Algorithmische lineare Algebra für Polynommatrizen

Bettina Krammer

FAKULTÄT FÜR MATHEMATIK DER UNIVERSITÄT REGENSBURG

REGENSBURGER MATHEMATISCHE SCHRIFTEN 32

Bestellungen sind an die

Universität Regensburg
NWF I - Mathematik
D-93040 Regensburg

zu richten.

Preis (ohne Gewähr) 8 EURO

Anschrift des Autors:

email: bkrammer@web.de

Bettina Krammer
Fakultät für Mathematik
Universität Regensburg
Universitätsstr. 31
D-93040 Regensburg

AMS Subject Classification: 15-04, 15A03, 15A06, 15A09, 15A99

Eingang des Manuskripts: Mai 2002

Bd. 32 ausgegeben: Juni 2002

ISSN 0179 - 9746 ISBN 3 - 88246 - 234 - 5

REGENSBURGER MATHEMATISCHE SCHRIFTEN 32

Algorithmische lineare Algebra für Polynommatrizen

Bettina Krammer

Fakultät für Mathematik der Universität Regensburg

Verzeichnis der Algorithmen

Wenn Sie einen Druckfehler finden, bitte bedenken Sie, daß er beabsichtigt war. Unser Blatt bringt für jeden etwas, denn es gibt immer Leute, die nach Fehlern suchen.

(Aus einer finnischen Tageszeitung)

Die vorliegende Arbeit stellt eine Kurzfassung meiner Diplomarbeit aus dem Jahre 2001 dar, welche aus dem noch jungen Gebiet der Computeralgebra stammt.

Will man die üblichen Anwendungen der linearen Algebra für Polynommatrizen effizient programmieren, so erweist sich dafür das allgemein bekannte *Gaußsche Eliminationsverfahren* als schlechte Grundlage, um solche Matrizen auf Zeilenstufen- oder Diagonalform zu bringen. Auf *Erwin Bareiss* und *Gennadi Malashonok* (u.a.) gehen wesentlich effizientere Methoden zurück, die das Wachstum der Koeffizienten begrenzt halten und die es außerdem ermöglichen, bestimmte Minoren zu berechnen. Diese Verfahren werden im folgenden gründlich ausgearbeitet und variiert; daneben finden sich im letzten Kapitel neue Algorithmen, um alle maximalen bzw. sogar alle Minoren einer bestimmten Ordnung zu berechnen.

In der vorliegenden Fassung liegt der Schwerpunkt darauf, die diesen Algorithmen zugrundeliegende Mathematik darzustellen, was in der Originalfassung noch durch zahlreiche Beispiele veranschaulicht wird. Darüberhinaus umfaßt die ursprüngliche Diplomarbeit den vollständigen source code zu allen vorgestellten Algorithmen, nebst einem ausführlichen Manual und nebst einem Vergleich aller Verfahren anhand verschiedener Beispielklassen. Wer daran interessiert ist, kann dies unter

`www.uni-regensburg.de/Fakultaeten/nat_Fak_I/Kreuzer/students.html`

finden oder kann mich unter `bkrammer@web.de` kontaktieren.

Abschließend möchte ich mich bei allen bedanken, die mich bei dieser Diplomarbeit unterstützt haben. Besonderer Dank geht an Herrn Prof. Dr. Martin Kreuzer, den Betreuer der Diplomarbeit, und an die Fakultät für Mathematik der Universität Regensburg, die diese Veröffentlichung ermöglicht hat.

Regensburg, im Mai 2002 *Bettina Krammer*

A graduate student at Trinity
Computed the square of infinity
But it gave him the fidgits
To put down the digits,
So he dropped math and took up divinity.
(Anonymous)

Wie der Titel "Algorithmische lineare Algebra für Polynommatrizen" bereits erahnen läßt, beschäftigt man sich im folgenden mit Matrizen über Integritätsringen, wobei sich das Hauptaugenmerk auf polynomiale Matrizen richtet. Es werden ohne Anspruch auf Vollständigkeit verschiedene Algorithmen vorgestellt und miteinander verglichen, um solche Matrizen auf Zeilenstufen- oder Diagonalform zu bringen. Dies kann geschehen, indem man ähnlich wie beim Gaußschen Eliminationsverfahren sukzessive Spalten ausräumt, aber auch dadurch, daß Zeilen oder sogar ganze Blockmatrizen auf einmal ausgeräumt werden. Diese Verfahren münden schließlich in die üblichen Anwendungen der linearen Algebra (Lösung linearer Gleichungssysteme etc.).

Die nötigen Grundkenntnisse aus der (linearen) Algebra werden vorausgesetzt, wobei dem Leser zumindest bekannt sein sollte, was eine Determinante ist und wie man sie berechnet (*Laplacesche Entwicklung*). Nachschlagen läßt sich dies beispielsweise unter [SchSt1], [SchSt2], [SchSt3] oder [F]. Ausblicke in die Computeralgebra und in die numerische Mathematik bieten [KR] oder [HH], jedoch sind keinerlei Computerkenntnisse erforderlich, um die folgenden Ausführungen zu verstehen.

Neben diesem kurzen Abriß des Inhalts wird später eine ausführlichere Beschreibung dessen folgen, was die vorliegende Arbeit nun eigentlich enthält. Zunächst einmal soll die naheliegende Frage beantwortet werden

0.1 Was ist ein Algorithmus?

Das Wort *Algorithmus* (wie auch die Bezeichnung *Algebra*) leitet sich vom Namen des persischen Mathematikers *Abu Jafar Mohammed ibn Musa al-Khowarizmi* ab, der im 9. Jahrhundert wirkte und eine Aufgabensammlung für Erbteilungen verfaßte. Als klassisches Beispiel eines Algorithmus läßt sich der *euklidische Algorithmus* aus dem Jahre 325 v. Chr. nennen, mit dem der größte gemeinsame Teiler zweier natürlicher Zahlen bestimmt werden kann.

Unter einem *Algorithmus* wird eine Vorschrift verstanden, die aus einer Menge eindeutiger Regeln besteht. Diese spezifizieren eine endliche Aufeinanderfolge von Operationen, so daß deren Ausführung die Lösung eines Problems aus einer speziellen Problemklasse liefert. Als Operationen sind dabei nicht nur die arithmetischen Grundoperationen "$+, -, \cdot, :$" u.ä., sondern im weiteren Sinne auch ganze Teilalgorithmen wie das Trigonalisieren oder Diagonalisieren von Matrizen zulässig.

Charakterisch für den Aufbau eines Algorithmus ist die *Eingabe* (der *Input*), die nötig ist für die anschließende Ausführung der *Rechenvorschrift* (der *Prozedur*), um schließlich die *Ausgabe* (den *Output*) zu erhalten. Dabei muß jeder Schritt der Rechenvorschrift exakt und eindeutig festgelegt sein (*Definitheit*), die Anzahl der Schritte muß außerdem endlich sein (*Finitheit*). Abgesehen davon soll der Algorithmus auf eine ganze Problemklasse anwendbar sein, deren Lösung im einzelnen nur eine Änderung der Eingabe erfordert (*Allgemeingültigkeit*).

Es gibt verschiedene Formen, einen Algorithmus zu beschreiben, beispielsweise durch ein *Flußdiagramm*. In der Originalarbeit liegen die Algorithmen zum einen als CoCoA-Funktionen vor, also als *Computerprogramme*, die die strengste Formulierung eines Algorithmus darstellen. Zum anderen wird eine freie Form der Darstellung gewählt, indem die Algorithmen als mathematische Sätze mit Beweis präsentiert werden. Den besten Beweis für die Korrektheit der Algorithmen bietet dabei natürlich die Tatsache, daß die entsprechenden Programme allesamt fehlerfrei funktionieren.

0.2 Was versteht man unter der Komplexität eines Algorithmus?

Oftmals gibt es zur Lösung derselben Aufgabe verschiedene Algorithmen. Um objektiv die Effizienz eines Algorithmus im Vergleich zu anderen Algorithmen abschätzen zu können, definiert man daher die *Komplexität* eines Algorithmus als die Abbildung $\mathbb{N} \to \mathbb{N}$, die jeder Anzahl $n \in \mathbb{N}$ von Eingabedaten die Anzahl der Grundoperationen (Additionen, Multiplikationen, Divisionen) beim Ablauf des Algorithmus zuordnet.

Bei der Ermittlung dieser Anzahl geht man stets vom schlimmstmöglichen Szenario (*worst case*) aus, wobei in der Regel die Anzahl der Additionen im Gegensatz zu den Multiplikationen oder Divisionen vernachlässigbar ist, da dafür nur verhältnismäßig wenig Rechenzeit aufgewandt werden muß. Für große n begnügt man sich statt der exakten Anzahl der Operationen i.a. mit einem etwas handlicheren Näherungswert, der oft auch mit dem *Landau-Symbol* O notiert wird.

Mit der so definierten Komplexität steht nun zwar ein Kriterium zur Verfügung, anhand dessen man unabhängig von der Implementierung oder vom Rechner Algorithmen miteinander vergleichen kann, dennoch läßt sich daraus nicht unbedingt schließen, welches Verfahren in der Praxis tatsächlich am besten taugt. Ein weiterer Aspekt ist der, daß bei der Komplexität die Codierungslänge der Eingabedaten in keiner Weise berücksichtigt wird. Im Hinblick auf die Rechenzeit macht es aber sehr wohl einen Unterschied, ob etwa zwei Polynome kleinen Grades mit kleinen Koeffizienten oder zwei Polynome sehr großen Grades miteinander multipliziert werden müssen.

0.3 Ein Blick in die Geschichte

Die Problemstellung, lineare Gleichungssysteme zu lösen oder Matrizen auf Zeilenstufen- oder Diagonalform zu bringen, ist keineswegs neu, sondern vielmehr noch immer für viele Anwendungen aktuell. So hat *Carl Friedrich Gauß* (1777–1855) schon 1810 im Zusammenhang mit Berechnungen in der Astronomie sein *Eliminationsverfahren* entwickelt, welches auch heute noch zu den Standardverfahren der linearen Algebra zählt. Im Jahre 1866 folgt

Charles Lutwidge Dodgson alias *Lewis Carroll* (1832–1898) mit einem Lösungsverfahren, das in dieser Arbeit aber nicht vorgestellt wird. Seit 1968 sind *Mehr-Schritt-Verfahren* bekannt, die auf *Erwin Bareiss* zurückgehen und die das Gaußsche Eliminationsverfahren als Spezialfall umfassen, demgegenüber jedoch eine erhebliche Verbesserung darstellen, was die Komplexität und auch das Wachstum der Koeffizienten betrifft. In den neunziger Jahren kommen schließlich diverse Vorschläge zur Diagonalisierung von *Gennadi Malashonok* hinzu. Daneben gibt es sicherlich eine Vielzahl von Autoren, die sich mit dieser Problematik beschäftigen.

Parallel dazu wird mit der Entwicklung von Rechenautomaten eine Idee verfolgt, die weiter zurückreicht, als man gemeinhin denkt. Eine *analytical engine*, die im Prinzip alle Konstruktionselemente eines modernen Computers enthält, wurde bereits ab 1833 von *Charles Babbage* (1791–1871) entworfen. Die befreundete *Ada Byron Lovelace* (1815–1852) kann wohl als erste Programmiererin der Weltgeschichte gelten; mit ihren *notes* ließen sich beispielsweise Bernoullische Zahlen berechnen. Beide waren ihrer Zeit (zu) weit voraus und konnten ihre Vorstellungen niemals vollständig verwirklicht sehen.

Erst Ende des 19. Jahrhunderts kommt der Stein ins Rollen, als *Hermann Hollerith* (1860–1929) mit den ersten Lochkarten-Maschinen den Grundstein zum späteren Weltkonzern IBM legt. Unabhängig voneinander entstehen schließlich in den 30'er und 40'er Jahren in Deutschland, England und in den USA die ersten leistungsfähigen Rechenmaschinen.

Auch wenn heute scheinbar die technischen Möglichkeiten bereit stehen, um in Sekundenschnelle oder sogar in einem Bruchteil von Sekunden eine Vielzahl komplizierter Rechenoperationen durchzuführen, so lohnt es sich dennoch, "günstige" Verfahren zur Trigonalisierung bzw. Diagonalisierung zu finden, die mit einer möglichst geringen Anzahl von Rechenoperationen auskommen und die das Wachstum der Koeffizienten begrenzt halten. Dadurch läßt sich, verglichen mit einem eher ungünstigen Verfahren wie etwa dem Gaußschen Eliminationsverfahren, die Rechenzeit drastisch verkürzen bzw. lassen sich in vielen Fällen Ergebnisse erhalten, die ein schwächeres Verfahren gar nicht erst schafft.

0.4 Was ist CoCoA?

> *Viele Menschen glauben, diese Maschine hätte ihre Ergebnisse in Zahlenform abzuliefern, und dementsprechend müsse ihre Arbeitsweise arithmetisch und numerisch sein statt algebraisch und analytisch. Das ist ein Irrtum. Die Maschine kann numerische Größen genau so anordnen und kombinieren, als wären es Buchstaben oder andere, allgemeine Symbole; in der Tat könnte sie ihre Ergebnisse in algebraischer Form ausgeben, wenn geeignete Vorkehrungen getroffen würden.*
>
> (Ada Lovelace über die *analytical engine* von Charles Babbage)

Das Akronym CoCoA ("Computations in Commutative Algebra") bezeichnet ein Computeralgebra-System, welches kostenlos unter der Adresse

`http://cocoa.dima.unige.it`

erhältlich ist. Alle in dieser Arbeit vorgestellten Algorithmen sind in der Programmiersprache CoCoAL (CoCoALanguage) geschrieben und lassen sich unter der derzeit aktuellen Version CoCoA-4.1 ausführen. Der vollständige source code nebst Manual findet sich unter

`www.uni-regensburg.de/Fakultaeten/nat_Fak_I/Kreuzer/students.html`.

Das Manual bietet eine ausführliche Dokumentation zu diesen selbstgeschriebenen CoCoA-Funktionen, wobei stets an konkreten Beispielen die Handhabung dieser Funktionen gezeigt wird.

0.5 Was enthält diese Arbeit?

In den ersten drei Kapiteln stellt diese Arbeit verschiedene Verfahren vor, um eine $n \times m$-Matrix A über einem Integritätsring R zu trigonalisieren bzw. zu diagonalisieren, wobei mittels elementarer Zeilenumformungen die Zeilenstufenspalten j_k ausgeräumt werden $(k = 1, \ldots, \operatorname{rang} A)$. Die zentrale Idee besteht darin, daß die Koeffizienten a_{ij} in den "trigonalisierten" bzw. "diagonalisierten" Zeilen ersetzt werden durch Determinanten der Form $\alpha_{ij}^{(k)} = [1 \ldots k\ i][j_1 \ldots j_k\ j](A)$ bzw. $\delta_{ij}^{(k)} = [1 \ldots k][j_1 \ldots \widehat{j_i}\ j \ldots j_k](A)$, die durch die Zeilen- und Spaltenauswahl von A definiert sind. Mit geeigneten Formeln, die im Laufe der Arbeit noch zu finden sein werden, lassen sich solche Determinanten rekursiv berechnen.

Es leuchtet auf Anhieb nicht unbedingt ein, warum diese Determinanten elementaren Zeilenumformungen gleichkommen. Ebenso erscheint es unglaublich, daß die mickrige Notation $\alpha_{ij}^{(k)}$ bzw. $\delta_{ij}^{(k)}$ schon die gesamte Information darüber enthalten soll, wie man mittels elementarer Zeilenumformungen in der Zeile i die Zeilenstufenspalten $j_1, \ldots, j_k$ bzw. $j_1 \ldots \widehat{j_i} \ldots j_k$ ausräumt, also die Information darüber, die Wievielfachen welcher Zeilen man zum Wievielfachen der alten i-ten Zeile addieren muß, um die neue i-te Zeile zu erhalten. Entwickelt man die genannten Determinanten aber nach der Spalte j, so bezeichnen die zugehörigen Kofaktoren die benötigten Vielfachen der beteiligten Zeilen $1, \ldots, k, i$ bzw. $1, \ldots, k$. Führt man dies für alle Spalten $j = 1, \ldots, m$ durch, dann ist dadurch eine elementare Zeilenumformung gegeben, mit deren Hilfe diejenigen Spalten ausgeräumt werden, die bei der Spaltenauswahl doppelt auftreten.

Als Nebeneffekt dieser Vorgehensweise sind die Verfahren prädestiniert zur Ermittlung gewisser oder sogar sämtlicher Minoren einer beliebigen Ordnung. Abgesehen davon lassen sich damit die üblichen Probleme der linearen Algebra bewältigen, wie etwa die Bestimmung des Rangs, der Determinanten und Inversen oder der Lösungen linearer Gleichungssysteme.

Anhand einiger Beispielklassen kann man schließlich die verschiedenen Verfahren miteinander vergleichen. Selbstverständlich läßt sich aber mit den gewählten Beispielen der Vergleich nicht vollständig ausschöpfen.

Weiter unten ist genauer aufgeschlüsselt, was die sechs Kapitel jeweils enthalten. Die einzelnen Abschnitte dieser Kapitel sind im allgemeinen so aufgebaut, daß ein Lemma vorbereitend auf den als mathematischen Satz formulierten Algorithmus hinführt, wobei stets alle Beweise vollständig angegeben sind. Während zu den theoretischen Überlegungen in der Regel der idealisierte Fall einer $n \times m$-Matrix betrachtet wird, für die $n \leq m$ gilt, deren Rang maximal ist und die ohne Zeilentausch auskommt, gelten die Algorithmen dann für den allgemeinstmöglichen Fall. Dabei lassen sich die vorher getroffenen Einschränkungen meist sehr einfach auflösen. Auf die Algorithmen folgen dann Bemerkungen zu ihrer Komplexität und zu ihrer Implementierung. In der Originalarbeit wird darüberhinaus zu jedem Algorithmus ein Beispiel vorgerechnet, welches für alle Algorithmen gleich ist und sich wie ein roter Faden durch die Arbeit zieht.

Was den zugrunde liegenden Integritätsring R betrifft, so kann dieser beliebig gewählt werden. Im Rahmen von CoCoA bieten sich dafür beispielsweise die Ringe $\mathbb{Z}$ und $\mathbb{Q}$ oder ein endlicher Körper $\mathbb{F}_p$ mit einer Primzahl $p \in \mathbb{N}_+$ an. In Frage kommen ferner Polynomringe in einer oder mehreren Unbestimmten über einem Integritätsring oder auch die zugehörigen

rationalen Funktionenkörper. Natürlich sind auch sonstige Integritätsringe zulässig, etwa bestimmte Restklassenringe.

Interessant sind aber letztlich nur die Fälle $R = \mathbb{Z},\ \mathbb{Z}[x_1, \ldots, x_r],\ \mathbb{F}_p[x_1, \ldots, x_r]$ mit einem $r \in \mathbb{N}_+$, denn die Trigonalisierung bzw. Diagonalisierung wird mit *bruchfreien* Verfahren vorgenommen, d.h. bei den verwendeten Rekursionsformeln kommt eine bruchfreie Division vor. Da die auftretenden Divisoren nicht notwendig Einheiten in R sind, führt man die bruchfreien Divisionen zwar genaugenommen im zugehörigen Quotientenkörper durch, jedoch liegt das Ergebnis wiederum in R.

Matrizen über den Ringen $R = \mathbb{Q},\ \mathbb{Q}[x_1, \ldots, x_r],\ \mathbb{Q}(x_1, \ldots, x_r)$ lassen sich sehr einfach auf die genannten Fälle zurückführen, indem man sie ggf. mit dem Hauptnenner multipliziert. Für den Fall $R = \mathbb{F}_p$ läßt sich genausogut das Gaußsche Verfahren (ohne Division) hernehmen, denn in diesem Fall wird das Wachstum der Koeffizienten automatisch durch die Arithmetik des endlichen Körpers beschränkt.

0.6 Was hat Kapitel I (Bareiss-Verfahren) zu bieten?

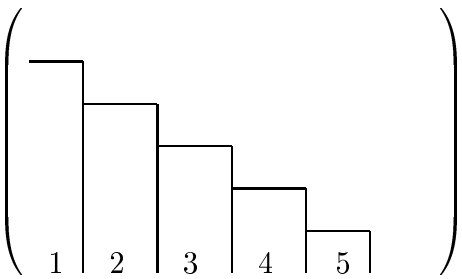
Wie das Gaußsche Eliminationsverfahren beruhen auch die *Bareiss-Mehrschritt-Verfahren* darauf, daß man sukzessive von links nach rechts eine oder sogar mehrere Spalten auf einmal unterhalb (bzw. unter- und oberhalb) der Diagonalen ausräumt, vgl. [B]. Allgemeiner als von Bareiss beschrieben, lassen sich diese Verfahren für völlig beliebige Matrizen (über einem Integritätsring) durchführen. Die Bareiss-Verfahren enthalten das Gauß-Verfahren als Spezialfall, wenn man bei ihrer Durchführung auf die Division verzichtet. Sie stellen demgegenüber jedoch aus mehreren Gründen eine entscheidende Verbesserung dar:

a) Das Wachstum der Koeffizienten wird beschränkt, indem eine bruchfreie Division durchgeführt wird.

b) Die rekursiv ermittelten Koeffizienten sind die Determinanten $\alpha_{ij}^{(k)}$ bzw. $\delta_{ij}^{(k)}$. Damit eröffnen sich weitere Anwendungsmöglichkeiten der Verfahren.

c) Die Komplexität des Bareiss-Mehr-Schritt-Verfahrens ist verglichen mit dem Gauß-Verfahren besser.

Für den Fall $n = 6$ zeigt eine grobe schematische Darstellung, in welcher Reihenfolge die Spalten ausgeräumt werden, um eine Matrix mit dem 1-Schritt-Verfahren auf Zeilenstufenform zu bringen:

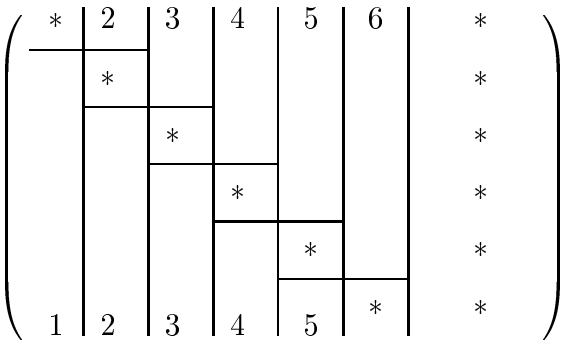


Mit einem Mehr-Schritt-Verfahren werden entsprechend mehrere Spalten auf einmal eliminiert. Diese Vorgehensweise entspricht der Konstruktion der Matrizenfolge $(B^{(k)})_{k=0\ldots n-1}$ mit

$$B^{(k)} = (b_{ij}^{(k)})_{\substack{i=1\ldots n\\ j=1\ldots m}} := \begin{pmatrix} (\alpha_{ij}^{(i-1)})_{\substack{i=1\ldots k\\ j=1\ldots m}} \\ \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\ldots n\\ j=1\ldots m}} \end{pmatrix}.$$

In einer solchen Matrix $B^{(k)}$ sind die ersten k Zeilenstufenspalten unter der Diagonalen ausgeräumt, so daß man von der Ausgangsmatrix $A = B^{(0)}$ schrittweise zur Zeilenstufenmatrix $B^{(n-1)} = (\alpha_{ij}^{(i-1)})$ gelangt.

Entsprechend erfolgt die Diagonalisierung in der unten angegebenen Reihenfolge:



Hierbei symbolisiert $*$ die verbleibenden Koeffizienten. Bei den Mehr-Schritt-Verfahren werden jeweils mehrere Spalten auf einmal ausgeräumt. Die zugehörige Matrizenfolge $(D^{(k)})_{k=0\ldots n}$ ist gegeben durch

$$D^{(k)} = (d_{ij}^{(k)})_{\substack{i=1\ldots n\\ j=1\ldots m}} = \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\ldots k\\ j=1\ldots m}} \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\ldots n\\ j=1\ldots m}} \end{pmatrix}.$$

In der obigen Matrix $D^{(k)}$ sind die ersten k Zeilenstufenspalten unter- und oberhalb der Diagonalen ausgeräumt, von der Ausgangsmatrix $A = D^{(0)}$ bis hin zur Diagonalmatrix $D^{(n)} = (\delta_{ij}^{(n)})$.

Die Formeln, mit denen sich die Matrizen $B^{(k)}$ und $D^{(k)}$ bzw. die betreffenden Determinanten $\alpha_{ij}^{(k)}$ und $\delta_{ij}^{(k)}$ rekursiv berechnen lassen, ergeben sich als unmittelbare Folgerung aus der *Sylvesteridentität*. Sie bilden das Kernstück der entsprechenden Algorithmen.

0.7 Was findet sich in Kapitel II (Malashonok-Verfahren)?

In diesem Kapitel werden weitere Methoden vorgestellt, um eine Matrix A zu diagonalisieren, nämlich das *Forward-Backup-Verfahren*, das *Malashonok-1-Schritt-Verfahren* und zuguterletzt das *Dichotomie-Verfahren*. Hartgesottene Leser, die vor einer schwer verständlichen Darstellung und unvollständigen Beweisen nicht zurückscheuen, können dies bei [Mal1] und [Mal2] nachlesen. Allgemeiner als von Malashonok dargestellt, lassen sich dabei die beiden erstgenannten Algorithmen wieder für beliebige Matrizen durchführen. Im Gegensatz zu den Bareiss-Verfahren wird die Ausräumung nun nicht nur spalten-, sondern auch zeilen- oder blockmatrizenweise vorgenommen. Zum Teil treten dabei neue Rekursionsformeln auf, die unter anderem auf der Multiplikation von Blockmatrizen basieren. Mit einer guten Matrizenmultiplikation kann somit die Komplexität der Diagonalisierung noch weiter verbessert werden.

Als erstes wird das *Forward-Backup-Verfahren* präsentiert, das sich in zwei Schritte aufspaltet. Der erste Schritt besteht darin, die gegebene Matrix A mit einem der bereits bekannten Bareiss-Verfahren zu trigonalisieren (*forward*). Besser als das von Malashonok propagierte 1-Schritt-Verfahren ist dazu das 2-Schritt-Verfahren (ein Mehr-Schritt-Verfahren) geeignet. Im zweiten Schritt werden dann von unten nach oben die Zeilen

ausgeräumt (*backup*). Ein einfaches Schema zeigt die Reihenfolge der Ausräumung:

$$\left(\begin{array}{c|c|c|c|c|c|c} * & & & & & 10 & * \\ \hline & * & & & & 9 & * \\ \hline & & * & & & 8 & * \\ \hline & & & * & & 7 & * \\ \hline & & & & * & 6 & * \\ \hline 1 & 2 & 3 & 4 & 5 & * & * \end{array}\right)$$

Hierbei symbolisiert $*$ die verbleibenden Koeffizienten. Der Forward-Prozedur entspricht die Konstruktion der bereits bekannten Matrizenfolge $(B^{(k)})_{k=0\ldots n-1}$ mit

$$B^{(k)} = (b_{ij}^{(k)})_{\substack{i=1\ldots n \\ j=1\ldots m}} := \begin{pmatrix} (\alpha_{ij}^{(i-1)})_{\substack{i=1\ldots k \\ j=1\ldots m}} \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\ldots n \\ j=1\ldots m}} \end{pmatrix}.$$

Zur Backup-Prozedur dient die Matrizenfolge $(F^{(k)})_{k=n\ldots 0}$ mit den Matrizen

$$F^{(k)} = (f_{ij}^{(k)})_{\substack{i=1\ldots n \\ j=1\ldots m}} := \begin{pmatrix} (\alpha_{ij}^{(i-1)})_{\substack{i=1\ldots k \\ j=1\ldots m}} \\ (\delta_{ij}^{(n)})_{\substack{i=k+1\ldots n \\ j=1\ldots m}} \end{pmatrix},$$

in denen für $k \leq n-1$ die oberen k Zeilen noch nicht diagonalisiert sind. Diese Folge führt von der Zeilenstufenmatrix $B^{(n-1)} = F^{(n)} = F^{(n-1)} = (\alpha_{ij}^{(i-1)})$ zur Diagonalmatrix $F^{(0)} = (\delta_{ij}^{(n)})$, indem sukzessive von unten nach oben die Zeilen ausgeräumt werden.

Mit dem *Malashonok-1-Schritt-Verfahren* werden, ausgehend vom ersten Zeilenstufenkoeffizienten a_{1j_1}, abwechselnd die darunterliegende Zeile und in den oberen Zeilen die danebenstehende Zeilenstufenspalte ausgeräumt. Damit erhält man also links oben eine diagonalisierte Blockmatrix, die sukzessive um die nächste Zeile und die nächste Zeilenstufenspalte erweitert wird, bis ganz A diagonalisiert ist. Ein grobes Schema zeigt, in welcher Reihenfolge die Zeilen bzw. Spalten ausgeräumt werden:

$$\left(\begin{array}{c|c|c|c|c|c|c} * & 2 & 4 & 6 & 8 & 10 & * \\ \hline 1 & * & & & & & * \\ \hline 3 & & * & & & & * \\ \hline 5 & & & * & & & * \\ \hline 7 & & & & * & & * \\ \hline 9 & & & & & * & * \end{array}\right)$$

Die zugehörige Matrizenfolge $(G^{(k)})_{k=0\ldots n}$ ist definiert durch

$$G^{(k)} = (g_{ij}^{(k)})_{\substack{i=1\ldots n \\ j=1\ldots m}} := \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\ldots k \\ j=1\ldots m}} \\ (a_{ij})_{\substack{i=k+1\ldots n \\ j=1\ldots m}} \end{pmatrix}.$$

Sie führt von der Ausgangsmatrix $A = G^{(0)}$ zur Diagonalmatrix $G^{(n)} = (\delta_{ij}^{(n)})$.

Zuletzt wird das komplizierteste Verfahren zur Diagonalisierung vorgestellt, nämlich das *Dichotomie-Verfahren*, welches als Spezialfälle die Forward-Backup- und Malashonok-1-Schritt-Verfahren umfaßt. Grob gesagt, beruht das Dichotomie-Verfahren zur Diagonalisierung darauf, die Matrix A in einen oberen und einen unteren Teil zu partitionieren und das Verfahren rekursiv auf die kleineren Teilmatrizen anzuwenden (*Divide et Impera-Methode*). Dabei kann die Partitionierung eine beliebige Zweiteilung sein oder speziell die Halbierung. In diesem Falle zeigt das folgende Schema die Reihenfolge der Ausräumung, wobei nun sogar ganze Blockmatrizen auf einmal eliminiert werden:

$*$	2		6				14		$*$
1	$*$								$*$
		$*$	5						$*$
3		4	$*$						$*$
				$*$	9		13		$*$
				8	$*$				$*$
						$*$	12		$*$
7				10		11	$*$		$*$

Dabei symbolisiert $*$ die verbleibenden Koffizienten. Wie bei allen vorangegangen Verfahren läßt sich auch hier die zugehörige Matrizenfolge exakt angeben, jedoch ist diese sehr kompliziert. Bei jedem Zwischenschritt bestehen die Matrizen aber wie üblich aus Determinanten der Form $\alpha_{ij}^{(k)}$ oder $\delta_{ij}^{(k)}$.

0.8 Was hält Kapitel III (Sasaki-Murao-Verfahren) bereit?

In diesem Kapitel werden die Verfahren aus den beiden vorangegangenen Kapiteln mit einer speziellen, von *Sasaki* und *Murao* 1982 beschriebenen Multiplikation $\otimes$ modifiziert, vgl. [SM]. Soll eine gegebene $n \times m$-Matrix A trigonalisiert bzw. diagonalisiert werden, so werden zunächst die Diagonalkoeffizienten durch unabhängige Variablen $X_1, \ldots, X_n$ substituiert. Dann wird bei jeder Rekursion statt der Division in R eine spezielle Multiplikation $\otimes$ im Erweiterungsring $R[X_1, \ldots, X_n]$ verwendet. Bei dem damit erzielten Endergebnis wird schließlich die Rücksubstitution durchgeführt.

Der Vorteil dieser Multiplikation $\otimes$ besteht darin, bei den rekursiv ermittelten Koeffizienten nur solche Terme zu berechnen, die auch tatsächlich zum Endergebnis beitragen. Anders ausgedrückt, statt am Produkt $f \cdot g$ zweier Polynome die Division mit Rest durchzuführen und damit den Quotienten q sowie den Rest r zu erhalten, kann man mittels $\otimes$ den Quotienten q ermitteln, ohne daß man das Produkt $f \cdot g$ und den Rest r berechnet.

Allgemeiner als von Sasaki und Murao dargestellt, läßt sich nicht nur das Bareiss-1-Schritt-Verfahren zur Trigonalisierung, sondern darüberhinaus lassen sich auch andere Algorithmen aus den Kapiteln I und II modifizieren. Allerdings sind die modifizierten Verfahren nicht für gänzlich beliebige Matrizen anwendbar, sondern nur für bestimmte Matrizen.

0.9 Was steckt in Kapitel IV (Vergleich der Verfahren)?

Theoretisch lassen sich die Verfahren aus den ersten drei Kapiteln anhand ihrer Komplexitäten vergleichen. Nicht immer ist aber ein nach theoretischen Erkenntnissen gutes Verfahren gleichermaßen zur praktischen Durchführung geeignet. In diesem Kapitel wird daher anhand verschiedener Beispielklassen (generische Matrizen, Hankelmatrizen, Zufallsmatrizen etc.) untersucht, wie gut die bekannten Algorithmen in der Praxis abschneiden, d.h. wenn man sie in CoCoA implementiert.

Zu diesem Zweck variiert man nicht nur die Art und Größe der Matrizen, sondern auch die zugrundeliegenden Integritätsringe. Dabei konzentriert man sich auf polynomiale Matrizen, wobei als Koeffizientenringe $\mathbb{Z}$ und $\mathbb{Z}/(32003)$ gewählt werden. Diese Beispiele sind in der Originalarbeit relativ ausführlich dargestellt, da man als Leser natürlich nicht jedes Beispiel am Computer nachvollzieht und da man sich außerdem meist nicht bewußt ist, daß die Umformung harmlos aussehender Matrizen in der Regel sehr unschöne Folgen hat. Betrachtet man beispielsweise generische Matrizen, so können beim Endergebnis Polynome auftreten, die aus Hunderten oder Tausenden von Monomen bestehen. In der vorliegenden Kurzfassung wird nur kurz auf diesen Vergleich der Verfahren eingegangen.

Im wesentlichen läßt sich bei den Testreihen die durch die Komplexität vorgegebene Ordnung der Algorithmen bestätigen, d.h. unter den Trigonalisierungsalgorithmen ist das Bareiss-2-Schritt-Verfahren mit zweimaliger Division empfehlenswert und unter bestimmten Voraussetzungen seine modifizierte Variante. Bei der Diagonalisierung sticht das Forward-Backup-Verfahren hervor, bei dem die Trigonalisierung mit dem eben genannten Bareiss-2-Schritt-Verfahren erledigt wird. Oft ist es dabei vorteilhaft, nicht nur die Forward-Prozedur, sondern auch die Backup-Prozedur zu modifizieren. Günstig kann auch ein Dichotomie-Verfahren sein, wofür eine Matrix jedoch bestimmte Voraussetzungen erfüllen muß.

0.10 Welche Anwendungen werden in Kapitel V aufgezeigt?

Mit Hilfe der Trigonalisierungs- und Diagonalisierungsverfahren aus den ersten drei Kapiteln lassen sich nun der Rang, die Determinante, Adjunkte und Inverse einer Matrix bestimmen. Eine weitere wichtige Anwendung aus der linearen Algebra stellt die Lösung homogener und inhomogener linearer Gleichungssysteme dar. Was die Lösung spezieller linearer Gleichungssysteme $Ax = b$ mit einer quadratischen, invertierbaren $n \times n$-Matrix A betrifft, so enthält die zur erweiterten Koeffizientenmatrix $(A \mid b)$ äquivalente Diagonalmatrix $(\delta_{ij}^{(n)})$ nichts anderes als die $(n+1)$ Determinanten, die man gemäß der *Cramerschen Regel* berechnen muß.

Diese Standardanwendungen aus der linearen Algebra sind in der ursprünglichen Diplomarbeit ausführlich dargestellt, in dieser Kurzfassung werden sie übersprungen.

0.11 Welche weiteren Anwendungen folgen in Kapitel VI?

Über die üblichen Problemstellungen der linearen Algebra hinaus läßt sich die Frage stellen, wie man möglichst günstig alle maximalen Minoren einer Matrix A oder sogar alle Minoren einer beliebigen Ordnung berechnen kann. Da deren Anzahl recht beträchtlich sein kann, sollte man unnötige Berechnungen tunlichst vermeiden. Andererseits kann es aber auch problematisch werden, wenn man allzuviele Zwischenergebnisse speichern will.

Was nun die maximalen Minoren anbelangt, so kennt man aus den ersten drei Kapiteln bereits diverse Verfahren, um Matrizenfolgen mit Einträgen der Form $\alpha_{ij}^{(k)}$ und $\delta_{ij}^{(k)}$ zu erzeugen. Diese Koeffizienten sind bis aufs Vorzeichen Minoren der Ordnung $k+1$ bzw. der Ordnung k und stellen somit nützliche Zwischenschritte auf dem Weg dahin dar,

alle maximalen Minoren zu berechnen. Jedes dieser besagten Verfahren schenkt einem dabei gewisse Rekursionsformeln, mit deren Hilfe sich solche Determinanten $\alpha_{ij}^{(k)}$ und $\delta_{ij}^{(k)}$ ermitteln lassen.

Da sich das Forward-Backup-Verfahren beim Vergleich aller Verfahren als gut erwiesen hat, soll es als Grundlage der Minorenberechnung dienen. Führt man diesen Algorithmus bzw. die Forward- und die Backup-Prozedur getrennt voneinander oft genug an geeigneten Matrizen durch, so kann man damit alle maximalen Minoren gewinnen. Anhand der dabei erzeugten Zeilenstufenmatrizen lassen sich nebenbei auch verschwindende Minoren enttarnen. Alternativ dazu reicht es aber auch, nur bestimmte maximale Minoren ermitteln, aus denen sich dann mittels linearer Relationen alle übrigen gewinnen lassen.

Um alle Minoren einer beliebigen Ordnung zu berechnen, kann man für jede mögliche Zeilenauswahl alle maximalen Minoren in diesen Zeilen bestimmen. Dabei nimmt man jedoch Doppelberechnungen von Zwischenergebnissen in Kauf, wenn eine Zeilenauswahl sich mit einer anderen überschneidet. Alternativ genügt es auch hier wieder, nur bestimmte Minoren zu berechnen und daraus dann über geeignete Relationen die restlichen.

Zu beachten ist bei der Verwendung des Forward-Backup-Verfahrens unter anderem, daß die damit hergestellten Diagonalmatrizen die maximalen Minoren nur bis aufs Vorzeichen enthalten. Daneben gilt es noch eine Vielzahl weiterer technischer Probleme zu lösen, wenn man dieses Verfahren tatsächlich zur Minorenberechnung einsetzt.

0.12 Eine Bemerkung zu den Notationen

Befaßt man sich mit Matrizen, dann liegt es in der Natur der Sache, daß die Indizes sich häufen und die Lesbarkeit einer Arbeit nicht eben erhöhen. Aus diesem Grunde ist die vorliegende Arbeit reichhaltig illustriert, indem explizit Schemata, Matrizen und in der Originalfassung Beispiele dargestellt sind. Ein weiteres Ärgernis ist die Tatsache, daß verschiedene Autoren sich selten auf eine gemeinsame Notation einigen, oder was noch schlimmer ist, dieselbe Schreibweise mit unterschiedlichen Bedeutungen belegen, so auch bei den Literaturquellen zu dieser Diplomarbeit.

Im Gegensatz zu den zugrunde liegenden Publikationen sind die Notationen in dieser Arbeit so gewählt, daß ihre Anzahl auf das notwendige Mindestmaß reduziert ist, daß sich aber trotzdem alle vorgestellten Rekursionsverfahren gleichermaßen damit beschreiben lassen. Dafür reichen nämlich die beiden Determinantentypen $\alpha_{ij}^{(k)}$ und $\delta_{ij}^{(k)}$ aus, die in ihrer Definition allgemeiner gefaßt sind als bei [B] oder [Mal1], [Mal2].

Anders als in diesen Veröffentlichungen wird in der vorliegenden Arbeit einerseits der Zusammenhang zwischen solchen Determinanten und den entsprechenden elementaren Zeilenumformungen transparent gemacht. Andererseits wird aber auch klar unterschieden zwischen diesen Determinanten und den Matrizenfolgen, in denen sie als Koeffizienten auftreten. Gleichzeitig soll mit der exakten Angabe solcher Matrizenfolgen zu jedem Zeitpunkt eines Rekursionsverfahrens offenliegen, was man da nun eigentlich berechnet hat.

Bei Bareiss bezeichnet $a_{ij}^{(k)}$ beispielsweise sowohl eine Determinante vom Typ $\alpha_{ij}^{(k)}$ als auch einen Koeffizienten aus der zu A äquivalenten Matrix $A^{(k)}$, in der die ersten k Spalten ausgeräumt sind. Mit solcherlei Doppelbezeichnungen begibt man sich generell auf dünnes Eis. Bei der Trigonalisierung mag dies noch angehen, spätestens bei der Diagonalisierung wird es brüchig. Beim Versuch, mit dieser Doppelnotation auch die Malashonok-Verfahren in den Griff zu bekommen, wird man, bildlich gesprochen, im kalten Wasser landen (siehe 0.6 und 0.7).

Lieber eine einzige Ursache verstehen
als König von Persien sein.
(Demokrit)

In diesem und allen noch folgenden Kapiteln bezeichne R stets einen Integritätsring und $A = (a_{ij})_{\substack{i=1\dots n \\ j=1\dots m}}$ eine $n \times m$-Matrix über R mit $n, m \in \mathbb{N}_+$. Für den Ring der $n \times m$-Matrizen über R verwendet man die Notation $\mathrm{Mat}_{n\times m}(R)$; gilt dabei $n = m$, so schreibt man auch kurz $\mathrm{Mat}_n(R)$.

Im ersten Kapitel lernt man verschiedene Verfahren kennen, um eine beliebige $n \times m$-Matrix A über einem Integritätsring R zu trigonalisieren bzw. zu diagonalisieren. Diese Verfahren laufen ähnlich wie das allgemein bekannte *Gaußsche Eliminationsverfahren* ab und schließen dieses als Spezialfall ein, vgl. [B].

Wie das *Gaußsche Eliminationsverfahren* beruhen auch die *Bareiss-Mehrschritt-Verfahren* darauf, daß man sukzessive von links nach rechts eine oder sogar mehrere Spalten auf einmal unterhalb (bzw. unter- und oberhalb) der Diagonalen ausräumt, und zwar wie üblich auf der Grundlage elementarer Zeilenumformungen. Diese Bareiss-Verfahren stellen gegenüber dem Gauß-Verfahren jedoch aus mehreren Gründen eine entscheidende Verbesserung dar:

a) Zum einen wird bei den Bareiss-Verfahren das Wachstum der Koeffizienten beschränkt, indem bei jeder Rekursion eine bruchfreie Division durchgeführt wird, wobei der Dividend, der Divisor und vor allem der Quotient jeweils in R liegen, ohne daß der Divisor unbedingt eine Einheit in R sein muß. Genaugenommen betrachtet man R bei der Durchführung solcher bruchfreier Divisionen also als Teilring des zugehörigen Quotientenkörpers.

b) Darüberhinaus wird sich zeigen, daß die rekursiv ermittelten Koeffizienten gewisse Determinanten sind, die sich aus bestimmten Zeilen und Spalten der Matrix A ergeben, die auf Zeilenstufen- bzw. Diagonalform gebracht werden soll. Aus diesem Grund lassen sich die Bareiss-Verfahren beispielweise auch einsetzen zur Minoren- oder Determinantenberechnung etc. Zu den Anwendungen siehe Kapitel V und VI.

c) Zum anderen lassen sich Berechnungen einsparen, wenn man mehrere Spalten auf einmal ausräumt; die Komplexität des Bareiss-Mehr-Schritt-Verfahrens stellt also gegenüber dem Gauß-Verfahren einen Fortschritt dar.

Völlig unklar erscheint dabei zunächst noch, wie man ganz allgemein bei jedem Rekursionsschritt geeignete Divisoren finden kann und wie die Rekursionsformeln zur Ermittlung der Koeffizienten beschaffen sind. Dies wird Gegenstand des ersten Abschnitts zu den Grundlagen sein. Der Zusammenhang zwischen den in b) angesprochenen Determinanten, die man mit $a_{ij}^{(k)}$ bzw. $\alpha_{ij}^{(k)}$ und mit $\delta_{ij}^{(k)}$ bezeichnet, und zwischen elementaren Zeilenumformungen wird im zweiten und fünften Abschnitt dieses Kapitels aufgezeigt werden. In den Abschnitten 3 bis 6 werden das 1-Schritt- und das 2-Schritt-Verfahren zur Trigonalisierung bzw. Diagonalisierung ausführlich erläutert werden. Allgemeiner als von Bareiss dargestellt, können diese Verfahren für beliebige Matrizen durchgeführt werden.

1 Grundlagen

In diesem Abschnitt werden Aussagen über gewisse Determinanten getroffen, die man aus ganz bestimmten Zeilen und Spalten einer Matrix $A \in \mathrm{Mat}_{n\times m}(R)$ erhält, und inbesondere darüber, wie sie sich rekursiv berechnen lassen. Als Grundlage geeigneter Rekursionsformeln dient die *Sylvesteridentität* aus Satz 1.2. Später zeigt sich, daß die Verwendung solcher Determinanten gleichbedeutend mit einer elementaren Zeilenumformung ist. Daher treten sie beim Bareiss-Verfahren in jedem Rekursionsschritt als Koeffizienten von Matrizen auf, die äquivalent zu A sind und schließlich Zeilenstufen- bzw. Diagonalform annehmen.

