf einem P4 - 2,4 GHz)



Laufzeit Analyse

Die Laufzeiten der Algorithmen sind proportional zu ihren Rekursionsaufrufen, dadurch lässt sich folgende Gleichung zur rechnerischen Bestimmung der Laufzeit des Laplace Schemas für eine 25x25 Matrix aufstellen:

$$\frac{t_{LP}(25)}{t_{LP}(13)} = \frac{R_{LP}(25)}{R_{LP}(13)}$$

Setzt man nun die Messdaten und die errechneten Aufwende ein, erhält man

$$\frac{t_{LP}(25)}{324sek} = \frac{11.141.420.304.415.739.821.629.898}{4.472.755.886}$$

$$\Leftrightarrow t_{LP}(25) \approx 324sek * 2.490.952.018.930.670$$

$$\Leftrightarrow t_{LP}(25) \approx 807.068.454.133.537.235 \text{ sek}$$

$$\Leftrightarrow t_{LP}(25) \approx 25{,}6 \; Milliarden \text{ Jahre}$$

der Messwert der Laufzeit des TVDS für eine 25x25 Matrix beträgt 13:28 min!

Topdown
Vorverarbeitendes
Determinanten
Schema

Optimierung des Laplace Schemas zur Determinanten Berechnung