1.1 Definitionen

Es bezeichne $A \in \mathrm{Mat}_{n\times m}(R)$ eine beliebige $n \times m$-Matrix über einem Integritätsring R.

a) Sei $k \in \{1,\ldots,n\}$. Ferner seien $i_1,\ldots,i_k \in \{1,\ldots,n\}$ und $j_1,\ldots,j_k \in \{1,\ldots,m\}$ beliebig. Mit der Schreibweise $[i_1 \ldots i_k][j_1 \ldots j_k](A)$ soll die Determinante bezeichnet werden, die man durch Auswahl der Zeilen $i_1,\ldots,i_k$ bzw. der Spalten $j_1,\ldots,j_k$ aus A erhält, also

$$[i_1 \ldots i_k][j_1 \ldots j_k](A) := \begin{vmatrix} a_{i_1 j_1} & \ldots & a_{i_1 j_k} \\ \vdots & \ddots & \vdots \\ a_{i_k j_1} & \ldots & a_{i_k j_k} \end{vmatrix}.$$

Dabei können die Indizes $i_1,\ldots,i_k$ bzw. $j_1,\ldots,j_k$ ungeordnet sein, d.h. es wird nicht gefordert, daß $i_1 < \ldots < i_k$ bzw. $j_1 < \ldots < j_k$ gilt. Außerdem dürfen Indizes doppelt genannt werden, d.h. es kann für $\lambda,\mu \in \{1,\ldots,k\}$ mit $\lambda \neq \mu$ durchaus $i_\lambda = i_\mu$ oder $j_\lambda = j_\mu$ gelten.
Sind die ausgewählten Zeilen- und Spaltenindizes paarweise verschieden und aufsteigend geordnet, d.h. ist $i_1 < \ldots < i_k$ und $j_1 < \ldots < j_k$, so wird $[i_1 \ldots i_k][j_1 \ldots j_k](A)$ bekanntlich als *Minor* von A bezeichnet. Gilt dabei sogar $i_\lambda = j_\lambda = \lambda$ für $\lambda = 1,\ldots,k$, so liegt ein sogenannter *Hauptminor* vor.

b) Für $k \in \mathbb{N},\ k \leq n-1, m$, für $i \in \{1,\ldots,n\}$ und $j \in \{1,\ldots,m\}$ definiert man speziell

$$a_{ij}^{(k)} := \begin{cases} a_{ij} & \text{falls } k = 0 \\ [1 \ldots k\ i][1 \ldots k\ j](A) & \text{sonst.} \end{cases}$$

Somit erhält man für $k > 0$ die Determinante $a_{ij}^{(k)}$, indem man zu der $k \times k$-Teilmatrix

links oben in A noch die i-te Zeile und die j-te Spalte dazukombiniert, also

$$a_{ij}^{(k)} = \left| \begin{array}{ccc|c} a_{11} & \dots & a_{1k} & a_{1j} \\ \vdots & \ddots & \vdots & \vdots \\ a_{k1} & \dots & a_{kk} & a_{kj} \\ \hline a_{i1} & \dots & a_{ik} & a_{ij} \end{array} \right| .$$

Für $k = 0$ ergibt $a_{ij}^{(k)}$ den Koeffizienten a_{ij}. Die oben definierten Determinanten $a_{ij}^{(k)}$ werden als Koeffizienten der Matrix

$$A^{(k)} := (a_{ij}^{(k)})_{\substack{i=1\dots n \\ j=1\dots m}}$$

aufgefaßt. Außerdem setzt man

$$a_{00}^{(-1)} := 1.$$

Die wichtigste Grundlage der später beschriebenen Bareiss-Verfahren stellt die nun folgende *Sylvesteridentität* dar. Es mag zunächst verwirrend erscheinen, daß darin auf der rechten Seite eine Determinante steht, deren Einträge selbst wiederum Determinanten sind, nämlich die eben definierten Minoren $a_{ij}^{(k)}$ der Ordnung $(k+1)$. Wie man diese $a_{ij}^{(k)}$ im Hinblick auf elementare Zeilenumformungen interpretieren und als Konsequenz für ein Rekursionsverfahren zur Trigonalisierung oder Diagonalisierung von Matrizen verwenden kann, wird im zweiten Abschnitt dieses Kapitels folgen.

1.2 Satz (Sylvesteridentität)
Sei $A \in \mathrm{Mat}_n(R)$ eine quadratische Matrix der Ordnung n über einem Integritätsring R mit $n \in \mathbb{N}_+$, wobei für $k \in \{0, \dots, n-1\}$ gelte $a_{kk}^{(k-1)} \neq 0$. Dann folgt

$$|A| \cdot (a_{kk}^{(k-1)})^{n-k-1} = |(a_{ij}^{(k)})_{i,j=k+1\dots n}| =$$

$$= \left| \begin{array}{ccc} a_{k+1,k+1}^{(k)} & \dots & a_{k+1,n}^{(k)} \\ \vdots & \ddots & \vdots \\ a_{n,k+1}^{(k)} & \dots & a_{n,n}^{(k)} \end{array} \right| = [k+1\dots n][k+1\dots n](A^{(k)}).$$

Beweis:
Der Fall $k = 0$ ist klar, da man auf beiden Seiten $|A|$ erhält. (Definitionsgemäß gilt dabei $a_{00}^{(-1)} = 1$.) Sei nun $1 \leq k \leq n-1$. Zunächst partitioniert man die $n \times n$-Matrix A so in Blockmatrizen, daß gilt

$$A = \left(\begin{array}{cc} B & C \\ D & E \end{array} \right) \begin{array}{l} \}k \text{ Zeilen} \\ \}(n-k) \text{ Zeilen} \end{array}$$

$$\underbrace{}_{k \text{ Spalten}} \quad \underbrace{}_{(n-k) \text{ Spalten}}$$

wobei B eine $k\times k$-, C eine $k\times(n-k)$-, D eine $(n-k)\times k$- und E eine $(n-k)\times(n-k)$-Matrix bezeichnet. Insbesondere gilt dabei für die $k \times k$-Blockmatrix B links oben

$$|B| = a_{kk}^{(k-1)} \neq 0$$

nach Voraussetzung, d.h. die Matrix B ist invertierbar (als Matrix über dem Quotientenkörper von R). Damit ergibt sich nun

$$A = \begin{pmatrix} B & C \\ D & E \end{pmatrix} = \begin{pmatrix} B & 0 \\ D & E_{n-k} \end{pmatrix} \cdot \begin{pmatrix} E_k & B^{-1}C \\ 0 & E - DB^{-1}C \end{pmatrix}.$$

Setzt man für die $(n-k) \times (n-k)$–Matrix rechts unten

$$H := E - DB^{-1}C,$$

so folgt nach den Rechenregeln für Determinanten

$$|A| = |B| \cdot |H|.$$

Multipliziert man beide Seiten mit $|B|^{n-k-1}$, so erhält man

$$|A| \cdot |B|^{n-k-1} = |B|^{n-k} \cdot |H| = ||B| \cdot H|,$$

letzteres aufgrund der Multilinearität der Determinante. Wegen $|B| = a_{kk}^{(k-1)}$ genügt zum Beweis der Sylvesteridentität zu zeigen, daß

$$|B| \cdot H = (a_{ij}^{(k)})_{i,j=k+1\ldots n}$$

gilt. Der Beweis erfolgt durch Koeffizientenvergleich. Weil $H = E - DB^{-1}C$ ist, ergibt sich in $|B| \cdot H$ koeffizientenweise für festes $i_0, j_0 \in \{k+1, \ldots, n\}$:

$$|B| \cdot (a_{i_0 j_0} - (DB^{-1}C)_{i_0 j_0}) = |B| \cdot (a_{i_0 j_0} - (a_{i_0 j})_{j=1\ldots k} \cdot B^{-1} \cdot (a_{i j_0})_{i=1\ldots k}) =: (*).$$

Gemäß der Partitionierung zu Beginn des Beweises gilt andererseits aber auch

$$a_{i_0 j_0}^{(k)} = [1 \ldots k\ i_0][1 \ldots k\ j_0](A) =$$

$$= \begin{vmatrix} B & (a_{i j_0})_{i=1\ldots k} \\ (a_{i_0 j})_{j=1\ldots k} & a_{i_0 j_0} \end{vmatrix} =$$

$$= |B| \cdot |a_{i_0 j_0} - (a_{i_0 j})_{j=1\ldots k} \cdot B^{-1} \cdot (a_{i j_0})_{i=1\ldots k}|,$$

und letzteres $= (*)$, da hier eine Determinante erster Ordnung berechnet wird. Insgesamt folgt also die Behauptung. Die zweite und dritte Gleichheit in der Behauptung sind lediglich andere Schreibweisen. □

1.3 Bemerkung

Die Sylvesteridentität (Satz 1.2) gilt auch dann, wenn der Hauptminor $a_{kk}^{(k-1)} = 0$ verschwindet, wobei zu beachten ist, daß $0^0 := 1$ gesetzt wird. Für das weitere ist dieser Fall aber belanglos.

Aus der Sylvesteridentität (Satz 1.2) folgt unmittelbar

1.4 Korollar (Rekursionsformel)
Sei $A \in \mathrm{Mat}_{n\times m}(R)$ eine $n \times m$-Matrix über einem Integritätsring R mit $n \leq m$. Ferner seien $k \in \{1,\ldots,n-1\}$ und $l \in \{0,\ldots,k-1\}$ mit $a_{ll}^{(l-1)} \neq 0$ beliebig.

a) Für $i \in \{k+1,\ldots,n\}$ und $j \in \{1,\ldots,m\}$ gilt:

$$a_{ij}^{(k)} = \frac{1}{(a_{ll}^{(l-1)})^{k-l}} \begin{vmatrix} a_{l+1,l+1}^{(l)} & \cdots & a_{l+1,k}^{(l)} & a_{l+1,j}^{(l)} \\ \vdots & \ddots & \vdots & \vdots \\ a_{k,l+1}^{(l)} & \cdots & a_{k,k}^{(l)} & a_{k,j}^{(l)} \\ a_{i,l+1}^{(l)} & \cdots & a_{i,k}^{(l)} & a_{i,j}^{(l)} \end{vmatrix} = \qquad (*)$$

$$= \frac{1}{(a_{ll}^{(l-1)})^{k-l}} [l+1\ldots k\ i][l+1\ldots k\ j](A^{(l)}).$$

Insbesondere gilt für $l = k-1$ folgende **1-Schritt-Formel**

$$a_{ij}^{(k)} = \frac{1}{a_{k-1,k-1}^{(k-2)}} \begin{vmatrix} a_{kk}^{(k-1)} & a_{kj}^{(k-1)} \\ a_{ik}^{(k-1)} & a_{ij}^{(k-1)} \end{vmatrix} = \frac{1}{a_{k-1,k-1}^{(k-2)}} \cdot [k\ i][k\ j](A^{(k-1)})$$

und für $l = k-2$ folgende **2-Schritt-Formel**

$$a_{ij}^{(k)} = \frac{1}{(a_{k-2,k-2}^{(k-3)})^2} \begin{vmatrix} a_{k-1,k-1}^{(k-2)} & a_{k-1,k}^{(k-2)} & a_{k-1,j}^{(k-2)} \\ a_{k,k-1}^{(k-2)} & a_{k,k}^{(k-2)} & a_{k,j}^{(k-2)} \\ a_{i,k-1}^{(k-2)} & a_{i,k}^{(k-2)} & a_{i,j}^{(k-2)} \end{vmatrix} =$$

$$= \frac{1}{(a_{k-2,k-2}^{(k-3)})^2} \cdot [k-1\ k\ i][k-1\ k\ j](A^{(k-2)}).$$

Entsprechend erhält man für $l = k-s$ eine **s-Schritt-Formel**, *welche die Berechnung einer Determinante der Ordnung $(s+1)$ erforderlich macht ($s \in \{1,\ldots,k\}$).*

b) $(a_{ll}^{(l-1)})^{k-l}$ ist ein Teiler der Determinante auf der rechten Seite von $(*)$ in a).

Beweis:

a) folgt sofort, indem man die Sylvesteridentität auf die Determinante $a_{ij}^{(k)}$ und den Hauptminor $a_{ll}^{(l-1)}$ anwendet.

b) gilt, da die Determinante $a_{ij}^{(k)} = [1\ldots k\ i][1\ldots k\ j](A)$ in R liegt, wenn sämtliche Koeffizienten von A in R liegen. □

Als nützlich wird sich später noch die folgende Beobachtung erweisen. (Zur Definition eines Kofaktors vergleiche [SchSt1], 2. Auflage, S.637.)

1.5 Proposition (Teilbarkeit der Kofaktoren)
Jeder Kofaktor der rechten Determinante aus obigem Korollar 1.4 a), nämlich der durch $(a_{ll}^{(l-1)})^{k-l}$ *teilbaren Determinante*

$$[l+1\ldots k\ i][l+1\ldots k\ j](A^{(l)}) = \begin{vmatrix} a_{l+1,l+1}^{(l)} & \cdots & a_{l+1,k}^{(l)} & a_{l+1,j}^{(l)} \\ \vdots & \ddots & \vdots & \vdots \\ a_{k,l+1}^{(l)} & \cdots & a_{k,k}^{(l)} & a_{k,j}^{(l)} \\ a_{i,l+1}^{(l)} & \cdots & a_{i,k}^{(l)} & a_{i,j}^{(l)} \end{vmatrix},$$

ist durch $(a_{ll}^{(l-1)})^{k-l-1}$ *teilbar.*
Beweis:
Daß der Kofaktor von $a_{ij}^{(l)}$ durch $(a_{ll}^{(l-1)})^{k-l-1}$ teilbar ist, folgt direkt, indem man die Rekursionsformel aus Korollar 1.4 a) auf $a_{kk}^{(k-1)}$ anwendet, genauer:

$$a_{kk}^{(k-1)} = \frac{1}{(a_{ll}^{(l-1)})^{k-l-1}} \begin{vmatrix} a_{l+1,l+1}^{(l)} & \cdots & a_{l+1,k}^{(l)} \\ \vdots & \ddots & \vdots \\ a_{k,l+1}^{(l)} & \cdots & a_{k,k}^{(l)} \end{vmatrix}.$$

Gemäß Korollar 1.4 b) liegt dies in R. Die Teilbarkeit aller anderen Kofaktoren folgt analog, indem man durch Zeilen- bzw. Spaltentausch den zugehörigen Koeffizienten an die Stelle von $a_{ij}^{(l)}$ bringt. Dadurch ändert sich der absolute Wert einer Determinante bekanntlich nicht. □

Im folgenden sollen die in diesem Abschnitt gewonnenen Erkenntnisse eingesetzt werden, um Verfahren zur Trigonalisierung bzw. Diagonalisierung von Matrizen zu erhalten.

1.6 Bemerkung (Algorithmen)
Sei A eine $n \times m$-Matrix über einem Integritätsring R. Ein Algorithmus zur Trigonalisierung bzw. zur Diagonalisierung von A beruht stets darauf, in $\mathrm{Mat}_{n\times m}(R)$ eine Folge von Matrizen $B^{(i)}$, $i \in I$, mit diesen Eigenschaften zu konstruieren:

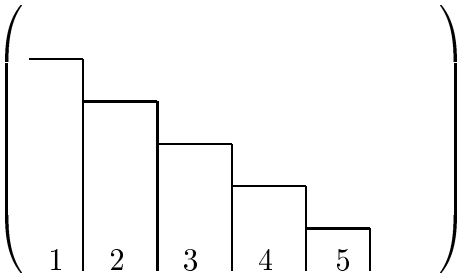
(1) Die Indexmenge $I = \{i_0, \ldots, i_p\}, p \in \mathbb{N}$, ist endlich.
(2) $B^{(i_0)} = A$.
(3) Für alle $i \in I$ ist $B^{(i)}$ eine zu A äquivalente Matrix.
(4) $B^{(i_p)}$ ist eine Matrix in Zeilenstufen- bzw. Diagonalform.
(5) Für $i = i_1, \ldots, i_p$ lassen sich die Matrizen $B^{(i)}$ rekursiv berechnen. Günstig ist es, wenn dazu nur eine Vorgängermatrix nötig ist, nicht mehrere.

Dabei sind die Matrizen $B^{(i)}$ i.a. nicht eindeutig bestimmt; sie liegen lediglich in einer Äquivalenzklasse mit A. Wie sie gewählt werden, hängt vom jeweiligen Verfahren ab und wird im folgenden ausführlich dargestellt werden.

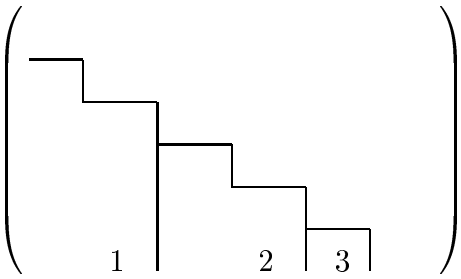
2 Trigonalisierung quadratischer Matrizen

In diesem Abschnitt sei $A = (a_{ij}) \in \mathrm{Mat}_n(R)$ stets eine quadratische Matrix über einem Integritätsring R mit der Ordnung $n \in \mathbb{N}_+$, deren Hauptminoren zudem alle ungleich 0 seien, d.h. die Voraussetzungen der Sylvesteridentität (Satz 1.2) seien erfüllt. Um diese Matrix mit dem 1-Schritt-Verfahren auf Zeilenstufenform zu bringen, räumt man beispielsweise

im Fall $n = 6$ in der Reihenfolge der Numerierung die Spalten aus:



Mit dem 2-Schritt-Verfahren ergibt sich die Reihenfolge



Das folgende Lemma stellt die entscheidende Verbindung her zwischen dem vorangegangenen Abschnitt und seinen Anwendungen, nämlich den Bareiss-Verfahren zur Trigonalisierung und Diagonalisierung. Es trägt wesentlich zum Verständnis aller weiteren Überlegungen bei, indem es aufzeigt, daß die Determinanten $a_{ij}^{(k)}$ Informationsträger von elementaren Zeilenumformungen darstellen.

2.1 Lemma (Zusammenhang von elementaren Zeilenumformungen und Determinanten)
Sei $A = (a_{ij})$ eine quadratische Matrix n-ter Ordnung ($n \in \mathbb{N}_+$) über einem Integritätsring R, deren Hauptminoren alle ungleich 0 sind.

a) Es sei $k \in \{1, \ldots, n-1\}$ beliebig, und außerdem sei $i \in \{1, \ldots, n\}$ mit $i > k$.
In A die i-te Zeile $(a_{ij})_{j=1\ldots n}$ durch die Zeile $(a_{ij}^{(k)})_{j=1\ldots n}$ zu ersetzen, ist gleichbedeutend damit, in der i-ten Zeile die Spalten 1 bis k auszuräumen mittels elementarer Zeilenumformungen der Zeilen 1 bis k und i.
Mit anderen Worten: die Determinante $a_{ij}^{(k)} = [1 \ldots k\ i][1 \ldots\ k\ j](A)$ läßt sich als Anleitung für elementare Zeilenumformungen interpretieren, um den Koeffizienten a_{ij} aus A neu zu berechnen. Bei $a_{ij}^{(k)}$ weist der hochgestellte Index (k) darauf hin, daß die ersten k Spalten ausgeräumt werden, und zwar mit Hilfe der ersten k Zeilen. Die tiefgestellten Indizes ij bezeichnen die Position des Koeffizienten, der ersetzt wird.

b) Die Matrix $(a_{ij}^{(i-1)})_{i,j=1\ldots n}$ ist eine obere Dreiecksmatrix, die äquivalent ist zu A.

Beweis:
a) Definitionsgemäß ist

$$a_{ij}^{(k)} = [1 \ldots k\ i][1 \ldots k\ j](A) = \begin{vmatrix} a_{11} & \ldots & a_{1k} & a_{1j} \\ \vdots & \ddots & \vdots & \vdots \\ a_{k1} & \ldots & a_{kk} & a_{kj} \\ a_{ij} & \ldots & a_{ik} & a_{ij} \end{vmatrix}.$$

Entwickelt man diese Determinante nach der letzten Spalte j, ergibt sich

$$a_{ij}^{(k)} = a_{ij} \cdot c_{ij} + \sum_{l=1}^{k} a_{lj} \cdot c_{lj} \qquad (*)$$

mit den zugehörigen Kofaktoren c_{ij} und $c_{lj}, l = 1, \ldots, k$. Diese Kofaktoren sind offensichtlich unabhängig vom Spaltenindex j. Die $c_{lj}, l = 1, \ldots, k$, hängen vom Zeilenindex i ab, während $c_{ij} = a_{kk}^{(k-1)}$ unabhängig von i ist. Läuft der Spaltenindex j nun von 1 bis n, so wird durch $(*)$ folgende elementare Zeilenumformung beschrieben:

$$\text{neue } i\text{-te Zeile } (a_{ij}^{(k)}) = c_{ij} \cdot \text{alte } i\text{-te Zeile } (a_{ij}) + \sum_{l=1}^{k} c_{lj} \cdot \text{alte } l\text{-te Zeile } (a_{lj}).$$

An dieser Stelle sieht man auch, warum die Voraussetzung $c_{ij} = a_{kk}^{(k-1)} \neq 0$ wichtig ist: ohne diese ginge bei der Zeilenumformung die i-te Zeile verloren.
Für einen Spaltenindex j mit $1 \leq j \leq k$ gilt außerdem

$$a_{ij}^{(k)} = [1 \ldots k \ i][1 \ldots k \ j](A) = 0,$$

denn die Spalte $j \in \{1, \ldots, k\}$ kommt doppelt vor, d.h. es werden tatsächlich die ersten k Spalten ausgeräumt. Damit folgt Behauptung a).

b) folgt sofort aus a). □

Da das vorangegangene Lemma grundlegend zum Verständnis aller weiteren Überlegungen beiträgt, wird es nun anhand eines einfachen Zahlenbeispiels ausgeleuchtet.

2.2 Beispiel zu Lemma 2.1

Betrachte die 4×4-Matrix

$$A = \begin{pmatrix} 2 & 3 & 1 & 0 \\ 4 & 2 & 2 & 2 \\ 2 & 3 & 0 & 2 \\ 2 & 0 & 2 & 2 \end{pmatrix}.$$

a) Um hierin die erste Spalte unterhalb der ersten Zeile auszuräumen, werden in den drei unteren Zeilen für $i = 2, 3, 4$ und $j = 1, \ldots, 4$ die Koeffizienten a_{ij} jeweils ersetzt durch $a_{ij}^{(1)} = [1 \ i][1 \ j](A)$. Beispielsweise ergibt sich die zweite Zeile für $j = 1, \ldots, 4$ durch

$$a_{2j}^{(1)} = [1 \ 2][1 \ j](A) = \begin{vmatrix} 2 & a_{1j} \\ 4 & a_{2j} \end{vmatrix} = 2 \cdot a_{2j} - 4 \cdot a_{1j}.$$

Speziell erhält man z.B. $a_{24}^{(1)} = \begin{vmatrix} 2 & 0 \\ 4 & 2 \end{vmatrix} = 4$. Insgesamt entsteht somit aus A die Matrix

$$B := \begin{pmatrix} 2 & 3 & 1 & 0 \\ 0 & -8 & 0 & 4 \\ 0 & 0 & -2 & 4 \\ 0 & -6 & 2 & 4 \end{pmatrix}.$$

Räumt man in der Matrix B mit Hilfe der zweiten Zeile die zweite Spalte in den beiden untersten Zeilen aus, so muß man für $i = 3, 4$ und $j = 1, \ldots, 4$ die Koeffizienten b_{ij} jeweils durch $[2 \ i][2 \ j](B)$ ersetzen. Man erhält also aus B die Matrix

$$C := \begin{pmatrix} 2 & 3 & 1 & 0 \\ 0 & -8 & 0 & 4 \\ 0 & 0 & 16 & -32 \\ 0 & 0 & -16 & -8 \end{pmatrix}.$$

Beispielsweise gilt dabei $c_{43} = [2\ 4][2\ 3](B) = \begin{vmatrix} -8 & 0 \\ -6 & 2 \end{vmatrix} = -16$. Für die dritte Zeile hätte man sich die Berechnung auch sparen können, denn dort war schon wie gewünscht $b_{32} = 0$.
Wie unschwer zu erkennen ist, entspricht diese Vorgehensweise, d.h. die Berechnung von Determinanten zweiter Ordnung, dem *Gaußschen Eliminationsverfahren.*

b) Um in der Matrix A die ersten beiden Spalten in den untersten beiden Zeilen auszuräumen, ersetzt man dort für $i = 3, 4$ und $j = 1, \ldots, 4$ die Koeffizienten a_{ij} durch $a_{ij}^{(2)} = [1\ 2\ i][1\ 2\ j](A)$. Beispielsweise ergibt sich die vierte Zeile für $j = 1, \ldots, 4$ durch

$$a_{4j}^{(2)} = [1\ 2\ 4][1\ 2\ j](A) = \begin{vmatrix} 2 & 3 & a_{1j} \\ 4 & 2 & a_{2j} \\ 2 & 0 & a_{4j} \end{vmatrix}.$$

Speziell ergibt sich z.B. $a_{43}^{(2)} = \begin{vmatrix} 2 & 3 & 1 \\ 4 & 2 & 2 \\ 2 & 0 & 2 \end{vmatrix} = -8$. Insgesamt erhält man somit

$$D := \begin{pmatrix} 2 & 3 & 1 & 0 \\ 4 & 2 & 2 & 2 \\ 0 & 0 & 8 & -16 \\ 0 & 0 & -8 & -4 \end{pmatrix}.$$

In b) räumt man also von A ausgehend zwei Spalten auf einmal aus, in a) dagegen über das Zwischenergebnis B sukzessive eine Spalte nach der anderen. Ersichtlich sind in der Matrix C die beiden untersten Zeilen jeweils das Doppelte (das a_{11}-fache) der entsprechenden Zeilen von D. Die Zeilen unterscheiden sich somit lediglich durch ein Vielfaches voneinander. Genau dieses Vielfache tritt bei der 1-Schritt-Rekursionsformel als Teiler auf, vgl. Korollar 1.4. Führt man in den beiden unteren Zeilen von C die besagte Division durch, erzielt man mit Vorgehensweise a) und b) dasselbe Ergebnis.

c) Ersetzt man in der Matrix A die zweite Zeile durch $(a_{2j}^{(1)})$, die dritte Zeile durch $(a_{3j}^{(2)})$ und die vierte Zeile durch $(a_{4j}^{(3)})$, so erhält man die Matrix

$$E := \begin{pmatrix} 2 & 3 & 1 & 0 \\ 0 & -8 & 0 & 4 \\ 0 & 0 & 8 & -16 \\ 0 & 0 & 0 & 20 \end{pmatrix},$$

also eine obere Dreiecksmatrix, wie in Lemma 2.1b) behauptet.

Zurück zum allgemeinen Fall, sei nach wie vor $A = (a_{ij}) \in \mathrm{Mat}_n(R)$ eine quadratische Matrix, deren Hauptminoren alle ungleich 0 seien, d.h. die Voraussetzungen der Sylvesteridentität (Satz 1.2) sind weiterhin erfüllt. Betrachte nun die Folge von Matrizen $A = B^{(0)}, B^{(1)}, \ldots, B^{(n-1)}$, welche allgemein für $k \in \{0, \ldots, n-1\}$ so definiert sind:

$$B^{(k)} := \begin{pmatrix} (a_{ij}^{(i-1)})_{\substack{i=1\ldots k \\ j=1\ldots n}} \\ (a_{ij}^{(k)})_{\substack{i=k+1\ldots n \\ j=1\ldots n}} \end{pmatrix} = \begin{pmatrix} (a_{1j}^{(0)})_{j=1\ldots m} \\ \vdots \\ (a_{kj}^{(k-1)})_{j=1\ldots m} \\ (a_{k+1,j}^{(k)})_{j=1\ldots m} \\ \vdots \\ (a_{nj}^{(k)})_{j=1\ldots m} \end{pmatrix}.$$

Wie üblich verwendet man dafür auch die Koeffizientenschreibweise

$$B^{(k)} = (b_{ij}^{(k)})_{i,j=1\ldots n}.$$

Dabei meint $a_{ij}^{(k)}$ die in 1.1 definierte Determinante; $b_{ij}^{(k)}$ bezeichnet einen Koeffizienten aus der Matrix $B^{(k)}$ und ist folgendermaßen definiert:

$$b_{ij}^{(k)} := \begin{cases} a_{ij}^{(i-1)} & \text{für } i = 1, \ldots, k, j = 1, \ldots, n \\ a_{ij}^{(k)} & \text{sonst.} \end{cases}$$

Außerdem setzt man

$$b_{00}^{(0)} := 1.$$

Aus dem vorangegangenen Lemma 2.1 folgt, daß $B^{(k)}$ zu A äquivalent ist und folgende Gestalt annimmt:

$$B^{(k)} = \begin{pmatrix} a_{11}^{(0)} & \cdots & \cdots & \cdots & \cdots & \cdots & a_{1n}^{(0)} \\ & a_{22}^{(1)} & \cdots & \cdots & \cdots & \cdots & a_{2n}^{(1)} \\ & & \ddots & & & & \vdots \\ & & & a_{k,k}^{(k-1)} & \cdots & \cdots & a_{k,n}^{(k-1)} \\ & \mathbf{0} & & & a_{k+1,k+1}^{(k)} & \cdots & a_{k+1,n}^{(k)} \\ & & & & \vdots & \ddots & \vdots \\ & & & & a_{n,k+1}^{(k)} & \cdots & a_{nn}^{(k)} \end{pmatrix}$$

In der Matrix $B^{(k)}$ sind also unterhalb der Diagonalen die ersten k Spalten ausgeräumt. Die Koeffizienten $a_{ii}^{(i-1)}$ auf der Diagonalen sind nach Voraussetzung ungleich 0 für alle Zeilen $i = 1, \ldots, k+1$, denn sie sind Hauptminoren.

Die Folge der Matrizen $B^{(k)}$ führt die Matrix A demnach in die zu A äquivalente obere Dreiecksmatrix $B^{(n-1)}$ über, indem sukzessive die Spalten unter der Diagonalen ausgeräumt werden. Damit ist der Zeitpunkt gekommen, an dem man die in Korollar 1.4 gefundenen Rekursionsformeln nutzen kann, um die Matrix $B^{(k)}$ aus einer Vorgängermatrix

zu berechnen. Zunächst untersucht man für $k = 1, \ldots, n-1$, wie man von der Matrix

$$B^{(k-1)} = \begin{pmatrix} (a_{ij}^{(i-1)})_{\substack{i=1\ldots k-1\\ j=1\ldots n}} \\ (a_{ij}^{(k-1)})_{\substack{i=k\ldots n\\ j=1\ldots n}} \end{pmatrix} =$$

$$= \begin{pmatrix} a_{11}^{(0)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & a_{1n}^{(0)} \\ & a_{22}^{(1)} & \cdots & \cdots & \cdots & \cdots & \cdots & a_{2n}^{(1)} \\ & & \ddots & & & & & \vdots \\ & & & a_{k-1,k-1}^{(k-2)} & \cdots & \cdots & \cdots & a_{k-1,n}^{(k-2)} \\ & & & & a_{k,k}^{(k-1)} & \cdots & \cdots & a_{k,n}^{(k-1)} \\ & \mathbf{0} & & & a_{k+1,k}^{(k-1)} & a_{k+1,k+1}^{(k-1)} & \cdots & a_{k+1,n}^{(k-1)} \\ & & & & \vdots & \vdots & \ddots & \vdots \\ & & & & a_{n,k}^{(k-1)} & a_{n,k+1}^{(k-1)} & \cdots & a_{nn}^{(k-1)} \end{pmatrix}$$

zur direkt nachfolgenden Matrix

$$B^{(k)} = \begin{pmatrix} (a_{ij}^{(i-1)})_{\substack{i=1\ldots k\\ j=1\ldots n}} \\ (a_{ij}^{(k)})_{\substack{i=k+1\ldots n\\ j=1\ldots n}} \end{pmatrix} =$$

$$= \begin{pmatrix} a_{11}^{(0)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & a_{1n}^{(0)} \\ & a_{22}^{(1)} & \cdots & \cdots & \cdots & \cdots & \cdots & a_{2n}^{(1)} \\ & & \ddots & & & & & \vdots \\ & & & a_{k-1,k-1}^{(k-2)} & \cdots & \cdots & \cdots & a_{k-1,n}^{(k-2)} \\ & & & & a_{k,k}^{(k-1)} & \cdots & \cdots & a_{k,n}^{(k-1)} \\ & \mathbf{0} & & & & a_{k+1,k+1}^{(k)} & \cdots & a_{k+1,n}^{(k)} \\ & & & & & \vdots & \ddots & \vdots \\ & & & & & a_{n,k+1}^{(k)} & \cdots & a_{nn}^{(k)} \end{pmatrix}$$

gelangen kann, in der im Vergleich zur Vorgängerin die k-te Spalte ausgeräumt ist.

2.3 Lemma (1-Schritt-Rekursion für quadratische Matrizen)
Die quadratischen Matrizen A, $B^{(k-1)}$ und $B^{(k)} \in \mathrm{Mat}_n(R)$ seien für $k = 1, \ldots, n-1$ wie oben. Außerdem sei

$$t := b_{k-1,k-1}^{(k-1)} \ (\neq 0 \text{ stets}).$$

Die Matrix $E = (e_{ij}) \in \mathrm{Mat}_n(R)$ werde folgendermaßen definiert:
(1) Die oberen Zeilen $i = 1, \ldots, k$ von E sind genau wie in $B^{(k-1)}$.
(2) Für die unteren Zeilen $i = k+1, \ldots, n$ gilt für $j = 1, \ldots, n$:

$$e_{ij} := \frac{1}{t} \cdot [k\ i][k\ j](B^{(k-1)}).$$

Dann gilt $E = B^{(k)}$.
Beweis:
Da $B^{(k)}$ und $B^{(k-1)}$ in den oberen k Zeilen nach Definition übereinstimmen, ist die Behauptung nur für die restlichen Zeilen $i = k+1, \ldots, n$ nachzuweisen. Für die besagten Zeilen gilt für $j = 1, \ldots, n$ nach Definition

$$b_{ij}^{(k)} = a_{ij}^{(k)},$$

und für letzteres folgt aus der 1-Schritt-Formel (Korollar 1.4)

$$a_{ij}^{(k)} = \frac{1}{a_{k-1,k-1}^{(k-2)}} [k\ i][k\ j](A^{(k-1)}), \qquad (*)$$

wobei nach Voraussetzung $a_{k-1,k-1}^{(k-2)} \neq 0$ gilt. Definitionsgemäß gilt

$$a_{k-1,k-1}^{(k-2)} = b_{k-1,k-1}^{(k-1)} = t$$

und außerdem

$$[k\ i][k\ j](A^{(k-1)}) = [k\ i][k\ j](B^{(k-1)}).$$

Damit folgt aus $(*)$ die Behauptung. □

Als nächstes überlegt man, wie man für gerades $k \in \{2, 4, \ldots, n-1\}$ aus der Matrix

$$B^{(k-2)} = \begin{pmatrix} (a_{ij}^{(i-1)})_{\substack{i=1\ldots k-2 \\ j=1\ldots n}} \\ (a_{ij}^{(k-2)})_{\substack{i=k-1\ldots n \\ j=1\ldots n}} \end{pmatrix} =$$

$$= \begin{pmatrix} a_{11}^{(0)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & a_{1n}^{(0)} \\ & a_{22}^{(1)} & \cdots & \cdots & \cdots & \cdots & \cdots & a_{2n}^{(1)} \\ & & \ddots & & & & & \vdots \\ & & & a_{k-1,k-1}^{(k-2)} & \cdots & \cdots & \cdots & a_{k-1,n}^{(k-2)} \\ & & & a_{k,k-1}^{(k-2)} & a_{k,k}^{(k-2)} & \cdots & \cdots & a_{k,n}^{(k-2)} \\ & \mathbf{0} & & a_{k+1,k-1}^{(k-2)} & a_{k+1,k}^{(k-2)} & a_{k+1,k+1}^{(k-2)} & \cdots & a_{k+1,n}^{(k-2)} \\ & & & \vdots & \vdots & \vdots & \ddots & \vdots \\ & & & a_{n,k-1}^{(k-2)} & a_{n,k}^{(k-2)} & a_{n,k+1}^{(k-2)} & \cdots & a_{nn}^{(k-2)} \end{pmatrix}$$

die übernächste Matrix

$$B^{(k)} = \begin{pmatrix} (a_{ij}^{(i-1)})_{\substack{i=1\ldots k\\ j=1\ldots n}} \\ (a_{ij}^{(k)})_{\substack{i=k+1\ldots n\\ j=1\ldots n}} \end{pmatrix} =$$

$$= \begin{pmatrix} a_{11}^{(0)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & a_{1n}^{(0)} \\ & a_{22}^{(1)} & \cdots & \cdots & \cdots & \cdots & \cdots & a_{2n}^{(1)} \\ & & \ddots & & & & & \vdots \\ & & & a_{k-1,k-1}^{(k-2)} & \cdots & \cdots & \cdots & a_{k-1,n}^{(k-2)} \\ & & & & a_{k,k}^{(k-1)} & \cdots & \cdots & a_{k,n}^{(k-1)} \\ & \mathbf{0} & & & & a_{k+1,k+1}^{(k)} & \cdots & a_{k+1,n}^{(k)} \\ & & & & & \vdots & \ddots & \vdots \\ & & & & & a_{n,k+1}^{(k)} & \cdots & a_{nn}^{(k)} \end{pmatrix}$$

erhält, bei der nun im Vergleich zu vorher die Spalten $k-1$ und k ausgeräumt sind.

2.4 Lemma (2-Schritt-Rekursion für quadratische Matrizen)
Die quadratische Matrix A sei wie eben, außerdem sei $n-1$ gerade. Betrachte in der obigen Situation für $k = 2, 4, \ldots, n-1$ die Matrix $B^{(k-2)}$ mit

$$t := b_{k-2,k-2}^{(k-2)} \ (\neq 0 \ \textit{stets}).$$

Die Matrix $E = (e_{ij}) \in \mathrm{Mat}_n(R)$ werde folgendermaßen definiert:

(1) Die oberen Zeilen $i = 1, \ldots, k-1$ sind genau wie in $B^{(k-2)}$.

(2) Für die unteren Zeilen $i = k+1, \ldots, n$ gilt für $j = 1, \ldots, n$:

$$e_{ij} := \frac{1}{t^2} \cdot [k-1 \ k \ i][k-1 \ k \ j](B^{(k-2)}).$$

(3) Die Zeile k ist für $j = 1, \ldots, n$ gegeben durch

$$e_{kj} := \frac{1}{t} \cdot [k-1 \ k][k-1 \ j](B^{(k-2)}).$$

Dann gilt $E = B^{(k)}$.

Beweis:
Beachtet man, daß jetzt auch die 2-Schritt-Rekursionsformel aus Korollar 1.4 verwendet wird und daß nach Definition

$$t = b_{k-2,k-2}^{(k-2)} = a_{k-2,k-2}^{(k-3)} \quad (\neq 0 \text{ nach Voraussetzung})$$

und

$$[k-1 \ k \ i][k-1 \ k \ j](B^{(k-2)}) = [k-1 \ k \ i][k-1 \ k \ j](A^{(k-2)})$$

gilt, so verläuft der Beweis analog zur 1-Schritt-Rekursion (Lemma 2.2). □

In den beiden vorangegangenen Lemmata hat man gesehen, wie man in der Folge der Matrizen $B^{(k)}, k = 0, \ldots, n-1$, durch Rechnung von einer Matrix zur nächsten oder übernächsten gelangen kann, wobei ausschließlich die Koeffizienten dieser Vorgängermatrix verwendet werden. Davon ausgehend ließe sich jetzt ein Algorithmus zur Trigonalisierung quadratischer Matrizen formulieren. Da jedoch das nächste Ziel heißt, ganz allgemein beliebige $n \times m$-Matrizen zu trigonalisieren, wird hier der Spezialfall quadratischer Matrizen nicht mehr weiter ausgeführt. Es genügt, die bereits angestellten Überlegungen zu verallgemeinern.

3 Trigonalisierung mit dem 1-Schritt-Verfahren

Wie zu Ende von Abschnitt 2 bereits angekündigt, sei ab sofort $A = (a_{ij})$ eine $n \times m$-Matrix über einem Integritätsring R mit $n, m \in \mathbb{N}_+$. Außerdem sei $n \leq m$, und der Rang von A sei maximal, also gleich n. (Dies erleichtert die folgenden theoretischen Betrachtungen und wird auf einfache Art und Weise in den folgenden Trigonalisierungsalgorithmen wieder aufgelöst werden.)

Angenommen, man kennt bereits ein Verfahren, um A zu trigonalisieren, z.B. das *Gaußsche Eliminationsverfahren*. Ist B die zu A äquivalente Matrix in Zeilenstufenform, die man durch dieses Verfahren erhält, so bezeichne für $k = 1, \ldots, n$ der Index j_k die Lage der Zeilenstufe in der k-ten Zeile von B, also den Spaltenindex des ersten Koeffizienten ungleich 0 in der Zeile k. Es gilt dann $j_1 < \ldots < j_n$, außerdem ist i.a. $j_k \neq k$ für $k = 1, \ldots, n$, denn in B können "lange" Zeilenstufen auftreten.

Diese Zeilenstufenindizes $j_k \in \{1, \ldots, m\}$, die für $k = 1, \ldots, n$ die Position der Zeilenstufen von B angeben, sind charakteristisch für A und in allen zu A äquivalenten trigonalisierten Matrizen gleich. Daher wird in Zukunft stets j_k geschrieben, um die k-te Zeilenstufe zu lokalisieren. Der Koeffizient an der Stelle (kj_k) wird Zeilenstufenkoeffizient genannt. Ferner setzt man $j_0 := 0$.

Führt man ein Trigonalisierungsverfahren durch, kann es eventuell nötig werden, Zeilen zu tauschen, um Zeilenstufen an der richtigen Position zu erhalten (*pivoting*). Das Endergebnis B hat also gegenüber A vertauschte Zeilen. Für die folgenden Überlegungen soll die Matrix A aber so beschaffen sein, daß ein Zeilentausch beim Trigonalisieren nicht nötig wird, daß sich also von vorneherein in jeder Zeile k von A die Zeilenstufe j_k befindet für $k = 1, \ldots, n$. (Warum diese Einschränkung gemacht wird, wird in Bemerkung 3.6 geklärt werden.)

Um die bisherigen Voraussetzungen noch einmal zusammenzufassen, so sei ab sofort A eine $n \times m$-Matrix über einem Integritätsring R mit den Eigenschaften

(V1) $n \leq m$,
(V2) $\operatorname{rang} A = n$ und
(V3) Zeilentausch ist nicht nötig.

Im Unterschied zu den quadratischen Matrizen gilt es jetzt stets zu berücksichtigen, daß sich für $k = 1, \ldots, n$ die Zeilenstufen nicht mehr unbedingt auf der Diagonalen an der Stelle (kk), sondern ganz allgemein an Position (kj_k) befinden. In Verallgemeinerung der Determinanten $a_{ij}^{(k)}$ verwendet man daher nun die Determinanten $\alpha_{ij}^{(k)}$, die wie folgt definiert sind.

3.1 Definition
Sei die Matrix A wie oben. Für $k \in \{0, \ldots, n-1\}$, $i \in \{1, \ldots, n\}$ und $j \in \{1, \ldots, m\}$

schreibt man

$$\alpha_{ij}^{(k)} := \begin{cases} a_{ij} & \text{falls } k = 0 \\ [1 \dots k\ i][j_1 \dots j_k\ j](A) & \text{sonst.} \end{cases}$$

Die so definierten Determinanten $\alpha_{ij}^{(k)}$ werden als Koeffizienten der Matrix

$$\mathcal{A}^{(k)} := (\alpha_{ij}^{(k)})_{\substack{i=1\dots n \\ j=1\dots m}}$$

aufgefaßt. Außerdem setzt man

$$\alpha_{00}^{(-1)} := 1.$$

Um die folgenden Überlegungen verständlicher zu machen, betrachtet man zunächst eine Verallgemeinerung von Lemma 2.1.

3.2 Bemerkungen (Merkregel für Determinanten und elementare Zeilenumformungen)
Sei die Matrix A eine $n \times m$-Matrix mit den Eigenschaften (V1) bis (V3).

a) Für $k \in \{1, \dots, n\}$ gilt

$$\alpha_{kj_k}^{(k-1)} = [1 \dots k][j_1 \dots j_k](A) \neq 0,$$

denn da die ausgewählten Spalten $j_1, \dots, j_k$ Zeilenstufenspalten sind, hat die Teilmatrix von A, die aus den Zeilen $1, \dots, k$ und den Spalten $j_1, \dots, j_k$ besteht, den maximalen Rang k.

b) Seien $k \in \{1, \dots, n-1\}$, $i \in \{k+1, \dots, n\}$ und $j \in \{1, \dots, m\}$ beliebig.
Die Determinante $\alpha_{ij}^{(k)} = [1 \dots k\ i][j_1 \dots j_k\ j](A)$ läßt sich als Anleitung verstehen, um in der Zeile i von A die Zeilenstufenspalten $j_1, \dots, j_k$ auszuräumen mit Hilfe der Zeilen $1, \dots, k$ und i.
Für alle Spalten j mit $1 \leq j < j_{k+1}$ nimmt $\alpha_{ij}^{(k)}$ den Wert 0 an, denn es werden nicht nur die Koeffizienten a_{ij} mit $j \in \{j_1, j_2, \dots, j_k\}$ ausgeräumt, sondern auch alle anderen Nichtzeilenstufenkoeffizienten a_{ij} mit $j \in \{1, 2, \dots, j_{k+1}-1\} \setminus \{j_1, j_2, \dots, j_k\}$.

3.3 Bemerkungen (verallgemeinerte Rekursionsformeln)
Sei A eine $n \times m$-Matrix mit den Eigenschaften (V1) bis (V3). Dann verallgemeinern sich die Rekursionsformeln aus Korollar 1.4, indem man dort statt $a_{\cdot\cdot}^{(\cdot)}$ die entsprechende Determinante $\alpha_{\cdot\cdot}^{(\cdot)}$ einsetzt. Analog verallgemeinert sich auch Proposition 1.5. Es folgt also

a) (*Rekursionsformeln*) Für $k \in \{1, \dots, n-1\}$, für $i \in \{k+1, \dots, n\}$, $j \in \{1, \dots, m\}$ und $l \in \{0, \dots, k-1\}$ gilt:

$$\alpha_{ij}^{(k)} = \frac{1}{(\alpha_{lj_l}^{(l-1)})^{k-l}} \begin{vmatrix} \alpha_{l+1,j_{l+1}}^{(l)} & \dots & \alpha_{l+1,j_k}^{(l)} & \alpha_{l+1,j}^{(l)} \\ \vdots & \ddots & \vdots & \vdots \\ \alpha_{k,j_{l+1}}^{(l)} & \dots & \alpha_{k,j_k}^{(l)} & \alpha_{k,j}^{(l)} \\ \alpha_{i,j_{l+1}}^{(l)} & \dots & \alpha_{i,j_k}^{(l)} & \alpha_{i,j}^{(l)} \end{vmatrix} =$$

$$= \frac{1}{(\alpha_{lj_l}^{(l-1)})^{k-l}} [l+1 \dots k\ i][j_{l+1} \dots j_k\ j](\mathcal{A}^{(l)}).$$

(Nach Bemerkung 3.2a) gilt stets $\alpha_{lj_l}^{(l-1)} \neq 0$.)

Insbesondere gilt für $l = k-1$ folgende **1-Schritt-Formel**

$$\alpha_{ij}^{(k)} = \frac{1}{\alpha_{k-1,j_{k-1}}^{(k-2)}} \begin{vmatrix} \alpha_{kj_k}^{(k-1)} & \alpha_{kj}^{(k-1)} \\ \alpha_{ij_k}^{(k-1)} & \alpha_{ij}^{(k-1)} \end{vmatrix} = \frac{1}{\alpha_{k-1,j_{k-1}}^{(k-2)}} [k\ i][j_k\ j](\mathcal{A}^{(k-1)})$$

und für $l = k-2$ folgende **2-Schritt-Formel**

$$\alpha_{ij}^{(k)} = \frac{1}{(\alpha_{k-2,j_{k-2}}^{(k-3)})^2} \begin{vmatrix} \alpha_{k-1,j_{k-1}}^{(k-2)} & \alpha_{k-1,j_k}^{(k-2)} & \alpha_{k-1,j}^{(k-2)} \\ \alpha_{k,j_{k-1}}^{(k-2)} & \alpha_{k,j_k}^{(k-2)} & \alpha_{k,j}^{(k-2)} \\ \alpha_{i,j_{k-1}}^{(k-2)} & \alpha_{i,j_k}^{(k-2)} & \alpha_{i,j}^{(k-2)} \end{vmatrix} =$$

$$= \frac{1}{(\alpha_{k-2,j_{k-2}}^{(k-3)})^2} [k-1\ k\ i][j_{k-1}\ j_k\ j](\mathcal{A}^{(k-2)}).$$

Entsprechend erhält man für $l = k-s$ eine **s-Schritt-Formel**, welche die Berechnung einer Determinante der Ordnung $(s+1)$ erforderlich macht ($s \in \{1, \ldots, k\}$).

b) $(\alpha_{lj_l}^{(l-1)})^{k-l}$ ist ein Teiler der Determinante auf der rechten Seite ganz oben in a).

c) *(Teilbarkeit der Kofaktoren)* Jeder Kofaktor der rechten, durch $(\alpha_{lj_l}^{(l-1)})^{k-l}$ teilbaren Determinante ganz oben in a) ist durch $(\alpha_{lj_l}^{(l-1)})^{k-l-1}$ teilbar.

Des weiteren verallgemeinert sich der bisher betrachtete Spezialfall quadratischer Matrizen in dem Sinne, daß man nun die Matrizenfolge $B^{(k)}$, $k = 0, \ldots, n-1$, betrachtet, die gegeben ist durch

$$B^{(k)} = (b_{ij}^{(k)})_{\substack{i=1\ldots n \\ j=1\ldots m}} := \begin{pmatrix} (\alpha_{ij}^{(i-1)})_{\substack{i=1\ldots k \\ j=1\ldots m}} \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\ldots n \\ j=1\ldots m}} \end{pmatrix}.$$

Außerdem setzt man

$$b_{0j_0}^{(0)} := 1.$$

Nach Bemerkung 3.2 sind diese Matrizen von der Form

$$B^{(k)} = \begin{pmatrix} (\alpha_{ij}^{(i-1)})_{\substack{i=1\ldots k\\ j=1\ldots m}} \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\ldots n\\ j=1\ldots m}} \end{pmatrix} =$$

$$= \begin{pmatrix} \alpha_{1j_1}^{(0)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{1m}^{(0)} \\ & & \alpha_{2j_2}^{(1)} & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{2m}^{(1)} \\ & & & \ddots & & & & & \vdots \\ & & & & \alpha_{k,j_k}^{(k-1)} & \cdots & \cdots & \cdots & \alpha_{k,m}^{(k-1)} \\ & & \mathbf{0} & & & \alpha_{k+1,j_{k+1}}^{(k)} & \cdots & \alpha_{k+1,m}^{(k)} \\ & & & & & \vdots & \ddots & \vdots \\ & & & & & \alpha_{n,j_{k+1}}^{(k)} & \cdots & \alpha_{nm}^{(k)} \end{pmatrix}.$$

In der Matrix $B^{(k)}$ sind also unterhalb der Diagonalen die ersten k Zeilenstufenspalten $j_1, j_2, \ldots, j_k$, somit die vorderen Spalten $1, \ldots, j_{k+1} - 1$, ausgeräumt.

Die Folge dieser Matrizen führt die Matrix $A = B^{(0)}$ in die Zeilenstufenmatrix $B^{(n-1)}$ über, indem sukzessive von links nach rechts eine Zeilenstufenspalte nach der anderen unterhalb der Diagonalen ausgeräumt wird. Wie bei den quadratischen Matrizen sollen mit Hilfe der Bareiss-Formeln die Matrizen der Folge $(B^{(k)})_{k=1,\ldots,n-1}$ rekursiv aus ihren Vorgängerinnen berechnet werden, indem ausschließlich die Koeffizienten der Vorgängermatrix verwendet werden, vgl. die Lemmata 2.3 und 2.4 (1-Schritt- bzw. 2-Schritt-Rekursion für quadratische Matrizen). Entsprechend gilt auch hier jetzt allgemein

3.4 Bemerkung (Merkregel für Rekursionsformeln)
Sei $k \in \{1, \ldots, n-1\}$. In obiger Situation gilt für die s-Schritt-Rekursion mit $s \in \{1, \ldots, k\}$: Beim Übergang von $B^{(k-s)}$ zu $B^{(k)}$ erhält man stets für $i = k+1, \ldots, n$ und $j = 1, \ldots, m$

$$b_{ij}^{(k)} = \frac{1}{(b_{k-s,j_{k-s}}^{(k-s)})^s} \cdot [k-s+1 \ldots k\ i][j_{k-s+1} \ldots j_k\ j](B^{(k-s)}).$$

Dabei gilt stets $b_{k-s,j_{k-s}} \neq 0$. In Worte gefaßt:

(1) Im Klammerausdruck mit den Spaltenindizes steht an letzter Stelle stets die Spalte, die neu berechnet werden soll, und davor die Spalten (Zeilenstufenindizes), die ausgeräumt werden sollen, z.B. in der später folgenden 2-Schritt-Rekursion (Lemma 4.1)

$$[j_{k-1}\ j_k\ j].$$

(2) Im Klammerausdruck mit den Zeilenindizes steht an letzter Stelle stets die Zeile, die man neu berechnen will, und davor die Zeilen, mit deren Hilfe man die gewünschten Spalten ausräumt, z.B. in Lemma 4.1

$$[k-1\ k\ i].$$

(3) Der Teiler ist der Zeilenstufenkoeffizient der Vorgängerzeile $(k-s)$ bzw. ist gleich 1, falls $k-s=0$ gilt. (Als Zeilenstufenkoeffizient ist der Teiler insbesondere ungleich 0.) Sein Exponent ist gleich der Schrittgröße s. In Lemma 4.1 erhält man also als Teiler

$$(b_{k-2,j_{k-2}}^{(k-2)})^2.$$

Was die obige Merkregel schon vorweggenommen hat, wird im folgenden detaillierter ausgeführt. Dazu betrachtet man die Folge der Matrizen $B^{(k)}$ und untersucht zunächst für $k=1,\ldots,n-1$ den Übergang von der Matrix

$$B^{(k-1)} = \begin{pmatrix} (\alpha_{ij}^{(i-1)})_{\substack{i=1\ldots k-1\\ j=1\ldots m}} \\ (\alpha_{ij}^{(k-1)})_{\substack{i=k\ldots n\\ j=1\ldots m}} \end{pmatrix} =$$

$$= \begin{pmatrix} \alpha_{1j_1}^{(0)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{1m}^{(0)} \\ & & \alpha_{2j_2}^{(1)} & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{2m}^{(1)} \\ & & & \ddots & & & & & \vdots \\ & & & & \alpha_{kj_k}^{(k-1)} & \cdots & \cdots & \cdots & \alpha_{k,m}^{(k-1)} \\ & & \mathbf{0} & & \alpha_{k+1,j_k}^{(k-1)} & \cdots & \alpha_{k+1,j_{k+1}}^{(k-1)} & \cdots & \alpha_{k+1,m}^{(k-1)} \\ & & & & \vdots & \ddots & \vdots & \ddots & \vdots \\ & & & & \alpha_{n,j_k}^{(k-1)} & \cdots & \alpha_{n,j_{k+1}}^{(k-1)} & \cdots & \alpha_{nm}^{(k-1)} \end{pmatrix}$$

zur direkt nachfolgenden Matrix

$$B^{(k)} = \begin{pmatrix} (\alpha_{ij}^{(i-1)})_{\substack{i=1\ldots k\\ j=1\ldots m}} \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\ldots n\\ j=1\ldots m}} \end{pmatrix} =$$

$$= \begin{pmatrix} \alpha_{1j_1}^{(0)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{1m}^{(0)} \\ & & \alpha_{2j_2}^{(1)} & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{2m}^{(1)} \\ & & & \ddots & & & & & \vdots \\ & & & & \alpha_{k,j_k}^{(k-1)} & \cdots & \cdots & \cdots & \alpha_{k,m}^{(k-1)} \\ & & \mathbf{0} & & & & \alpha_{k+1,j_{k+1}}^{(k)} & \cdots & \alpha_{k+1,m}^{(k)} \\ & & & & & & \vdots & \ddots & \vdots \\ & & & & & & \alpha_{n,j_{k+1}}^{(k)} & \cdots & \alpha_{nm}^{(k)} \end{pmatrix}$$

in der im Vergleich zu vorher die Zeilenstufenspalte j_k ausgeräumt ist.

3.5 Lemma (1-Schritt-Rekursion zur Trigonalisierung)
Betrachte in der obigen Situation für $k = 1, \ldots, n-1$ die Matrix $B^{(k-1)}$ mit

$$t := b_{k-1,j_{k-1}}^{(k-1)} \ (\neq 0 \ stets).$$

Die Matrix $E = (e_{ij}) \in \mathrm{Mat}_{n\times m}(R)$ sei wie folgt definiert:
(1) Die oberen Zeilen $i = 1, \ldots, k$ von E sind genau wie in $B^{(k-1)}$.
(2) Für die unteren Zeilen $i = k+1, \ldots, n$ gilt für $j = 1, \ldots, m$:

$$e_{ij} := \frac{1}{t} \cdot [k\ i][j_k\ j](B^{(k-1)}).$$

Dann gilt $E = B^{(k)}$.
Beweis:
Dies folgt sofort aus der Merkregel 3.4. □

3.6 Bemerkung (Zeilentausch)
Sei $A \in \mathrm{Mat}_{n\times m}(R)$ eine beliebige $n \times m$-Matrix mit $n \leq m$. Die Folge der Matrizen, die man aus A durch sukzessive Spaltenausräumung erhält, wobei möglicherweise Zeilen vertauscht werden, werde mit $\widetilde{B}^{(k)}$, $k = 0, \ldots, n-1$, bezeichnet.

Angenommen, es gilt $a_{11} = 0$ und $a_{21} \neq 0$. Dann muß man zuerst die Zeilen 1 und 2 vertauschen, um die Matrix $\widetilde{B}^{(0)}$ zu erhalten und damit die nächste Matrix $\widetilde{B}^{(1)}$ berechnen zu können. Die neu berechneten Einträge von $\widetilde{B}^{(1)}$ sind dann aber nicht mehr von der Form $\alpha_{ij}^{(1)} = [1\ i][j_1\ j](A)$, sondern von der Form $[2\ i][j_1\ j](A)$. Allgemein gesprochen, muß man also bei jedem Zeilentausch in der Rekursionsformel den Klammerausdruck mit den Zeilenindizes entsprechend der Zeilenpermutation ändern. Für theoretische Betrachtungen wäre dies ungünstig. Aus diesem Grunde wird stets die Vereinbarung getroffen, daß die Matrix A ohne Zeilentausch auskommt.

Mit Hilfe der obigen Präsentation eines Rekursionsschrittes (Lemma 3.5) läßt sich nun leicht ein Algorithmus für das 1-Schritt-Verfahren formulieren, um beliebige $n\times m$-Matrizen zu trigonalisieren. Dabei darf man nicht vergessen, die zu Beginn der Überlegungen getroffenen Einschränkungen mitzubedenken, d.h. im Gegensatz zu vorher dürfen jetzt die Zeilenanzahl n, die Spaltenanzahl m und der Rang von A beliebig sein, und auch ein Zeilentausch soll nicht mehr ausgeschlossen werden.

Eine kurze Vorbemerkung zur Benennung der Algorithmen: es bezeichne stets
`B1, B2` – Bareiss-1- bzw. 2-Schritt
`T, D` – Trigonalisierung bzw. Diagonalisierung
`OD, MD` – ohne bzw. mit Division
`Row` – Zeilenalgorithmus.
Ferner umschließt die Bezeichnung eines Algorithmus auch die Angabe der Argumente, d.h. der Eingabewerte, die die zugehörige CoCoA-Funktion verarbeitet. In diesem Sinne ist eine CoCoA-Funktion, also ein Computerprogramm, wie eine mathematische Funktion zu verstehen, die den Eingabewerten eindeutig bestimmte Ausgabewerte zuordnet.

3.7 Satz (Algorithmus B1TMD)
Sei $A = (a_{ij})$ eine beliebige $n \times m$-Matrix über einem Integritätsring R. Die Unteralgorithmen `Pivot` *und* `B1TMDRow` *seien wie anschließend definiert. Betrachte die folgenden Instruktionen:*

(1) Setze $k := 1$, $\lambda := 1$ *und* $t := 1$.
(2) (Evtl. Zeilentausch) Falls $\lambda > m$ *oder* $k > n$ *ist, gib das Ergebnis* A *aus. Andernfalls ersetze* A *durch das Ergebnis von* $\texttt{Pivot}(A, \lambda, k)$.
(3) (Lange Zeilenstufe) Ist $a_{k\lambda} = 0$, *dann erhöhe* λ *um* 1 *und fahre fort bei Schritt (2).*
(4) (Spalte j_k *ausräumen) Setze* $j_k := \lambda$. *Ersetze die unteren Zeilen* $i = k+1, \ldots, n$ *von* A *jeweils durch das Ergebnis von* $\texttt{B1TMDRow}(A, i, j_k, k, t)$. *(Falls* $k = n$ *gilt, ist dies als leere Anweisung zu verstehen.)*
Setze dann $t := a_{kj_k}$ *und erhöhe* k *sowie* λ *jeweils um* 1. *Fahre bei Schritt (2) fort.*

Dies ist ein mit $\texttt{B1TMD}(A)$ *bezeichneter Algorithmus, der als Ergebnis eine zu* A *äquivalente Matrix in Zeilenstufenform ausgibt.*

Hierbei bedeute $\texttt{Pivot}(A, \lambda, k)$ *den folgenden Zeilentauschalgorithmus, um in der Matrix* A *an die Position* k *eine Zeile mit* $a_{k\lambda} \neq 0$ *zu tauschen, sofern dies möglich ist.*
(P1) Falls $a_{k\lambda} = 0$ *ist, suche die nächste Zeile* l *mit* $k < l \leq n$, *für die* $a_{l\lambda} \neq 0$ *gilt. Falls eine solche Zeile* l *existiert, dann multipliziere sie mit* -1 *und vertausche die beiden Zeilen* l *und* k.
(P2) Gib das Ergebnis A *aus.*

Der Algorithmus $\texttt{B1TMDRow}(A, i, j_k, k, t)$ *berechnet die* i*-te Zeile von* A *neu, d.h. räumt in Zeile* i *mit Hilfe von Zeile* k *die Spalte* j_k *aus:*
(R1) (Berechnen) Für $j = j_k+1, \ldots, m$ *ersetze* a_{ij} *durch* $[k\ i][j_k\ j](A) = a_{kj_k} \cdot a_{ij} - a_{ij_k} \cdot a_{kj}$. *Ersetze dann den neuberechneten Koeffizienten* a_{ij} *durch* a_{ij}/t.
(R2) (Ausräumen) Setze $a_{ij_k} := 0$.
(R3) Gib das Ergebnis $(a_{ij})_{j=1\ldots m}$ *aus, d.h. die* i*-te Zeile von* A.

Beweis:
Die Instruktionen (P1), (P2) bzw. (R1) bis (R3) können offensichtlich nur endlich oft durchlaufen werden. Dabei ist klar, daß `Pivot` wie behauptet die Zeilen tauscht. Daß `B1TMDRow` die i-te Zeile von A wie gewünscht neu berechnet, folgt sofort aus der 1-Schritt-Rekursion (Lemma 3.5).

Was `B1TMD` betrifft, so wird bei jedem Durchlauf von Schritt (3) λ um 1 erhöht, und bei jedem Durchlauf von Schritt (4) werden k und λ jeweils um 1 erhöht. Nach endlich vielen Schritten wird also das Abbruchkriterium in Schritt (2) erreicht.

Das Ergebnis von `B1TMD` stellt wie behauptet eine trigonalisierte Matrix dar, die zur ursprünglichen Matrix äquivalent ist, und zwar aufgrund der vorangegangenen Betrachtungen und aufgrund von folgendem: Mit dem Algorithmus `Pivot` in Schritt (2) und mit Schritt (3) wird dafür gesorgt, daß die Zeilenstufen korrekt gefunden werden, wenn ein Zeilentausch nötig ist oder wenn eine "lange" Zeilenstufe auftritt. (Insbesondere dient die Variable λ dazu, die jeweilige Zeilenstufenspalte j_k zu ermitteln.) Schritt (4) ist die direkte Anwendung der 1-Schritt-Rekursion (Lemma 3.5). Das Abbruchkriterium in Schritt (2) stellt außerdem sicher, daß bei beliebiger Zeilen- und Spaltenanzahl und bei beliebigem Rang von A die Instruktionen genau dann beendet werden, wenn die gewünschte Zeilenstufenform erreicht ist. □

3.8 Bemerkung (Komplexität von B1TMD)

Sei $A \in \mathrm{Mat}_{n\times m}(R)$ eine beliebige $n \times m$-Matrix maximalen Ranges, wobei für die Zeilenstufen $j_k = k$ für $k = 1, \ldots, \operatorname{rang} A$ gelte. Betrachtet man den Algorithmus `B1TMD` aus Satz 3.7, so ergeben sich dort bei jedem Durchlauf von Schritt (4)

$(n-k)(m-k)$ Additionen,

$2(n-k)(m-k)$ Multiplikationen,

$(n-k)(m-k)$ Divisionen.

Ist A eine $n \times m$-Matrix mit $n \leq m$ und $\operatorname{rang} A = n$, so erhält man insgesamt also $(k = 1, \ldots, n-1\ )$

$$\approx \tfrac{1}{2}mn^2 - \tfrac{1}{6}n^3 \text{ Additionen},$$
$$\approx mn^2 - \tfrac{1}{3}n^3 \text{ Multiplikationen},$$
$$\approx \tfrac{1}{2}mn^2 - \tfrac{1}{6}n^3 \text{ Divisionen}.$$

Ist A dagegen eine $n \times m$-Matrix mit $n > m$ und $\operatorname{rang} A = m$, so ergeben sich insgesamt $(k = 1, \ldots, m-1\ )$

$$\approx \tfrac{1}{2}mn^2 - \tfrac{1}{6}m^3 \text{ Additionen},$$
$$\approx mn^2 - \tfrac{1}{3}m^3 \text{ Multiplikationen},$$
$$\approx \tfrac{1}{2}mn^2 - \tfrac{1}{6}m^3 \text{ Divisionen}.$$

Hierbei ist die Anzahl der Additionen vernachlässigbar, da dafür nur relativ wenig Rechenzeit aufgewandt werden muß. Kostspielig sind vor allem Multiplikationen und Divisionen. Zur Kalkulation der Komplexitäten vergleiche auch die Formelsammlung im Anhang.

3.9 Bemerkungen (Algorithmus B1TOD und Gaußsches Eliminationsverfahren)

Sei A eine beliebige $n \times m$-Matrix über einem Integritätsring R.

a) *(Algorithmus* `B1TOD`*)* Bei dem Algorithmus `B1TMD` aus Satz 3.7 kann auf die Division verzichtet werden. Läßt man die Variable t weg und ersetzt man in Schritt (4) den Unteralgorithmus `B1TMDRow`(A, i, j_k, k, t) durch `B1TODRow`(A, i, j_k, k), so erhält man den Algorithmus `B1TOD`(A), der eine zu A äquivalente Matrix in Zeilenstufenform zum Ergebnis hat. Dabei ist `B1TODRow`(A, i, j_k, k) genau wie `B1TMDRow`(A, i, j_k, k, t) definiert, nur daß in Schritt (R1) die Division durch t entfällt.

b) *(Komplexität von* `B1TOD`*)* Bis auf die Divisionen ist die Komplexität von `B1TOD` genau so, wie in Bemerkung 3.8 für `B1TMD` angegeben.

c) Der in a) beschriebene Algorithmus `B1TOD` entspricht gerade dem allgemein bekannten *Gaußschen Eliminationsverfahren.* Ohne Division wachsen bei jedem Rekursionsschritt die Koeffizienten jedoch i.a. sehr viel stärker als beim Bareiss-Verfahren, welches die Division mit Teilern, also bruchfrei, durchführt.

3.10 Bemerkungen zur Implementierung (B1TMD, B1TOD)

Die folgenden Punkte gelten auch für die später noch dargestellten Algorithmen:

a) Die Unteralgorithmen, die Zeilentausch oder Zeilenberechnungen vornehmen, sind deshalb aus den Algorithmen ausgegliedert, um diese einerseits übersichtlicher zu gestalten. Andererseits sind diese Unteralgorithmen aber auch noch für künftige Algorithmen verwendbar, die auf diese Art nach dem Baukastenprinzip zusammengestellt werden können.

b) (*overwriting*) Es ist nicht nötig, sich die gesamte Folge der Matrizen $B^{(k)}$ zu merken, denn für jeden Rekursionsschritt wird jeweils nur die Vorgängermatrix benötigt. Um Speicherplatz zu sparen, kann man also, wie im Algorithmus `B1TMD` (Satz 3.7) geschehen, die Variable A immer wieder überschreiben. Dies gilt auch für andere Variablen, z.B. k, λ, t. Dabei ist besonders zu beachten, daß man nur solche Koeffizienten überschreibt, deren alten Wert man nicht mehr für weitere Berechnungen benötigt; beispielsweise darf man in Satz 3.7 den Schritt (R2) keinesfalls vor Schritt (R1) durchführen, denn in (R1) wird der alte Wert von a_{ij_k} zur Berechnung von a_{ij} benötigt. Erst danach darf man $a_{ij_k} = 0$ setzen.

c) *(Zuweisungen)* Es ist stets günstiger, Zuweisungen statt Berechnungen vorzunehmen. Daher werden die Spalten, die ausgeräumt werden sollen, 0 gesetzt.

d) Führt man das Verfahren ohne Division durch, kann man bei dem Algorithmus aus Satz 3.7 in Schritt (4) die Neuberechnung von Zeilen i entfallen lassen, in denen schon wie gewünscht $a_{ij_k} = 0$ steht. Im Verfahren mit Division müssen dagegen auch solche Zeilen neu berechnet werden, damit sich anschließend die richtigen Minoren in der Zeile befinden (andernfalls wäre danach die Division i.a. nicht mehr bruchfrei).

e) Ist R ein Polynomring, so erweist es sich bei CoCoA i.a. als günstiger, die Division mit dem Divisionsalgorithmus `DivAlg` durchzuführen statt mit dem Operator "/".

f) Beim Unteralgorithmus `Pivot` wird bei einem Zeilentausch in Schritt (P1) eine der beiden vertauschten Zeilen mit -1 multipliziert. Dies ist zwar nicht unbedingt nötig, stellt aber sicher, daß die bei jedem Schritt rekursiv ermittelten Koeffizienten dasselbe Vorzeichen tragen wie die entsprechenden Determinanten der ursprünglichen Matrix A. (Sei beispielsweise A eine quadratische Matrix der Ordnung n. Die dazu äquivalente obere Dreiecksmatrix, die man mit `B1TMD` ermittelt, werde mit B bezeichnet. Möglicherweise müssen im Laufe des Verfahrens mehrmals Zeilen vertauscht werden. Da aber bei jedem Zeilentausch der Vorzeichenwechsel der Determinante berücksichtigt wird, gilt am Ende $b_{nn} = |A|$.)

g) In CoCoA werden Multiplikationen der Form "$0 \cdot x$" mit einem $x \in R$ oder Divisionen der Form "$0/y$" mit einem $y \in R \setminus \{0\}$ ohne Zeitverlust durchgeführt, d.h. die CPU-Zeit beträgt dafür 0 sec. Für Multiplikationen der Form "$1 \cdot y$" oder Divisionen der Form "$y/1$" mit $y \in R \setminus \{0\}$ ergibt sich eine sehr geringe CPU-Zeit > 0 sec. Daher entfällt bei `B1TMDRow` die Division, wenn $t = 1$ gilt. Darüberhinaus wird auch auf Multiplikationen verzichtet, bei denen ein Faktor gleich 0 ist.

4 Trigonalisierung mit dem 2-Schritt-Verfahren

Es liege weiterhin die Situation wie in Abschnitt 3 vor, d.h. A sei eine $n \times m$-Matrix über einem Integritätsring R mit den folgenden Eigenschaften:

(V1) $n \leq m$,
(V2) $\operatorname{rang} A = n$ und
(V3) Zeilentausch ist nicht nötig.

Auch die Matrizenfolge $(B^{(k)})_{k=0,\ldots,n-1}$ sei wie gehabt mit

$$B^{(k)} = (b_{ij}^{(k)})_{\substack{i=1\ldots n\\ j=1\ldots m}} := \begin{pmatrix} (\alpha_{ij}^{(i-1)})_{\substack{i=1\ldots k\\ j=1\ldots m}} \\ \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\ldots n\\ j=1\ldots m}} \end{pmatrix}.$$

Außerdem setzt man wieder

$$b_{0j_0}^{(0)} := 1.$$

Statt wie im vorangegangen Abschnitt die Matrix A auf Zeilenstufenform zu bringen, indem man eine Zeilenstufenspalte nach der anderen ausräumt, werden beim 2-Schritt-Verfahren nun immer zwei Spalten auf einmal ausgeräumt. Welche Rekursionsformeln ergeben sich in diesem Falle?

Dazu betrachtet man wieder die Folge der Matrizen $B^{(k)}$ und insbesondere für gerades $k \in \{2, 4, \ldots, n-1\}$ den Übergang von der Matrix

$$B^{(k-2)} = \begin{pmatrix} (\alpha_{ij}^{(i-1)})_{\substack{i=1\ldots k-2\\ j=1\ldots m}} \\ (\alpha_{ij}^{(k-2)})_{\substack{i=k-1\ldots n\\ j=1\ldots m}} \end{pmatrix} =$$

$$= \begin{pmatrix} \alpha_{1j_1}^{(0)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{1m}^{(0)} \\ & & \alpha_{2j_2}^{(1)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{2m}^{(1)} \\ & & & \ddots & & & & & & & \vdots \\ & & & & \alpha_{k-1,j_{k-1}}^{(k-2)} & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{k-1,m}^{(k-2)} \\ & & & & \alpha_{k,j_{k-1}}^{(k-2)} & \cdots & \alpha_{k,j_k}^{(k-2)} & \cdots & \cdots & \cdots & \alpha_{k,m}^{(k-2)} \\ & & \mathbf{0} & & \alpha_{k+1,j_{k-1}}^{(k-2)} & \cdots & \alpha_{k+1,j_k}^{(k-2)} & \cdots & \alpha_{k+1,j_{k+1}}^{(k-2)} & \cdots & \alpha_{k+1,m}^{(k-2)} \\ & & & & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ & & & & \alpha_{n,j_{k-1}}^{(k-2)} & \cdots & \alpha_{n,j_k}^{(k-2)} & \cdots & \alpha_{n,j_{k+1}}^{(k-2)} & \cdots & \alpha_{nm}^{(k-2)} \end{pmatrix}$$

zur übernächsten Matrix

$$B^{(k)} = \begin{pmatrix} (\alpha_{ij}^{(i-1)})_{\substack{i=1\ldots k\\ j=1\ldots m}} \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\ldots n\\ j=1\ldots m}} \end{pmatrix} =$$

$$= \begin{pmatrix} \alpha_{1j_1}^{(0)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{1m}^{(0)} \\ & & \alpha_{2j_2}^{(1)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{2m}^{(1)} \\ & & & \ddots & & & & & & & \vdots \\ & & & & \alpha_{k-1,j_{k-1}}^{(k-2)} & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{k-1,m}^{(k-2)} \\ & & & & & & \alpha_{k,j_k}^{(k-1)} & \cdots & \cdots & \cdots & \alpha_{k,m}^{(k-1)} \\ & & \mathbf{0} & & & & & & \alpha_{k+1,j_{k+1}}^{(k)} & \cdots & \alpha_{k+1,m}^{(k)} \\ & & & & & & & & \vdots & \ddots & \vdots \\ & & & & & & & & \alpha_{n,j_{k+1}}^{(k)} & \cdots & \alpha_{nm}^{(k)} \end{pmatrix}$$

in der im Vergleich zu vorher die beiden Zeilenstufenspalten j_{k-1} und j_k ausgeräumt sind.

4.1 Lemma (2-Schritt-Rekursion zur Trigonalisierung)
Sei A wie eben, und sei $n-1$ gerade. Betrachte in der obigen Situation für $k = 2, 4, \ldots, n-1$ die Matrix $B^{(k-2)}$ mit

$$t := b_{k-2,j_{k-2}}^{(k-2)} \ (\neq 0 \ stets).$$

Die Matrix $E = (e_{ij}) \in \mathrm{Mat}_{n\times m}(R)$ sei folgendermaßen definiert:

(1) Die oberen Zeilen $i = 1, \ldots, k-1$ von E sind genau wie in $B^{(k-2)}$.

(2) Für die unteren Zeilen $i = k+1, \ldots, n$ gilt für $j = 1, \ldots, m$:

$$e_{ij} := \frac{1}{t^2} \cdot [k-1 \ k \ i][j_{k-1} \ j_k \ j](B^{(k-2)}).$$

(3) Für die Zeile k gilt für $j = 1, \ldots, m$:

$$e_{kj} := \frac{1}{t} \cdot [k-1 \ k][j_{k-1} \ j](B^{(k-2)}).$$

Dann gilt $E = B^{(k)}$.

Beweis:
Dies folgt sofort aus der Merkregel 3.4. □

4.2 Bemerkungen zur 2-Schritt-Rekursion (Determinantenentwicklung)

a) *(Determinantenentwicklung)* Die Matrix $B := B^{(k-2)}$ sei genau wie in obigem Lemma 4.1. In 4.1 (2) tritt eine Determinante dritter Ordnung auf, nämlich

$$[k-1 \ k \ i][j_{k-1} \ j_k \ j](B) = \begin{vmatrix} b_{k-1,j_{k-1}} & b_{k-1,j_k} & b_{k-1,j} \\ b_{k,j_{k-1}} & b_{k,j_k} & b_{k,j} \\ b_{i,j_{k-1}} & b_{i,j_k} & b_{i,j} \end{vmatrix}.$$

Entwickelt man diese nach der letzten Spalte j, so erhält man

$$\varepsilon := [k-1 \ k \ i][j_{k-1} \ j_k \ j](B) = c_0 \cdot b_{ij} - c_{i1} \cdot b_{k,j} + c_{i2} \cdot b_{k-1,j}$$

mit den Faktoren

$$c_0 := [k-1 \ k][j_{k-1} \ j_k](B)$$
$$c_{i1} := [k-1 \ i][j_{k-1} \ j_k](B)$$
$$c_{i2} := [k \ i][j_{k-1} \ j_k](B).$$

Die c_0, c_{i1}, c_{i2} sind — bis auf das Vorzeichen von c_{i1} — die zugehörigen Kofaktoren. Wegen der Teilbarkeit der Kofaktoren gilt

$$\frac{\varepsilon}{t^2} = \frac{1}{t} \cdot \Big(\underbrace{\frac{c_0}{t}}_{\in R} \cdot b_{ij} - \underbrace{\frac{c_{i1}}{t}}_{\in R} \cdot b_{k,j} + \underbrace{\frac{c_{i2}}{t}}_{\in R} \cdot b_{k-1,j} \Big) \quad \in R.$$

b) *(Zeilenstufenkoeffizient)* Insbesondere für den Zeilenstufenkoeffizienten der k-ten Zeile gilt aufgrund von Lemma 4.1(3) und Bemerkung a)

$$b_{kj_k}^{(k)} = c_0/t.$$

4.3 Satz (Algorithmus B2TMD)
Sei $A = (a_{ij})$ eine beliebige $n \times m$-Matrix über einem Integritätsring R. Die Unteralgorithmen `Pivot` *und* `B1TMDRow` *seien wie beim Algorithmus* `B1TMD` *aus Satz 3.7,* `C0Pivot` *und* `B2TMDRow` *seien wie anschließend definiert. Betrachte die folgenden Instruktionen:*

(1) Setze $k := 1$, $\lambda := 1$ und $t := 1$.

(2) (Evtl. Zeilentausch) Falls $\lambda > m$ oder $k > n$ ist, gib das Ergebnis A aus. Andernfalls ersetze A durch das Ergebnis von `Pivot`(A, λ, k).

(3) (Lange Zeilenstufe) Ist $a_{k\lambda} = 0$, dann erhöhe λ um 1 und fahre fort bei Schritt (2).

(4) (Letzte Spalte ausräumen) Setze $j_k := \lambda$. Falls $j_k = m$ ist, dann ersetze die unteren Zeilen $i = k+1, \ldots, n$ der Matrix A jeweils durch das Ergebnis von `B1TMDRow`(A, i, j_k, k, t). *Erhöhe λ um 1 und fahre bei Schritt (2) fort.*

(5) (Evtl. Zeilentausch) Setze $j_k := \lambda$. Führe `C0Pivot`(A, j_k, k) *aus und erhalte daraus A und c_0.*

(6) (Lange Zeilenstufe, Spalten $j_k, j_k + 1$ ausräumen) Falls $c_0 = 0$ ist, dann ersetze die unteren Zeilen $i = k+1, \ldots, n$ von A jeweils durch das Ergebnis von `B1TMDRow`(A, i, j_k, k, t). *Setze dann $t := a_{kj_k}$, erhöhe k um 1, λ um 2 und fahre bei Schritt (2) fort.*

(7) (Spalten $j_k, j_k + 1$ ausräumen) Setze $j_{k+1} := j_k + 1$ und $t_2 := t^2$. Ersetze zunächst die unteren Zeilen $i = k+2, \ldots, n$ von A jeweils durch das Ergebnis von `B2TMDRow`(A, i, j_k, k, c_0, t_2). *Ersetze dann die Zeile $(k+1)$ durch das Ergebnis von* `B1TMDRow`$(A, k+1, j_k, k, t)$. *Setze $t := a_{k+1,j_k+1}$, erhöhe k und λ jeweils um 2 und fahre bei Schritt (2) fort.*

Dies ist ein Algorithmus namens `B2TMD`(A), *der eine zu A äquivalente Matrix in Zeilenstufenform erzeugt.*

Hierbei bedeute `C0Pivot`(A, j_k, k) *den folgenden Zeilentauschalgorithmus, um — sofern möglich — an Position $(k+1)$ eine Zeile zu tauschen, deren Zeilenstufe an der Stelle $(k+1, j_k+1)$ liegt.*

(P1) Berechne $c_0 := [k\ k+1][j_k\ j_k+1](A) = a_{kj_k} \cdot a_{k+1,j_k+1} - a_{k+1,j_k} \cdot a_{k,j_k+1}$. Falls $c_0 = 0$ ist, suche die nächste Zeile l mit $k+1 < l \leq n$, für die $c_0 := [k\ l][j_k\ j_k+1](A) \neq 0$ gilt. Falls eine solche Zeile l existiert, dann multipliziere sie mit -1 und vertausche die Zeilen l und $(k+1)$.

(P2) Gib als Ergebnis A und c_0 aus.

In Schritt (7) bezeichne `B2TMDRow`(A, i, j_k, k, c_0, t_2) *folgenden Algorithmus, welcher die i-te Zeile von A neu berechnet, d.h. es werden in der i-Zeile mit Hilfe der Zeilen $k, k+1$ die Spalten $j_k, j_k + 1$ ausgeräumt.*

(R1) (Kofaktor berechnen) Berechne $c_1 := [k\ i][j_k\ j_k+1](A) = a_{kj_k} \cdot a_{i,j_k+1} - a_{ij_k} \cdot a_{k,j_k+1}$.

(R2) (Kofaktor berechnen) Berechne dann $c_2 := [k+1\ i][j_k\ j_k+1](A) = a_{k+1,j_k} \cdot a_{i,j_k+1} - a_{ij_k} \cdot a_{k+1,j_k+1}$.

(R3) (Determinante entwickeln) Für $j = j_k + 2, \ldots, m$ ersetze den Koeffizienten a_{ij} zunächst durch $c_0 \cdot a_{ij} - c_1 \cdot a_{k+1,j} + c_2 \cdot a_{kj}$. Ersetze daraufhin den neuen Koeffizienten a_{ij} durch a_{ij}/t_2.

(R4) (Ausräumen) Setze $a_{ij_k} := 0$ und $a_{i,j_k+1} := 0$.

(R5) Gib das Ergebnis $(a_{ij})_{j=1\ldots m}$ aus, d.h. die i-te Zeile von A.

Beweis:
Offensichtlich werden die Anweisungen (P1), (P2) bzw. (R1) bis (R5) nur endlich oft wiederholt. Daß sie zu den behaupteten Ergebnissen führen, folgt sofort aus der 2-Schritt-Rekursion (Lemma 4.1 bzw. Bemerkung 4.2).

Was `B2TMD` angeht, so wird bei jedem Durchlauf von Schritt (3), (4), (6) oder (7) die Variable λ erhöht, bei jedem Durchlauf von Schritt (6) oder (7) wird k erhöht. Nach endlich vielen Wiederholungen wird also das Abbruchkriterium in Schritt (2) erreicht.

Analog zum Beweis von Satz 3.7 (`B1TMD`) und aufgrund der vorangegangenen und der folgenden Überlegungen ist das Ergebnis eine trigonalisierte Matrix, die zur ursprünglichen Matrix äquivalent ist. Schritt (4) wird durchlaufen, falls der 2er-Schritt nicht aufgeht und am Ende nur noch die letzte Spalte m ausgeräumt werden muß – dann ist man fertig. Schritt (5) ist nötig, da der gemäß 1-Schritt-Formel neu zu berechnende Koeffizient a_{k+1,j_k+1} gleich c_0/t ist, vgl. Bemerkung 4.2b). Ist nun $c_0 = 0$, muß evtl. ein Zeilentausch durchgeführt werden, um an der Stelle $(k+1, j_k+1)$ eine Zeilenstufe zu erhalten. Läßt sich diese nicht finden, d.h. ist $c_0 = 0$ stets, bedeutet dies gerade, daß mit der 1-Schritt-Formel schon beide Spalten $j_k, j_k + 1$ ausgeräumt werden, vgl. Schritt(6). Schritt (7) folgt direkt aus der 2-Schritt-Rekursion (Lemma 4.1 bzw. Bemerkung 4.2). □

4.4 Bemerkung (Komplexität von B2TMD)
Sei $A \in \mathrm{Mat}_{n\times m}(R)$ ein $n \times m$-Matrix maximalen Ranges, wobei für die Lage der Zeilenstufen $j_k = k$ für $k = 1, \ldots, \operatorname{rang} A$ gelte. Betrachtet man den Algorithmus `B2TMD` aus Satz 4.3, so ergeben sich dort bei jedem Durchlauf von Schritt (7)

$$2(n-k)(m-k) + (m-k+1) + 2(n-k) + 1 \text{ Additionen,}$$
$$3(n-k)(m-k) + 2(m-k+1) + 4(n-k) + 3 \text{ Multiplikationen,}$$
$$(n-k)(m-k) + (m-k+1) \text{ Divisionen.}$$

Bezeichnet A eine $n \times m$-Matrix mit $n \le m$ und mit $\operatorname{rang} A = n$, so führt dies insgesamt ($k = 1, 3, \ldots, n-1$, k ungerade) zu

$$\approx \tfrac{1}{2}mn^2 - \tfrac{1}{6}n^3 \text{ Additionen,}$$
$$\approx \tfrac{3}{4}mn^2 - \tfrac{1}{4}n^3 \text{ Multiplikationen,}$$
$$\approx \tfrac{1}{4}mn^2 - \tfrac{1}{12}n^3 \text{ Divisionen.}$$

Ist A dagegen eine $n\times m$-Matrix mit $n > m$ und mit $\operatorname{rang} A = m$, so ergeben sich insgesamt ($k = 1, \ldots, m-1$, k ungerade)

$$\approx \tfrac{1}{2}mn^2 - \tfrac{1}{6}m^3 \text{ Additionen,}$$
$$\approx \tfrac{3}{4}mn^2 - \tfrac{1}{4}m^3 \text{ Multiplikationen,}$$
$$\approx \tfrac{1}{4}mn^2 - \tfrac{1}{12}m^3 \text{ Divisionen.}$$

4.5 Bemerkungen (Algorithmus B2TOD)
Sei $A \in \mathrm{Mat}_{n\times m}(R)$ eine beliebige $n \times m$-Matrix über einem Integritätsring R.

a) *(Algorithmus* `B2TOD`*)* Der Algorithmus `B2TMD` aus Satz 4.3 läßt sich auch ohne Division durchführen, d.h. man läßt die Teiler t sowie t_2 weg und ersetzt in Schritt (4), (6) und (7) `B1TMDRow` durch `B1TODRow` (aus Bemerkung 3.9) und außerdem in Schritt (7) `B2TMDRow`(A, i, j_k, k, c_0, t_2) durch `B2TODRow`(A, i, j_k, k, c_0). Letzteres ist genau wie `B2TMDRow` definiert, außer daß in Schritt (R3) die Division durch t_2 entfällt. Damit erhält man den Algorithmus `B2TOD`(A), der als Ergebnis eine zu A äquivalente Matrix in Zeilenstufenform ausgibt.

b) *(Komplexität von* `B2TOD`*)* Bis auf die Divisionen erhält man für `B2TOD` dieselbe Komplexität, wie in Bemerkung 4.4 für den Algorithmus `B2TMD` angegeben.

Nutzt man bei der Berechnung der Determinanten dritter Ordnung die Teilbarkeit der Kofaktoren aus (Bemerkung 4.2a), so folgt als Variante des Algorithmus `B2TMD` aus Satz 4.3

4.6 Satz (Algorithmus B2TMDMD)
Sei $A \in \mathrm{Mat}_{n\times m}(R)$ eine beliebige $n \times m$-Matrix über einem Integritätsring R. Betrachte den Algorithmus `B2TMD` *aus Satz 4.3 und nehme folgende Änderungen vor:*

Lasse in Schritt (7) den Teiler t_2 weg und ersetze `B2TMDRow`(A, i, j_k, k, c_0, t_2) *durch den Algorithmus* `B2TMDMDRow`(A, i, j_k, k, c_0, t)*, welcher wie folgt definiert ist:*

(R1) (Kofaktor berechnen) Ersetze c_0 durch c_0/t.

(R2) (Kofaktor berechnen) Berechne $c_1 := [k\ i][j_k\ j_k+1](A) = a_{kj_k} \cdot a_{i,j_k+1} - a_{ij_k} \cdot a_{k,j_k+1}$. Ersetze c_1 durch c_1/t.

(R3) (Kofaktor berechnen) Berechne zunächst $c_2 := [k+1\ i][j_k\ j_k+1](A) = a_{k+1,j_k} \cdot a_{i,j_k+1} - a_{ij_k} \cdot a_{k+1,j_k+1}$. Ersetze c_2 durch c_2/t.

(R4) (Determinante entwickeln) Für $j = j_k + 2, \ldots, m$ ersetze den Koeffizienten a_{ij} durch $c_0 \cdot a_{ij} - c_1 \cdot a_{k+1,j} + c_2 \cdot a_{kj}$. Ersetze dann den neuen Koeffizienten a_{ij} durch a_{ij}/t.

(R5) (Ausräumen) Setze $a_{ij_k} := 0$ und $a_{i,j_k+1} := 0$.

(R6) Gib das Ergebnis $(a_{ij})_{j=1\ldots m}$ aus, d.h. die i-te Zeile von A.

Dies ist ebenfalls ein Algorithmus, welcher mit `B2TMDMD`(A) *bezeichnet wird und als Ergebnis eine zu A äquivalente Matrix in Zeilenstufenform ausgibt. Sein Ergebnis stimmt mit dem von* `B2TMD` *überein.*

Beweis:
Die Behauptung folgt sofort aus der Teilbarkeit der Kofaktoren (Bemerkung 3.3c). Statt nämlich erst die Determinante dritter Ordnung zu berechnen und dann durch t^2 zu teilen — wie beim Algorithmus `B2TMD` bzw. `B2TMDRow` (Satz 4.3) — kann man erst die Kofaktoren durch t teilen (Schritt (R1) bis (R3)) und dann das Ergebnis aus Schritt (R4) erneut durch t dividieren. □

4.7 Bemerkung (Komplexität von B2TMDMD)
Sei $A \in \mathrm{Mat}_{n\times m}(R)$ eine beliebige $n \times m$-Matrix maximalen Ranges mit $j_k = k$ für alle $k = 1, \ldots, \operatorname{rang} A$. Beim Algorithmus `B2TMDMD` aus Satz 4.6 benötigt man für jeden Durchlauf von Schritt (7)

$$2(n-k)(m-k) + (m-k+1) + 2(n-k) + 1 \text{ Additionen,}$$
$$3(n-k)(m-k) + 2(m-k+1) + 4(n-k) + 2 \text{ Multiplikationen,}$$
$$(n-k)(m-k) + (m-k+1) + 2(n-k) + 1 \text{ Divisionen.}$$

Ist A eine $n \times m$-Matrix mit $n \le m$, so ergeben sich insgesamt ($k = 1, 3, \ldots, n-1$, k ungerade)

$$\approx \tfrac{1}{2}mn^2 - \tfrac{1}{6}n^3 \text{ Additionen,}$$
$$\approx \tfrac{3}{4}mn^2 - \tfrac{1}{4}n^3 \text{ Multiplikationen,}$$
$$\approx \tfrac{1}{4}mn^2 - \tfrac{1}{12}n^3 \text{ Divisionen.}$$

Bezeichnet A dagegen eine $n \times m$-Matrix mit $n > m$ und mit $\operatorname{rang} A = m$, so ergeben sich insgesamt ($k = 1, \ldots, m-1$, k ungerade)

$$\approx \tfrac{1}{2}mn^2 - \tfrac{1}{6}m^3 \text{ Additionen,}$$
$$\approx \tfrac{3}{4}mn^2 - \tfrac{1}{4}m^3 \text{ Multiplikationen,}$$
$$\approx \tfrac{1}{4}mn^2 - \tfrac{1}{12}m^3 \text{ Divisionen.}$$

4.8 Bemerkungen zur Implementierung (B2TMD, B2TOD, B2TMDMD)
Betrachte zunächst den Algorithmus `B2TMD` aus Satz 4.3.

a) (*overwriting*) In Schritt (7) darf die Zeile $(k+1)$ erst zuletzt umgeformt werden, denn für `B2TMDRow` wird die alte $(k+1)$-te Zeile benötigt.

b) *(Zuweisungen)* Parameter, die öfter benötigt werden, sollten nicht jedesmal neu berechnet werden, sondern als Variable zur Verfügung stehen. Beispielsweise wird in Schritt (5) in `C0Pivot` schon c_0 berechnet, welches später noch benötigt wird, etwa für den Unteralgorithmus `B2TMDRow`.
In Schritt (6) kann man die Spalten j_k, j_k+1 beide 0 setzen, denn im Grunde genommen hat man mit `C0Pivot` in Schritt (5) schon berechnet, daß die Spalte (j_k+1) ausgeräumt wird. In Schritt (7) kann man bei der Berechnung der $(k+1)$-ten Zeile $a_{k+1,j_{k+1}} := c_0/t$ setzen. Dazu muß man `B1TMDRow` so modifizieren, daß es, falls es aus Schritt (6) oder (7) aufgerufen wird, den Koeffizienten a_{k+1,j_k+1} nicht berechnet, sondern den entsprechenden Wert 0 bzw. c_0/t zuweist.

c) Bei `B2TMD` wird nur dann Schritt (7) durchlaufen, wenn die Zeilenstufenspalten j_k und $j_{k+1} = j_k+1$ direkt nebeneinander liegen. Man könnte es auch anders machen. Stellt man nämlich in Schritt (6) fest, daß $c_0 = 0$ ist, also sich in der Spalte (j_k+1) nicht die $(k+1)$-te Zeilenstufe befindet, könnte man in der Spalte (j_k+2) weitersuchen, ob man dort $c_0 = [k\ k+1][j_k\ j_k+2](A) \neq 0$ findet, usw. bis schließlich die Position $(k+1, j_{k+1})$ der Zeilenstufe gefunden ist. Dann könnte man statt Schritt (6) das echte 2-Schritt-Verfahren, wie in Lemma 4.1 beschrieben, durchführen. Dazu müßte man das in Schritt (7) verwendete `B2TMDRow` entsprechend abändern.

d) Auch bei `C0Pivot` wird wie bei `Pivot` bei einem Zeilentausch eine der getauschten Zeilen mit -1 multipliziert, um das Vorzeichen der Determinante zu erhalten.

Analog gelten a) bis d) auch für den Algorithmus `B2TMDMD` aus Satz 4.6. Für den Algorithmus `B2TOD` aus Bemerkung 4.5 gelten a) bis d) ebenfalls entsprechend, nur in b) muß berücksichtigt werden, daß man bei der Berechnung der Zeile $(k+1)$ `B1TODRow` so modifiziert, daß $a_{k+1,j_k+1} := c_0$ gesetzt wird.

4.9 Bemerkung (Mehrschritt-Verfahren)
Um eine $n \times m$-Matrix A aus $\mathrm{Mat}_{n\times m}(R)$ zu trigonalisieren, ließe sich analog zu den beschriebenen 1- und 2-Schritt-Verfahren ein s-Schritt-Verfahren mit $n-1 \geq s \geq 3$ konstruieren. Hierzu wird es allerdings nötig, Determinanten der Ordnung $(s+1)$ zu berechnen, vergleiche die Rekursionsformeln (3.3) bzw. die Merkregel (3.4). Diese Mehrschritt-Verfahren werden hier aber nicht mehr weiter ausgeführt.

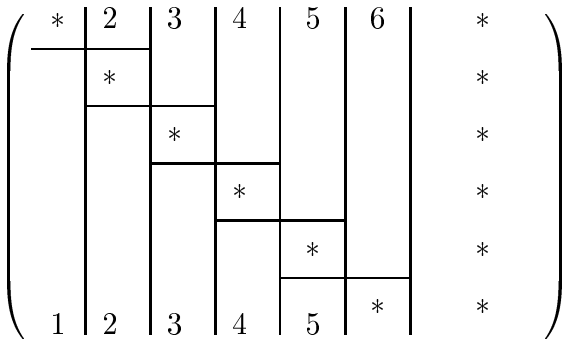
5 Diagonalisierung mit dem 1-Schritt-Verfahren

Nachdem man sich in den vorangegangenen Abschnitten bereits mit den Bareiss-Verfahren vertraut machen konnte, soll bei der Diagonalisierung der Spezialfall quadratischer Matrizen übersprungen und gleich der allgemeine Fall beliebiger $n \times m$-Matrizen betrachtet werden. Sei also ab sofort $A \in \mathrm{Mat}_{n\times m}(R)$ eine $n \times m$-Matrix über einem Integritätsring R. Für die folgenden Überlegungen gelte wieder wie zuvor

(V1) $n \leq m$,
(V2) $\operatorname{rang} A = n$ und
(V3) Zeilentausch ist unnötig.

Für die später dargestellten Algorithmen werden diese Voraussetzungen nicht benötigt. Sie dienen lediglich dazu, die folgenden Überlegungen nicht unnötig zu komplizieren.

Zur Diagonalisierung mit dem 1-Schritt-Verfahren werden sukzessive von links nach rechts die Spalten ober- und unterhalb der Zeilenstufendiagonalen ausgeräumt. Ein Schema für den vereinfachten Fall einer $6 \times m$-Matrix, deren Zeilenstufen alle auf der Diagonalen liegen, veranschaulicht die Reihenfolge der Ausräumung:



Hierbei symbolisiert $*$ die verbleibenden Koeffizienten.

In den vorausgegangenen Abschnitten hat man gelernt, daß bei der Trigonalisierung Koeffizienten der Form $\alpha_{ij}^{(k)}$ auftreten. Welche Koeffizienten erhält man nun bei der Diagonalisierung?

5.1 Definitionen
Die Matrix A sei wie oben.

a) Für $k \in \{0, \ldots, n\}$ schreibt man

$$\delta^{(k)} := \begin{cases} 1 & \text{falls } k = 0 \\ [1 \ldots k][j_1 \ldots j_k](A) & \text{sonst .} \end{cases}$$

b) Sei $k \in \{1, \ldots, n\}$. Für $i = 1, \ldots, k$ und $j = 1, \ldots, m$ bezeichne $\delta_{ij}^{(k)}$ die Determinante, die man aus $\delta^{(k)}$ erhält, indem man dort für die Spalte j_i die Spalte j eintauscht, also

$$\delta_{ij}^{(k)} := [1 \ldots k][j_1 \ldots j_{i-1}\ j\ j_{i+1} \ldots j_k](A) = [1 \ldots k][j_1 \ldots \widehat{j_i}\ j \ldots j_k](A),$$

wobei die Notation $\widehat{\cdot}$ die Auslassung bedeute. Außerdem setzt man für $i = 1, \ldots, n$ und für $j = 1, \ldots, m$

$$\delta_{ij}^{(0)} := 1.$$

5.2 Bemerkung
Mit den aus Abschnitt 3 bekannten Determinanten $\alpha_{ij}^{(k)} := [1 \ldots k\ \ i][j_1 \ldots j_k\ \ j](A)$ gilt insbesondere für $k \in \{1, \ldots, n\}$ und $j \in \{1, \ldots, m\}$

$$\delta_{kj}^{(k)} = [1 \ldots k][j_1 \ldots j_{k-1}\ j](A) = \alpha_{kj}^{(k-1)},$$

und speziell für den Zeilenstufenindex j_k erhält man den Zeilenstufenkoeffizienten

$$\delta_{kj_k}^{(k)} = [1 \ldots k][j_1 \ldots j_k](A) = \delta^{(k)} = \alpha_{kj_k}^{(k-1)} \quad (\neq 0).$$

So, wie in Lemma 2.1 der Zusammenhang zwischen den Determinanten $a_{ij}^{(k)}$ und der Trigonalisierung von Matrizen aufgezeigt wird, folgt nun eine entsprechende Aussage über die Determinanten $\delta_{ij}^{(k)}$ im Zusammenhang mit der Diagonalisierung von Matrizen.

5.3 Lemma (Matrizenfolge zur Diagonalisierung)
Sei $A \in \mathrm{Mat}_{n\times m}(R)$ eine $n \times m$-Matrix wie oben.

a) (Elementare Zeilenumformungen) Seien $k \in \{1,\ldots,n\}$ und $i \in \{1,\ldots,k\}$ beliebig. In der Matrix A die i-te Zeile $(a_{ij})_{j=1\ldots m}$ durch die Zeile $(\delta_{ij}^{(k)})_{j=1\ldots m}$ zu ersetzen, ist gleichbedeutend damit, in der i-ten Zeile die Spalten $j_1, j_2, \ldots, \widehat{j_i}, \ldots, j_k$ auszuräumen mittels elementarer Zeilenumformungen der Zeilen 1 bis k.
Mit anderen Worten: die Determinante $\delta_{ij}^{(k)} = [1\ldots k][j_1\ldots\widehat{j_i}\ j\ldots j_k](A)$ läßt sich als Anleitung für elementare Zeilenumformungen interpretieren, um den Koeffizienten a_{ij} aus A neu zu berechnen. Bei $\delta_{ij}^{(k)}$ weisen der hochgestellte Index (k) und der tiefgestellte Zeilenindex i darauf hin, daß die ersten k Zeilenstufenspalten (außer der j_i-ten) ausgeräumt werden, und zwar mit Hilfe der ersten k Zeilen. Die tiefgestellten Indizes i und j bezeichnen außerdem die Position desjenigen Koeffizienten, der neu berechnet wird.

b) (Matrizenfolge) Für $k = 0,\ldots,n$ sei die Matrix $D^{(k)}$ aus $\mathrm{Mat}_{n\times m}(R)$ wie folgt definiert:

$$D^{(k)} = (d_{ij}^{(k)}) := \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\ldots k\\ j=1\ldots m}} \\ \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\ldots n\\ j=1\ldots m}} \end{pmatrix}$$

In dieser Matrix sind oberhalb und unterhalb der Diagonalen die ersten k Zeilenstufenspalten $j_1, j_2, \ldots, j_k$ ausgeräumt. Insbesondere gilt $D^{(0)} = A$ und schließlich $D^{(n)} = (\delta_{ij}^{(n)})_{\substack{i=1\ldots n\\ j=1\ldots m}}$. Letzteres ist eine zu A äquivalente Diagonalmatrix.

c) Sei für $k = 1,\ldots,n$ die Matrix $D^{(k)} = (d_{ij}^{(k)})$ wie in a). In dieser Matrix gilt für die Zeilenstufenkoeffizienten der ersten k Zeilen $i = 1,\ldots,k$

$$d_{1j_1}^{(k)} = d_{2j_2}^{(k)} = \ldots = d_{kj_k}^{(k)} = \delta^{(k)} = d_{kj_k}^{(k-1)}.$$

Mit anderen Worten: sämtliche Zeilenstufenkoeffizienten der ersten k Zeilen sind gleich $\delta^{(k)}$, und dieses ist bereits aus der Vorgängermatrix $D^{(k-1)}$ bekannt, nämlich als Zeilenstufenkoeffizient $d_{kj_k}^{(k-1)}$ der Zeile k.

Beweis:

a) Es gilt für $i = 1,\ldots,k$ und $j = 1,\ldots m$

$$\delta_{ij}^{(k)} = [1\ldots k][j_1\ldots\widehat{j_i}\ j\ldots j_k](A) = [1\ldots\widehat{i}\ldots k\ i][j_1\ldots\widehat{j_i}\ldots j_k\ j](A),$$

denn da i und j beide von derselben Position aus nach hinten getauscht werden, stellt dies insgesamt eine gerade Permutation dar. Aus der rechten Seite folgt sofort für alle Spalten $j = 1,\ldots,m$, daß mit der Determinante $\delta_{ij}^{(k)}$ in der Zeile i die Spalten $j_1,\ldots,\widehat{j_i},\ldots,j_k$ ausgeräumt werden, und zwar mittels elementarer Zeilenumformungen der Zeilen $1,\ldots,k$, vgl. Lemma 2.1.

b) Für die Matrix

$$D^{(k)} = \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\ldots k\\ j=1\ldots m}} \\ \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\ldots n\\ j=1\ldots m}} \end{pmatrix}$$

gilt, daß nach a) wie gewünscht über der Zeilenstufendiagonalen mit den Determinanten $\delta_{ij}^{(k)}$ in den Zeilen $i = 1,\ldots,k$ die Spalten $j_1,\ldots,\widehat{j_i},\ldots,j_k$ ausgeräumt werden.

Unterhalb werden mit den Determinanten $\alpha_{ij}^{(k)}$ in den Zeilen $i = k+1, \ldots, n$ bekanntlich die Spalten $j_1, \ldots, j_k$ ausgeräumt. Nach a) ist speziell die Matrix $D^{(n)} = (\delta_{ij}^{(n)})$ eine zu A äquivalente Diagonalmatrix.

c) folgt sofort aus der Definition der Matrizen $D^{(k)}$ und aus Bemerkung 5.2. □

Sei ab sofort die Folge der Matrizen $D^{(k)} = (d_{ij}^{(k)})$ für $k = 0, \ldots, n$ so, wie in Lemma 5.3b) definiert, also

$$D^{(k)} = \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\ldots k \\ j=1\ldots m}} \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\ldots n \\ j=1\ldots m}} \end{pmatrix} =$$

$$= \begin{pmatrix}
\delta_{1j_1}^{(k)} & \cdots & * & 0 & * \cdots * & 0 & * \cdots * & \delta_{1j_{k+1}}^{(k)} & \cdots & \delta_{1m}^{(k)} \\
 & \ddots & \vdots & \vdots & \vdots \quad \vdots & \vdots & \vdots \quad \vdots & \vdots & & \vdots \\
 & & * & 0 & \vdots \quad \vdots & \vdots & \vdots \quad \vdots & \vdots & \ddots & \vdots \\
 & & & \delta_{k-1,j_{k-1}}^{(k)} & * \cdots * & 0 & \vdots \quad \vdots & \vdots & & \vdots \\
 & & & & & \delta_{k,j_k}^{(k)} & * \cdots * & \delta_{kj_{k+1}}^{(k)} & \cdots & \delta_{k,m}^{(k)} \\
 & & \mathbf{0} & & & & & \alpha_{k+1,j_{k+1}}^{(k)} & \cdots & \alpha_{k+1,m}^{(k)} \\
 & & & & & & & \vdots & \ddots & \vdots \\
 & & & & & & & \alpha_{n,j_{k+1}}^{(k)} & \cdots & \alpha_{nm}^{(k)}
\end{pmatrix}$$

Hierbei symbolisiert $*$ aus Platzgründen dazwischenliegende Koeffizienten, die ungleich 0 sein können. Außerdem setzt man

$$d_{0j_0}^{(0)} := 1.$$

Nach dem vorangegangenen Lemma 5.3 sind in der Matrix $D^{(k)}$ die ersten k Zeilenstufenspalten, nämlich $j_1, \ldots, j_k$, oberhalb und unterhalb der Diagonalen ausgeräumt, insbesondere sind somit unter der Diagonalen die vorderen Spalten $1, \ldots, j_{k+1} - 1$ ausgeräumt. Die Matrix nimmt also die angegebene Gestalt an. Somit führt die Folge dieser Matrizen die Matrix $A = D^{(0)}$ in die Diagonalmatrix $D^{(n)} = (\delta_{ij}^{(n)})$ über, indem sukzessive von links nach rechts über und unter der Diagonalen die Zeilenstufenspalten ausgeräumt werden.

Für die eingangs schematisch dargestellte $6 \times m$-Matrix, deren Zeilenstufen alle auf der Diagonalen liegen, sieht der Übergang von einer Matrix $D^{(k-1)}$ zur nächsten Matrix $D^{(k)}$, $k = 1, \ldots, n$, wie folgt aus:

$$D^{(k-1)} = \begin{pmatrix} (\delta_{ij}^{(k-1)})_{\substack{i=1\ldots k-1 \\ j=1\ldots m}} \\ (\alpha_{ij}^{(k-1)})_{\substack{i=k\ldots n \\ j=1\ldots m}} \end{pmatrix} = \left(\begin{array}{c|c|c|c|c|c|c}
\delta^{(k-1)} & & & * & * & \cdots & * \\ \hline
 & \delta^{(k-1)} & & * & * & \cdots & * \\ \hline
 & & \delta^{(k-1)} & * & * & \cdots & * \\ \hline
 & & & \delta^{(k)} & * & \cdots & * \\ \hline
 & & & * & * & \cdots & * \\
 & & & * & * & \cdots & *
\end{array}\right)$$

Daraus erhält man durch Ausräumung der k-ten Spalte die Matrix

$$D^{(k)} = \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\dots k\\ j=1\dots m}} \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\dots n\\ j=1\dots m}} \end{pmatrix} = \left(\begin{array}{ccccc|ccc} \delta^{(k)} & & & & & * & \cdots & * \\ & \delta^{(k)} & & & & * & \cdots & * \\ & & \delta^{(k)} & & & * & \cdots & * \\ & & & \delta^{(k)} & & * & \cdots & * \\ & & & & & * & \cdots & * \\ & & & & & * & \cdots & * \end{array}\right)$$

Hierbei symbolisiert $*$ die verbleibenden Koeffizienten, die ungleich 0 sein können.

Wie üblich stellt sich nun wieder die Frage, wie man diese Matrizen rekursiv berechnen kann. Dazu betrachtet man diese Matrizenfolge allgemein für den Fall, daß schrittweise eine Zeilenstufenspalte nach der anderen ausgeräumt wird, man betrachtet also für $k = 1, \dots, n$ den Übergang von der Matrix

$$D^{(k-1)} = \begin{pmatrix} (\delta_{ij}^{(k-1)})_{\substack{i=1\dots k-1\\ j=1\dots m}} \\ (\alpha_{ij}^{(k-1)})_{\substack{i=k\dots n\\ j=1\dots m}} \end{pmatrix} =$$

$$= \begin{pmatrix} \delta_{1j_1}^{(k-1)} & \cdots & * & 0 & * \cdots * & \delta_{1j_k}^{(k-1)} & \cdots & \delta_{1j_{k+1}}^{(k-1)} & \cdots & \delta_{1m}^{(k-1)} \\ & \ddots & \vdots & \vdots & \vdots \quad \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ & & * & 0 & \vdots \quad \vdots & \vdots & & \vdots & & \vdots \\ & & & \delta_{k-1,j_{k-1}}^{(k-1)} & * \cdots * & \delta_{k-1,j_k}^{(k-1)} & \cdots & \delta_{k-1,j_{k+1}}^{(k-1)} & \cdots & \delta_{k-1,m}^{(k-1)} \\ & & & & & \alpha_{k,j_k}^{(k-1)} & \cdots & \alpha_{kj_{k+1}}^{(k-1)} & \cdots & \alpha_{k,m}^{(k-1)} \\ & & \mathbf{0} & & & \alpha_{k+1,j_k}^{(k-1)} & \cdots & \alpha_{k+1,j_{k+1}}^{(k-1)} & \cdots & \alpha_{k+1,m}^{(k-1)} \\ & & & & & \vdots & \ddots & \vdots & \ddots & \vdots \\ & & & & & \alpha_{nj_k}^{(k-1)} & \cdots & \alpha_{n,j_{k+1}}^{(k-1)} & \cdots & \alpha_{nm}^{(k-1)} \end{pmatrix}$$

zur direkten Nachfolgerin

$$D^{(k)} = \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\ldots k\\ j=1\ldots m}} \\ \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\ldots n\\ j=1\ldots m}} \end{pmatrix} =$$

$$= \begin{pmatrix} \delta_{1j_1}^{(k)} & \cdots & * & 0 & * \cdots * & 0 & * \cdots * & \delta_{1j_{k+1}}^{(k)} & \cdots & \delta_{1m}^{(k)} \\ & \ddots & \vdots & \vdots & \vdots \quad \vdots & \vdots & \vdots \quad \vdots & \vdots & & \vdots \\ & & * & 0 & \vdots \quad \vdots & \vdots & \vdots \quad \vdots & \vdots & \ddots & \vdots \\ & & & \delta_{k-1,j_{k-1}}^{(k)} & * \cdots * & 0 & \vdots \quad \vdots & \vdots & & \vdots \\ & & & & & \delta_{k,j_k}^{(k)} & * \cdots * & \delta_{kj_{k+1}}^{(k)} & \cdots & \delta_{k,m}^{(k)} \\ & & \mathbf{0} & & & & & \alpha_{k+1,j_{k+1}}^{(k)} & \cdots & \alpha_{k+1,m}^{(k)} \\ & & & & & & & \vdots & \ddots & \vdots \\ & & & & & & & \alpha_{n,j_{k+1}}^{(k)} & \cdots & \alpha_{nm}^{(k)} \end{pmatrix}$$

in der verglichen mit vorher die j_k-te Spalte über und unter der Diagonalen ausgeräumt ist. Dabei gilt in $D^{(k-1)}$ für die oberen $(k-1)$ Zeilenstufenkoeffizienten

$$\delta_{1j_1}^{(k-1)} = \ldots = \delta_{k-1,j_{k-1}}^{(k-1)} = \delta^{(k-1)} \neq 0$$

und in $D^{(k)}$ für die oberen k Zeilenstufenkoeffizienten

$$\delta_{1j_1}^{(k)} = \ldots = \delta_{kj_k}^{(k)} = \delta^{(k)} = \alpha_{kj_k}^{(k-1)} \neq 0.$$

5.4 Lemma (1-Schritt-Rekursion zur Diagonalisierung)
Die Matrizen A und $D^{(k-1)}$ seien für $k = 1, \ldots, n$ wie oben. Außerdem sei

$$t := d_{k-1,j_{k-1}}^{(k-1)} \ (\neq 0 \ \textit{stets}).$$

Die Matrix $E = (e_{ij}) \in \mathrm{Mat}_{n\times m}(R)$ werde folgendermaßen definiert:

(1) Die Zeile k ist genau wie in $D^{(k-1)}$.

(2) Die oberen Zeilen $i = 1, \ldots, k-1$ und die unteren Zeilen $i = k+1, \ldots, n$ definiert man für $j = 1, \ldots, m$ durch

$$e_{ij} := \frac{1}{t} \cdot [k\ i][j_k\ j](D^{(k-1)}).$$

Dann gilt $E = D^{(k)}$.

Beweis:
Die Behauptung ist nur für die oberen Zeilen $1, \ldots, k-1$ von $D^{(k)}$ nachzuweisen, denn für die k-te Zeile ist sie klar nach Bemerkung 5.2, und für die unteren Zeilen folgt die Behauptung aus der 1-Schritt-Rekursion zur Trigonalisierung (Lemma 3.4).

Um in $D^{(k-1)}$ in den oberen Zeilen $i = 1, \ldots, k-1$ die Spalte j_k mit Hilfe der Zeile k auszuräumen und dabei eine mit $(\widetilde{d}_{ij})_{j=1\ldots m}$ bezeichnete Zeile zu erhalten, kann man folgende elementare Zeilenumformung vornehmen: für $j = 1, \ldots, m$ berechnet man

$$\widetilde{d}_{ij} := [k\ i][j_k\ j](D^{(k-1)}) = \begin{vmatrix} \alpha_{kj_k}^{(k-1)} & \alpha_{kj}^{(k-1)} \\ \delta_{ij_k}^{(k-1)} & \delta_{ij}^{(k-1)} \end{vmatrix} = \delta^{(k)}\delta_{ij}^{(k-1)} - \delta_{ij_k}^{(k-1)}\alpha_{kj}^{(k-1)}. \qquad (*)$$

In dieser neuberechneten Zeile i sind die Spalten $j_1, \ldots, \widehat{j_i}, \ldots, j_k$ ausgeräumt, und zwar (rekursiv) unter Verwendung der Zeilen $1, \ldots, k$. Insbesondere gilt für den Zeilenstufenkoeffizienten

$$\widetilde{d}_{ij_i} = \delta^{(k)}\delta^{(k-1)} - \delta_{ij_k}^{(k-1)} \cdot 0 \quad (\neq 0). \qquad (**)$$

Andererseits sind auch in $D^{(k)}$ in den Zeilen $i = 1, \ldots, k-1$ die Spalten $j_1, \ldots, \widehat{j_i}, \ldots, j_k$ ausgeräumt, ebenfalls mit Hilfe der Zeilen $1, \ldots, k$. Es gilt für $j = 1, \ldots, m$

$$d_{ij}^{(k)} = \delta_{ij}^{(k)}, \qquad (***)$$

also gilt insbesondere für den Zeilenstufenkoeffizienten

$$d_{ij_i}^{(k)} = \delta_{ij_i}^{(k)} = \delta^{(k)}. \qquad (****)$$

Ein Koeffizientenvergleich von $(**)$ und $(****)$ ergibt

$$d_{ij_i}^{(k)} = \frac{1}{\delta^{(k-1)}} \cdot \widetilde{d}_{ij_i},$$

wobei $\delta^{(k-1)} \neq 0$ gilt. Da die durch $(*)$ und $(***)$ gegebenen Zeilen sich nur durch ein Vielfaches voneinander unterscheiden können, folgt insgesamt für alle $i = 1, \ldots, k-1$ und alle $j = 1, \ldots, m$

$$d_{ij}^{(k)} = \frac{1}{\delta^{(k-1)}} \cdot \widetilde{d}_{ij} = \frac{1}{t} \cdot [k\ i][k\ j](D^{(k-1)}),$$

also die Behauptung. □

Es zeigt sich in Lemma 5.4, daß man für beliebiges $k \in \{1, \ldots, n\}$ und $j \in \{1, \ldots, m\}$ zur Berechnung aller Koeffizienten $d_{ij}^{(k)}$, d.h. nicht nur zur Berechnung der Determinanten $\alpha_{ij}^{(k)}$ in den unteren Zeilen $i = k+1, \ldots, n$, sondern auch zur Berechnung der Determinanten $\delta_{ij}^{(k)}$ in den oberen Zeilen $i = 1, \ldots, k-1$, die bereits bekannte 1-Schritt-Rekursionsformel verwenden darf, vgl. dazu Lemma 3.5 (1-Schritt-Rekursion zur Trigonalisierung). Induktiv folgt daraus für $s \in \{1, \ldots k\}$, daß zur Berechnung von $\delta_{ij}^{(k)}$, $i = 1, \ldots, k-s$, die bekannte s-Schritt-Rekursionsformel für den Übergang von $D^{(k-s)}$ zu $D^{(k)}$ gilt. Analog zu den Rekursionformeln für Determinanten vom Typ $\alpha_{ij}^{(k)}$ (Bemerkung 3.3) erhält man daher Rekursionsformeln zur Berechnung der Determinanten $\delta_{ij}^{(k)}$, die hier vollständigkeitshalber angegeben werden. (Beachte, daß dazu für $i \in \{1, \ldots, l\}$ in den Formeln aus 3.3 anstelle von $\alpha_{i.}^{(\cdot)}$ stets $\delta_{i.}^{(\cdot)}$ und insbesondere anstelle des Divisors $\alpha_{lj_l}^{(l-1)}$ stets $\delta_{ij_i}^{(l)} = \delta^{(l)} \neq 0$ eingesetzt wird).

5.5 Korollar (allgemeine Rekursionsformeln)
Sei A eine $n \times m$-Matrix mit den Eigenschaften (V1) bis (V3).

a) (*Rekursionsformeln*) Seien $k \in \{1,\ldots,n\}$ und $l \in \{0,\ldots,k-1\}$ beliebig. Für $i \in \{1,\ldots,l\}$ und $j \in \{1,\ldots,m\}$ gilt:

$$\delta_{ij}^{(k)} = \frac{1}{(\delta^{(l)})^{k-l}} \begin{vmatrix} \alpha_{l+1,j_{l+1}}^{(l)} & \ldots & \alpha_{l+1,j_k}^{(l)} & \alpha_{l+1,j}^{(l)} \\ \vdots & \ddots & \vdots & \vdots \\ \alpha_{k,j_{l+1}}^{(l)} & \ldots & \alpha_{k,j_k}^{(l)} & \alpha_{k,j}^{(l)} \\ \delta_{i,j_{l+1}}^{(l)} & \ldots & \delta_{i,j_k}^{(l)} & \delta_{i,j}^{(l)} \end{vmatrix} =$$

$$= \frac{1}{(\delta^{(l)})^{k-l}} [l+1 \ldots k \; i][j_{l+1} \ldots j_k \; j](D^{(l)}).$$

Insbesondere gilt für $l = k-1$ folgende **1-Schritt-Formel**

$$\delta_{ij}^{(k)} = \frac{1}{\delta^{(k-1)}} \begin{vmatrix} \alpha_{kj_k}^{(k-1)} & \alpha_{kj}^{(k-1)} \\ \delta_{ij_k}^{(k-1)} & \delta_{ij}^{(k-1)} \end{vmatrix} = \frac{1}{\delta^{(k-1)}} [k \; i][j_k \; j](D^{(k-1)})$$

und für $l = k-2$ folgende **2-Schritt-Formel**

$$\delta_{ij}^{(k)} = \frac{1}{(\delta^{(k-2)})^2} \begin{vmatrix} \alpha_{k-1,j_{k-1}}^{(k-2)} & \alpha_{k-1,j_k}^{(k-2)} & \alpha_{k-1,j}^{(k-2)} \\ \alpha_{k,j_{k-1}}^{(k-2)} & \alpha_{k,j_k}^{(k-2)} & \alpha_{k,j}^{(k-2)} \\ \delta_{i,j_{k-1}}^{(k-2)} & \delta_{i,j_k}^{(k-2)} & \delta_{i,j}^{(k-2)} \end{vmatrix} =$$

$$= \frac{1}{(\delta^{(k-2)})^2} [k-1 \; k \; i][j_{k-1} \; j_k \; j](D^{(k-2)}).$$

Entsprechend erhält man für $l = k-s$ eine **s-Schritt-Formel**, welche die Berechnung einer Determinante der Ordnung $(s+1)$ erforderlich macht ($s \in \{1,\ldots,k\}$).

b) $(\delta^{(l)})^{k-l}$ ist ein Teiler der Determinante auf der rechten Seite ganz oben in a).

c) *(Teilbarkeit der Kofaktoren)* Jeder Kofaktor der rechten, durch $(\delta^{(l)})^{k-l}$ teilbaren Determinante ganz oben in a) ist durch $(\delta^{(l)})^{k-l-1}$ teilbar.

5.6 Satz (Algorithmus B1DMD)
Sei A eine beliebige $n \times m$-Matrix über einem Integritätsring R. Der Unteralgorithmus `Pivot` *sei wie in Satz 3.7,* `B1DMDRow` *sei wie nachfolgend definiert. Betrachte die folgenden Instruktionen:*

(1) Setze $k := 1$, $\lambda := 1$ und $t := 1$.

(2) (Evtl. Zeilentausch) Falls $\lambda > m$ oder $k > n$ ist, gib das Ergebnis A aus. Andernfalls ersetze A durch das Ergebnis von `Pivot`(A, λ, k).

(3) (Lange Zeilenstufe) Ist $a_{k\lambda} = 0$, dann erhöhe λ um 1 und fahre fort bei Schritt (2).

(4) (Spalte j_k ausräumen) Setze $j_k := \lambda$. Ersetze dann alle Zeilen i von A außer der k-ten jeweils durch das Ergebnis von `B1DMDRow`(A, i, j_k, k, t). *Setze $t := a_{kj_k}$, erhöhe k und λ jeweils um 1 und fahre bei Schritt (2) fort.*

Dies ist ein mit `B1DMD`(A) *bezeichneter Algorithmus, der als Ergebnis eine zu A äquivalente Matrix in Diagonalform ausgibt.*

Die Neuberechnung der i-ten Zeile von A erfolgt dabei in Schritt (4) mit dem Algorithmus `B1DMDRow`(A, i, j_k, k, t)*, d.h. es wird in der i-ten Zeile mit Hilfe der Zeile k die Spalte* j_k *ausgeräumt.*

(R1) (Berechnen) Für $j = j_k+1, \ldots, m$ *ersetze* a_{ij} *durch* $[k\ i][j_k\ j](A) = a_{kj_k} \cdot a_{ij} - a_{ij_k} \cdot a_{kj}$*. Ersetze dann den neuen Koeffizienten* a_{ij} *durch* a_{ij}/t*.*

(R2) (Ausräumen) Setze $a_{ij_k} := 0$*.*

(R3) (Über k-Zeile zusätzlich berechnen) Falls $i < k$ *ist, dann ersetze für alle Spalten* $j \in \{j_i + 1, \ldots, j_k - 1\}$ *erst* a_{ij} *durch* $[k\ i][j_k\ j](A) = a_{kj_k} \cdot a_{ij}$*. Ersetze dann den neuen Koeffizienten* a_{ij} *durch* a_{ij}/t*. Setze den Zeilenstufenkoeffizienten* $a_{ij_i} := a_{kj_k}$*.*

(R4) Gib das Ergebnis $(a_{ij})_{j=1\ldots m}$ *aus, d.h. die i-te Zeile von A.*

Beweis:
Die Instruktionen unterscheiden sich von denen des Algorithmus `B1TMD` (Satz 3.7) nur durch die Verwendung des Unteralgorithmus `B1DMDRow` in Schritt (4). Letzterer ergibt sich als Anwendung der 1-Schritt-Rekursion (Lemma 5.4), insbesondere wird in Schritt (R3) aufgrund von Lemma 5.3b) der Zeilenstufenkoeffizient $a_{ij_i} = a_{kj_k}$ gesetzt. Die Behauptung folgt ansonsten analog zu Satz 3.7 (`B1TMD`). □

5.7 Bemerkung (Komplexität von B1DMD)
Sei $A \in \mathrm{Mat}_{n\times m}(R)$ eine beliebige $n \times m$-Matrix. Außerdem sei der Rang von A maximal mit $j_k = k$ für $k = 1, \ldots, \operatorname{rang} A$.
Bei jeder Wiederholung von Schritt (4) beim Algorithmus `B1DMD` aus Satz 5.6 ergeben sich

$(n-1)(m-k)$ Additionen,

$2(n-1)(m-k)$ Multiplikationen,

$(n-1)(m-k)$ Divisionen.

Ist A eine $n \times m$-Matrix mit $n \leq m$ und $\operatorname{rang} A = n$, folgen insgesamt also $(k = 1, \ldots, n)$

$\approx mn^2 - \frac{1}{2}n^3$ Additionen,

$\approx 2mn^2 - n^3$ Multiplikationen,

$\approx mn^2 - \frac{1}{2}n^3$ Divisionen.

Bezeichnet A dagegen eine $n \times m$-Matrix mit $n > m$ und $\operatorname{rang} A = m$, so ergeben sich insgesamt $(k = 1, \ldots, m)$

$\approx \frac{1}{2}m^2 n$ Additionen,

$\approx m^2 n$ Multiplikationen,

$\approx \frac{1}{2}m^2 n$ Divisionen.

5.8 Bemerkungen (Algorithmus B1DOD und Gaußsches Eliminationsverfahren)
Sei A eine beliebige $n \times m$-Matrix über einem Integritätsring R.

a) *(Algorithmus* `B1DOD`*)* Verzichtet man beim Algorithmus `B1DMD` aus Satz 5.6 auf die Division, d.h. läßt man den Teiler t weg und ersetzt man in Schritt (4) den Algorithmus `B1DMDRow`(A, i, j_k, k, t) durch `B1DODRow`(A, i, j_k, k), so erhält man den Algorithmus `B1DOD`A, dessen Ergebnis ebenfalls eine zu A äquivalente Matrix in Diagonalform darstellt. Dabei ist `B1DODRow` genau wie `B1DMDRow` definiert, nur daß in Schritt (R1) die Division durch t entfällt und (R3) ersetzt wird durch den folgenden Schritt
(R3') (Über k-Zeile zusätzlich berechnen) Falls $i < k$ ist, dann ersetze a_{ij} durch
$[k\ i][j_k\ j](A) = a_{kj_k} \cdot a_{ij}$ für $j = 1, \ldots, j_k - 1$.

b) Der in a) beschriebene Algorithmus `B1DOD` entspricht dem allgemein bekannten *Gaußschen Eliminationsverfahren.*

c) *(Komplexität von* `B1DOD`*)* Man erhält bis auf die Divisionen dieselbe Komplexität, wie in Bemerkung 5.7 für den Algorithmus `B1DMD` ermittelt worden ist.

5.9 Bemerkung zur Implementierung (B1DMD)
Betrachte den Algorithmus `B1DMD` aus Satz 5.6.
(Zuweisungen) Aus Lemma 5.3 b) weiß man, daß die oberen Zeilenstufenkoeffizienten alle gleich a_{kj_k} sind. Daher wird bei dem Unteralgorithmus `B1DMDRow` in Schritt (R3) die entsprechende Zuweisung vorgenommen. Würde man dies nicht tun und jedesmal auch den Zeilenstufenkoeffizienten a_{ij_i} berechnen, hätte man am Ende schlimmstenfalls n-mal dieselbe Determinante der Ordnung n ausgerechnet.

6 Diagonalisierung mit dem 2-Schritt-Verfahren

Genau wie im vorigen Abschnitt erfülle die Matrix A aus $\mathrm{Mat}_{n\times m}(R)$ die Voraussetzungen
(V1) $n \le m$,
(V2) $\operatorname{rang} A = n$ und
(V3) Zeilentausch sei unnötig.

Man betrachtet wieder dieselbe Folge von Matrizen $D^{(k)}$, $k = 0, \ldots, n$, wie in Abschnitt 5, also die Matrizen

$$D^{(k)} = \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\ldots k\\ j=1\ldots m}} \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\ldots n\\ j=1\ldots m}} \end{pmatrix}.$$

Außerdem sei wieder

$$d_{0j_0}^{(0)} := 1.$$

Anhand dieser Folge wird nun das 2-Schritt-Verfahren zur Diagonalisierung untersucht, d.h. bei jedem Rekursionsschritt werden jetzt zwei Spalten auf einmal ausgeräumt. Ein Schema für den vereinfachten Fall einer $6 \times m$-Matrix, deren Zeilenstufen alle auf der Diagonalen liegen, veranschaulicht die Reihenfolge der Ausräumung:

$$\left(\begin{array}{c|c|c|c|c} * & 1 & 2 & 3 & *\cdots* \\ \hline & * & & & *\cdots* \\ \hline & & * & & *\cdots* \\ \hline & & & * & *\cdots* \\ \hline & & & * & *\cdots* \\ \hline & & & * & *\cdots* \\ \hline & 1 & 2 & 3 & *\cdots* \end{array}\right)$$

Hierbei symbolisiert $*$ die verbleibenden Koeffizienten. Am Ende steht auf der gesamten Diagonalen die Determinante $\delta^{(6)}$.

Ein Rekursionsschritt läuft folgendermaßen ab: ausgehend beispielsweise von

$$D^{(2)} = \begin{pmatrix} (\delta_{ij}^{(2)})_{\substack{i=1\ldots 2\\ j=1\ldots m}} \\ (\alpha_{ij}^{(2)})_{\substack{i=3\ldots 6\\ j=1\ldots m}} \end{pmatrix} = \left(\begin{array}{cc|cc|ccc|c} \delta^{(2)} & & * & * & * & \cdots & \cdots & * \\ \hline & \delta^{(2)} & * & * & * & \cdots & \cdots & * \\ \hline & & \delta^{(3)} & * & * & \cdots & \cdots & * \\ & & * & * & * & \cdots & \cdots & * \\ \hline & & * & * & * & \cdots & \cdots & * \\ & & * & * & * & \cdots & \cdots & * \end{array}\right)$$

werden zunächst nur die beiden oberen und unteren Zeilen umgeformt, so daß die folgende Matrix entsteht, in der in den betreffenden Zeilen die Spalten 3 und 4 ausgeräumt sind:

$$\begin{pmatrix} (\delta_{ij}^{(4)})_{\substack{i=1\dots2\\j=1\dots m}} \\ (\alpha_{ij}^{(2)})_{\substack{i=3\dots4\\j=1\dots m}} \\ (\alpha_{ij}^{(4)})_{\substack{i=5\dots6\\j=1\dots m}} \end{pmatrix} = \left(\begin{array}{cc|cc|cccc} \delta^{(4)} & & & & * & \cdots & \cdots & * \\ & \delta^{(4)} & & & * & \cdots & \cdots & * \\ \hline & & \delta^{(3)} & * & * & \cdots & \cdots & * \\ & & * & * & * & \cdots & \cdots & * \\ \hline & & & & * & \cdots & \cdots & * \\ & & & & * & \cdots & \cdots & * \end{array}\right)$$

Daraus erhält man durch Ausräumung der dritten Spalte in der vierten Zeile die Matrix

$$\begin{pmatrix} (\delta_{ij}^{(4)})_{\substack{i=1\dots2\\j=1\dots m}} \\ (\alpha_{ij}^{(2)})_{\substack{i=3\\j=1\dots m}} \\ (\delta_{ij}^{(4)})_{\substack{i=4\\j=1\dots m}} \\ (\alpha_{ij}^{(4)})_{\substack{i=5\dots6\\j=1\dots m}} \end{pmatrix} = \left(\begin{array}{cc|cc|cccc} \delta^{(4)} & & & & * & \cdots & \cdots & * \\ & \delta^{(4)} & & & * & \cdots & \cdots & * \\ \hline & & \delta^{(3)} & * & * & \cdots & \cdots & * \\ & & & \delta^{(4)} & * & \cdots & \cdots & * \\ \hline & & & & * & \cdots & \cdots & * \\ & & & & * & \cdots & \cdots & * \end{array}\right)$$

und schließlich durch Ausräumung der vierten Spalte in der dritten Zeile wie gewünscht die Matrix, in der die ersten vier Spalten ausgeräumt sind:

$$D^{(4)} = \begin{pmatrix} (\delta_{ij}^{(4)})_{\substack{i=1\dots4\\j=1\dots m}} \\ (\alpha_{ij}^{(2)})_{\substack{i=5\dots6\\j=1\dots m}} \end{pmatrix} = \left(\begin{array}{cc|cc|cccc} \delta^{(4)} & & & & * & \cdots & \cdots & * \\ & \delta^{(4)} & & & * & \cdots & \cdots & * \\ \hline & & \delta^{(4)} & & * & \cdots & \cdots & * \\ & & & \delta^{(4)} & * & \cdots & \cdots & * \\ \hline & & & & * & \cdots & \cdots & * \\ & & & & * & \cdots & \cdots & * \end{array}\right)$$

Bei dieser Matrix sind die Diagonalkoeffizienten der ersten vier Zeilen alle gleich dem Hauptminor vierter Ordnung $\delta^{(4)}$.

Allgemein gilt es nun wieder zu überlegen, wie man für gerades $k \in \{2, 4, \dots, n\}$ aus der

Matrix

$$D^{(k-2)} = \begin{pmatrix} (\delta_{ij}^{(k-2)})_{\substack{i=1\dots k-2\\ j=1\dots m}} \\ (\alpha_{ij}^{(k-2)})_{\substack{i=k-1\dots n\\ j=1\dots m}} \end{pmatrix} =$$

$$= \begin{pmatrix} \delta_{1j_1}^{(k-2)} & \cdots & * & \delta_{1j_{k-1}}^{(k-2)} & \cdots & \delta_{1j_k}^{(k-2)} & \cdots & \delta_{1j_{k+1}}^{(k-2)} & \cdots & \delta_{1m}^{(k-2)} \\ & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ & & * & \delta_{k-2,j_{k-1}}^{(k-2)} & \cdots & \delta_{k-2,j_k}^{(k-2)} & \cdots & \delta_{k-2,j_{k+1}}^{(k-2)} & \cdots & \delta_{k-2,m}^{(k-2)} \\ & & & \alpha_{k-1,j_{k-1}}^{(k-2)} & \cdots & \alpha_{k-1,j_k}^{(k-2)} & \cdots & \alpha_{k-1,j_{k+1}}^{(k-2)} & \cdots & \alpha_{k-1,m}^{(k-2)} \\ & \mathbf{0} & & \alpha_{k,j_{k-1}}^{(k-2)} & \cdots & \alpha_{k,j_k}^{(k-2)} & \cdots & \alpha_{k,j_{k+1}}^{(k-2)} & \cdots & \alpha_{k,m}^{(k-2)} \\ & & & \alpha_{k+1,j_{k-1}}^{(k-2)} & \cdots & \alpha_{k+1,j_k}^{(k-2)} & \cdots & \alpha_{k+1,j_{k+1}}^{(k-2)} & \cdots & \alpha_{k+1,m}^{(k-2)} \\ & & & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ & & & \alpha_{n,j_{k-1}}^{(k-2)} & \cdots & \alpha_{nj_k}^{(k-2)} & \cdots & \alpha_{n,j_{k+1}}^{(k-2)} & \cdots & \alpha_{nm}^{(k-2)} \end{pmatrix}$$

die übernächste Matrix

$$D^{(k)} = \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\dots k\\ j=1\dots m}} \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\dots n\\ j=1\dots m}} \end{pmatrix} =$$

$$= \begin{pmatrix} \delta_{1j_1}^{(k)} & \cdots & * & 0 & * \cdots * & 0 & * \cdots * & \delta_{1,j_{k+1}}^{(k)} & \cdots & \delta_{1m}^{(k)} \\ & \ddots & \vdots & \vdots & \vdots \quad \vdots & \vdots & \vdots \quad \vdots & \vdots & & \vdots \\ & & * & 0 & \vdots \quad \vdots & \vdots & \vdots \quad \vdots & \vdots & \ddots & \vdots \\ & & & \delta_{k-1,j_{k-1}}^{(k)} & * \cdots * & 0 & \vdots \quad \vdots & \vdots & & \vdots \\ & & & & & \delta_{k,j_k}^{(k)} & * \cdots * & \delta_{k,j_{k+1}}^{(k)} & \cdots & \delta_{k,m}^{(k)} \\ & & \mathbf{0} & & & & & \alpha_{k+1,j_{k+1}}^{(k)} & \cdots & \alpha_{k+1,m}^{(k)} \\ & & & & & & & \vdots & \ddots & \vdots \\ & & & & & & & \alpha_{n,j_{k+1}}^{(k)} & \cdots & \alpha_{nm}^{(k)} \end{pmatrix}$$

berechnen kann, in der im Vergleich zu vorher die beiden Zeilenstufenspalten j_{k-1} und j_k ober- und unterhalb der Diagonalen ausgeräumt sind. Dabei gilt in der Matrix $D^{(k-2)}$ für die oberen $(k-2)$ Zeilenstufenkoeffizienten

$$\delta_{1j_1}^{(k-2)} = \ldots = \delta_{k-2,j_{k-2}}^{(k-2)} = \delta^{(k-2)} \neq 0$$

und in der Matrix $D^{(k)}$ für die oberen k Zeilenstufenkoeffizienten

$$\delta_{1j_1}^{(k)} = \ldots = \delta_{kj_k}^{(k)} = \delta^{(k)} \neq 0.$$

6.1 Lemma (2-Schritt-Rekursion zur Diagonalisierung)
Sei die Matrix A wie eben, und sei n gerade. Betrachte in der obigen Situation für $k = 2,\ 4, \ldots, n$ die Matrix $D^{(k-2)}$ mit

$$t := d_{k-2,j_{k-2}}^{(k-2)}\ (\neq 0 \text{ stets}).$$

Die Matrizen $E = (e_{ij})$ und $\widetilde{E} = (\widetilde{e}_{ij}) \in \mathrm{Mat}_{n\times m}(R)$ seien folgendermaßen definiert:

(1) Die oberen Zeilen $i = 1, \ldots, k-2$ und die unteren Zeilen $i = k+1, \ldots, n$ sind gegeben durch

$$e_{ij} := \widetilde{e}_{ij} := \frac{1}{t^2} \cdot [k-1\ k\ i][j_{k-1}\ j_k\ j](D^{(k-2)}).$$

(2) Für die Zeile k gilt für $j = 1, \ldots, m$

$$e_{kj} := \widetilde{e}_{kj} := \frac{1}{t} \cdot [k-1\ k][j_{k-1}\ j](D^{(k-2)}).$$

(3) Für die Zeile $(k-1)$ gilt für $j = 1, \ldots, m$

$$\widetilde{e}_{k-1,j} := d_{k-1,j}^{(k-2)}$$

$$e_{k-1,j} := \frac{1}{\widetilde{e}_{k-1,j_{k-1}}} \cdot [k\ k-1][j_k\ j](\widetilde{E}).$$

Dann gilt $E = D^{(k)}$.

Beweis:
Da nach Korollar 5.5 die 2-Schritt-Rekursionsformel nicht nur für die unteren Zeilen $i = k+1, \ldots, n$, sondern auch für die oberen Zeilen $i = 1, \ldots, k-2$ gilt, ist die Behauptung für diese Zeilen klar. Für die Zeile k folgt die Behauptung aus der 1-Schritt-Rekursion wegen

$$e_{kj} = \widetilde{e}_{kj} = \frac{1}{t} \cdot [k-1\ k][j_{k-1}\ j](D^{(k-2)}) = \alpha_{k-1,j}^{(k-1)} = \delta_{k-1,j}^{(k)} = d_{k-1,j}^{(k)}.$$

Man erhält $\widetilde{E}$ also, indem man in $D^{(k-2)}$ in allen Zeilen außer der $(k-1)$-ten die Spalten j_{k-1} und j_k ausräumt. Die Matrix $\widetilde{E}$ ist bis auf die Zeile $(k-1)$, die aus $D^{(k-2)}$ stammt, genau wie $D^{(k)}$, also von folgender Form

$$\widetilde{E} = \begin{pmatrix}
\delta_{1j_1}^{(k)} & \cdots & * & 0 & * \cdots * & 0 & * \cdots * & \delta_{1,j_{k+1}}^{(k)} & \cdots & \delta_{1m}^{(k)} \\
 & \ddots & \vdots & \vdots & \vdots\ \ \vdots & \vdots & \vdots\ \ \vdots & \vdots & \ddots & \vdots \\
 & & * & 0 & \vdots\ \ \vdots & 0 & \vdots\ \ \vdots & \delta_{k-2,j_{k+1}}^{(k)} & & \delta_{k-2,m}^{(k)} \\
 & & & \alpha_{k-1,j_{k-1}}^{(k-2)} & * \cdots * & \alpha_{k-1,j_k}^{(k-2)} & * \cdots * & \alpha_{k-1,j_{k+1}}^{(k-2)} & \cdots & \alpha_{k-1,m}^{(k-2)} \\
 & & & & & \delta_{k,j_k}^{(k)} & \cdots & \delta_{k,j_{k+1}}^{(k)} & \cdots & \delta_{k,m}^{(k)} \\
 & & \mathbf{0} & & & & & \alpha_{k+1,j_{k+1}}^{(k)} & \cdots & \alpha_{k+1,m}^{(k)} \\
 & & & & & & & \vdots & \ddots & \vdots \\
 & & & & & & & \alpha_{n,j_{k+1}}^{(k)} & \cdots & \alpha_{nm}^{(k)}
\end{pmatrix}$$

Für die Zeilen $(k-1)$ und k von $\widetilde{E}$ gilt für alle Spalten $j = 1, \ldots, m$

$$\alpha_{k-1,j}^{(k-2)} = \delta_{k-1,j}^{(k-1)} \qquad \text{und} \qquad \delta_{kj}^{(k)} = \alpha_{kj}^{(k-1)}.$$

Eingesetzt in $\widetilde{E}$ folgt daraus

$$\widetilde{E} = \begin{pmatrix}
\delta_{1j_1}^{(k)} & \cdots & * & 0 & * \cdots * & 0 & * \cdots * & \delta_{1,j_{k+1}}^{(k)} & \cdots & \delta_{1m}^{(k)} \\
 & \ddots & \vdots & \vdots & \vdots \quad \vdots & \vdots & \vdots \quad \vdots & \vdots & & \vdots \\
 & & * & 0 & \vdots \quad \vdots & 0 & \vdots \quad \vdots & \delta_{k-2,j_{k+1}}^{(k)} & & \delta_{k-2,m}^{(k)} \\
 & & & \delta_{k-1,j_{k-1}}^{(k-1)} & * \cdots * & \delta_{k-1,j_k}^{(k-1)} & * \cdots * & \delta_{k-1,j_{k+1}}^{(k-1)} & \cdots & \delta_{k-1,m}^{(k-1)} \\
 & & & & & \alpha_{k,j_k}^{(k-1)} & \cdots & \alpha_{k,j_{k+1}}^{(k-1)} & \cdots & \alpha_{k,m}^{(k-1)} \\
 & & \mathbf{0} & & & & & \alpha_{k+1,j_{k+1}}^{(k)} & \cdots & \alpha_{k+1,m}^{(k)} \\
 & & & & & & & \vdots & \ddots & \vdots \\
 & & & & & & & \alpha_{n,j_{k+1}}^{(k)} & \cdots & \alpha_{nm}^{(k)}
\end{pmatrix}$$

Damit liegt für die Zeilen $(k-1)$ und k eine Situation wie in Lemma 5.4 (1-Schritt-Rekursion) vor, d.h. man darf in gewohnter Weise die 1-Schritt-Rekursionsformel (vgl. Korollar 5.5a) anwenden, um in $\widetilde{E}$ in der $(k-1)$-ten Zeile die Spalte j_k auszuräumen und damit letztendlich $E = D^{(k)}$ zu erhalten. □

6.2 Bemerkungen zur 2-Schritt-Rekursion (Determinantenentwicklung)

a) Wie man in Lemma 6.1 sieht, darf man in gewohnter Weise die bekannten Rekursionsformeln verwenden. Die zur Trigonalisierung aufgestellte Merkregel 3.4 gilt also nicht nur für die Matrizen der Form $B^{(k)}$, sondern analog auch für die Matrizen der Form $D^{(k)}$.

b) *(Determinantenentwicklung)* Sei $D := D^{(k-2)}$. Die in Lemma 6.1 (2) auftretende Determinante dritter Ordnung wird genau wie in Bemerkung 4.2 entwickelt. Damit erhält man

$$\varepsilon := [k-1\ k\ i][j_{k-1}\ j_k\ j](D) = \begin{vmatrix} d_{k-1,j_{k-1}} & d_{k-1,j_k} & d_{k-1,j} \\ d_{k,j_{k-1}} & d_{k,j_k} & d_{k,j} \\ d_{i,j_{k-1}} & d_{i,j_k} & d_{i,j} \end{vmatrix} =$$

$$= c_0 \cdot d_{ij} - c_{i1} \cdot d_{k,j} + c_{i2} \cdot d_{k-1,j}$$

mit den Faktoren

$c_0 := [k-1\ k][j_{k-1}\ j_k](D)$,

$c_{i1} := [k-1\ i][j_{k-1}\ j_k](D)$ und

$c_{i2} := [k\ i][j_{k-1}\ j_k](D)$.

Die $c_0,\ c_{i1},\ c_{i2}$ sind — bis auf das Vorzeichen von c_{i1} — die zugehörigen Kofaktoren. Wegen der Teilbarkeit der Kofaktoren gilt

$$\frac{\varepsilon}{t^2} = \frac{1}{t} \cdot \Big(\underbrace{\frac{c_0}{t}}_{\in R} \cdot d_{ij} - \underbrace{\frac{c_{i1}}{t}}_{\in R} \cdot d_{k,j} + \underbrace{\frac{c_{i2}}{t}}_{\in R} \cdot d_{k-1,j} \Big) \quad \in R.$$

c) *(Zeilenstufenkoeffizienten)* Insbesondere für den Zeilenstufenkoeffizienten der k-ten Zeile folgt aus Lemma 6.1(3) und Bemerkung b)

$$d^{(k)}_{kj_k} = c_0/t.$$

Nach Lemma 5.3b) folgt für alle $i = 1, \ldots, k$

$$d^{(k)}_{ij_i} = c_0/t,$$

d.h. die Zeilenstufenkoeffizienten der ersten k Zeilen sind alle gleich c_0/t.

6.3 Satz (Algorithmus B2DMD)
Sei A eine beliebige $n \times m$-Matrix über einem Integritätsring R. Der Unteralgorithmus `Pivot` *sei wie in Satz 3.7,* `COPivot` *wie in Satz 4.3 und* `B1DMDRow` *wie in Satz 5.6,* `B2DMDRow` *sei wie nachfolgend definiert. Betrachte die folgenden Instruktionen:*

(1) Setze $k := 1$, $\lambda := 1$ und $t := 1$.
(2) (Evtl. Zeilentausch) Falls $\lambda > m$ oder $k > n$ ist, gib das Ergebnis A aus. Andernfalls ersetze A durch das Ergebnis von `Pivot`(A, λ, k).
(3) (Lange Zeilenstufe) Ist $a_{k\lambda} = 0$, dann erhöhe λ um 1 und fahre fort bei Schritt (2).
(4) (Letzte Spalte ausräumen) Setze $j_k := \lambda$. Falls $j_k = m$ oder $k = n$ ist, dann ersetze alle Zeilen $i \in \{1, \ldots, n\} \backslash \{k\}$ von A jeweils durch das Ergebnis von `B1DMDRow`(A, i, j_k, k, t), *erhöhe k und λ jeweils um 1 und fahre bei Schritt (2) fort.*
(5) (Evtl. Zeilentausch) Setze $j_k := \lambda$. Führe `COPivot`(A, j_k, k) *aus und erhalte daraus A und c_0.*
(6) (Lange Zeilenstufe, Spalten $j_k, j_k + 1$ ausräumen) Falls $c_0 = 0$ ist, dann ersetze alle Zeilen $i \in \{1, \ldots, n\} \backslash \{k\}$ von A jeweils durch das Ergebnis von `B1DMDRow`(A, i, j_k, k, t). *Setze dann $t := a_{kj_k}$, erhöhe k um 1 und λ um 2 und fahre bei Schritt (2) fort.*
(7) (Spalten $j_k, j_k + 1$ ausräumen) Setze $j_{k+1} := j_k + 1$. Berechne dann $t_2 := t^2$ und $\delta := c_0/t$. Ersetze alle Zeilen $i \in \{1, \ldots, n\} \setminus \{k, k+1\}$ jeweils durch das Ergebnis von `B2DMDRow`$(A, i, j_k, k, c_0, t_2, \delta)$. *Ersetze die Zeile $(k+1)$ durch das Ergebnis von* `B1DMDRow`$(A, k+1, j_k, k, t)$. *Ersetze zuletzt die Zeile k durch das Ergebnis von* `B1DMDRow`$(A, k, j_k + 1, k + 1, a_{kj_k})$. *Setze $t := a_{k+1,j_k+1}$ und erhöhe k sowie λ jeweils um 2. Fahre dann bei Schritt (2) fort.*

Dies stellt einen mit `B2DMD`(A) *bezeichneten Algorithmus dar, der eine zu A äquivalente Matrix in Diagonalform ausgibt.*

In Schritt (7) bedeute `B2DMDRow`$(A, i, j_k, k, c_0, t_2, \delta)$ *den folgenden Algorithmus, welcher die i-te Zeile von A gemäß 2-Schritt-Formel berechnet, d.h. in der Zeile i mit Hilfe der Zeilen $k, k+1$ die Spalten $j_k, j_k + 1$ ausräumt.*

(R1) (Kofaktor berechnen) Berechne $c_1 := [k\ i][j_k\ j_k + 1](A) = a_{kj_k} \cdot a_{i,j_k+1} - a_{ij_k} \cdot a_{k,j_k+1}$.
(R2) (Kofaktor berechnen) Berechne dann $c_2 := [k+1\ i][j_k\ j_k + 1](A) = a_{k+1,j_k} \cdot a_{i,j_k+1} - a_{ij_k} \cdot a_{k+1,j_k+1}$.

(R3) (Determinante entwickeln) Für $j = j_k + 2, \ldots, m$ ersetze den Koeffizienten a_{ij} durch $c_0 \cdot a_{ij} - c_1 \cdot a_{k+1,j} + c_2 \cdot a_{kj}$. Ersetze dann den neuen Koeffizienten a_{ij} durch a_{ij}/t_2.

(R4) (Ausräumen) Setze $a_{ij_k} := 0$ und $a_{i,j_k+1} := 0$.

(R5) (Über k-Zeile zusätzlich) Falls $i < k$ ist, berechne für alle Nichtzeilenstufenspalten $j \in \{j_i + 1, \ldots, j_k - 1\} \setminus \{j_{i+1}, j_{i+2}, \ldots, j_{k-1}\}$ den Koeffizienten a_{ij} genau wie in Schritt (R3) neu. Setze den Zeilenstufenkoeffizienten $a_{ij_i} := \delta$.

(R6) Gib das Ergebnis $(a_{ij})_{j=1\ldots m}$ aus, d.h. die i-te Zeile von A.

Beweis:
Schritt (7) ist die direkte Anwendung der 2-Schritt-Rekursion (Lemma 6.1 und Bemerkung 6.2). Ansonsten verläuft der Beweis analog zu Satz 4.3 (`B2TMD`) und Satz 5.6 (`B1DMD`). □

6.4 Bemerkung (Komplexität von B2DMD)

Sei $A \in \mathrm{Mat}_{n\times m}(R)$ eine beliebige $n \times m$-Matrix mit maximalem Rang und mit $j_k = k$ für $k = 1, \ldots, \operatorname{rang} A$. Beim Algorithmus `B2DMD` aus Satz 6.3 benötigt man bei jedem Durchlauf von Schritt (7)

$2(n-2)(m-k) + 2(m-k+1) + 2(n-2)$ Additionen,

$3(n-2)(m-k) + 4(m-k+1) + 4(n-2)$ Multiplikationen,

$(n-2)(m-k) + 2(m-k+1)$ Divisionen.

Ist A eine $n \times m$-Matrix mit $n \le m$ und mit $\operatorname{rang} A = n$, so ergeben sich insgesamt ($k = 1, 3, \ldots, n-1$, k ungerade)

$\approx mn^2 - \frac{1}{2}n^3$ Additionen,

$\approx \frac{3}{2}mn^2 - \frac{3}{4}n^3$ Multiplikationen,

$\approx \frac{1}{2}mn^2 - \frac{1}{4}n^3$ Divisionen.

Bezeichnet A dagegen eine $n \times m$-Matrix mit $n > m$ und mit $\operatorname{rang} A = m$, so benötigt man insgesamt ($k = 1, \ldots, m-1$, k ungerade)

$\approx \frac{1}{2}m^2n$ Additionen,

$\approx \frac{3}{4}m^2n$ Multiplikationen,

$\approx \frac{1}{4}m^2n$ Divisionen.

6.5 Bemerkungen (Algorithmus B2DOD)

Sei A eine beliebige $n \times m$-Matrix über einem Integritätsring R.

a) *(Algorithmus `B2DOD`)* Bei dem Algorithmus `B2DMD` aus Satz 6.3 kann man auf die Division verzichten. Dazu läßt man die Variablen t, t_2 und δ weg; ferner wird der Unteralgorithmus `B1DMDRow` ersetzt durch `B1DODRow` aus Bemerkung 5.8 und `B2DMDRow` durch `B2DODRow`. Dabei ist `B2DODRow`(A, i, j_k, k, c_0) genau wie `B2DMDRow`$(A, i, j_k, k, c_0, t_2, \delta)$ definiert, nur daß Schritt (R3) durch (R3') und Schritt (R5) durch (R5') ersetzt wird wie folgt:

(R3') (Determinante entwickeln) Für $j = j_k + 2, \ldots, m$ ersetze den Koeffizienten a_{ij} durch $c_0 \cdot a_{ij} - c_1 \cdot a_{k+1,j} + c_2 \cdot a_{kj}$.

(R5') (Über k-Zeile zusätzlich berechnen) Falls $i < k$ ist, dann berechne den Koeffizienten a_{ij} genau wie in Schritt (R3') für alle Nichtzeilenstufenspalten $j \in \{j_i + 1, \ldots, j_k - 1\} \setminus \{j_{i+1}, j_{i+2}, \ldots, j_{k-1}\}$.

Dadurch erhält man den Algorithmus `B2DOD`(A), der als Ergebnis ebenfalls eine zu A äquivalente Matrix in Diagonalform ausgibt.

b) *(Komplexität von `B2DOD`)* Bis auf die Anzahl der Divisionen entspricht die Komplexität von `B2DOD` der von `B2DMD` aus Bemerkung 6.4.

Nutzt man bei der Berechnung der Determinanten dritter Ordnung die Teilbarkeit der Kofaktoren aus (Bemerkung 6.2b), so folgt als Variante des Algorithmus `B2DMD` aus Satz 6.3

6.6 Satz (Algorithmus B2DMDMD)
Sei A eine beliebige $n \times m$-Matrix über einem Integritätsring R. Betrachte den Algorithmus `B2DMD` *aus Satz 6.3 und nehme folgende Änderungen vor:*

Ersetze in Schritt (7) den Algorithmus `B2DMDRow`$(A, i, j_k, k, c_0, t, \delta)$ *durch den Algorithmus* `B2DMDMDRow`$(A, i, j_k, k, t, \delta)$*, welcher wie folgt definiert ist:*

(R1) (Kofaktor berechnen) Berechne $c_1 := [k\ i][j_k\ j_k+1](A) = a_{kj_k} \cdot a_{i,j_k+1} - a_{ij_k} \cdot a_{k,j_k+1}$. *Ersetze* c_1 *durch* c_1/t.

(R2) (Kofaktor berechnen) Berechne dann $c_2 := [k+1\ i][j_k\ j_k+1](A) = a_{k+1,j_k} \cdot a_{i,j_k+1} - a_{ij_k} \cdot a_{k+1,j_k+1}$. *Ersetze* c_2 *durch* c_2/t.

(R3) (Determinante entwickeln) Für $j = j_k+2, \ldots, m$ *ersetze den Koeffizienten* a_{ij} *zuerst durch* $\delta \cdot a_{ij} - c_1 \cdot a_{k+1,j} + c_2 \cdot a_{kj}$. *Ersetze den neuen Koeffizienten* a_{ij} *durch* a_{ij}/t.

(R4) (Ausräumen) Setze $a_{ij_k} := 0$ *und* $a_{i,j_k+1} := 0$.

(R5) (Über k-Zeile zusätzlich) Gilt $i < k$*, so berechne für alle Spalten* $j \in \{j_i+1, \ldots, j_k-1\} \setminus \{j_{i+1}, j_{i+2}, \ldots, j_{k-1}\}$ *den Nichtzeilenstufenkoeffizienten* a_{ij} *genau wie in Schritt (R3). Setze außerdem den Zeilenstufenkoeffizienten* $a_{ij_i} := \delta$.

(R6) Gib das Ergebnis $(a_{ij})_{j=1\ldots m}$ *aus, d.h. die i-te Zeile von A.*

Damit ist ein Algorithmus namens `B2DMDMD`(A) *gegeben, der eine zu A äqivalente Matrix in Diagonalform ausgibt. Sein Ergebnis stimmt mit dem von* `B2DMD` *überein.*

Beweis:
Die Behauptung sofort aus der Teilbarkeit der Kofaktoren. Da die Variable $\delta = c_0/t$ schon vorliegt, muß dabei für den Kofaktor c_0 die Division nicht noch mal eigens durchgeführt werden. □

6.7 Bemerkung (Komplexität von B2DMDMD)
Sei $A \in \mathrm{Mat}_{n\times m}(R)$ eine beliebige $n \times m$-Matrix. Außerdem sei der Rang von A maximal und $j_k = k$ für $k = 1, \ldots, \operatorname{rang} A$. Für den Algorithmus `B2DMDMD` aus Satz 6.6 ergeben sich bei jeder Wiederholung von Schritt (7)

$2(n-2)(m-k) + 2(m-k+1) + 2(n-2)$ Additionen,
$3(n-2)(m-k) + 4(m-k+1) + 4(n-2)$ Multiplikationen,
$(n-2)(m-k) + 2(m-k+1) + 2(n-2)$ Divisionen.

Ist A eine $n \times m$-Matrix mit $n \le m$ und mit $\operatorname{rang} A = n$, so benötigt man insgesamt ($k = 1, 3, \ldots, n-1$, k ungerade)

$\approx mn^2 - \frac{1}{2}n^3$ Additionen,
$\approx \frac{3}{2}mn^2 - \frac{3}{4}n^3$ Multiplikationen,
$\approx \frac{1}{2}mn^2 - \frac{1}{4}n^3$ Divisionen.

Bezeichnet A dagegen eine $n \times m$-Matrix mit $n > m$ und mit $\operatorname{rang} A = m$, so ergeben sich insgesamt ($k = 1, \ldots, m-1$, k ungerade)

$\approx \frac{1}{2}m^2n$ Additionen,
$\approx \frac{3}{4}m^2n$ Multiplikationen,
$\approx \frac{1}{4}m^2n$ Divisionen.

6.8 Bemerkungen zur Implementierung (B2DMD, B2DOD, B2DMDMD)
Zum Algorithmus `B2DMD` aus Satz 6.3 läßt sich sagen:

a) (*overwriting*) In Schritt (7) müssen zunächst die Zeilen $i \in \{1, \ldots, n\} \setminus \{k, k+1\}$ umgeformt werden, daraufhin die Zeile $(k+1)$ und erst ganz zuletzt die Zeile k. Für

`B2DMDRow` werden nämlich die alten Zeilen $k, k+1$ benötigt, für die Umformung der Zeile $(k+1)$ wird die alte k-te Zeile gebraucht, und schließlich für die Neuberechnung der k-ten Zeile die neue Zeile $(k+1)$.

b) *(Zuweisungen)* Die oberen $(k+1)$ Zeilenstufenkoeffizienten werden gleich δ gesetzt. Wie bereits in Bemerkung 4.8b) beschrieben, kann man auch hier wieder `B1DMDRow` so modifizieren, daß es bei einem Aufruf in Schritt (6) oder (7) den Koeffizienten a_{k+1,j_k+1} nicht berechnet, sondern den Wert 0 bzw. $\delta = c_0/t$ zuweist.

c) Wie bei `B2TMD` wird auch hier nur dann Schritt (7) ausgeführt, wenn die beiden Zeilenstufenspalten j_k und $j_{k+1} = j_k + 1$ direkt nebeneinanderliegen.

Analog gilt dies auch für die beiden Algorithmen `B2DOD` und `B2DMDMD` aus Bemerkung 6.5 bzw. aus Satz 6.6.

d) Die zur Trigonalisierung eingesetzten Zeilenprogramme (`B1TMDRow` etc.) lassen sich durch die entsprechenden Zeilenprogramme zur Diagonalisierung ersetzen, da diese eine Verallgemeinerung darstellen.

6.9 Bemerkung (Mehrschritt-Verfahren)
Wie schon bei der Trigonalisierung ließe sich auch zur Diagonalisierung ein Mehr-Schritt-Verfahren mit einer größeren Schrittspanne als 2 konstruieren, vgl. Bemerkung 4.9. Dies wird hier aber nicht mehr weiter ausgeführt.

"Old Minty," sagte er, als er seinen Vater sah, "hat uns erzählt, daß 'mathema' Wissen heißt und 'pathema' Leiden, da hab ich vorgeschlagen, man könnte das Fach doch 'Pathematik' nennen."

(Ruth Rendell, *Schuld verjährt nicht*)

Nach wie vor bezeichne R einen Integritätsring, und nach wie vor soll für die theoretischen Betrachtungen die Matrix A aus $\mathrm{Mat}_{n\times m}(R)$ die drei üblichen Voraussetzungen erfüllen:

(V1) $n \leq m$,
(V2) $\operatorname{rang} A = n$ und
(V3) Zeilentausch ist unnötig.

In diesem Kapitel werden nun weitere Methoden vorgestellt, um A zu diagonalisieren, nämlich das *Forward-Backup-Verfahren*, das *Malashonok-1-Schritt-Verfahren* und zuguterletzt das *Dichotomie-Verfahren*, vgl. [Mal1] und [Mal2]. Allgemeiner als von Malashonok dargestellt, lassen sich die Algorithmen wieder für beliebige Matrizen durchführen.

Als Grundlagen genügen auch weiterhin die in Kapitel I gewonnenen Erkenntnisse, insbesondere reichen die bisherigen Notationen aus, um die folgenden Verfahren zu beschreiben. Vor allem sollte man dabei im Hinterkopf behalten, daß die Determinanten der Form $\alpha_{ij}^{(k)}$ Koeffizienten in "trigonalisierten" Zeilen darstellen und daß Determinanten der Form $\delta_{ij}^{(k)}$ in "diagonalisierten" Zeilen auftreten. Dabei bedeutet der Index k, daß die Spalten $j_1, j_2, \ldots, j_k$ bzw. die Spalten $j_1, \ldots, \widehat{j_i}, \ldots, j_k$ ausgeräumt werden.

Bei den bisher bekannten Bareiss-Verfahren wird A diagonalisiert, indem sukzessive von links nach rechts die Spalten ausgeräumt werden, bis als Endergebnis die Diagonalmatrix

$$\left(\delta_{ij}^{(n)}\right)_{\substack{i=1\ldots n\\ j=1\ldots m}}$$

erreicht ist. Dies geschieht, indem man eine Folge von Matrizen $D^{(k)}$ konstruiert, die für $k = 0, \ldots, n$ von folgender Form sind:

$$D^{(k)} = (d_{ij}^{(k)}) = \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\ldots k\\ j=1\ldots m}} \\ \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\ldots n\\ j=1\ldots m}} \end{pmatrix}.$$

(Wenn auf Bareiss-Verfahren Bezug genommen wird, sind jetzt stets diejenigen mit Division gemeint.)

Die drei folgenden Verfahren nach Malashonok führen zwar zum selben Endergebnis, gehen bei der Ausräumung aber in anderer Reihenfolge vor. Anschaulich heißt dies, daß die Ausräumung nicht nur spaltenweise, sondern auch zeilen- oder blockweise vorgenommen wird. Für die Konstruktion der zugehörigen Matrizenfolge bedeutet dies, daß nach wie vor die Zeileneinträge der Matrizen von der Form $\alpha_{ij}^{(k)}$ oder $\delta_{ij}^{(k)}$ sind. Verglichen mit oben werden nun aber deren Reihenfolge, d.h. der Ablauf der Ausräumung, und deren Indizes k, d.h. die ausgeräumten Spalten, variiert.

Für solche Matrizenfolgen lassen sich nur zum Teil die bereits bekannten Bareiss-Rekursionsformeln weiter verwenden, zum Teil müssen neue Rekursionsformeln gefunden werden. Diese zusätzlich benötigten Rekursionsformeln basieren unter anderem auf der Multiplikation von Matrizen. Kennt man also ein gutes Verfahren zur Matrizenmultiplikation, kann so die Komplexität der Diagonalisierung noch weiter verbessert werden. Momentan ist beispielsweise ein Algorithmus zur Matrizenmultiplikation mit der Komplexität $O(n^{\log_2 7}) \approx O(n^{2,81})$ bekannt, vgl. [Str], oder sogar ein Algorithmus mit der Komplexität $O(n^{2,37})$, vgl. [CW].

Als erstes wird nun das *Forward-Backup-Verfahren* präsentiert, das in zwei Schritten durchgeführt wird. Der erste Schritt besteht darin, A mit einem der bereits bekannten Bareiss-Verfahren zu trigonalisieren (*forward*). Im zweiten Schritt werden dann von unten nach oben die Zeilen ausgeräumt (*backup*).

Für den Fall, daß A beispielsweise eine $6 \times m$-Matrix ist, deren Zeilenstufenkoeffizienten alle auf der Diagonalen liegen, werden somit die Zeilen und Spalten in der Reihenfolge der Numerierung ausgeräumt:

$$\left(\begin{array}{c|c|c|c|c|c|c}
* & & & & & 10 & * \\ \hline
 & * & & & & 9 & * \\ \hline
 & & * & & & 8 & * \\ \hline
 & & & * & & 7 & * \\ \hline
 & & & & * & 6 & * \\ \hline
1 & 2 & 3 & 4 & 5 & * & *
\end{array}\right)$$

Hierbei symbolisiert $*$ die verbleibenden Koeffizienten auf der Diagonalen bzw. die jeweils verbleibenden letzten $(m-6)$ Koeffizienten. Am Ende steht auf der gesamten Diagonalen die Determinante $\delta^{(6)}$.

7 Diagonalisierung mit dem Forward-Backup-Verfahren

Zur Vorbereitung der späteren Beweise, in denen Determinanten des unten angegebenen Typs evaluiert werden, dient folgende

7.1 Proposition
Sei $k \in \mathbb{N}_+$, und sei ferner

$$E = \left(\begin{array}{ccc|c} d & & \mathbf{0} & c_1 \\ & \ddots & & \vdots \\ \mathbf{0} & & d & c_k \\ \hline a_1 & \cdots & a_k & b \end{array}\right)$$

eine $(k+1)\times(k+1)$-Matrix über einem Integritätsring R, wobei in der $k\times k$-Teilmatrix links oben nur auf der Diagonalen $d \in R$ stehe und sonst 0. Dann gilt

$$|E| = d^{k-1}\cdot(bd - \sum_{i=1}^{k} a_i c_i) = d^{k-1}\cdot\Big[\, bd - (\, a_1 \quad \dots \quad a_k\,)\cdot\begin{pmatrix} c_1 \\ \vdots \\ c_k \end{pmatrix}\Big].$$

Beweis:
Entwickelt man $|E|$ nach der letzten Spalte, dann ergibt sich

$$|E| = b\cdot\begin{vmatrix} d & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & d \end{vmatrix} + \sum_{i=1}^{k} c_i\cdot\widetilde{c_i}, \qquad (*)$$

wobei $\widetilde{c_i}$ jeweils den Kofaktor von c_i bezeichne und für $i = 1,\dots,k$ von folgender Form ist:

$$\widetilde{c_i} = (-1)^{i+k+1}\cdot\begin{vmatrix} d & & & & & & \\ & \ddots & & & & \mathbf{0} & \\ & & d & & & & \\ & & & 0 & d & & \\ & \mathbf{0} & & & & \ddots & \\ & & & & & & d \\ a_1 & \cdots & \cdots & a_i & \cdots & \cdots & a_k \end{vmatrix}.$$

Entwickelt man $\widetilde{c_i}$ nach der Spalte i, folgt

$$\widetilde{c_i} = (-1)^{i+k+1}\cdot(-1)^{i+k}\cdot a_i\cdot d^{k-1} = -a_i\cdot d^{k-1}. \qquad (**)$$

Insgesamt ergibt sich also aus $(*)$ und $(**)$

$$|E| = bd^k - \sum_{i=1}^{k} a_i\cdot c_i\cdot d^{k-1} = d^{k-1}\ (bd - \sum_{i=1}^{k} a_i c_i)$$

wie behauptet. Die zweite Gleichheit der Behauptung ist klar. □

Nach wie vor erfülle die Matrix $A \in \mathrm{Mat}_{n\times m}(R)$ die drei Voraussetzungen
(V1) $n \le m$,
(V2) $\operatorname{rang} A = n$ und

(V3) Zeilentausch sei unnötig.

Diese Voraussetzungen dienen wie schon in Kapitel I dazu, die folgenden Überlegungen einfacher darstellen zu können. Für die Algorithmen (Satz 7.5) sind sie nicht nötig.

Wie im ersten Kapitel (Bareiss-Verfahren) wird A zunächst trigonalisiert, indem sukzessive von links nach rechts die Spalten unter der Zeilenstufendiagonalen ausgeräumt werden (daher die Bezeichnung *forward*). Als Ergebnis erhält man eine Matrix in Zeilenstufenform, nämlich die Matrix

$$(\alpha_{ij}^{(i-1)})_{\substack{i=1\dots n\\ j=1\dots m}}.$$

Anschließend werden von unten nach oben die Zeilen ausgeräumt (*backup*). Dieses letztgenannte, bisher noch unbekannte **Backup-Verfahren** wird nun näher untersucht. Dazu betrachtet man die Folge der Matrizen $F^{(k)}$, $k = n, n-1, \dots, 0$, die wie folgt definiert sind. (Hier läuft ausnahmsweise die Indexmenge absteigend von n bis 0.)

$$F^{(k)} := \begin{pmatrix} (\alpha_{ij}^{(i-1)})_{\substack{i=1\dots k\\ j=1\dots m}} \\ (\delta_{ij}^{(n)})_{\substack{i=k+1\dots n\\ j=1\dots m}} \end{pmatrix} =$$

$$= \begin{pmatrix}
\alpha_{1j_1}^{(0)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{1m}^{(0)} \\
\ddots & & & & & & & \vdots \\
 & \alpha_{kj_k}^{(k-1)} * & \alpha_{kj_{k+1}}^{(k-1)} * & \alpha_{kj_{k+2}}^{(k-1)} & * \cdots * & \alpha_{kj_{n-1}}^{(k-1)} * & \alpha_{kj_n}^{(k-1)} * & \alpha_{km}^{(k-1)} \\
 & & \delta_{k+1,j_{k+1}}^{(n)} * & 0 & * \cdots * & 0 * & 0 * & \delta_{k+1,m}^{(n)} \\
 & & & \delta_{k+2,j_{k+2}}^{(n)} & * \cdots * & 0 * & 0 * & \delta_{k+2,m}^{(n)} \\
 & \mathbf{0} & & & \ddots & \vdots\ \vdots & \vdots\ \vdots & \vdots \\
 & & & & * & 0 * & 0 * & \delta_{n-2,m}^{(n)} \\
 & & & & & \delta_{n-1,j_{n-1}}^{(n)} * & 0 * & \delta_{n-1,m}^{(n)} \\
 & & & & & & \delta_{n,j_n}^{(n)} * & \delta_{n,m}^{(n)}
\end{pmatrix}$$

Aus Platzgründen symbolisiert $*$ jeweils die dazwischenliegenden Koeffizienten, die ungleich 0 sein können. Die Matrix $F^{(k)}$ ist in den ersten k Zeilen schon trigonalisiert, aber noch nicht diagonalisiert, und in den unteren Zeilen $k+1, \dots, n$ bereits fertig diagonalisiert. Insbesondere gilt für die erste bzw. letzte Matrix der Folge $(F^{(k)})_{k=n\dots 0}$

$$F^{(n)} = \left(\alpha_{ij}^{(i-1)}\right)_{\substack{i=1\dots n\\ j=1\dots m}} = \begin{pmatrix} (\alpha_{ij}^{(i-1)})_{\substack{i=1\dots n-1\\ j=1\dots m}} \\ (\delta_{ij}^{(n)})_{\substack{i=n\\ j=1\dots m}} \end{pmatrix} = F^{(n-1)}$$

$$F^{(0)} = \left(\delta_{ij}^{(n)}\right)_{\substack{i=1\dots n\\ j=1\dots m}}.$$

Damit bilden die Matrizen $F^{(k)}$ für $k = n, \dots, 0$ eine Folge, wie man sie für das Backup-Verfahren benötigt, um ausgehend von der Zeilenstufenmatrix $F^{(n)}$ sukzessive von unten nach oben die Zeilen auszuräumen und somit die Diagonalmatrix $F^{(0)}$ herzustellen.

Für die schematisch dargestellte $6\times m$-Matrix von vorhin sieht der Übergang von einer Matrix $F^{(k)}$ zur nachfolgenden Matrix $F^{(k-1)}$, $k=n-1,n-2,\ldots,1$ folgendermaßen aus:

$$F^{(k)}=\begin{pmatrix}(\alpha_{ij}^{(i-1)})_{\substack{i=1\ldots k\\ j=1\ldots m}}\\ (\delta_{ij}^{(n)})_{\substack{i=k+1\ldots n\\ j=1\ldots m}}\end{pmatrix}=\left(\begin{array}{c|c|c|c|c|c|c}
* & * & \cdots & \cdots & \cdots & * & * \\ \hline
 & \delta^{(k-1)} & * & \cdots & \cdots & * & * \\ \hline
 & & \delta^{(k)} & * & \cdots & * & * \\ \hline
 & & & \delta^{(n)} & & & * \\ \hline
 & & & & \delta^{(n)} & & * \\ \hline
 & & & & & \delta^{(n)} & *
\end{array}\right)$$

Daraus erhält man durch Ausräumen der k-ten Zeile die Matrix

$$F^{(k-1)}=\begin{pmatrix}(\alpha_{ij}^{(i-1)})_{\substack{i=1\ldots k-1\\ j=1\ldots m}}\\ (\delta_{ij}^{(n)})_{\substack{i=k\ldots n\\ j=1\ldots m}}\end{pmatrix}=\left(\begin{array}{c|c|c|c|c|c|c}
* & * & \cdots & \cdots & \cdots & * & * \\ \hline
 & \delta^{(k-1)} & * & \cdots & \cdots & * & * \\ \hline
 & & \delta^{(n)} & & & & * \\ \hline
 & & & \delta^{(n)} & & & * \\ \hline
 & & & & \delta^{(n)} & & * \\ \hline
 & & & & & \delta^{(n)} & *
\end{array}\right)$$

Allgemein gilt es nun wieder zu überlegen, wie man diese Matrizen rekursiv berechnen kann. Dazu betrachtet man für $k=n-1,n-2,\ldots,1$ die Matrix

$$F^{(k)}=\begin{pmatrix}(\alpha_{ij}^{(i-1)})_{\substack{i=1\ldots k\\ j=1\ldots m}}\\ (\delta_{ij}^{(n)})_{\substack{i=k+1\ldots n\\ j=1\ldots m}}\end{pmatrix}=$$

$$=\begin{pmatrix}
\alpha_{1j_1}^{(0)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{1m}^{(0)}\\
\ddots & & & & & & & \vdots\\
\alpha_{k-1,j_{k-1}}^{(k-2)} & * & \alpha_{k-1,j_k}^{(k-2)} & * & \alpha_{k-1,j_{k+1}}^{(k-2)} & * \cdots * & \alpha_{k-1,j_{n-1}}^{(k-2)} * & \alpha_{k-1,j_n}^{(k-2)} * & \alpha_{k-1,m}^{(k-2)}\\
 & & \alpha_{k,j_k}^{(k-1)} & * & \alpha_{k,j_{k+1}}^{(k-1)} & * \cdots * & \alpha_{k,j_{n-1}}^{(k-1)} * & \alpha_{k,j_n}^{(k-1)} * & \alpha_{k,m}^{(k-1)}\\
 & & & & \delta_{k+1,j_{k+1}}^{(n)} & * \cdots * & 0 * & 0 * & \delta_{k+1,m}^{(n)}\\
 & \mathbf{0} & & & & \ddots & \vdots\ \vdots & \vdots\ \vdots & \vdots\\
 & & & & & * & 0\ \vdots & \vdots\ \vdots & \vdots\\
 & & & & & & \delta_{n-1,j_{n-1}}^{(n)} * & 0 * & \delta_{n-1,m}^{(n)}\\
 & & & & & & & \delta_{n,j_n}^{(n)} * & \delta_{n,m}^{(n)}
\end{pmatrix}$$

wobei für die Zeilenstufenkoeffizienten der unteren Zeilen $k+1,\ldots,n$

$$\delta^{(n)}_{k+1,j_{k+1}} = \ldots = \delta^{(n)}_{n-1,j_{n-1}} = \delta^{(n)}_{nj_n} = \delta^{(n)} \neq 0$$

gilt und für den k-ten Zeilenstufenkoeffizienten $\alpha^{(k-1)}_{kj_k} = \delta^{(k)}$. Die direkt nachfolgende Matrix ist von der Form

$$F^{(k-1)} = \begin{pmatrix} (\alpha^{(i-1)}_{ij})_{\substack{i=1\ldots k-1\\ j=1\ldots m}} \\ (\delta^{(n)}_{ij})_{\substack{i=k\ldots n\\ j=1\ldots m}} \end{pmatrix} =$$

$$= \begin{pmatrix} \alpha^{(0)}_{1j_1} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha^{(0)}_{1m} \\ \ddots & & & & & & & \vdots \\ & \alpha^{(k-2)}_{k-1,j_{k-1}} * & \alpha^{(k-2)}_{k-1,j_k} * & \alpha^{(k-2)}_{k-1,j_{k+1}} & * \cdots * & \alpha^{(k-2)}_{k-1,j_{n-1}} * & \alpha^{(k-2)}_{k-1,j_n} * & \alpha^{(k-2)}_{k-1,m} \\ & & \delta^{(n)}_{k,j_k} * & 0 & * \cdots * & 0 * & 0 * & \delta^{(n)}_{k,m} \\ & & & \delta^{(n)}_{k+1,j_{k+1}} & * \cdots * & 0 * & 0 * & \delta^{(n)}_{k+1,m} \\ & \mathbf{0} & & & \ddots & \vdots\ \vdots & \vdots\ \vdots & \vdots \\ & & & & * & 0 \ \vdots & \vdots\ \vdots & \vdots \\ & & & & & \delta^{(n)}_{n-1,j_{n-1}} * & 0 * & \delta^{(n)}_{n-1,m} \\ & & & & & & \delta^{(n)}_{n,j_n} * & \delta^{(n)}_{n,m} \end{pmatrix}$$

Hier gilt für die Zeilenstufenkoeffizienten der unteren Zeilen $k,\ldots,n$

$$\delta^{(n)}_{kj_k} = \delta^{(n)}_{k+1,j_{k+1}} = \ldots = \delta^{(n)}_{n-1,j_{n-1}} = \delta^{(n)}_{nj_n} = \delta^{(n)} \neq 0.$$

7.2 Lemma (Backup-Rekursion)
Sei A eine $n \times m$-Matrix mit den Eigenschaften (V1) bis (V3). Betrachte in der obigen Situation für $k = n-1, n-2, \ldots, 1$ die Matrix $F = (f_{ij}) := F^{(k)}$, in der der k-te Zeilenstufenkoeffizient bezeichnet wird mit

$$t := f_{kj_k} \ (\neq 0 \ \textit{stets}).$$

Die Matrix $E = (e_{ij}) \in \mathrm{Mat}_{n\times m}(R)$ sei wie folgt definiert:
(1) Die Zeile k ist für $j = 1,\ldots,m$ gegeben durch

$$e_{kj} := \frac{1}{t} \cdot \Big[\, f_{nj_n} f_{kj} \ - \ (\, f_{kj_{k+1}} \quad \ldots \quad f_{kj_{n-1}} \quad f_{kj_n} \,) \cdot \begin{pmatrix} f_{k+1,j} \\ \vdots \\ f_{n-1,j} \\ f_{nj} \end{pmatrix} \Big].$$

(2) Alle anderen Zeilen von E sind genau wie in F.
Dann gilt $E = F^{(k-1)}$.

Beweis:
Da die Matrizen $F = F^{(k)}$ und $F^{(k-1)}$ bis auf die k-te Zeile identisch sind, ist die Behauptung nur für die Zeile k nachzuweisen. Um in F in der k-ten Zeile die Zeilenstufenspalten $j_{k+1}, \ldots, j_{n-1}, j_n$ mit Hilfe der Zeilen $k+1, \ldots, n$ auszuräumen, berechnet man mittels Proposition 7.1 für $j = 1, \ldots, m$ folgende Determinante:

$$g_{kj} := [k+1 \ldots n \ k][j_{k+1} \ldots j_{n-1} \ j_n \ j](F) =$$

$$= \begin{vmatrix} \delta^{(n)} & & & \mathbf{0} & f_{k+1,j} \\ & \ddots & & & \vdots \\ & & \ddots & & f_{n-1,j} \\ \mathbf{0} & & & \delta^{(n)} & f_{nj} \\ f_{kj_{k+1}} & \cdots & f_{kj_{n-1}} & f_{kj_n} & f_{kj} \end{vmatrix} =$$

$$= (\delta^{(n)})^{n-k-1} \cdot \Big[\ \delta^{(n)} \ f_{kj} \ - \ (\ f_{kj_{k+1}} \quad \cdots \quad f_{kj_n} \ ) \begin{pmatrix} f_{k+1,j} \\ \vdots \\ f_{nj} \end{pmatrix} \Big].$$

Zur Vereinfachung setzt man mit $\delta^{(n)} \neq 0$

$$\widetilde{g}_{kj} := \frac{1}{(\delta^{(n)})^{n-k-1}} \cdot g_{kj} = \delta^{(n)} \ f_{kj} \ - \ (\ f_{kj_{k+1}} \quad \cdots \quad f_{kj_n} \ ) \begin{pmatrix} f_{k+1j} \\ \vdots \\ f_{nj} \end{pmatrix}. \qquad (*)$$

Die neu berechnete k-te Zeile von F wird mit $(\widetilde{g}_{kj})_{j=1\ldots m}$ bezeichnet. In ihr sind die Spalten $j_1, \ldots, \widehat{j_k} \ldots, j_{n-1}, j_n$ ausgeräumt, und zwar unter Verwendung der Zeilen $1, \ldots, n$. Insbesondere für den Zeilenstufenkoeffizienten der Zeile k folgt daraus mit $\delta^{(n)}, \ t \neq 0$

$$\widetilde{g}_{kj_k} = \delta^{(n)} \cdot f_{kj_k} \ - \ (\ f_{kj_{k+1}} \quad \cdots \quad f_{kj_n} \ ) \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} = \delta^{(n)} \cdot t \quad (\neq 0). \qquad (**)$$

Andererseits sind auch in $F^{(k-1)}$ in der Zeile k die Spalten $j_1, \ldots, \widehat{j_k}, \ldots, j_n$ ausgeräumt, ebenfalls mit Hilfe der Zeilen $1, \ldots, n$. Definitionsgemäß gilt für $j = 1, \ldots, m$

$$f_{kj}^{(k-1)} = \delta_{kj}^{(n)}. \qquad (***)$$

Insbesondere für den Zeilenstufenkoeffizienten gilt

$$f_{kj_k}^{(k-1)} = \delta_{kj_k}^{(n)} = \delta^{(n)}. \qquad (****)$$

Ein Koeffizientenvergleich von $(**)$ und $(****)$ zeigt

$$f_{kj_k}^{(k-1)} = \frac{1}{t} \cdot \widetilde{g}_{kj_k}.$$

Da sich die beiden Zeilen $(\widetilde{g}_{kj})_{j=1\ldots m}$ und $(f_{kj}^{(k-1)})_{j=1\ldots m}$ nur durch ein Vielfaches voneinander unterscheiden können, folgt daraus für alle $j = 1, \ldots, m$

$$f_{kj}^{(k-1)} = \frac{1}{t} \cdot \widetilde{g}_{kj},$$

also mit $(*)$ und $\delta^{(n)} = f_{n,j_n}$ die Behauptung. □

7.3 Korollar (allgemeine Rekursionsformel)
Sei A eine $n \times m$-Matrix mit den Eigenschaften (V1) bis (V3), und sei $k \in \{1, \ldots, n-1\}$ beliebig. Dann gilt für $i = 1, \ldots, k$ und $j = 1, \ldots, m$

$$\delta_{ij}^{(n)} = \frac{1}{\delta^{(k)}} \cdot \Big[\; \delta^{(n)} \cdot \delta_{ij}^{(k)} \; - \; \big(\delta_{ij_{k+1}}^{(k)} \quad \cdots \quad \delta_{ij_{n-1}}^{(k)} \quad \delta_{ij_n}^{(k)} \big) \cdot \begin{pmatrix} \delta_{k+1,j}^{(n)} \\ \vdots \\ \delta_{n-1,j}^{(n)} \\ \\ \delta_{nj}^{(n)} \end{pmatrix} \Big] =$$

$$= \frac{1}{\delta^{(k)}} \cdot \Big[\; \delta^{(n)} \cdot \delta_{ij}^{(k)} \; - \; \sum_{r=k+1}^{n} \delta_{ij_r}^{(k)} \cdot \delta_{rj}^{(n)} \; \Big].$$

Beweis:
Für $i = k$ folgt die Behauptung sofort aus obigem Lemma 7.2, indem man in

$$f_{kj}^{(k-1)} = \frac{1}{f_{kj_k}} \cdot \Big[\; f_{nj_n} f_{kj} \; - \; \big(\, f_{kj_{k+1}} \quad \cdots \quad f_{kj_{n-1}} \quad f_{kj_n} \, \big) \cdot \begin{pmatrix} f_{k+1,j} \\ \vdots \\ f_{n-1,j} \\ f_{nj} \end{pmatrix} \Big]$$

die auftretenden Koeffizienten $f_{\cdot\cdot}^{(\cdot)}$ durch ihre Werte $\delta_{\cdot\cdot}^{(\cdot)}$ ersetzt. Für $i = 1, \ldots, k-1$ folgt die Behauptung analog, indem man in A die Zeilen i und k und die Spalten j_i und j_k vertauscht. □

7.4 Bemerkungen (Merkregel für Backup-Rekursionsformel)

a) (*Elementare Zeilenumformung*) Wenn j von 1 bis m läuft, beschreibt die in Lemma 7.2 (1) gegebene Formel mit $t = f_{kj_k} \neq 0$ die elementare Zeilenumformung:

$$\text{neue } k\text{-te Zeile } (f_{kj}^{(k-1)}) =$$
$$= \frac{1}{t} \cdot \Big[f_{nj_n} \cdot \text{alte } k\text{-te Zeile } (f_{kj}) - \sum_{i=k+1}^{n} f_{kj_i} \cdot \text{alte } i\text{-te Zeile } (f_{ij}) \Big].$$

b) (*Merkregel*) Seien $k \in \{n-1, n-2, \ldots, 1\}$ und $j \in \{1, \ldots, m\}$ beliebig. Zur Berechnung des neuen Koeffizienten $f_{kj}^{(k-1)}$ aus der alten Matrix $F = F^{(k)}$ geht man wie folgt vor, vgl. Lemma 7.2:
   (1) Man multipliziert den alten Koeffizienten f_{kj} mit dem Zeilenstufenkoeffizienten der letzten Zeile.
   (2) Davon subtrahiert man das Produkt aus der partiellen Zeile
   (Koeffizienten der Zeile k in den auszuräumenden Spalten $j_{k+1}, \ldots, j_{n-1}, j_n$)

und der partiellen Spalte, die unter dem alten Koeffizienten f_{kj} liegt.

(3) Dieses Ergebnis wird durch den Zeilenstufenkoeffizienten der Zeile k geteilt.

Im folgenden bezeichne $< \cdot, \cdot >$ das Standardskalarprodukt. Aus dem vorangegangenen Lemma 7.2 ergibt sich jetzt für beliebige Matrizen

7.5 Satz (Algorithmen B1DMDFB, B2DMDFB, B2DMDMDFB)
Sei A eine beliebige $n \times m$-Matrix über einem Integritätsring R, wobei $r := \operatorname{rang} A$ den Rang von A bezeichne. Die Trigonalisierungsalgorithmen `B1TMD, B2TMD` *und* `B2TMDMD` *seien wie in Satz 3.7, Satz 4.3 bzw. Satz 4.6, der Unteralgorithmus* `BackupRow` *sei wie anschließend definiert.*

a) (Algorithmus `B1DMDFB`*) Betrachte die folgenden Instruktionen:*

(1) (forward: trigonalisieren) Ersetze A durch das Ergebnis von `B1TMD`(A)*. Ermittle die Zeilenstufen $j_1, \ldots, j_r$ von A und setze $k := r$.*

(2) (backup) Falls $k = 1$ ist, gib das Ergebnis A aus. Andernfalls ersetze die Zeile $(k-1)$ durch das Ergebnis von `BackupRow`$(A, \{j_k, j_{k+1}, \ldots, j_r\}, k, r)$*. Verkleinere k um 1 und wiederhole dann Schritt (2).*

Dies ist ein mit `B1DMDFB`(A) *bezeichneter Algorithmus, der als Ergebnis eine diagonalisierte Matrix ausgibt, die zu A äquivalent ist.*

Dabei berechnet der Unteralgorithmus `BackupRow`$(A, \{j_k, j_{k+1}, \ldots, j_r\}, k, r)$ *die $(k-1)$-te Zeile von A neu, d.h. räumt in ihr die Spalten $j_k, j_{k+1}, \ldots, j_r$ mit Hilfe der Zeilen $k, k+1, \ldots, r$ aus.*

(R1) (Berechnen) Ersetze für alle Nichtzeilenstufenspalten $j \in \{j_{k-1}+1, \ldots, m\} \setminus \{j_k, j_{k+1}, \ldots, j_r\}$ den Koeffizienten $a_{k-1,j}$ erst durch den Koeffizienten $a_{rj_r} \cdot a_{k-1,j} - < (a_{k-1,j_k}, a_{k-1,j_{k+1}}, \ldots, a_{k-1,j_r}), (a_{kj}, a_{k+1,j}, \ldots, a_{rj}) >$. Ersetze dann den neuen Koeffizienten $a_{k-1,j}$ durch $\frac{a_{k-1,j}}{a_{k-1,j_{k-1}}}$. Setze den Zeilenstufenkoeffizienten $a_{k-1,j_{k-1}} := a_{rj_r}$.

(R2) (Ausräumen) Setze für $l = k, \ldots, r$ jeweils $a_{k-1,j_l} := 0$.

(R3) Gib das Ergebnis $(a_{k-1,j})_{j=1\ldots m}$, also die $(k-1)$-te Zeile von A aus.

b) (Algorithmen `B2DMDFB` *und* `B2DMDMDFB`*) Ersetzt man in a) in Schritt (1) den Algorithmus* `B1TMD` *durch* `B2TMD` *bzw.* `B2TMDMD`*, so erhält man die mit* `B2DMDFB`(A) *bzw.* `B2DMDMDFB`(A) *bezeichneten Algorithmen, die ebenfalls eine zu A äquivalente Matrix in Diagonalform zum Ergebnis haben.*

Beweis:

a) Die Anweisungen (R1) bis (R3) werden offensichtlich nur endlich oft wiederholt. Nach Lemma 7.2 (Backup-Rekursion) führen sie zum behaupteten Ergebnis.
Auch die Instruktionen (1) und (2) werden nur endlich oft wiederholt, da bei jedem Durchlauf von Schritt (2) k um 1 reduziert wird, bis vom Startwert $r < \infty$ schließlich 1 erreicht ist. Daß das Ergebnis wie behauptet ist, folgt sofort aus den vorangegangenen Überlegungen und daraus, daß Schritt (2) die Anwendung der Backup-Rekursion aus Lemma 7.2 ist. Dabei spielt es keine Rolle, ob in A Zeilen getauscht werden müssen oder ob der Rang nicht maximal ist, denn dies wird zum einen schon bei der Trigonalisierung berücksichtigt, zum anderen werden die Schritte (1) und (2) in Abhängigkeit vom Rang durchgeführt.

b) Es kommt nur darauf an, daß in Schritt (1) eine trigonalisierte Matrix der Form (modulo Zeilentausch)

$$\begin{pmatrix} (\alpha_{ij}^{(i-1)})_{\substack{i=1\ldots r \\ j=1\ldots m}} \\ 0 \end{pmatrix}$$

hergestellt wird. Mit welchem der drei angegebenen Trigonalisierungs-Verfahren man dies bewerkstelligt, ist letztlich egal. □

7.6 Bemerkungen (Komplexität von B1DMDFB, B2DMDFB, B2DMDMDFB) Sei $A \in \mathrm{Mat}_{n\times m}(R)$ eine beliebige $n \times m$-Matrix, wobei der Rang von A maximal sei und $j_k = k$ für $k = 1, \ldots, \operatorname{rang} A$ gelte. Bei jeder Durchführung von `BackupRow` in Schritt (2) aus Satz 7.5 ergeben sich

$(n-k)(m-k)+1$ Additionen,

$(n-k+1)(m-k)$ Multiplikationen,

$(m-k)$ Divisionen.

Ist A nun eine $n \times m$-Matrix mit $n \leq m$ und $\operatorname{rang} A = n$, so folgt daraus:

a) (*Komplexität der Backup-Prozedur*) Insgesamt benötigt man $(k = n, \ldots, 2)$

$\approx \frac{1}{2}mn^2 - \frac{1}{6}n^3$ Additionen,

$\approx \frac{1}{2}mn^2 - \frac{1}{6}n^3$ Multiplikationen,

$\approx mn - \frac{1}{2}n^2$ Divisionen.

b) (*Komplexitäten von* `B1DMDFB, B2DMDFB, B2DMDMDFB`) Nimmt man die Komplexitäten der Trigonalisierungsverfahren (Bemerkung 3.8, 4.4 bzw. 4.7) und die Komplexität des Backup-Verfahrens aus a) zusammen, so erhält man insgesamt für eine $n \times m$-Matrix mit $n \leq m$:

(Komplexität von `B1DMDFB`*)*

$\approx mn^2 - \frac{2}{3}n^3$ Additionen,

$\approx \frac{3}{2}mn^2 - \frac{5}{6}n^3$ Multiplikationen,

$\approx \frac{1}{2}mn^2 - \frac{1}{6}n^3$ Divisionen.

(Komplexität von `B2DMDFB`*)*

$\approx mn^2 - \frac{2}{3}n^3$ Additionen,

$\approx \frac{5}{4}mn^2 - \frac{3}{4}n^3$ Multiplikationen,

$\approx \frac{1}{4}mn^2 - \frac{1}{12}n^3$ Divisionen.

(Komplexität von `B2DMDMDFB`*)*

Diese Komplexität ist gleich der Komplexität von `B2DMDFB`.

Bezeichnet A eine $n \times m$-Matrix mit $n > m$ und $\operatorname{rang} A = m$, so folgt:

c) (*Komplexität der Backup-Prozedur*) Insgesamt ergeben sich $(k = m, \ldots, 2)$

$\approx \frac{1}{2}mn^2 - \frac{1}{6}m^3$ Additionen,

$\approx \frac{1}{2}mn^2 - \frac{1}{6}m^3$ Multiplikationen,

$\approx \frac{1}{2}m^2$ Divisionen.

d) (*Komplexitäten von* `B1DMDFB, B2DMDFB, B2DMDMDFB`) Nimmt man die Komplexitäten der Trigonalisierungsverfahren und die Komplexität des Backup-Verfahrens aus c) zusammen, so erhält man insgesamt für eine $n \times m$-Matrix mit $n > m$:

(Komplexität von `B1DMDFB`*)*

$\approx mn^2 - \frac{2}{3}m^3$ Additionen,

$\approx \frac{3}{2}mn^2 - \frac{5}{6}m^3$ Multiplikationen,

$\approx \frac{1}{2}mn^2 - \frac{1}{6}m^3$ Divisionen.

(Komplexität von `B2DMDFB`*)*

$\approx mn^2 - \frac{2}{3}m^3$ Additionen,

$\approx \frac{5}{4}mn^2 - \frac{3}{4}m^3$ Multiplikationen,

$\approx \frac{1}{4}mn^2 - \frac{1}{12}m^3$ Divisionen.

(Komplexität von `B2DMDMDFB`*)*

Diese Komplexität ist gleich der Komplexität von `B2DMDFB`.

7.7 Bemerkungen zur Implementierung (B1DMDFB, B2DMDFB und B2DMDMDFB)

Betrachte die in Satz 7.5 gegebenen Algorithmen `B1DMDFB`, `B2DMDFB` und `B2DMDMDFB`.

a) *(Zuweisungen)* Wie in Schritt (R1) beschrieben, wird der Zeilenstufenkoeffizient der k-ten Zeile gleich a_{rj_r} gesetzt, um die Berechnung zu sparen.

b) (*Algorithmus* `BackupDMD`) Schritt (2) ist als eigene CoCoA-Funktion `BackupDMD` programmiert worden. Somit kann man diese Funktion auch unabhängig von den Algorithmen `B1DMDFB`, `B2DMDFBN` oder `B2DMDMDFB` auf eine Zeilenstufenmatrix anwenden, die mit einem Bareiss-Verfahren mit Division hergestellt worden ist und mit der Backup-Prozedur diagonalisiert weden soll.

c) In diesem Fall wurde darauf verzichtet, ein entsprechendes Verfahren ohne Division zu programmieren, welches ohnehin weniger effektiv wäre und ansonsten nicht weiter von Interesse ist. Möchte man auf die Division verzichten, genügt es nämlich nicht, im Rekursionsschritt aus Lemma 7.2 (1) einfach die Division durch t wegzulassen. Dies würde sogar zu einem falschen Ergebnis führen. Betrachtet man nämlich den Beweis von Lemma 7.2, so ergibt sich dort ohne Division ein sehr viel komplizierteres Ergebnis für den Koeffizienten $g_{kj} = [k+1 \ldots n \ k][j_{k+1} \ldots \ j_n \ j](A)$, weil dann die Diagonalelemente i.a. nicht mehr alle gleich $\delta^{(n)}$ sind, sondern unterschiedliche Werte $f_{k+1,j_{k+1}}, \ldots, f_{n,j_n}$ annehmen können. Als Konsequenz erhielte man also in 7.2 (1) mit $e_{kj} = g_{kj}$ ein ganz anderes Ergebnis.

8 Diagonalisierung mit dem Malashonok-1-Schritt-Verfahren

Nach wie vor sei A eine $n \times m$-Matrix über einem Integritätsring R mit den drei üblichen Voraussetzungen

(V1) $n \leq m$,

(V2) $\operatorname{rang} A = n$ und

(V3) Zeilentausch sei unnötig.

Mit dem folgenden *Malashonok-1-Schritt-Verfahren* werden, ausgehend vom ersten Zeilenstufenkoeffizienten a_{1j_1}, abwechselnd die darunterliegende Zeile und in den oberen Zeilen die danebenstehende Zeilenstufenspalte ausgeräumt. Damit erhält man also links oben eine diagonalisierte Blockmatrix, die sukzessive um die nächste Zeile und die nächste Zeilenstufenspalte erweitert wird, bis ganz A diagonalisiert ist.

Ein grobes Schema zeigt, in welcher Reihenfolge die Zeilen bzw. Spalten ausgeräumt

werden, falls A eine $6 \times m$-Matrix ist, deren Zeilenstufen alle auf der Diagonalen liegen.

$$\left(\begin{array}{c|c|c|c|c|c|c} * & 2 & 4 & 6 & 8 & 10 & * \\ \hline 1 & * & & & & & * \\ \hline 3 & & * & & & & * \\ \hline 5 & & & * & & & * \\ \hline 7 & & & & * & & * \\ \hline 9 & & & & & * & * \end{array}\right)$$

Hierbei symbolisiert $*$ die verbleibenden Koeffizienten auf der Diagonalen bzw. die jeweils verbleibenden letzten $(m-6)$ Koeffizienten. Am Ende steht auf der gesamten Diagonalen die Determinante $\delta^{(6)}$.

Zur Vorbereitung späterer Beweise benötigt man zunächst

8.1 Proposition (Rekursionsformeln)
Die Matrix $A = (a_{ij})$ aus $\mathrm{Mat}_{n\times m}(R)$ sei wie oben.

a) Seien $k \in \{0, \ldots, n-1\}$, $i \in \{k+1, \ldots, n\}$ und $j \in \{1, \ldots, m\}$ beliebig. Dann gilt

$$\begin{aligned} \alpha_{ij}^{(k)} &= a_{ij} \cdot \delta^{(k)} \; - \; \sum_{\lambda=1}^{k} a_{ij_\lambda} \cdot \delta_{\lambda j}^{(k)} = \\ &= a_{ij} \cdot \delta^{(k)} \; - \; (\, a_{ij_1} \; a_{ij_2} \ldots a_{ij_k} \,) \begin{pmatrix} \delta_{1j}^{(k)} \\ \delta_{2j}^{(k)} \\ \vdots \\ \delta_{k,j}^{(k)} \end{pmatrix} . \end{aligned}$$

b) Seien $k \in \{1, \ldots, n\}$ und $j \in \{1, \ldots, m\}$ beliebig. Dann gilt

$$\begin{aligned} \delta_{kj}^{(k)} &= a_{kj} \cdot \delta^{(k-1)} \; - \; \sum_{\lambda=1}^{k-1} a_{kj_\lambda} \cdot \delta_{\lambda j}^{(k-1)} = \\ &= a_{kj} \cdot \delta^{(k-1)} \; - \; (\, a_{kj_1} \; a_{kj_2} \ldots a_{kj_{k-1}} \,) \begin{pmatrix} \delta_{1j}^{(k-1)} \\ \delta_{2j}^{(k-1)} \\ \vdots \\ \delta_{k-1,j}^{(k-1)} \end{pmatrix} . \end{aligned}$$

Beweis:

a) Im Fall $k = 0$ erhält man auf beiden Seiten a_{ij}. Sei nun $1 \le k \le n-1$. Die Behauptung

folgt sofort, wenn man $\alpha_{ij}^{(k)}$ nach der letzten Zeile i entwickelt, nämlich:

$$\alpha_{ij}^{(k)} = [1\dots k\ i][j_1\dots j_k\ j](A) = \begin{vmatrix} a_{1j_1} & \cdots & a_{1j_k} & a_{1j} \\ \vdots & \ddots & \vdots & \vdots \\ a_{k,j_1} & \cdots & a_{kj_k} & a_{k,j} \\ a_{ij_1} & \cdots & a_{ij_k} & a_{ij} \end{vmatrix} = \qquad (*)$$

$$= a_{ij}\cdot[1\dots k][j_1\dots j_k](A) \ + \ \sum_{\lambda=1}^{k} a_{ij_\lambda}\cdot c_\lambda$$

mit den zugehörigen Kofaktoren $c_\lambda,\ \lambda = 1,\dots,k$, für die wiederum gilt:

$$c_\lambda = (-1)^{k+1+\lambda}\ [1\dots k][j_1\dots \widehat{j_\lambda}\dots j_k\ j](A).$$

Tauscht man hier die Spalte j von der hintersten Position k an die Position λ, so folgt

$$c_\lambda = (-1)^{k+1+\lambda}\ (-1)^{k-\lambda}\ [1\dots k][j_1\dots \widehat{j_\lambda}\ j\dots j_k](A) = -\ \delta_{\lambda j}^{(k)}.$$

Damit und mit $[1\dots k][j_1\dots j_k](A) = \delta^{(k)}$ folgt aus $(*)$ die Behauptung a). Die zweite Gleichheit der Behauptung ist klar.

b) folgt sofort aus a), da $\delta_{kj}^{(k)} = \alpha_{kj}^{(k-1)}$ gilt. □

Um A zu diagonalisieren, werden nun abwechselnd Zeilen und Spalten ausgeräumt. Dazu betrachtet man die Folge von Matrizen $A =: G^{(0)} = G^{(1)}, G^{(2)}, \dots, G^{(n)}$, die allgemein für $k = 1,\dots,n$ von folgender Form sind

$$G^{(k)} := \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\dots k\\ j=1\dots m}} \\ \\ (a_{ij})_{\substack{i=k+1\dots n\\ j=1\dots m}} \end{pmatrix} =$$

$$= \begin{pmatrix} \delta_{1j_1}^{(k)} & * & 0 & * & \cdots & 0 & \cdots & * & 0 & * & 0 & * & \delta_{1,j_{k+1}}^{(k)} & \cdots & \delta_{1m}^{(k)} \\ & & \delta_{2j_2}^{(k)} & * & \cdots & 0 & \cdots & * & 0 & * & 0 & * & \delta_{2,j_{k+1}}^{(k)} & \cdots & \delta_{2m}^{(k)} \\ & & & \ddots & & & \vdots & \vdots & \vdots & \vdots & * & \vdots & \vdots & & \vdots \\ & & & & \ddots & & \vdots & \vdots & \vdots & \vdots & * & \vdots & \vdots & \ddots & \vdots \\ & & & & & \ddots & \vdots & \vdots & \vdots & \vdots & * & \vdots & \vdots & & \vdots \\ & & \mathbf{0} & & & & * & 0 & * & 0 & * & \delta_{k-2,j_{k+1}}^{(k)} & \cdots & \delta_{k-2,m}^{(k)} \\ & & & & & & \delta_{k-1,j_{k-1}}^{(k)} & * & 0 & * & \delta_{k-1,j_{k+1}}^{(k)} & \cdots & \delta_{k-1,m}^{(k)} \\ & & & & & & & & \delta_{kj_k}^{(k)} & * & \delta_{k,j_{k+1}}^{(k)} & \cdots & \delta_{km}^{(k)} \\ a_{k+1,j_1} & \cdots & & \cdots & & a_{k+1,j_{k-1}} & \cdots & a_{k+1,j_k} & \cdots & a_{k+1,j_{k+1}} & \cdots & & a_{k+1,m} \\ \vdots & & & & & & & & & & & & \vdots \\ a_{nj_1} & \cdots & & \cdots & & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & & a_{nm} \end{pmatrix}$$

Dabei symbolisiert $*$ wieder die dazwischenliegenden Koeffizienten, die ungleich 0 sein können. $G^{(k)}$ bezeichnet also die Matrix, in der die $k \times j_k$-Blockmatrix links oben schon diagonalisiert ist und die unteren Zeilen $k+1, \ldots, n$ unverändert genau wie in A sind. Die Folge der so definierten Matrizen führt A in Diagonalform über, indem die bereits diagonalisierte Blockmatrix links oben sukzessive um die nächste Zeile und die nächste Zeilenstufenspalte erweitert wird, d.h. indem abwechselnd die darunterliegende Zeile und die danebenstehende Zeilenstufenspalte ausgeräumt werden.

Für die $6 \times m$-Matrix von vorhin veranschaulichen die folgenden schematischen Darstellungen den Übergang von einer Matrix $G^{(k-1)}$ zur nächsten, für $k = 2, \ldots, n$.

$$G^{(k-1)} = \begin{pmatrix} (\delta_{ij}^{(k-1)})_{\substack{i=1\ldots k-1\\ j=1\ldots m}} \\ (a_{ij})_{\substack{i=k\ldots n\\ j=1\ldots m}} \end{pmatrix} = \left(\begin{array}{c|c|c|c|ccc} \delta^{(k-1)} & & & * & * & \cdots & * \\ \hline & \delta^{(k-1)} & & * & * & \cdots & * \\ \hline & & \delta^{(k-1)} & * & * & \cdots & * \\ \hline * & * & * & * & * & \cdots & * \\ \hline * & \cdots & \cdots & \cdots & \cdots & \cdots & * \\ * & \ldots & \ldots & \cdots & \cdots & \cdots & * \end{array}\right)$$

In dieser Matrix steht in den ersten $(k-1)$ Zeilen der Hauptminor $\delta^{(k-1)}$ auf der Diagonalen. Durch Ausräumen der k-ten Zeile erhält man daraus zunächst die Matrix

$$\begin{pmatrix} (\delta_{ij}^{(k-1)})_{\substack{i=1\ldots k-1\\ j=1\ldots m}} \\ (\delta_{kj}^{(k)})_{j=1\ldots m} \\ (a_{ij})_{\substack{i=k+1\ldots n\\ j=1\ldots m}} \end{pmatrix} = \left(\begin{array}{c|c|c|c|ccc} \delta^{(k-1)} & & & * & * & \cdots & * \\ \hline & \delta^{(k-1)} & & * & * & \cdots & * \\ \hline & & \delta^{(k-1)} & * & * & \cdots & * \\ \hline & & & \delta^{(k)} & * & \cdots & * \\ \hline * & \cdots & \cdots & \cdots & \cdots & \cdots & * \\ * & \ldots & \ldots & \cdots & \cdots & \cdots & * \end{array}\right)$$

und schließlich durch Ausräumen der k-ten Spalte in den oberen $(k-1)$ Zeilen die gewünschte Matrix

$$G^{(k)} = \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\ldots k\\ j=1\ldots m}} \\ (a_{ij})_{\substack{i=k+1\ldots n\\ j=1\ldots m}} \end{pmatrix} = \left(\begin{array}{c|c|c|c|ccc} \delta^{(k)} & & & & * & \cdots & * \\ \hline & \delta^{(k)} & & & * & \cdots & * \\ \hline & & \delta^{(k)} & & * & \cdots & * \\ \hline & & & \delta^{(k)} & * & \cdots & * \\ \hline * & \cdots & \cdots & \cdots & \cdots & \cdots & * \\ * & \ldots & \ldots & \cdots & \cdots & \cdots & * \end{array}\right)$$

in der nun in den ersten k Zeilen der Hauptminor $\delta^{(k)}$ als Diagonalkoeffizient steht.

Wie kann man für $k = 2, \ldots, n$ diese Matrizen rekursiv berechnen? Betrachte dazu allgemein die beiden direkt aufeinander folgenden Matrizen

$$G^{(k-1)} = \begin{pmatrix} (\delta_{ij}^{(k-1)})_{\substack{i=1\ldots k-1\\ j=1\ldots m}} \\ \\ (a_{ij})_{\substack{i=k\ldots n\\ j=1\ldots m}} \end{pmatrix} =$$

$$= \begin{pmatrix} \delta_{1j_1}^{(k-1)} & * \cdots 0 \cdots * & & 0 & * & 0 & * & \delta_{1,j_k}^{(k-1)} & * & \cdots & \delta_{1m}^{(k-1)} \\ & \ddots & & \vdots & \vdots & \vdots & * & \vdots & \vdots & & \vdots \\ & & \ddots & \vdots & \vdots & \vdots & * & \vdots & \vdots & \ddots & \vdots \\ & & & 0 & \vdots & \vdots & * & \vdots & \vdots & & \vdots \\ & \mathbf{0} & & \delta_{k-2,j_{k-2}}^{(k-1)} & * & 0 & * & \delta_{k-2,j_k}^{(k-1)} & * & \cdots & \delta_{k-2,m}^{(k-1)} \\ & & & & & \delta_{k-1,j_{k-1}}^{(k-1)} & * & \delta_{k-1,j_k}^{(k-1)} & * & \cdots & \delta_{k-1,m}^{(k-1)} \\ a_{k,j_1} & \cdots & & a_{kj_{k-2}} & \cdots & a_{kj_{k-1}} & \cdots & a_{k,j_k} & \cdots & \cdots & a_{k,m} \\ a_{k+1,j_1} & \cdots & & \cdots & & \cdots & \cdots & \cdots & \cdots & \cdots & a_{k+1,m} \\ \vdots & & & & & & & & & & \vdots \\ a_{nj_1} & \cdots & & \cdots & & \cdots & \cdots & \cdots & \cdots & \cdots & a_{nm} \end{pmatrix}$$

wobei für die Zeilenstufenkoeffizienten der oberen $(k-1)$ Zeilen gilt

$$\delta_{1j_1}^{(k-1)} = \ldots = \delta_{k-1,j_{k-1}}^{(k-1)} = \delta^{(k-1)} \neq 0.$$

Die nachfolgende Matrix ist von der Form

$$G^{(k)} = \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\ldots k\\ j=1\ldots m}} \\ \\ (a_{ij})_{\substack{i=k+1\ldots n\\ j=1\ldots m}} \end{pmatrix} =$$

$$= \begin{pmatrix} \delta_{1j_1}^{(k)} & * \cdots 0 \cdots * & & 0 & * & 0 & * & 0 & * & \cdots & \delta_{1m}^{(k)} \\ & \ddots & & \vdots & \vdots & \vdots & \vdots & \vdots & * & & \vdots \\ & & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & * & \ddots & \vdots \\ & & & 0 & \vdots & \vdots & \vdots & \vdots & * & & \vdots \\ & & & \delta_{k-2,j_{k-2}}^{(k)} & * & 0 & * & 0 & * & \cdots & \delta_{k-2,m}^{(k)} \\ & \mathbf{0} & & & & \delta_{k-1,j_{k-1}}^{(k)} & * & 0 & * & \cdots & \delta_{k-1,m}^{(k)} \\ & & & & & & & \delta_{k,j_k}^{(k)} & * & \cdots & \delta_{k,m}^{(k)} \\ a_{k+1,j_1} & \cdots & & \cdots & & \cdots & \cdots & \cdots & \cdots & \cdots & a_{k+1,m} \\ \vdots & & & & & & & & & & \vdots \\ a_{nj_1} & \cdots & & \cdots & & \cdots & \cdots & \cdots & \cdots & \cdots & a_{nm} \end{pmatrix}$$

Hier gilt für die Zeilenstufenkoeffizienten der oberen k Zeilen

$$\delta^{(k)}_{1j_1} = \ldots = \delta^{(k)}_{k-1,j_{k-1}} = \delta^{(k)}_{kj_k} = \delta^{(k)} \neq 0.$$

8.2 Lemma (Malashonok-1-Schritt-Rekursion)
Sei A eine $n \times m$-Matrix mit den Eigenschaften (V1) bis (V3). Betrachte in der obigen Situation für $k = 2, \ldots, n$ die Matrix $G := G^{(k-1)}$, in der der $(k-1)$-te Zeilenstufenkoeffizient bezeichnet werde mit

$$t := g_{k-1,j_{k-1}} \ (\neq 0 \ stets).$$

Die Matrizen $E, \widetilde{E} \in \mathrm{Mat}_{n\times m}(R)$ seien folgendermaßen definiert:

(1) Die Zeile k ist für $j = 1, \ldots, m$ gegeben durch

$$e_{kj} := \widetilde{e}_{kj} := t \cdot g_{kj} - (\, g_{kj_1} \quad \cdots \quad g_{kj_{k-2}} \quad g_{kj_{k-1}} \,) \begin{pmatrix} g_{1j} \\ \vdots \\ g_{k-2,j} \\ g_{k-1,j} \end{pmatrix}.$$

(2) Die unteren Zeilen $i = k+1, \ldots, n$ sind genau wie in G.

(3) Für die oberen Zeilen $i = 1, \ldots, k-1$ gilt für $j = 1, \ldots, m$:
In $\widetilde{E}$ sind die Zeilen genau wie in G.
In E gilt

$$e_{ij} := \frac{1}{t}[k \ i][j_k \ j](\widetilde{E}).$$

Dann ist $E = G^{(k)}$.

Beweis:
Für die Zeile k folgt die Behauptung sofort aus Proposition 8.1b). Ausführlich heißt dies

$$g_{k-1,j_{k-1}} \cdot g_{kj} - (\, g_{kj_1} \quad \cdots \quad g_{kj_{k-2}} \quad g_{kj_{k-1}} \,) \begin{pmatrix} g_{1j} \\ \vdots \\ g_{k-2,j} \\ g_{k-1,j} \end{pmatrix} =$$

$$= \delta^{(k-1)} \cdot a_{kj} - (\, a_{kj_1} \quad \cdots \quad a_{kj_{k-2}} \quad a_{kj_{k-1}} \,) \begin{pmatrix} \delta^{(k-1)}_{1j} \\ \vdots \\ \delta^{(k-1)}_{k-2,j} \\ \delta^{(k-1)}_{k-1,j} \end{pmatrix} = \delta^{(k)}_{kj} = g^{(k)}_{kj}.$$

Da die Matrizen $G = G^{(k-1)}$ und $G^{(k)}$ in den unteren Zeilen $k+1, \ldots, n$ übereinstimmen,

bleibt die Behauptung nur noch für die oberen Zeilen $1, \ldots, k-1$ zu zeigen. Es gilt

$$\widetilde{E} = \begin{pmatrix}
\delta_{1j_1}^{(k-1)} & * \cdots 0 \cdots * & 0 & * & 0 & * & \delta_{1,j_k}^{(k-1)} & * & \cdots & \delta_{1m}^{(k-1)} \\
 & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & * & & \vdots \\
 & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & * & \ddots & \vdots \\
 & \ddots & 0 & \vdots & \vdots & \vdots & \vdots & * & & \vdots \\
\mathbf{0} & & \delta_{k-2,j_{k-2}}^{(k-1)} & * & 0 & * & \delta_{k-2,j_k}^{(k-1)} & * & \cdots & \delta_{k-2,m}^{(k-1)} \\
 & & & & \delta_{k-1,j_{k-1}}^{(k-1)} & * & \delta_{k-1,j_k}^{(k-1)} & * & \cdots & \delta_{k-1,m}^{(k-1)} \\
 & & & & & & \delta_{k,j_k}^{(k)} & * & \cdots & \delta_{k,m}^{(k)} \\
a_{k+1,j_1} & \cdots & \cdots & & \cdots & \cdots & \cdots & \cdots & \cdots & a_{k+1,m} \\
\vdots & & & & & & & & & \vdots \\
a_{nj_1} & \cdots & \cdots & & \cdots & \cdots & \cdots & \cdots & \cdots & a_{nm}
\end{pmatrix}$$

Aus der Bareiss-1-Schritt-Rekursion folgt hier sofort $E = G^{(k)}$. □

8.3 Bemerkung (Merkregel für 1-Schritt-Formel)
Betrachte in der Situation von Lemma 8.2 den Übergang von der alten Matrix $G = G^{(k-1)}$ zur neuen Matrix $G^{(k)}$. Für $k \in \{2, \ldots, n\}$ und $j \in \{1, \ldots, m\}$ gilt:

a) *(Elementare Zeilenumformung)* Wenn j von 1 bis m läuft, beschreibt Lemma 8.2 (1) mit $t = g_{k-1,j_{k-1}} \neq 0$ die folgende elementare Zeilenumformung:

$$\text{neue } k\text{-te Zeile } (g_{kj}^{(k)}) = t \cdot \text{alte } k\text{-te Zeile } (g_{kj}) - \sum_{i=1}^{k-1} g_{kj_i} \cdot \text{alte } i\text{-te Zeile } (g_{ij}).$$

b) (*Merkregel*) Den neu zu berechnenden Koeffizienten $g_{kj}^{(k)}$ erhält man aus der Vorgängermatrix $G = G^{(k-1)}$ wie folgt:
   (1) Man multipliziert den alten Koeffizienten g_{kj} mit dem Zeilenstufenkoeffizienten der Vorgängerzeile $(k-1)$.
   (2) Davon subtrahiert man das Produkt aus der partiellen Zeile (Koeffizienten der Zeile k in den auszuräumenden Spalten $j_1, j_2, \ldots, j_{k-1}$) und der partiellen Spalte, die über dem alten Koeffizienten g_{kj} liegt.

c) *(Merkregel)* Die neuen Koeffizienten in den oberen Zeilen $1, \ldots, k-1$ erhält man mit der bekannten Bareiss-1-Schritt-Rekursionsformel.

Als Anwendung der Malashonok-1-Schritt-Rekursion aus Lemma 8.2 folgt nun

8.4 Satz (Algorithmus Mal1DMD)
Sei $A \in \mathrm{Mat}_{n\times m}(R)$ eine beliebige $n \times m$-Matrix über einem Integritätsring R. Die beiden Unteralgorithmen `Pivot` *und* `B1DMDRow` *seien wie in Satz 3.7 bzw. Satz 5.6,* `Mal1Pivot` *und* `Mal1DMDRow` *seien wie unten angegeben. Betrachte die folgenden Instruktionen:*
(1) Setze $k := 1$, $\lambda := 1$ und ersetze A durch `Pivot`(A, λ, k).
(2) (Erste Zeilenstufe finden, evtl. Zeilentausch) Falls $a_{k\lambda} = 0$ ist, dann erhöhe λ um 1. Falls $\lambda > m$ gilt, gib das Ergebnis A aus, andernfalls ersetze A durch `Pivot`(A, λ, k) *und fahre fort bei Schritt (2) bzw. (3).*

(3) (Erste Zeilenstufe gefunden) Falls $a_{k\lambda} \neq 0$ *gilt, setze* $j_k := \lambda$ *und erhöhe* λ *um 1. Falls* $\lambda > m$ *gilt, fahre fort bei Schritt (6).*

(4) (Evtl. Zeilentausch) Falls $k \geq n$ *ist, gib das Ergebnis* A *aus, falls* $\lambda > m$ *gilt, fahre fort bei Schritt (6). Andernfalls führe* $\texttt{Mal1Pivot}(A, \lambda, \{j_1, \ldots, j_k\}, k)$ *durch und erhalte daraus* A, λ *und* c.

(5) ($(k+1)$*-te Zeile ausräumen,* j_{k+1}*-te Spalte ausräumen) Falls* $\lambda \leq m$ *gilt, dann setze* $j_{k+1} := \lambda$. *Ersetze in diesem Fall ferner die Zeile* $(k+1)$ *durch das Ergebnis von* $\texttt{Mal1DMDRow}(A, \{j_1, \ldots, j_k, j_{k+1}\}, k, c)$ *und setze* $t := a_{kj_k}$. *Ersetze dann die oberen Zeilen* $i = 1, \ldots, k$ *jeweils durch das Ergebnis von* $\texttt{B1DMDRow}(A, i, j_{k+1}, k{+}1, t)$. *Erhöhe* k *und* λ *jeweils um 1 und fahre fort bei Schritt (4).*

(6) (Alle überflüssigen unteren Zeilen vollständig ausräumen) Falls $\lambda > m$ *gilt, dann setze alle unteren Zeilen* $i = k+1, \ldots, n$ *gleich 0. Gib das Ergebnis* A *aus.*

Dies ist ein Algorithmus mit der Bezeichnung $\texttt{Mal1DMD}(A)$*, dessen Ergebnis eine zu* A *äquivalente Matrix in Diagonalform ist.*

Dabei bedeute $\texttt{Mal1Pivot}(A, \lambda, \{j_1, \ldots, j_k\}, k)$ *den folgenden Zeilentauschalgorithmus, um die nächste Zeilenstufe* j_{k+1} *zu finden, falls sie existiert. In diesem Fall werden die Matrix* A *mit eventuell getauschten Zeilen, die nächste Zeilenstufe* λ *und der nächste Zeilenstufenkoeffizient* c *als Ergebnis ausgegeben. Falls keine Zeilenstufe mehr existiert, werden die unveränderte Matrix* A*,* $\lambda = m+1$ *und* $c = 0$ *ausgegeben.*

(P1) $(a_{k+1,\lambda}$ *berechnen)*
Berechne $c := a_{kj_k} \cdot a_{k+1,\lambda} - \ <(a_{k+1,j_1}, a_{k+1,j_2}, \ldots, a_{k+1,j_k}), (a_{1\lambda}, a_{2\lambda}, \ldots, a_{k\lambda})>$. *Falls* $c \neq 0$ *ist, gib als Ergebnis* A*,* λ *und* c *aus.*

(P2) (Evtl. Zeilentausch) Andernfalls suche in den darunterliegenden Zeilen $i = k{+}2, \ldots, n$*, ob sich dort* $c := a_{kj_k} \cdot a_{i,\lambda} - <(a_{i,j_1}, a_{i,j_2}, \ldots, a_{i,j_k}), (a_{1\lambda}, a_{2\lambda}, \ldots, a_{k\lambda})> \neq 0$ *finden läßt. Falls eine solche Zeile* i *existiert, dann multipliziere die Zeile* i *mit* -1*, ersetze* c *durch* $-c$*, tausche die Zeilen* k *und* i *und gib als Ergebnis* A*,* λ *und* c *aus.*

(P3) (Nächste Spalte durchsuchen) Erhöhe λ *um 1. Falls* $\lambda > m$ *gilt, dann gib das Ergebnis* A*,* λ *und* $c \quad (= 0)$ *aus, andernfalls fahre fort bei Schritt (P1).*

Die $(k{+}1)$*-te Zeile von* A *wird mit dem Algorithmus* $\texttt{Mal1DMDRow}(A, \{j_1, \ldots, j_k, j_{k+1}\}, k, c)$ *neu berechnet, d.h. es werden mit Hilfe der oberen Zeilen* $1, \ldots, k$ *die Spalten* $j_1, j_2, \ldots, j_k$ *ausgeräumt:*

(R1) (Berechnen) Ersetze für alle Spalten $j = j_{k+1}+1, \ldots, m$ *den Koeffizienten* $a_{k+1,j}$ *durch* $a_{kj_k} \cdot a_{k+1,j} - <(a_{k+1,j_1}, a_{k+1,j_2}, \ldots, a_{k+1,j_k}), (a_{1j}, a_{2j}, \ldots, a_{kj})>$. *Setze dann den Zeilenstufenkoeffizienten* $a_{k+1,j_{k+1}} := c$.

(R2) (Ausräumen) Für alle $j \in \{1, \ldots, j_{k+1}-1\}$ *setze* $a_{k+1,j} := 0$.

(R3) Gib das Ergebnis $(a_{k+1,j})_{j=1\ldots m}$*, also die* $(k+1)$*-te Zeile von* A *aus.*

Beweis:

Offensichtlich werden die Schritte (P1) bis (P3) bzw. (R1) bis (R3) nur endlich oft wiederholt. Aus der Malashonok-1-Schritt-Rekursion (Lemma 8.2) folgt sofort, daß die besagten Schritte zum behaupteten Ergebnis führen.

Was die Anweisungen (1) bis (6) betrifft, so wird bei jeder Wiederholung von Schritt (2), (3) oder (5) die Variable λ erhöht, bzw. wird bei jeder Wiederholung von Schritt (5) die Variable k erhöht. Daher muß nach endlich vielen Schritten das Abbruchkriterium in Schritt (4) bzw. (6) erreicht werden.

Daß das Ergebnis von `Mal1DMD` wie behauptet ist, folgt aus den vorangegangenen und den folgenden Überlegungen. Mit den Schritten (2), (3) und (4) wird jeweils sichergestellt, daß die Zeilenstufen korrekt gefunden werden und evtl. ein Zeilentausch durchgeführt wird. Falls $A = 0$ gilt, wird schon in Schritt (2) das Ergebnis $A = 0$ ausgegeben. Schritt (5)

ist die Anwendung der Malashonok-1-Schritt-Rekursion (Lemma 8.2) unter Verwendung der mit `Mal1Pivot` gefundenen Parameter $\lambda = j_{k+1}$ und $c = a_{k+1,j_{k+1}}$. Schritt (6) wird durchgeführt, falls sich herausstellt, daß keine weitere Zeilenstufe mehr existiert und also alle unteren Zeilen $k+1, \ldots, n$ komplett ausgeräumt werden. □

8.5 Bemerkung (Komplexität von Mal1DMD)
Sei $A \in \mathrm{Mat}_{n\times m}(R)$ eine beliebige $n \times m$-Matrix, wobei der Rang von A maximal sei und $k = j_k$ für $k = 1, \ldots, \operatorname{rang} A$ gelte. Betrachtet man den Algorithmus `Mal1DMD` aus Satz 8.4, so ergeben sich dort bei jeder Wiederholung von Schritt (5)

$(k-1)(m-k) + k(m-k+1)$ Additionen,

$2(k-1)(m-k) + k(m-k+1)$ Multiplikationen,

$(k-1)(m-k)$ Divisionen.

Ist A eine $n \times m$-Matrix mit $n \leq m$ und mit $\operatorname{rang} A = n$, so benötigt man insgesamt $(k = 1, \ldots, n-1)$

$\approx mn^2 - \frac{2}{3}n^3$ Additionen,

$\approx \frac{3}{2}mn^2 - n^3$ Multiplikationen,

$\approx \frac{1}{2}mn^2 - \frac{1}{3}n^3$ Divisionen.

Bezeichnet A dagegen eine $n \times m$-Matrix mit $n > m$ und $\operatorname{rang} A = m$, so ergeben sich insgesamt $(k = 1, \ldots, m-1)$

$\approx \frac{1}{3}m^3$ Additionen,

$\approx \frac{1}{2}m^3$ Multiplikationen,

$\approx \frac{1}{6}m^3$ Divisionen.

8.6 Bemerkungen zur Implementierung (Mal1DMD)

a) Mit Hilfe des Unteralgorithmus `Mal1Pivot` wird die nächste Zeilenstufe j_{k+1} gesucht. Falls eine solche existiert, wird mit c bereits der nächste Zeilenstufenkoeffizient $a_{k+1,j_{k+1}}$ berechnet, andernfalls gilt $c = 0$, wenn keine weitere Zeilenstufe mehr existiert. Daher wird in Schritt (R1) die entsprechende Zuweisung für den Zeilenstufenkoeffizienten vorgenommen bzw. in Schritt (6) alles ausgeräumt.

b) Wie üblich wird bei `Mal1Pivot` eine der beiden vertauschten Zeilen mit -1 multipliziert, um das Vorzeichen der Determinante zu erhalten.

c) Auch hier wird wie schon beim Forward-Backup-Verfahren darauf verzichtet, einen entsprechenden Algorithmus ohne Division zu programmieren.

9 Grundlagen für das Dichotomie-Verfahren

Zuletzt wird das komplizierteste Verfahren zur Diagonalisierung vorgestellt, nämlich das *Dichotomie-Verfahren*, welches als Spezialfälle die in Abschnitt 7 und 8 behandelten Forward-Backup- und Malashonok-1-Schritt-Verfahren umfaßt.

Für die folgenden Betrachtungen sei ab sofort $A \in \mathrm{Mat}_{n\times m}(R)$ eine $n \times m$-Matrix über einem Integritätsring R mit den Eigenschaften:

(V1) $n \leq m$,
(V2) $\operatorname{rang} A = n$,
(V3) Zeilentausch ist nicht nötig und
(V4) für $k = 1, \ldots, n$ gilt $j_k = k$, d.h. die Zeilenstufen liegen alle auf der Hauptdiagonalen.

Anders als bei den bisherigen Verfahren tritt nun neben den üblichen Voraussetzungen (V1) bis (V3) eine weitere Voraussetzung (V4) auf. Im Gegensatz zu vorher dienen diese vier Voraussetzungen nun aber nicht nur dazu, die theoretischen Betrachtungen zu vereinfachen, sondern sie werden auch bei den in Abschnitt 10 folgenden Algorithmen beibehalten. Diese Algorithmen sind also weniger allgemein als die bereits bekannten Verfahren.

Grob gesagt, beruht das Dichotomie-Verfahren zur Diagonalisierung darauf, die Matrix A in einen oberen und einen unteren Teil zu partitionieren und das Verfahren rekursiv auf die kleineren Teilmatrizen anzuwenden. (Diese Vorgehensweise, ein Problem in eine Folge leicht lösbarer Teilprobleme zu zerlegen, nennt man auch *Divide et Impera-Methode.*)

Der Begriff *Dichotomie* kommt aus dem Griechischen und bedeutet Zweiteilung. Im folgenden ist mit "Dichotomie" eine beliebige Zweiteilung gemeint, d.h. die beiden Teilmatrizen von A müssen nicht dieselbe Zeilenanzahl besitzen. Speziell kann man zur Partitionierung bei jeder Rekursion die Teilmatrizen solange halbieren, bis man schließlich einzeilige Teilmatrizen erhält. In diesem Falle soll A für theoretische Betrachtungen die zusätzliche Voraussetzung

(V5) $n = 2^p$ für ein $p \in \mathbb{N}_+$

erfüllen, damit die Halbierung aufgeht.

9.1 Proposition (Rekursionsschritt des Dichotomie-Verfahrens)
Sei A eine $n \times m$-Matrix mit den Eigenschaften (V1) bis (V4). Betrachte die folgenden fünf Schritte:

(0) (Partitionieren) Falls die Matrix A einzeilig ist, gib das Ergebnis A aus. Andernfalls wähle ein $\mu \in \{1, \ldots, n-1\}$ und partitioniere die Matrix $A = \binom{A^{(o)}}{A^{(u)}}$ in eine obere Hälfte $A^{(o)} := (a_{ij})_{\substack{i=1\ldots\mu \\ j=1\ldots m}}$ und eine untere Hälfte $A^{(u)} := (a_{ij})_{\substack{i=\mu+1\ldots n \\ j=1\ldots m}}$.

(1) (Diagonalisieren links oben) Ersetze die obere Hälfte $A^{(o)}$ durch $(\delta_{ij}^{(\mu)})_{\substack{i=1\ldots\mu \\ j=1\ldots m}}$.

(2) (Ausräumen links unten) Ersetze die untere Hälfte $A^{(u)}$ durch $(\alpha_{ij}^{(\mu)})_{\substack{i=\mu+1\ldots n \\ j=1\ldots m}}$.

(3) (Diagonalisieren rechts unten) Ersetze die untere Hälfte $A^{(u)}$ durch $(\delta_{ij}^{(n)})_{\substack{i=\mu+1\ldots n \\ j=1\ldots m}}$.

(4) (Ausräumen rechts oben) Ersetze die obere Hälfte $A^{(o)}$ durch $(\delta_{ij}^{(n)})_{\substack{i=1\ldots\mu \\ j=1\ldots m}}$ und gib das Ergebnis A aus.

Dies stellt einen Algorithmus dar, welcher eine zu A äquivalente Matrix in Diagonalform zum Ergebnis hat.

Beweis:
Offensichtlich erhält man nach Wahl der Koeffizienten mit jedem der (endlich vielen) Schritte (0) bis (4) eine zu A äquivalente Matrix. Als Ergebnis wird die Matrix $(\delta_{ij}^{(n)})$ $(= A$ für $n = 1)$ ausgegeben, also eine zu A äquivalente Matrix in Diagonalform wie behauptet. □

Da die vorausgegangene Proposition 9.1 eine wichtige Grundlage des Dichotomie-Verfahrens bildet, wird nun noch einmal nachvollzogen, was die fünf Schritte (0) bis (4) eigentlich bewirken.

9.2 Beispiel (Rekursionsschritt des Dichotomie-Verfahrens)
Sei A eine $n \times m$-Matrix mit den Eigenschaften (V1) bis (V4). Der Rekursionsschritt des Dichotomie-Verfahrens aus Proposition 9.1, angewandt auf A, besteht aus den folgenden 5 Schritten.

(0) (*Partitionieren*) Falls die Matrix A einzeilig ist, ist sie schon diagonalisiert. Die weiteren Schritte (1) bis (4) sind also nicht nötig. Andernfalls partitioniere die Matrix A in eine obere und eine untere Hälfte. Die Partitionierung sei beliebig mit einem $\mu \in \{1, \ldots, n-1\}$. Damit zerfällt die Matrix

$$A = \begin{pmatrix} \overbrace{B}^{\mu \text{ Spalten}} & \overbrace{C}^{n-\mu \text{ Spalten}} & \overbrace{D}^{(m-n) \text{ Spalten}} \\ E & F & G \end{pmatrix} \begin{matrix} \} \ \mu \text{ Zeilen} \\ \} \ n-\mu \text{ Zeilen} \end{matrix}$$

in die sechs Blockmatrizen B, C, D, E, F, G.

(1) (*Diagonalisieren links oben*) Die obere Hälfte $(B \mid C \mid D)$ wird diagonalisiert, d.h. B wird durch ein Vielfaches der Einheitsmatrix ersetzt. Als Ergebnis folgt

$$A^{(1)} := \begin{pmatrix} \overbrace{b \cdot E_\mu}^{\mu \text{ Spalten}} & \overbrace{C^{(1)}}^{n-\mu \text{ Spalten}} & \overbrace{D^{(1)}}^{(m-n) \text{ Spalten}} \\ E & F & G \end{pmatrix} \begin{matrix} \} \ \mu \text{ Zeilen} \\ \} \ n-\mu \text{ Zeilen} \end{matrix}$$

Dabei gilt $b := |B| = \delta^{(\mu)}$ und $(C^{(1)} \mid D^{(1)}) := (\delta_{ij}^{(\mu)})_{\substack{i=1\ldots\mu \\ j=\mu+1\ldots m}}$.

(2) (*Ausräumen links unten*) Die Blockmatrix E wird ausgeräumt, F und G werden ersetzt durch $F^{(2)}$ bzw. $G^{(2)}$. Das Ergebnis wird bezeichnet mit

$$A^{(2)} := \begin{pmatrix} \overbrace{b \cdot E_\mu}^{\mu \text{ Spalten}} & \overbrace{C^{(1)}}^{n-\mu \text{ Spalten}} & \overbrace{D^{(1)}}^{(m-n) \text{ Spalten}} \\ 0 & F^{(2)} & G^{(2)} \end{pmatrix} \begin{matrix} \} \ \mu \text{ Zeilen} \\ \} \ n-\mu \text{ Zeilen} \end{matrix}$$

mit $(F^{(2)} \mid G^{(2)}) := (\alpha_{ij}^{(\mu)})_{\substack{i=\mu+1\ldots n \\ j=\mu+1\ldots m}}$.

(3) (*Diagonalisieren rechts unten*) Die Blockmatrix $F^{(2)}$ wird durch ein Vielfaches der Einheitsmatrix ersetzt, $G^{(2)}$ wird ersetzt durch $G^{(3)}$. Als Ergebnis erhält man

$$A^{(3)} := \begin{pmatrix} \overbrace{b \cdot E_\mu}^{\mu \text{ Spalten}} & \overbrace{C^{(1)}}^{n-\mu \text{ Spalten}} & \overbrace{D^{(1)}}^{(m-n) \text{ Spalten}} \\ 0 & f \cdot E_{n-\mu} & G^{(3)} \end{pmatrix} \begin{matrix} \} \ \mu \text{ Zeilen} \\ \} \ n-\mu \text{ Zeilen} \end{matrix}$$

mit $f := \begin{vmatrix} B & C \\ E & F \end{vmatrix} = \delta^{(n)}$ und mit der Blockmatrix $G^{(3)} := (\delta_{ij}^{(n)})_{\substack{i=\mu+1\ldots n \\ j=m-n+1\ldots m}}$.

(4) (*Ausräumen rechts oben*) Die Blockmatrix $C^{(1)}$ wird ausgeräumt, $D^{(1)}$ wird ersetzt durch $D^{(4)}$, b wird ersetzt durch f. Als Endergebnis ergibt sich die Diagonalmatrix

$$A^{(4)} := \begin{pmatrix} \overbrace{f \cdot E_\mu}^{\mu \text{ Spalten}} & \overbrace{0}^{n-\mu \text{ Spalten}} & \overbrace{D^{(4)}}^{(m-n) \text{ Spalten}} \\ 0 & f \cdot E_{n-\mu} & G^{(3)} \end{pmatrix} \begin{matrix} \} \ \mu \text{ Zeilen} \\ \} \ n-\mu \text{ Zeilen} \end{matrix}$$

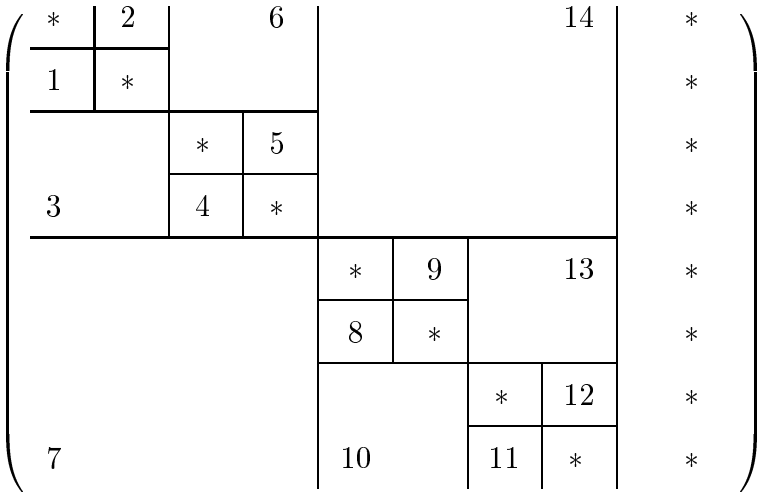
mit der Blockmatrix $D^{(4)} := (\delta_{ij}^{(n)})_{\substack{i=1\dots\mu \\ j=m-n+1\dots m}}$. Es gilt $A^{(4)} = (\delta_{ij}^{(n)})_{\substack{i=1\dots n \\ j=1\dots m}}$.

Wie auch bei allen anderen Diagonalisierungsvefahren mit Division wird hier das Endergebnis $(\delta_{ij}^{(n)})_{\substack{i=1\dots n \\ j=1\dots m}}$ erreicht, wobei in den Zwischenschritten (1) bis (4) jeweils Determinanten der Form $\alpha_{ij}^{(k)}$ bzw. der Form $\delta_{ij}^{(k)}$ (rekursiv) berechnet werden müssen. In den Ausräumungsschritten (2) und (4) werden jetzt nicht nur Zeilen und Spalten, sondern ganze Blockmatrizen auf einmal ausgeräumt. Die dafür nötigen Rekursionsformeln müssen im folgenden noch ermittelt werden. Um die Diagonalisierungsschritte (1) und (3) durchzuführen, kann man den Rekursionsschritt aus Proposition 9.1 wiederum auf die Teilmatrizen $(B \mid C \mid D)$ bzw. $(F^{(2)} \mid G^{(2)})$ anwenden.

Im folgenden versteht man unter "Dichotomie-Verfahren" stets das Verfahren, das durch den Rekursionsschritt aus Proposition 9.1 gegeben ist, wobei für jeden Diagonalisierungschritt (1) bzw. (3) der angegebene Rekursionsschritt wiederum auf die entsprechenden Teilmatrizen angewandt wird. Ein Diagonalisierungsschritt verzweigt sich also solange, bis bei der Partitionierung die obere Hälfte einzeilig ist.

In welcher Reihenfolge sukzessive die Blöcke ausgeräumt werden, wenn man zur Partitionierung stets die Halbierung wählt und jeden Diagonalisierungsschritt rekursiv durchführt, wird nun veranschaulicht.

9.3 Beispiel (Halbierung)
Sei A eine $n \times m$-Matrix mit den Eigenschaften (V1) bis (V5). Als Partitionierung wählt man die Halbierung. Anschaulich dargestellt für eine $8 \times m$-Matrix A, wird diese diagonalisiert, indem in der Reihenfolge der Numerierung die Blockmatrizen ausgeräumt werden:



Hierbei symbolisiert $*$ die verbleibenden Koffizienten auf der Diagonalen bzw. die in jeder Zeile verbleibenden letzten $(m-8)$ Koeffizienten. Auf der gesamten Diagonalen steht am Ende die Determinante $\delta^{(8)}$. Die angegebene Reihenfolge der Ausräumung ergibt sich, da der Rekursionsschritt (mit den Zwischenschritten (0) bis (4)) so lange wiederholt werden muß, bis bei der Partitionierung die obere Hälfte einzeilig ist. Ausführlich bedeutet dies für die obige $8 \times m$-Matrix A:

(0) Halbiere A.
(1) Diagonalisiere (rekursiv) die Zeilen $1, \dots, 4$.
  (1.0) Halbiere die obere Hälfte von A.
  (1.1) Diagonalisiere (rekursiv) die Zeilen 1,2.
    (1.1.0) Halbiere das oberste Viertel von A.

(1.1.1) Diagonalisiere die Zeile 1.

(1.1.2) Räume die Blockmatrix Nummer 1 aus.

(1.1.3) Diagonalisiere die Zeile 2.

(1.1.4) Räume die Blockmatrix Nummer 2 aus.

(1.2) Räume die Blockmatrix Nummer 3 aus.

(1.3) Diagonalisiere (rekursiv) die Zeilen 3,4.

(1.3.0) Halbiere das zweite Viertel von A.

(1.3.1) Diagonalisiere die Zeile 3.

(1.3.2) Räume Blockmatrix Nummer 4 aus.

(1.3.3) Diagonalisiere die Zeile 4.

(1.3.4) Räume Blockmatrix Nummer 5 aus.

(1.4) Räume die Blockmatrix Nummer 6 aus.

(2) Räume Blockmatrix Nummer 7 aus.

(3) Diagonalisiere (rekursiv) die Zeilen 5,...,8. Dieser Schritt verzweigt sich analog zu Schritt (1).

(4) Zuletzt wird Blockmatrix Nummer 14 ausgeräumt.

Bezeichnet man mit $E^{(k)}$ die zu A äquivalente Matrix, die man als Ergebnis eines jeden Schrittes $k \in \{0, 1, 1.0, 1.1, 1.1.0, 1.1.1, \ldots, 4\}$ erhält, so wird das Dichotomie-Verfahren repräsentiert durch die Matrizenfolge $A = E^{(0)} = E^{(1.0)} = E^{(1.1.0)} = E^{(1.1.1)}$, $E^{(1.1.2)} = E^{(1.1.3)}$, $E^{(1.1.4)} = E^{(1.1)}$, $E^{(1.2)} = E^{(1.3.0)} = E^{(1.3.1)}$, $E^{(1.3.2)} = E^{(1.3.3)}$, $E^{(1.3.4)} = E^{(1.3)}$, $E^{(1.4)} = E^{(1)}$, $E^{(2)}$, $E^{(3)} = \ldots = E^{(3.1.1)}$, $\ldots$, $E^{(3.4)} = E^{(3)}$, $E^{(4)}$. Für jede dieser Matrizen lassen sich die Koeffizienten anhand von Proposition 9.1 präzise angeben, aber es läßt sich bereits erahnen, daß dies eine komplizierte Angelegenheit ist.

Nachdem man sich mit Hilfe der Beispiele 9.2 und 9.3 eine Vorstellung davon verschafft hat, wie man beim Dichotomie-Verfahren vorgeht, stellen sich wie üblich die Fragen:

a) Welche Folge von zu A äquivalenten Matrizen repräsentiert das Dichotomie-Verfahren, d.h. insbesondere welche Koeffizienten der Form $\alpha_{ij}^{(k)}$ oder $\delta_{ij}^{(k)}$ müssen bei jedem Schritt ermittelt werden?

b) Wie lassen sich die in a) gefundenen Matrizen rekursiv berechnen?

Um die Frage a) zu beantworten, so zeigt sich in Beispiel 9.3, daß sich die Gesamtfolge der zu A äquivalenten Matrizen zwar exakt angeben läßt, jedoch unschön anzusehen ist. Daher wird darauf verzichtet, eine (zudem von der Partitionierung abhängige) Matrizenfolge für das gesamte Verfahren anzugeben, da die damit gewonnene Allgemeinheit auf Kosten der Verständlichkeit geht. Es genügt, im folgenden für eine bestimmte Matrix, nämlich die anschließend definierte Matrix $H^{(k)}$, eine Matrizenfolge für einen Rekursionsschritt explizit anzugeben. Da jeder Rekursionsschritt in die Schritte (0) bis (4) zerfällt, müssen dazu fünf Matrizen konstruiert werden, wodurch sich die Darstellung der Matrizenfolge wesentlich umfangreicher gestaltet als bei den bisherigen Verfahren. Anhand dieser Folge läßt sich die Frage b) nach den Rekursionsformeln so allgemeingültig beantworten, daß man anschließend einen Algorithmus für das Dichotomie-Verfahren formulieren kann.

Die $n \times m$-Matrix A erfülle auch weiterhin die Voraussetzungen (V1) bis (V4). Für

$k \in \{0, \ldots, n\}$ definiert man die $n \times m$-Matrix

$$H^{(k)} := \left(\begin{array}{c|c} \delta^{(k)} \cdot E_k & \mathbf{0} \\ \hline (\alpha_{ij}^{(k)})_{\substack{i=k+1\ldots n \\ j=1\ldots m}} & \end{array} \right) =$$

$$= \left(\begin{array}{c|c} \delta^{(k)} \cdot E_k & \mathbf{0} \\ \hline \mathbf{0} & (\alpha_{ij}^{(k)}) \end{array} \right) \begin{array}{l} \}k \text{ Zeilen } (i=1,\ldots,k) \\ \}(n-k) \text{ Zeilen } (i=k+1,\ldots,n) \end{array}$$

$$\underbrace{\qquad}_{\substack{k \text{ Spalten} \\ j=1,\ldots,k}} \quad \underbrace{\qquad}_{\substack{(m-k) \text{ Spalten} \\ j=k+1,\ldots,m}}$$

Insbesondere gilt $H^{(0)} = A$. Zur Abkürzung schreibt man auch

$$H^{(k)} := \left(\begin{array}{c|c} \delta^{(k)} \cdot E_k & \mathbf{0} \\ \hline \mathbf{0} & T \end{array} \right)$$

mit der $(n-k) \times (m-k)$-Blockmatrix

$$T := (\alpha_{ij}^{(k)})_{\substack{i=k+1,\ldots,n \\ j=k+1,\ldots,m}} .$$

In der obigen Matrix $H^{(k)}$ sind also die ersten k Zeilen diagonalisiert mit dem Zeilenstufenkoeffizienten $\delta^{(k)} \neq 0$. In den nächsten $(n-k)$ Zeilen sind die ersten k Spalten ausgeräumt mit den Koeffizienten $\alpha_{ij}^{(k)}$.

Im folgenden untersucht man nun für $k \in \{0, \ldots, n-1\}$ den Rekursionsschritt zur Diagonalisierung von T, also der Blockmatrix rechts unten in $H^{(k)}$. Für die Zeilenanzahl $(n-k)$ von T gelte o.E. $n-k \geq 2$, denn falls T einzeilig ist, ist T schon diagonalisiert. Warum betrachtet man, wenn man T diagonalisieren will, die Matrix $H^{(k)}$ und nicht gleich von vornherein die kleinere Matrix T? Und warum genügt es, die Matrizen T bzw. $H^{(k)}$ zu betrachten, um einen Rekursionsschritt des Dichotomie-Verfahrens zu untersuchen und allgemein gültige Rekursionsformeln zu finden? Dies geschieht aufgrund der folgenden Überlegungen:

9.4 Bemerkungen (Matrizenfolge)
Sei A eine $n \times m$-Matrix mit den Eigenschaften (V1) bis (V4). Betrachte in einem beliebigen Stadium des Dichotomie-Verfahrens das mit B bezeichnete Zwischenergebnis, also eine zu A äquivalente Matrix, und zwar zu dem Zeitpunkt, an dem an B einer der Diagonalisierungsschritte (1) oder (3) (rekursiv) durchgeführt werden soll.

a) Die nun als nächstes zu diagonalisierende Teilmatrix von B ist von der Gestalt

$$\widetilde{T} := (\alpha_{ij}^{(k)})_{\substack{i=k+1,\ldots,l \\ j=k+1,\ldots,m}}$$

mit $k \in \{0, \ldots, n\}$, $l \in \{1, \ldots, n\}$ und $k \leq l$, vgl. die Darstellung des Rekursionsschritts in Proposition 9.1 oder Beispiel 9.2.

b) Darüberhinaus ist für $k \neq 0$ in B die Zeile k, also die Vorgängerzeile der zu diagonalisierenden Teilmatrix $\widetilde{T}$, stets von der Gestalt

$$(b_{kj})_{j=1,\ldots,m} = (\delta_{kj}^{(k)})_{j=1,\ldots,m}.$$

Insbesondere nimmt also der Zeilenstufenkoeffizient b_{kk} den Wert $\delta^{(k)}$ an. Bis auf den Koeffizienten $b_{kk} = \delta^{(k)}$, der in die Rekursionsformeln eingeht (vgl. das noch folgende Lemma 9.5), sind die Koeffizienten in den oberen k Zeilen von B völlig belanglos. Sie bleiben unberührt bei der Diagonalisierung von $\widetilde{T}$. Wie die oberen k Zeilen von B beschaffen sind, hängt jeweils von der Art der Partitionierung ab. Bei der Halbierung wie in Beispiel 9.2 gilt

$$(b_{ij})_{\substack{i=1,\ldots,k \\ j=1,\ldots,m}} = (\delta_{ij}^{(k)})_{\substack{i=1,\ldots,k \\ j=1,\ldots,m}}.$$

In den Bemerkungen 10.11 und 10.12 wird man andere Partitionierungen kennenlernen, für die die oberen $(k-1)$ Zeilen von B anders als bei der Halbierung aussehen.

c) Die unteren Zeilen $l+1,\ldots,n$ von B sind genau wie in der ursprünglichen Matrix A, da sie bei der Diagonalisierung der darüberliegenden Blockmatrix $\widetilde{T}$ nicht angetastet werden.

d) Zusammenfassend gilt für $k = 0$ also $B = A$ und für $k \neq 0$

$$B = \left(\begin{array}{c} (b_{ij})_{i=1,\ldots,k-1} \\ \hline (\delta_{kj}^{(k)}) \\ \hline (\alpha_{ij}^{(k)})_{i=k+1,\ldots,l} \\ \hline (a_{ij})_{i=l+1,\ldots,n} \end{array}\right) = \left(\begin{array}{c|c|c} \multicolumn{3}{c}{(b_{ij})} \\ \hline \mathbf{0} & \delta^{(k)} & (\delta_{kj}^{(k)}) \\ \hline \mathbf{0} & \mathbf{0} & \widetilde{T} \\ \hline \multicolumn{3}{c}{(a_{ij})} \end{array}\right) \begin{array}{l} \}i=1,\ldots,k-1 \\ \}i=k \\ \}i=k+1,\ldots,l \\ \}i=l+1,\ldots,n \end{array}$$

$$\underbrace{\qquad}_{j=1,\ldots,k-1} \quad \underbrace{\qquad}_{j=k} \quad \underbrace{\qquad}_{j=k+1,\ldots,m}$$

Wegen der obigen Bemerkungen a) bis c) genügt es zur Ermittlung der Rekursionsformeln, die Matrix

$$H^{(k)} := \left(\begin{array}{c|c} \delta^{(k)} \cdot E_k & \mathbf{0} \\ \hline \mathbf{0} & T \end{array}\right) \begin{array}{l} \}i=1,\ldots,k \\ \}i=k+1,\ldots,n \end{array}$$

$$\underbrace{\qquad}_{j=1,\ldots,k} \quad \underbrace{\qquad}_{j=k+1,\ldots,m}$$

zu betrachten, die man aus B erhält, indem man die oberen k Zeilen von B einfachheitshalber durch $(\delta^{(k)} \cdot E_k \mid 0)$ ersetzt und l durch n. Um es noch einmal deutlich zu sagen, so ist die Matrix $H^{(k)}$ i.a. nicht äquivalent zu A. Aber die gewünschte Neuberechnung von $\widetilde{T}$ läßt sich offensichtlich sofort aus der im folgenden hergeleiteten Neuberechnung von T schlußfolgern.

e) Es ist günstig, mit $H^{(k)}$ eine Matrix derselben Spaltenanzahl wie A und derselben Anzahl an Vorgängerzeilen von T bzw. $\widetilde{T}$ zu betrachten, denn einerseits kann man dadurch für $i = k+1,\ldots,n$ und $j = 1,\ldots,m$ einfach

$$h_{ij}^{(k)} = \alpha_{ij}^{(k)}$$

schreiben. Andererseits wird an der Matrix $H^{(k)}$ besser veranschaulicht, welchen Fortschritt die Spaltenausräumung in A insgesamt nimmt, als wenn man nur isoliert die kleinere Matrix T betrachtet.

Betrachte nun also für $k \in \{0, \ldots, n-1\}$ die Matrix $H^{(k)}$. Um die durch k gegebene Partitionierung zu verfeinern, sei ferner $\mu \in \{1, \ldots, n-k-1\}$. Es erleichtert das Verständnis der folgenden Matrizen, wenn man sich stets vor Augen führt, daß Koeffizienten der Form $\delta_{ij}^{(k)}$ in "diagonalisierten" Zeilen i auftreten, in denen die Spalten $j_1, \ldots, \widehat{j_i}, \ldots, j_k$ ausgeräumt sind. Koeffizienten der Form $\alpha_{ij}^{(k)}$ befinden sich in "trigonalisierten" Zeilen, in denen die Spalten $j_1, \ldots, j_k$ ausgeräumt sind. (Nach Voraussetzung (V4) gilt sogar stets $j_k = k$ für $k = 1, \ldots, n$.)

(0) Partitionieren

Partitioniert man die untere Hälfte von $H^{(k)}$ mittels μ, so erhält man die Matrix

$$H^{(k\mu)} := \left(\begin{array}{c|c} \delta^{(k)} \cdot E_k & \mathbf{0} \\ \hline (\alpha_{ij}^{(k)})_{\substack{i=k+1\ldots k+\mu \\ j=1\ldots m}} & \\ \hline (\alpha_{ij}^{(k)})_{\substack{i=k+\mu+1\ldots n \\ j=1\ldots m}} & \end{array}\right) =$$

$$= \left(\begin{array}{c|c|c|c} \delta^{(k)} \cdot E_k & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline & (\alpha_{ij}^{(k)}) & (\alpha_{ij}^{(k)}) & (\alpha_{ij}^{(k)}) \\ \hline \mathbf{0} & (\alpha_{ij}^{(k)}) & (\alpha_{ij}^{(k)}) & (\alpha_{ij}^{(k)}) \end{array}\right) \begin{array}{l} \left.\vphantom{\begin{array}{c}a\\a\end{array}}\right\} k \text{ Zeilen } (i=1,\ldots,k) \\ \left.\vphantom{\begin{array}{c}a\\a\end{array}}\right\} \mu \text{ Zeilen } (i=k+1,\ldots,k+\mu) \\ \left.\vphantom{\begin{array}{c}a\\a\end{array}}\right\} (n-k-\mu) \text{ Zeilen } (i=k+\mu+1,\ldots,n) \end{array}$$

$$\underbrace{\phantom{\begin{array}{c}a\\a\end{array}}}_{\substack{k \text{ Spalten} \\ (j=1,\ldots,k)}} \underbrace{\phantom{\begin{array}{c}a\\a\end{array}}}_{\substack{\mu \text{ Spalten} \\ (j=k+1,\ldots,k+\mu)}} \underbrace{\phantom{\begin{array}{c}a\\a\end{array}}}_{\substack{(n-k-\mu) \text{ Spalten} \\ (j=k+\mu+1,\ldots,n)}} \underbrace{\phantom{\begin{array}{c}a\\a\end{array}}}_{\substack{(m-n) \text{ Spalten} \\ (j=n+1,\ldots,m)}}$$

Abkürzend schreibt man dafür auch

$$H^{(k\mu)} := \left(\begin{array}{c|c|c|c} \delta^{(k)} \cdot E_k & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline & U & V & W \\ \hline \mathbf{0} & X & Y & Z \end{array}\right) \begin{array}{l} \left.\vphantom{\begin{array}{c}a\\a\end{array}}\right\} k \text{ Zeilen} \\ \left.\vphantom{\begin{array}{c}a\\a\end{array}}\right\} \mu \text{ Zeilen} \\ \left.\vphantom{\begin{array}{c}a\\a\end{array}}\right\} (n-k-\mu) \text{ Zeilen} \end{array}$$

$$\underbrace{\phantom{\begin{array}{c}a\\a\end{array}}}_{k \text{ Spalten}} \underbrace{\phantom{\begin{array}{c}a\\a\end{array}}}_{\mu \text{ Spalten}} \underbrace{\phantom{\begin{array}{c}a\\a\end{array}}}_{(n-k-\mu) \text{ Spalten}} \underbrace{\phantom{\begin{array}{c}a\\a\end{array}}}_{(m-n) \text{ Spalten}}$$

mit den entsprechenden Blockmatrizen U, V, W, X, Y, Z. Für diese Matrix gilt

$$H^{(k\mu)} = H^{(k)}.$$

(1) Diagonalisieren links oben

In $H^{(k\mu)}$ werden die Zeilen $i = k+1, \ldots, k+\mu$ diagonalisiert, folglich kann man dort die Koeffizienten ersetzen durch $\delta_{ij}^{(k+\mu)}$. Damit erhält man

$$H^{(k\mu,1)} := \left(\begin{array}{c|c} \delta^{(k)} \cdot E_k & \mathbf{0} \\ \hline (\delta_{ij}^{(k+\mu)})_{\substack{i=k+1\ldots k+\mu \\ j=1\ldots m}} & \\ \hline (\alpha_{ij}^{(k)})_{\substack{i=k+\mu+1\ldots n \\ j=1\ldots m}} & \end{array}\right) =$$

$$= \left(\begin{array}{c|c|c|c} \delta^{(k)} \cdot E_k & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline & \delta^{(k+\mu)} \cdot E_\mu & (\delta_{ij}^{(k+\mu)}) & (\delta_{ij}^{(k+\mu)}) \\ \hline \mathbf{0} & (\alpha_{ij}^{(k)}) & (\alpha_{ij}^{(k)}) & (\alpha_{ij}^{(k)}) \end{array}\right) \begin{array}{l} \} \ i=1,\ldots,k \\ \} \ i=k+1,\ldots,k+\mu \\ \} \ i=k+\mu+1,\ldots,n \end{array}$$

$$\underbrace{\phantom{\delta^{(k)} \cdot E_k}}_{j=1,\ldots,k} \quad \underbrace{\phantom{\delta^{(k+\mu)} \cdot E_\mu}}_{j=k+1,\ldots,k+\mu} \quad \underbrace{\phantom{(\delta_{ij}^{(k+\mu)})}}_{j=k+\mu+1,\ldots,n} \quad \underbrace{\phantom{(\delta_{ij}^{(k+\mu)})}}_{j=n+1,\ldots,m}$$

Zur Abkürzung schreibt man dafür auch

$$H^{(k\mu,1)} := \left(\begin{array}{c|c|c|c} \delta^{(k)} \cdot E_k & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline & \delta^{(k+\mu)} \cdot E_\mu & V^{(1)} & W^{(1)} \\ \hline \mathbf{0} & X & Y & Z \end{array}\right) \begin{array}{l} \} \ k \text{ Zeilen} \\ \} \ \mu \text{ Zeilen} \\ \} \ (n-k-\mu) \text{ Zeilen} \end{array}$$

$$\underbrace{\phantom{\delta^{(k)} \cdot E_k}}_{k \text{ Spalten}} \quad \underbrace{\phantom{\delta^{(k+\mu)} \cdot E_\mu}}_{\mu \text{ Spalten}} \quad \underbrace{\phantom{V^{(1)}}}_{(n-k-\mu) \text{ Spalten}} \quad \underbrace{\phantom{W^{(1)}}}_{(m-n) \text{ Spalten}}$$

mit den entsprechenden Blockmatrizen $V^{(1)}, W^{(1)}, X, Y, Z$.

(2) Ausräumen links unten

Räumt man in der obigen Matrix $H^{(k\mu,1)}$ die Blockmatrix X aus, d.h. räumt man die Spalten $j = k+1\ldots,k+\mu$ in den Zeilen $i = k+\mu+1,\ldots,n$ aus, so werden in den betreffenden Zeilen die Koeffizienten durch $\alpha_{ij}^{(k+\mu)}$ ersetzt. Man erhält also die Matrix

$$H^{(k\mu,2)} := \left(\begin{array}{c|c} \delta^{(k)}\cdot E_k & \mathbf{0} \\ \hline (\delta_{ij}^{(k+\mu)})_{\substack{i=k+1\ldots k+\mu\\ j=1\ldots m}} & \\ \hline (\alpha_{ij}^{(k+\mu)})_{\substack{i=k+\mu+1\ldots n\\ j=1\ldots m}} & \end{array}\right) =$$

$$= \left(\begin{array}{c|c|c|c} \delta^{(k)}\cdot E_k & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline & \delta^{(k+\mu)}\cdot E_\mu & (\delta_{ij}^{(k+\mu)}) & (\delta_{ij}^{(k+\mu)}) \\ \hline \mathbf{0} & \mathbf{0} & (\alpha_{ij}^{(k+\mu)}) & (\alpha_{ij}^{(k+\mu)}) \end{array}\right) \begin{array}{l} \left.\vphantom{\begin{array}{c}a\\a\end{array}}\right\} i=1,\ldots,k \\ \left.\vphantom{\begin{array}{c}a\\a\end{array}}\right\} i=k+1,\ldots,k+\mu \\ \left.\vphantom{\begin{array}{c}a\\a\end{array}}\right\} i=k+\mu+1,\ldots,n \end{array}$$

$$\underbrace{\qquad}_{j=1,\ldots,k}\ \underbrace{\qquad}_{j=k+1,\ldots,k+\mu}\ \underbrace{\qquad}_{j=k+\mu+1,\ldots,n}\ \underbrace{\qquad}_{j=n+1,\ldots,m}$$

Zur Abkürzung schreibt man dafür auch

$$H^{(k\mu,2)} := \left(\begin{array}{c|c|c|c} \delta^{(k)}\cdot E_k & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline & \delta^{(k+\mu)}\cdot E_\mu & V^{(1)} & W^{(1)} \\ \hline \mathbf{0} & \mathbf{0} & Y^{(2)} & Z^{(2)} \end{array}\right) \begin{array}{l} \left.\vphantom{\begin{array}{c}a\\a\end{array}}\right\} k \text{ Zeilen} \\ \left.\vphantom{\begin{array}{c}a\\a\end{array}}\right\} \mu \text{ Zeilen} \\ \left.\vphantom{\begin{array}{c}a\\a\end{array}}\right\} (n-k-\mu) \text{ Zeilen} \end{array}$$

$$\underbrace{\qquad}_{k \text{ Spalten}}\ \underbrace{\qquad}_{\mu \text{ Spalten}}\ \underbrace{\qquad}_{(n-k-\mu) \text{ Spalten}}\ \underbrace{\qquad}_{(m-n) \text{ Spalten}}$$

mit den entsprechenden Blockmatrizen $V^{(1)}, W^{(1)}, Y^{(2)}, Z^{(2)}$.

(3) Diagonalisieren rechts unten

Diagonalisiert man in der obigen Matrix $H^{(k\mu,2)}$ die Zeilen $i = k+\mu+1, \ldots, n$, so werden dort die Koeffizienten durch $\delta_{ij}^{(n)}$ ersetzt. Man erhält somit

$$H^{(k\mu,3)} := \left(\begin{array}{c|c} \delta^{(k)} \cdot E_k & \mathbf{0} \\ \hline (\delta_{ij}^{(k+\mu)})_{\substack{i=k+1\ldots k+\mu \\ j=1\ldots m}} & \\ \hline (\delta_{ij}^{(n)})_{\substack{i=k+\mu+1\ldots n \\ j=1\ldots m}} & \end{array}\right) =$$

$$= \left(\begin{array}{c|c|c|c} \delta^{(k)} \cdot E_k & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline & \delta^{(k+\mu)} \cdot E_\mu & (\delta_{ij}^{(k+\mu)}) & (\delta_{ij}^{(k+\mu)}) \\ \hline \mathbf{0} & \mathbf{0} & \delta^{(n)} \cdot E_{n-k-\mu} & (\delta_{ij}^{(n)}) \end{array}\right) \begin{array}{l} \} i=1,\ldots,k \\ \} i=k+1,\ldots,k+\mu \\ \} i=k+\mu+1,\ldots,k+\lambda \end{array}$$

$$\underbrace{}_{j=1,\ldots,k} \quad \underbrace{}_{j=k+1,\ldots,k+\mu} \quad \underbrace{}_{j=k+\mu+1,\ldots,n} \quad \underbrace{}_{j=n+1,\ldots,m}$$

Zur Abkürzung schreibt man dafür auch

$$H^{(k\mu,3)} := \left(\begin{array}{c|c|c|c} \delta^{(k)} \cdot E_k & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline & \delta^{(k+\mu)} \cdot E_\mu & V^{(1)} & W^{(1)} \\ \hline \mathbf{0} & \mathbf{0} & \delta^{(n)} \cdot E_{n-k-\mu} & Z^{(3)} \end{array}\right) \begin{array}{l} \} k \text{ Zeilen} \\ \} \mu \text{ Zeilen} \\ \} (n-k-\mu) \text{ Zeilen} \end{array}$$

$$\underbrace{}_{k \text{ Spalten}} \quad \underbrace{}_{\mu \text{ Spalten}} \quad \underbrace{}_{(n-k-\mu) \text{ Spalten}} \quad \underbrace{}_{(m-n) \text{ Spalten}}$$

mit den entsprechenden Blockmatrizen $V^{(1)}, W^{(1)}, Z^{(3)}$.

(4) Ausräumen rechts oben

Wird in der obigen Matrix $H^{(k\mu,3)}$ die Blockmatrix $V^{(1)}$ ausgeräumt, d.h. werden die Spalten $j = k+\mu+1, \ldots, n$ in den Zeilen $i = k+1, \ldots, k+\mu$ ausgeräumt, so werden dort die Koeffizienten durch $\delta_{ij}^{(n)}$ ersetzt. Man erhält also

$$H^{(k\mu,4)} := \left(\begin{array}{c} \delta^{(k)} \cdot E_k \mid \mathbf{0} \\ \hline (\delta_{ij}^{(n)})_{\substack{i=k+1\ldots k+\mu \\ j=1\ldots m}} \\ \hline (\delta_{ij}^{(n)})_{\substack{i=k+\mu+1\ldots n \\ j=1\ldots m}} \end{array}\right) =$$

$$= \left(\begin{array}{c|c|c|c} \delta^{(k)} \cdot E_k & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline & \delta^{(n)} \cdot E_\mu & \mathbf{0} & (\delta_{ij}^{(n)}) \\ \hline \mathbf{0} & \mathbf{0} & \delta^{(n)} \cdot E_{n-k-\mu} & (\delta_{ij}^{(n)}) \end{array}\right) \begin{array}{l} \} i=1,\ldots,k \\ \} i=k+1,\ldots,k+\mu \\ \} i=k+\mu+1,\ldots,n \end{array}$$

$$\underbrace{\quad}_{j=1,\ldots,k} \quad \underbrace{\quad}_{j=k+1,\ldots,k+\mu} \quad \underbrace{\qquad}_{j=k+\mu+1,\ldots,n} \quad \underbrace{\qquad}_{j=n+1,\ldots,m}$$

Zur Abkürzung schreibt man dafür auch

$$H^{(k\mu,4)} := \left(\begin{array}{c|c|c|c} \delta^{(k)} \cdot E_k & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline & \delta^{(n)} \cdot E_\mu & \mathbf{0} & W^{(4)} \\ \hline \mathbf{0} & \mathbf{0} & \delta^{(n)} \cdot E_{n-k-\mu} & Z^{(3)} \end{array}\right) \begin{array}{l} \} k \text{ Zeilen} \\ \} \mu \text{ Zeilen} \\ \} (n-k-\mu) \text{ Zeilen} \end{array}$$

$$\underbrace{\quad}_{k \text{ Spalten}} \quad \underbrace{\quad}_{\mu \text{ Spalten}} \quad \underbrace{\qquad}_{(n-k-\mu) \text{ Spalten}} \quad \underbrace{\qquad}_{(m-n) \text{ Spalten}}$$

mit den entsprechenden Blockmatrizen $W^{(4)}$ und $Z^{(3)}$.

Wie lauten nun die Rekursionsformeln, um in den Ausräumungsschritten (2) und (4) die Matrizen aus ihren direkten Vorgängerinnen zu berechnen?

9.5 Lemma (Dichotomie-Rekursion zeilenweise)
Die $n \times m$-Matrix A erfülle die Voraussetzungen (V1) bis V(4). Betrachte in der obigen Situation für $k \in \{0, \ldots, n-1\}$ und für $\mu \in \{1, \ldots, n-k-1\}$ die fünf Matrizen $H^{(k\mu)}$, $H := H^{(k\mu,1)}$, $H^{(k\mu,2)}$, $\widetilde{H} := H^{(k\mu,3)}$ und $H^{(k\mu,4)}$. Ferner seien

$$t_k := \begin{cases} 1 & \text{falls } k = 0 \\ h_{kk} & \text{sonst} \end{cases}, \quad t_\mu := h_{k+\mu,k+\mu} = \widetilde{h}_{k+\mu,k+\mu} \quad \text{und} \quad t_n := \widetilde{h}_{nn}.$$

Die Matrizen E und $\widetilde{E} \in \mathrm{Mat}_{n\times m}(R)$ seien wie folgt definiert:

(1) In E gilt für die Zeilen $i = k+\mu+1, \ldots, n$ und die Spalten $j = 1, \ldots, m$

$$e_{ij} := \frac{1}{t_k} \cdot \Big[\, t_\mu \cdot h_{ij} - \begin{pmatrix} h_{i,k+1} & h_{i,k+2} & \ldots & h_{i,k+\mu} \end{pmatrix} \cdot \begin{pmatrix} h_{k+1,j} \\ h_{k+2,j} \\ \vdots \\ h_{k+\mu,j} \end{pmatrix} \Big].$$

Alle anderen Zeilen von E sind genau wie in H.

(2) In $\widetilde{E}$ gilt für die Zeilen $i = k+1, \ldots, k+\mu$ und die Spalten $j = 1, \ldots, m$

$$\widetilde{e}_{ij} := \frac{1}{t_\mu} \Big[\, t_n \cdot \widetilde{h}_{ij} - \begin{pmatrix} \widetilde{h}_{i,k+\mu+1} & \widetilde{h}_{i,k+\mu+2} & \ldots & \widetilde{h}_{i,n} \end{pmatrix} \cdot \begin{pmatrix} \widetilde{h}_{k+\mu+1,j} \\ \widetilde{h}_{k+\mu+2,j} \\ \vdots \\ \widetilde{h}_{nj} \end{pmatrix} \Big].$$

Alle anderen Zeilen von $\widetilde{E}$ sind genau wie in $\widetilde{H}$.

Dann gilt $E = H^{(k\mu,2)}$ und $\widetilde{E} = H^{(k\mu,4)}$.

Beweis:
Beachte im folgenden, daß nach Voraussetzung (V4) $j_s = s$ für $s = 1, \ldots, n$ gilt und daß ferner definitionsgemäß gilt

$$t_k = \delta^{(k)} \neq 0, \quad t_\mu = \delta^{(k+\mu)} \neq 0 \quad \text{und} \quad t_n = \delta^{(n)} \neq 0.$$

Zeige nun zunächst $E = H^{(k\mu,2)}$ und betrachte dazu die Matrix H. Um darin in den Zeilen $i = k+\mu+1, \ldots, n$ die Spalten $k+1, \ldots, k+\mu$ mit Hilfe der Zeilen $k+1, \ldots, k+\mu$ auszuräumen, berechnet man für $j = 1, \ldots, m$ die folgende Determinante (mittels Proposition 7.1)

$$g_{ij} := [k+1 \ldots k+\mu\ i][k+1 \ldots k+\mu\ j](H) = \begin{vmatrix} \delta^{(k+\mu)} & & \mathbf{0} & \delta^{(k+\mu)}_{k+1,j} \\ & \ddots & & \vdots \\ \mathbf{0} & & \delta^{(k+\mu)} & \delta^{(k+\mu)}_{k+\mu,j} \\ \alpha^{(k)}_{i,k+1} & \cdots & \alpha^{(k)}_{i,k+\mu} & \alpha^{(k)}_{ij} \end{vmatrix} =$$

$$= t_\mu^{\mu-1} \cdot \Big(t_\mu \cdot \alpha^{(k)}_{ij} - \sum_{r=k+1}^{k+\mu} \alpha^{(k)}_{ir} \cdot \delta^{(k+\mu)}_{rj} \Big).$$

Setze zur Vereinfachung

$$\widetilde{g}_{ij} := \frac{1}{t_\mu^{\mu-1}} \cdot g_{ij} = t_\mu \cdot \alpha_{ij}^{(k)} - \sum_{r=k+1}^{k+\mu} \alpha_{ir}^{(k)} \cdot \delta_{rj}^{(k+\mu)}. \qquad (*)$$

Nach Proposition 8.1a) gilt für $i = k+1, \ldots, n$ und $j = 1, \ldots, m$

$$\alpha_{ij}^{(k)} = \delta^{(k)} \cdot a_{ij} - \sum_{s=1}^{k} a_{is} \cdot \delta_{sj}^{(k)}.$$

Setzt man dies oben in $(*)$ ein, so erhält man

$$\widetilde{g}_{ij} = t_k \cdot \underbrace{t_\mu \cdot a_{ij}}_{=:\pi} - \underbrace{t_\mu \cdot \sum_{s=1}^{k} a_{is} \cdot \delta_{sj}^{(k)}}_{=:\rho} - t_k \cdot \underbrace{\sum_{r=k+1}^{k+\mu} a_{ir} \cdot \delta_{rj}^{(k+\mu)}}_{=:\sigma} + \underbrace{\sum_{r=k+1}^{k+\mu} \sum_{s=1}^{k} a_{is} \cdot \delta_{sr}^{(k)} \cdot \delta_{rj}^{(k+\mu)}}_{=:\tau}.$$

Vereinfacht ausgedrückt mit den Variablen π, ρ, σ und τ, folgt also

$$\widetilde{g}_{ij} = t_k \cdot \pi - \rho - t_k \cdot \sigma + \tau. \qquad (**)$$

Um es noch einmal zusammenzufassen: mit $\widetilde{g}_{ij}$ sind die Spalten $k+1, \ldots, k+\mu$ ausgeräumt worden, und zwar ausgehend von der Matrix H, in der schon die Spalten $1, \ldots, k$ ausgeräumt sind. Insgesamt sind in der Zeile $(\widetilde{g}_{ij})$ also die Spalten $1, \ldots, k+\mu$ ausgeräumt, und zwar rekursiv unter Verwendung der ersten Zeilen $1, \ldots, k+\mu$. Andererseits sind auch in $H^{(k\mu,2)}$ die Spalten $1, \ldots, k+\mu$ ausgeräumt, ebenfalls mit Hilfe der Zeilen $1, \ldots, k+\mu$. Es gilt wieder nach Proposition 8.1.a) (vgl. oben)

$$h_{ij}^{(k\mu,2)} = \alpha_{ij}^{(k+\mu)} = a_{ij} \cdot t_\mu - \sum_{s=1}^{k+\mu} a_{is} \cdot \delta_{sj}^{(k+\mu)} = \pi - \sigma - \sum_{s=1}^{k} a_{is} \cdot \delta_{sj}^{(k+\mu)}. \qquad (***)$$

Nach der allgemeinen Backup-Rekursionsformel (Korollar 7.3) gilt für $s = 1, \ldots, k$ und $j = 1, \ldots, m$

$$\delta_{sj}^{(k+\mu)} = \frac{1}{t_k} \cdot \Big(t_\mu \cdot \delta_{sj}^{(k)} - \sum_{r=k+1}^{k+\mu} \delta_{sr}^{(k)} \cdot \delta_{rj}^{(k+\mu)} \Big).$$

Oben in $(***)$ eingesetzt, ergibt sich damit

$$h_{ij}^{(k\mu,2)} = \pi - \sigma - \frac{1}{t_k} \cdot t_\mu \cdot \sum_{s=1}^{k} a_{is} \cdot \delta_{sj}^{(k)} + \frac{1}{t_k} \cdot \sum_{s=1}^{k} \sum_{r=k+1}^{k+\mu} a_{is} \cdot \delta_{sr}^{(k)} \cdot \delta_{rj}^{(k+\mu)}.$$

Mit den obigen Variablen π, ρ, σ und τ vereinfacht sich dies zu

$$h_{ij}^{(k\mu,2)} = \pi - \sigma - \frac{1}{t_k} \cdot \rho + \frac{1}{t_k} \cdot \tau. \qquad (****)$$

Es folgt also aus $(**)$ und $(****)$ für $i = k+1, \ldots, n$ und $j = 1, \ldots, m$

$$h_{ij}^{(k\mu,2)} = \frac{1}{t_k} \cdot \widetilde{g}_{ij},$$

also die Behauptung mit $(*)$, wenn man dort $\alpha_{ir}^{(k)} = h_{ij}$ für $i = k+1, \ldots, n$, $r = 1, \ldots, m$ und $\delta_{rj}^{(k+\mu)} = h_{rj}$ für $r = k+1, \ldots, k+\mu$, $j = 1, \ldots, m$ einsetzt.

Jetzt bleibt noch die zweite Behauptung zu zeigen, d.h. $H^{(k\mu,4)} = \widetilde{E}$. Dies folgt unmittelbar aus der allgemeinen Backup-Rekursionsformel (Korollar 7.3). Es gilt nämlich für $i = k+1, \ldots, k+\mu$ und $j = 1, \ldots, m$

$$h_{ij}^{(k\mu,4)} = \delta_{ij}^{(n)} = \frac{1}{\delta^{(k+\mu)}} \cdot \Big[\delta^{(n)} \cdot \delta_{ij}^{(k+\mu)} - (\delta_{i,k+\mu+1}^{(k+\mu)} \ldots \delta_{i,n}^{(k+\mu)}) \cdot \begin{pmatrix} \delta_{k+\mu+1,j}^{(n)} \\ \vdots \\ \delta_{nj}^{(n)} \end{pmatrix} \Big] =$$

$$= \frac{1}{t_\mu} \cdot \Big[t_n \cdot \widetilde{h}_{ij} - (\widetilde{h}_{i,k+\mu+1} \ldots \widetilde{h}_{i,n}) \cdot \begin{pmatrix} \widetilde{h}_{k+\mu+1,j} \\ \vdots \\ \widetilde{h}_{nj} \end{pmatrix} \Big]$$

wie behauptet. □

9.6 Korollar (allgemeine Rekursionsformeln)
Sei A eine $n \times m$-Matrix mit den Eigenschaften (V1) bis (V3), und sei $k \in \{0, \ldots, n-1\}$. Ferner sei $\mu \in \{k+1, \ldots, n\}$. Dann gilt für $i = k+\mu+1, \ldots, n$ und $j = 1, \ldots, m$:

$$\alpha_{ij}^{(k+\mu)} = \frac{1}{\delta^{(k)}} \cdot \Big[\ \delta^{(k+\mu)} \cdot \alpha_{ij}^{(k)} - \begin{pmatrix} \alpha_{i,k+1}^{(k)} & \alpha_{i,k+2}^{(k)} & \ldots & \alpha_{i,k+\mu}^{(k)} \end{pmatrix} \cdot \begin{pmatrix} \alpha_{k+1,j}^{(k)} \\ \alpha_{k+2,j}^{(k)} \\ \vdots \\ \alpha_{k+\mu,j}^{(k)} \end{pmatrix} \ \Big] =$$

$$= \frac{1}{\delta^{(k)}} \cdot \Big[\ \delta^{(k+\mu)} \cdot \alpha_{ij}^{(k)} - \sum_{r=k+1}^{k+\mu} \alpha_{ir}^{(k)} \cdot \alpha_{rj}^{(k)} \Big].$$

Beweis:
Sei zunächst $j_k = k$ für alle $k = 1, \ldots, n$. Nach obigem Lemma 9.5 gilt für die Zeilen $i = k+\mu+1, \ldots, n$ und die Spalten $j = 1, \ldots, m$

$$h_{ij}^{(k\mu,2)} = \frac{1}{t_k} \cdot \Big[\ t_\mu \cdot h_{ij} - \begin{pmatrix} h_{i,k+1} & h_{i,k+2} & \ldots & h_{i,k+\mu} \end{pmatrix} \cdot \begin{pmatrix} h_{k+1,j} \\ h_{k+2,j} \\ \vdots \\ h_{k+\mu,j} \end{pmatrix} \ \Big].$$

Ersetzt man alle auftretenden Koeffizienten durch ihre Werte $\alpha_{\ldots}^{(\cdot)}$ bzw. $\delta^{(\cdot)}$, so folgt sofort die Behauptung. Für beliebige $j_1, \ldots, j_k$ folgt die Behauptung analog. □

9.7 Bemerkungen (Merkregeln für zeilenweise Dichotomie-Rekursion)
Betrachte in der Situation von Lemma 9.5 zunächst den Übergang von der alten Matrix $H = H^{(k\mu,1)}$ zur neuen Matrix $H^{(k\mu,2)}$:

a) (*Elementare Zeilenumformung*) Wenn j von 1 bis m läuft, beschreibt Lemma 9.5 (1) die folgende elementare Zeilenumformung:

$$\text{neue } i\text{-te Zeile } (h_{ij}^{(k\mu,2)}) = \\ = \frac{1}{t_k} \cdot \Big[\, t_\mu \cdot \text{alte } i\text{-te Zeile } (h_{ij}) - \sum_{r=k+1}^{k+\mu} h_{ir} \cdot \text{alte } r\text{-te Zeile } (h_{rj}) \Big].$$

b) *(Merkregel)* Den neuen Koeffizienten $h_{ij}^{(k\mu,2)}$ berechnet man wie folgt aus der alten Matrix H:

(1) Man multipliziert den alten Koeffizienten h_{ij} mit dem Zeilenstufenkoeffizienten t_μ der zuletzt diagonalisierten Teilmatrix.

(2) Davon subtrahiert man das Produkt aus der partiellen Zeile

(Koeffizienten in der Zeile i, die ausgeräumt werden sollen)

und der partiellen Spalte in der nächsten Blockmatrix direkt über dem alten Koeffizienten h_{ij}.

(3) Falls $k \neq 0$ gilt, teilt man das Ergebnis durch den Zeilenstufenkoeffizienten t_k der zuvorletzt diagonalisierten Teilmatrix. (Für $k = 0$ gilt $t_k = 1$.)

Betrachte nun den Übergang von der alten Matrix $\widetilde{H} = H^{(k\mu,3)}$ zur neuen Matrix $H^{(k\mu,4)}$.

c) *(Elementare Zeilenumformung)* Als elementare Zeilenumformung ausgedrückt, besagt Lemma 9.5 (2), wenn j von 1 bis m läuft:

$$\text{neue } i\text{-te Zeile } (h_{ij}^{(k\mu,4)}) = \\ = \frac{1}{t_\mu} \cdot \Big[\, t_n \cdot \text{alte } i\text{-te Zeile } (\widetilde{h}_{ij}) - \sum_{r=k+\mu+1}^{n} \widetilde{h}_{ir} \cdot \text{alte } r\text{-te Zeile } (\widetilde{h}_{rj}) \Big].$$

d) *(Merkregel)* Den neuen Koeffizienten $h_{ij}^{(k\mu,4)}$ berechnet man wie folgt:

(1) Man multipliziert den alten Koeffizienten $\widetilde{h}_{ij}$ mit dem Zeilenstufenkoeffizienten t_n der zuletzt diagonalisierten Teilmatrix.

(2) Davon subtrahiert man das Produkt aus der partiellen Zeile

(Koeffizienten der Zeile i, die ausgeräumt werden sollen)

und der partiellen Spalte in der nächsten Blockmatrix direkt unter dem alten Koeffizienten $\widetilde{h}_{ij}$.

(3) Das Ergebnis dividiert man durch den Zeilenstufenkoeffizienten t_μ der zuvorletzt diagonalisierten Blockmatrix.

Faßt man die zeilenweise gegebenen Rekursionsformeln (Lemma 9.5) zusammen zu einer Multiplikation von Blockmatrizen, so folgt unmittelbar

9.8 Korollar (matrizenweise Dichotomie-Rekursion)

Sei A eine $n \times m$-Matrix über einem Integritätsring R mit den Eigenschaften (V1) bis (V4). Betrachte für $k \in \{0, \ldots, n-1\}$ und $\mu \in \{1, \ldots, n-k\}$ die Folge der Matrizen

$$H := H^{(k\mu,1)} = \left(\begin{array}{c|c|c|c} t_k \cdot E_k & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & t_\mu \cdot E_\mu & V^{(1)} & W^{(1)} \\ \hline \mathbf{0} & X & Y & Z \end{array}\right)$$

$$H^{(k\mu,2)} = \left(\begin{array}{c|c|c|c} t_k \cdot E_k & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & t_\mu \cdot E_\mu & V^{(1)} & W^{(1)} \\ \hline \mathbf{0} & \mathbf{0} & Y^{(2)} & Z^{(2)} \end{array}\right)$$

$$\widetilde{H} := H^{(k\mu,3)} = \left(\begin{array}{c|c|c|c} t_k \cdot E_k & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & t_\mu \cdot E_\mu & V^{(1)} & W^{(1)} \\ \hline \mathbf{0} & \mathbf{0} & t_n \cdot E_{n-k-\mu} & Z^{(3)} \end{array}\right)$$

$$H^{(k\mu,4)} = \left(\begin{array}{c|c|c|c} t_k \cdot E_k & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & t_n \cdot E_\mu & \mathbf{0} & W^{(4)} \\ \hline \mathbf{0} & \mathbf{0} & t_n \cdot E_{n-k-\mu} & Z^{(3)} \end{array}\right)$$

Ferner seien $t_k := \begin{cases} 1 & \text{falls } k = 0 \\ h_{kk} & \text{sonst} \end{cases}$, $t_\mu := h_{k+\mu,k+\mu} = \widetilde{h}_{k+\mu,k+\mu}$ *und* $t_n := \widetilde{h}_{nn}$.

(1) In $H^{(k\mu,2)}$ *gilt:*

$$(Y^{(2)} \mid Z^{(2)}) = \frac{1}{t_k} \cdot \Big[t_\mu \cdot (Y \mid Z) - X \cdot (V^{(1)} \mid W^{(1)}) \Big].$$

(2) In $H^{(k\mu,4)}$ *gilt:*

$$W^{(4)} = \frac{1}{t_\mu} \cdot \Big[\, t_n \cdot W^{(1)} - \; V^{(1)} \cdot Z^{(3)} \Big].$$

Beweis:
Dies folgt sofort aus den zeilenweise gegebenen Rekursionsformeln (Lemma 9.5), wenn man sie zusammenfaßt zu einer Multiplikation von Blockmatrizen.

Wie man sieht, basieren die Rekursionsformeln nun auf der Multiplikation bestimmter Blockmatrizen, d.h. hier läßt sich der Vorteil einer günstigen Matrizenmultiplikation ausspielen, wie sie beispielsweise bei [Str] oder [CW] zu finden ist.

9.9 Bemerkung (Merkregeln für matrizenweise Dichotomie-Rekursion)

a) Betrachte in Korollar 9.8 den Übergang von der alten Matrix $H = H^{(k\mu,1)}$ zur neuen Matrix $H^{(k\mu,2)}$. Die neue Teilmatrix $(Y^{(2)} \mid Z^{(2)})$ berechnet man wie folgt aus H:
   (1) Man multipliziert die alte Teilmatrix $(Y \mid Z)$ mit dem Zeilenstufenkoeffizienten t_μ der zuletzt diagonalisierten Teilmatrix.

(2) Davon subtrahiert man das Produkt aus der Blockmatrix, die ausgeräumt werden soll, und der nächsten Teilmatrix $(V^{(1)} \mid W^{(1)})$ direkt über der alten Teilmatrix.
(3) Falls $k \neq 0$ gilt, teilt man das Ergebnis durch den Zeilenstufenkoeffizienten t_k der zuvorletzt diagonalisierten Teilmatrix. (Für $k = 0$ gilt $t_k = 1$.)

Anders ausgedrückt, lassen sich (1) und (2) auch so verstehen, daß man wie bei der Berechnung einer Determinante zweiter Ordnung vorgeht, um aus der alten Teilmatrix

$$\left(\begin{array}{c|c} t_\mu \cdot E_\mu & (V^{(1)} \mid W^{(1)}) \\ \hline X & (Y \mid Z) \end{array}\right)$$

die neue Teilmatrix

$$(Y^{(2)} \mid Z^{(2)}) = \frac{1}{t_k} \cdot \Big[t_\mu \cdot E_\mu \cdot (Y \mid Z) - X \cdot (V^{(1)} \mid W^{(1)}) \Big]$$

zu erhalten. Man geht also wie bei der Bareiss-1-Schritt-Rekursion zur Trigonalisierung vor, wobei an die Stelle der Koeffizienten Blockmatrizen treten.

b) Betrachte den Übergang von der alten Matrix $\widetilde{H} = H^{(k\mu,3)}$ zur neuen Matrix $H^{(k\mu,4)}$. Die neue Blockmatrix $W^{(4)}$ berechnet man wie folgt aus $\widetilde{H}$:
(1) Man multipliziert die alte Blockmatrix $W^{(1)}$ mit dem Zeilenstufenkoeffizient t_n der zuletzt diagonalisierten Teilmatrix.
(2) Davon subtrahiert man das Produkt aus der Blockmatrix, die ausgeräumt werden soll, und der nächsten Blockmatrix direkt unter der alten Blockmatrix.
(3) Das Ergebnis teilt man durch den Zeilenstufenkoeffizienten t_μ der zuvorletzt diagonalisierten Blockmatrix.

Analog zu a) lassen sich auch hier (1) und (2) so verstehen, daß man ähnlich wie bei der Berechnung einer Determinante zweiter Ordnung (mit vertauschter Reihenfolge der Summanden) vorgeht, um aus der alten Teilmatrix

$$\left(\begin{array}{c|c} V^{(1)} & W^{(1)} \\ \hline t_n \cdot E_{n-\mu-k} & Z^{(3)} \end{array}\right)$$

die neue Teilmatrix

$$W^{(4)} = \frac{1}{t_\mu} \cdot \Big[t_n \cdot E_{n-\mu-k} \cdot W^{(1)} - V^{(1)} \cdot Z^{(3)} \Big]$$

zu erhalten. Man geht also wieder wie bei der Bareiss-1-Schritt-Rekursion zur Diagonalisierung vor, wobei Blockmatrizen die Stelle der Koeffizienten einnehmen.

Damit sind nun im wesentlichen die Grundlagen geschaffen, um im nächsten Abschnitt dieses Kapitels Algorithmen für das Dichotomie-Verfahren formulieren zu können.

10 Diagonalisierung mit dem Dichotomie-Verfahren

Nach wie vor sei A eine $n \times m$-Matrix über einem Integritätsring R mit den vier Voraussetzungen

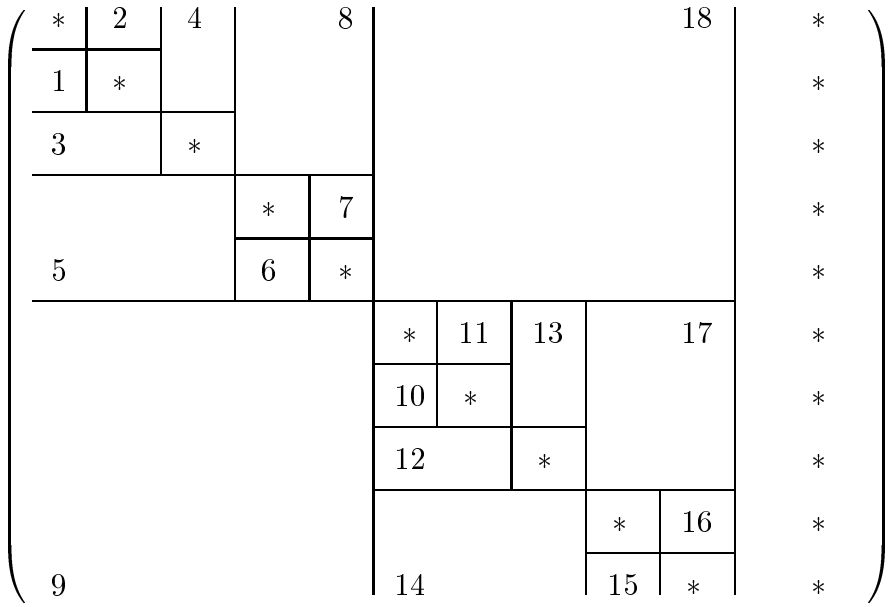
(V1) $n \leq m$,
(V2) $\operatorname{rang} A = n$,
(V3) Zeilentausch ist nicht nötig und
(V4) für $k = 1, \ldots, n$ gilt $j_k = k$.

Anders als bei früheren Verfahren werden diese Voraussetzungen auch zu den Dichotomie-Algorithmen benötigt, d.h. diese Verfahren sind weniger allgemein als andere.

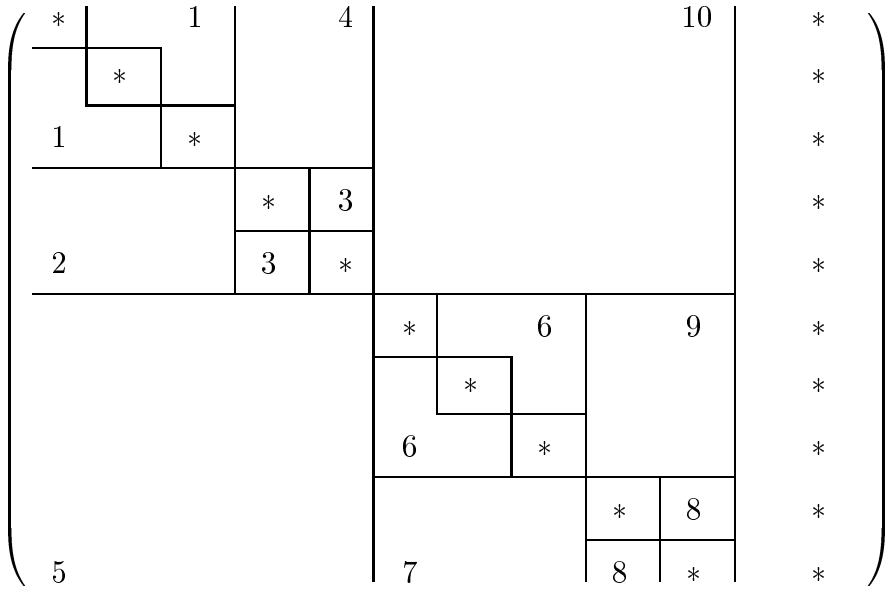
Basierend auf dem vorangegangen Abschnitt 9 bieten sich nun mehrere Varianten an, um die Matrix A zu diagonalisieren. Zum einen kann man aufgrund von Lemma 9.5 bzw. Merkregel 9.7 entweder wie gewohnt zeilenweise oder aufgrund von Korollar 9.8 bzw. Merkregel 9.9 matrizenweise vorgehen. Zum anderen kann man die Partitionierung beliebig wählen. Für die vier folgenden Algorithmen werde als Partitionsmethode die Halbierung verwendet. Die Einschränkung, daß die Zeilenanzahl n von A eine 2-Potenz ist, gilt nun nicht mehr. Ist aber $n \in \mathbb{N}_+$ beliebig, läßt sich i.a. nicht mehr erwarten, daß die Halbierung aufgeht. Hier sind mehrere Lösungsmöglichkeiten denkbar:

10.1 Bemerkungen (Halbierung bei beliebiger Zeilenanzahl)

a) *(Halbierung mit Rest oben bis zur Einzeiligkeit)*
Geht die Halbierung nicht auf, wird die überzählige Zeile zur oberen Hälfte dazugefügt. Die Halbierung der Teilmatrizen findet solange statt, bis schließlich die obere Hälfte einzeilig ist. Am Beispiel einer 10×12-Matrix ergibt sich somit folgende Reihenfolge der Ausräumung:

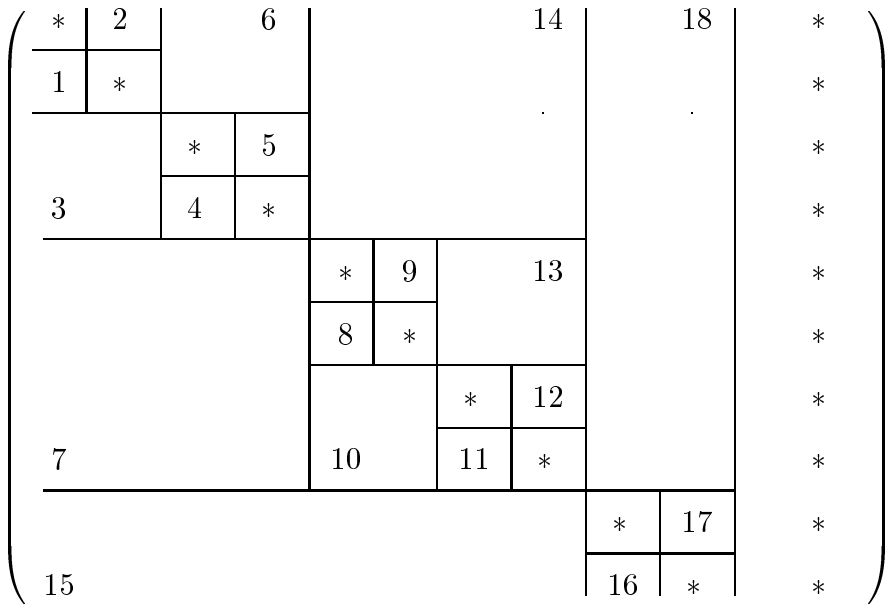


b) *(Halbierung mit Rest oben bis zur s-Zeiligkeit)*
Die Möglichkeit a) läßt sich so variieren, daß nicht bis zur Einzeiligkeit, sondern bis zu einer bestimmten Zeilenanzahl s mit $n \geq s \geq 1$ halbiert wird. Die Teilmatrizen mit s Zeilen kann man dann mit einem der bereits bekannten Diagonalisierungsverfahren aus den Abschnitten 3, 4 oder 5 diagonalisieren. Am Beispiel einer 10×12-Matrix ergibt sich mit $s = 3$ folgende Reihenfolge der Ausräumung:



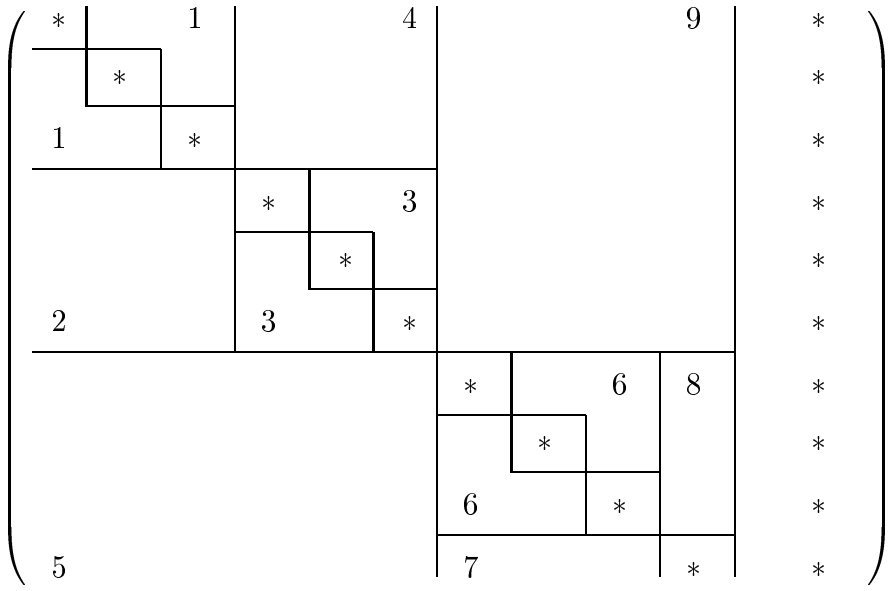
c) *(Verdopplung von Einzeiligkeit aus)*

Ausgehend von der ersten Zeile wird solange die Blockgröße verdoppelt, bis die ganze Matrix A erfaßt ist. Am Beispiel einer 10×12-Matrix ergibt sich somit folgende Reihenfolge der Ausräumung:



d) *(Verdopplung von s-Zeiligkeit aus)*

Die Möglichkeit c) läßt sich so variieren, daß ausgehend von den oberen s Zeilen die Blockgröße verdoppelt wird, wobei $n \geq s \geq 1$ gilt. Die s-zeiligen Teilmatrizen können mit einem der bereits bekannten Verfahren diagonalisiert werden. Am Beispiel einer 10×12-Matrix ergibt sich mit $s = 3$ folgende Reihenfolge der Ausräumung:



Wie man sieht, führt jede dieser vier Varianten zu einem anderen Ablauf der Ausräumung. Verfährt man nach den Varianten 10.1b) bzw. 10.1d), partitioniert man die gegebene Matrix A also nur bis zu einer gewissen Blockgröße s, so kann man zur Diagonalisierung der (maximal s-zeiligen) Teilmatrizen einen der bekannten Algorithmen wählen, nämlich `B1DMD` aus Satz 5.6, `B2DMD` aus Satz 6.3, `B2DMDMD` aus Satz 6.6, `B1DMDFB, B2DMDFB, B2DMDMDFB` aus Satz 7.5 oder `Mal1DMD` aus Satz 8.4. Der ausgewählte Algorithmus werde mit `Diag` bezeichnet, also

$$\texttt{Diag} \in L := \{\texttt{B1DMD}, \texttt{B2DMD}, \texttt{B2DMDMD}, \texttt{B1DMDFB}, \texttt{B2DMDFB}, \texttt{B2DMDMDFB}, \texttt{Mal1DMD}\}.$$

Diagonalisiert man mit dem Algorithmus `Diag` eine Teilmatrix $\widetilde{T}$ von A, so startet die Division nicht mit dem Teiler $t = 1$, sondern mit dem Teiler t, der sich im Dichotomie-Algorithmus ergibt. Um dies deutlich zu machen, schreibt man dafür $\texttt{Diag}(\widetilde{T}, t)$, vgl. dazu auch die noch folgenden Bemerkungen zur Implementierung (10.10).

Eine kurze Vorbemerkung zur Bezeichnung der Algorithmen: Es bezeichnet

- `H` — Halbierung, vgl. Bemerkung 10.1a) und b)
- `V` — Verdopplung, vgl. Bemerkung 10.1c) und d)
- `R` — Rekursionsformeln zeilenweise, vgl. Lemma 9.5 bzw. Merkregel 9.7
- `M` — Rekursionsformeln matrizenweise, vgl. Korollar 9.8 bzw. Merkregel 9.9.

Zunächst wird das Dichotomie-Verfahren mit der Halbierungsmethode aus Bemerkung 10.1a) bzw. 10.1b) und den zeilenweisen Rekursionsformeln aus Lemma 9.5 durchgeführt. Dabei bezeichne $\lfloor n \rfloor = \max\{m \in \mathbb{N} : m \leq n\}$ den *Gaußschen Ganzteil* von n, und $< \cdot, \cdot >$ bezeichne wie üblich das Standardskalarprodukt.

10.2 Satz (Algorithmus MalDichHR)
Sei A eine $n \times m$-Matrix über einem Integritätsring R mit $n \leq m$ und $\operatorname{rang} A = n$*, wobei außerdem noch $j_k = k$ für $k = 1, \ldots, n$ gelte und wobei ein Zeilentausch nicht nötig sei. Ferner seien $s \in \{1, \ldots, n\}$ und* $\texttt{Diag} \in L$ *beliebig. Die Algorithmen* `AusrLURow` *und* `AusrRORow` *seien wie nachfolgend angegeben. Betrachte die folgenden Instruktionen:*

(0) Setze $t_u := 1$.
(1) Setze $P := (n)$ und $k_o := n$.

(2) (Partitionieren) Falls $k_o > s$ gilt, dann ersetze k_o durch den Ganzteil $\lfloor \frac{k_o+1}{2} \rfloor$, füge k_o im Tupel P an erster Position ein und wiederhole Schritt (2).

(3) (Blockende der oberen Hälfte) Setze $k_o := p_1$, wobei p_1 den ersten Koeffizienten des Tupels P bezeichne, und streiche p_1 aus P.

(4) (Diagonalisieren links oben) Falls $s > 1$ gilt, dann ersetze die obere Hälfte $\widetilde{T} := (a_{ij})_{\substack{i=1\dots k_o \\ j=1\dots m}}$ von A durch das Ergebnis von $\mathtt{Diag}(\widetilde{T}, t_u)$.

(5) (Blockende der unteren Hälfte) Falls P leer ist, dann setze $k_u := k_o$, andernfalls setze $k_u := p_1$.

(6) (Ausräumen links unten) Falls $k_o < n$ gilt, dann ersetze die Zeilen $i = k_o + 1, \dots, k_u$ jeweils durch das Ergebnis von $\mathtt{AusrLURow}(A, i, k_o, t_u)$.

(7) (Rekursiv diagonalisieren rechts unten) Falls $k_o < n$ gilt, dann setze $t_o := a_{k_o,k_o}$ und ersetze die Teilmatrix $\widetilde{T} := (a_{ij})_{\substack{i=k_o+1\dots k_u \\ j=k_o+1\dots m}}$ durch die Matrix, die man als Ergebnis erhält, wenn man die Schritte (1) bis (9) für die $(k_u - k_o) \times (m - k_o)$-Matrix $\widetilde{T}$ mit $t_u := t_o$ durchführt.
(Die Wiederholung dieser Schritte ist so zu verstehen, daß s und $\mathtt{Diag}$ wie zuvor sind und daß t_u wie gerade eben definiert ist. Statt A wird nun aber $\widetilde{T}$ betrachtet, d.h. insbesondere daß die Zeilenanzahl jetzt $(k_u - k_o)$ statt n beträgt. Die Variablen (P, k_o, k_u, t_o etc.), die bis jetzt in den Schritten (1) bis (7) mit Bezug auf A ermittelt worden sind, werden bei der Wiederholung der Schritte (1) bis (9) für $\widetilde{T}$ aber nicht verändert, da sich die dabei auftretenden Variablen auf $\widetilde{T}$ und nicht auf A beziehen.)

(8) (Ausräumen rechts oben) Im Fall $k_o < n$ ersetze die Zeilen $i = 1, \dots, k_o$ jeweils durch das Ergebnis von $\mathtt{AusrRORow}(A, i, k_o, k_u, t_o)$. Setze $t_u := 1$, ersetze k_o durch k_u und fahre fort bei Schritt (5).

(9) Gib das Ergebnis A aus.

Dies stellt einen Algorithmus namens $\mathtt{MalDichHR}(A)$ dar, der eine zu A äquivalente Matrix in Diagonalform, nämlich $(\delta_{ij}^{(n)})$, zum Ergebnis hat.

Hierbei bezeichne $\mathtt{AusrLURow}(A, i, k_o, t_u)$ den Algorithmus, um in A in der Zeile i die Spalten $1, \dots, k_o$ auszuräumen.

(R1) (Berechnen) Ersetze für die Spalten $j = k_o + 1, \dots, m$ den Koeffizienten a_{ij} zunächst durch $a_{11} \cdot a_{ij} - <(a_{i1}, \dots, a_{ik_o}), (a_{1j}, \dots, a_{k_oj})>$. Ersetze dann den neuen Koeffizienten a_{ij} durch a_{ij}/t_u.

(R2) (Ausräumen) Für $j = 1, \dots, k_o$ setze $a_{ij} := 0$.

(R3) Gib das Ergebnis $(a_{ij})_{j=1\dots m}$ aus, also die i-te Zeile von A.

Mit dem Algorithmus $\mathtt{AusrRORow}(A, i, k_o, k_u, t_0)$ werden in der Matrix A in der Zeile i die Spalten $k_o + 1, \dots, k_u$ ausgeräumt.

(R1) (Berechnen) Ersetze für die Spalten $j = k_u + 1, \dots, m$ den Koeffizienten a_{ij} zunächst durch $a_{k_u,k_u} \cdot a_{ij} - <(a_{i,k_o+1}, \dots, a_{ik_u}), (a_{k_o+1,j}, \dots, a_{k_u,j})>$. Ersetze dann den neuen Koeffizienten a_{ij} durch a_{ij}/t_o.

(R2) (Ausräumen) Für $j = k_o + 1, \dots, k_u$ setze $a_{ij} := 0$.

(R3) Setze den Zeilenstufenkoeffizienten $a_{ii} := a_{k_u,k_u}$.

(R4) Gib das Ergebnis $(a_{ij})_{j=1\dots m}$, also die i-te Zeile von A aus.

Beweis:
Da die Zeilenanzahl von A endlich ist, können die angegeben Schritte nur endlich oft wiederholt werden. Daß das Ergebnis die Diagonalmatrix $(\delta_{ij}^{(n)})$ ist, folgt sofort aus den in Abschnitt 9 angestellten Überlegungen. Insbesondere sind die Algorithmen $\mathtt{AusrLURow}$ und $\mathtt{AusrRORow}$ Anwendungen von Lemma 9.5 (Dichotomie-Rekursion zeilenweise). □

10.3 Bemerkung (Komplexität von MalDichHR)
Sei die Matrix $A \in \mathrm{Mat}_{n\times m}(R)$ eine $n \times m$-Matrix wie in Satz 10.2, wobei zusätzlich für die Zeilenanzahl $n = 2^p$ mit einem $p \in \mathbb{N}_+$ gelte. Ferner sei $s = 1$. (Für s=1 wird ein Algorithmus `Diag` $\in L$ im Algorithmus `MalDichHR` gar nicht benötigt.)

Für eine Teilmatrix der Größe $\frac{n}{2^{k-1}} \times m_l$ mit $k = 1, \ldots, p$ und $m_l := m - l \cdot \frac{n}{2^{k-1}}$ für $l = 0, \ldots, 2^{k-1} - 1$ werden in Schritt (6) bzw. Schritt (8) Teilmatrizen der Größe $\frac{n}{2^k} \times (m_l - \frac{n}{2^k})$ bzw. der Größe $\frac{n}{2^k} \times (m_l - \frac{n}{2^{k-1}})$ neu berechnet. Damit ergeben sich bei der Durchführung von Schritt (6) jeweils

$s_{kl}^{(6)} := \frac{n}{2^k}(m_l - \frac{n}{2^k})(\frac{n}{2^k} + 1)$ Additionen,

$s_{kl}^{(6)}$ Multiplikationen,

$r_{kl}^{(6)} := \frac{n}{2^k}(m_l - \frac{n}{2^k})$ Divisionen,

und in Schritt (8) jeweils

$s_{kl}^{(8)} := \frac{n}{2^k}(m_l - \frac{n}{2^{k-1}})(\frac{n}{2^k} + 1)$ Additionen,

$s_{kl}^{(8)}$ Multiplikationen,

$r_{kl}^{(8)} := \frac{n}{2^k}(m_l - \frac{n}{2^{k-1}})$ Divisionen.

Bei der Diagonalisierung von ganz A wird der Diagonalisierungsschritt 10.2 (7) rekursiv mit dem Dichotomie-Algorithmus durchgeführt, vgl. zur Veranschaulichung Beispiel 9.3. Daher ergeben sich insgesamt (mit $n = 2^p \iff p = \log_2 n$)

$$\sum_{k=1}^{p} \sum_{l=0}^{2^{k-1}-1} (s_{kl}^{(6)} + s_{kl}^{(8)}) \approx \tfrac{1}{3}mn^2 + mnp - \tfrac{5}{21}n^3 - \tfrac{1}{2}n^2p - 2mn + \tfrac{1}{2}n^2$$

$$\approx \tfrac{1}{3}mn^2 - \tfrac{5}{21}n^3 \quad \text{Additionen bzw. Multiplikationen}$$

$$\sum_{k=1}^{p} \sum_{l=0}^{2^{k-1}-1} (r_{kl}^{(6)} + r_{kl}^{(8)}) \approx pmn - 2mn + \tfrac{1}{2}n^2 - \tfrac{1}{2}n^2p \quad \text{Divisionen.}$$

Verwendet man zur Partitionierung die Halbierung wie in Bemerkung 10.1a) bzw. 10.1b) und verwendet man statt der zeilenweisen Rekursionformeln die matrizenweisen Rekursionsformeln aus Korollar 9.8, so folgt

10.4 Satz (Algorithmus MalDichHM)
Sei A eine $n \times m$-Matrix über einem Integritätsring R mit $n \le m$ und $\operatorname{rang} A = n$*, wobei außerdem noch $j_k = k$ für $k = 1, \ldots, n$ gelte und wobei ein Zeilentausch nicht nötig sei. Ferner seien $s \in \{1, \ldots, n\}$ und* `Diag` $\in L$ *beliebig. Die Algorithmen* `AusrLUMat` *und* `AusrROMat` *seien wie nachfolgend angegeben. Betrachte die folgenden Instruktionen:*

(0) Setze $t_u := 1$.

(1) Setze $P := (n)$ und $k_o := n$.

(2) (Partitionieren) Falls $k_o > s$ gilt, dann ersetze k_o durch den Ganzteil $\lfloor \frac{k_o+1}{2} \rfloor$, füge k_o im Tupel P an erster Position ein und wiederhole Schritt (2).

(3) (Blockende der oberen Hälfte) Setze $k_o := p_1$, wobei p_1 den ersten Koeffizienten des Tupels P bezeichne, und streiche p_1 aus P.

(4) (Diagonalisieren links oben) Falls $s > 1$ gilt, dann ersetze die obere Hälfte $\widetilde{T} := (a_{ij})_{\substack{i=1\ldots k_o \\ j=1\ldots m}}$ von A durch das Ergebnis von `Diag`$(\widetilde{T}, t_u)$.

(5) (Blockende der unteren Hälfte) Falls P leer ist, dann setze $k_u := k_o$, andernfalls setze $k_u := p_1$.

(6) (Ausräumen links unten) Im Fall $k_o < n$ ersetze die Teilmatrix $(a_{ij})_{\substack{i=k_o+1\dots k_u \\ j=1\dots m}}$ durch das Ergebnis von `AusrLUMat`(A, t_u, k_o, k_u).

(7) (Rekursiv diagonalisieren rechts unten) Falls $k_o < n$ gilt, dann setze $t_o := a_{k_o,k_o}$. Ersetze die Teilmatrix $\widetilde{T} := (a_{ij})_{\substack{i=k_o+1\dots k_u \\ j=k_o+1\dots m}}$ durch die Matrix, die man als Ergebnis erhält, wenn man die Schritte (1) bis (9) für die $(k_u - k_o) \times (m - k_o)$-Matrix $\widetilde{T}$ mit $t_u := t_o$ durchführt. (Diese Rekursion ist wieder so wie in Satz 10.2 (7) zu verstehen.)

(8) (Ausräumen rechts oben) Im Fall $k_o < n$ ersetze die Teilmatrix $(a_{ij})_{\substack{i=1\dots k_o \\ j=1\dots m}}$ durch das Ergebnis von `AusrROMat`(A, t_o, k_o, k_u). *Setze $t_u := 1$, ersetze k_o durch k_u und fahre fort bei Schritt (5).*

(9) Gib das Ergebnis A aus.

Dies stellt einen Algorithmus mit der Bezeichnung `MalDichHM`(A) *dar, der eine zu A äquivalente Matrix in Diagonalform, nämlich die Matrix $(\delta_{ij}^{(n)})$, zum Ergebnis hat.*

Hierbei wird mit dem Algorithmus `AusrLUMat`(A, t_u, k_o, k_u) *in A die Blockmatrix in den Zeilen $k_o + 1, \dots, k_u$ und den Spalten $1, \dots, k_o$ ausgeräumt.*

(R1) (Berechnen) Setze $B := (a_{ij})_{\substack{i=k_o+1\dots k_u \\ j=1\dots k_o}}$. Ersetze die Blockmatrix B zunächst durch die Blockmatrix $a_{11} \cdot (a_{ij})_{\substack{i=k_o+1\dots k_u \\ j=k_o+1\dots m}} - B \cdot (a_{ij})_{\substack{i=1\dots k_o \\ j=k_o+1\dots m}}$. Ersetze dann die neue Blockmatrix B durch $\frac{1}{t_u} \cdot B$.

(R2) (Ausräumen) Ersetze B durch die erweiterte $(k_u - k_o) \times m$-Matrix $(0 \mid B)$ mit der $(k_u - k_o) \times k_o$-Matrix 0.

(R3) Gib als Ergebnis die $(k_u - k_o) \times m$-Matrix B aus.

Mit dem Algorithmus `AusrROMat`(A, t_o, k_o, k_u) *wird in A die Blockmatrix in den Zeilen $1, \dots, k_o$ und den Spalten $k_o + 1, \dots, k_u$ ausgeräumt.*

(R1) (Berechnen) Setze $B := (a_{ij})_{\substack{i=1\dots k_o \\ j=k_o+1\dots k_u}}$. Ersetze die Blockmatrix B zunächst durch die Matrix $a_{k_o+1,k_o+1} \cdot (a_{ij})_{\substack{i=1\dots k_o \\ j=k_u+1\dots m}} - B \cdot (a_{ij})_{\substack{i=k_o+1\dots k_u \\ j=k_u+1\dots m}}$. Ersetze dann die neue Blockmatrix B durch $\frac{1}{t_o} \cdot B$.

(R2) (Ausräumen) Ersetze die Blockmatrix B durch die erweiterte $k_o \times m$-Matrix $(0 \mid B)$ mit der $k_o \times (m - k_u)$-Matrix 0.

(R3) Für $i = 1, \dots, k_o$ setze die Zeilenstufenkoeffizienten $b_{ii} := a_{k_o+1,k_o+1}$.

(R4) Gib als Ergebnis die $k_o \times m$-Matrix B aus.

Beweis:
Die Instruktionen sind identisch mit denen des Algorithmus `MalDichHR` aus Satz 10.2, bis auf die Schritte (6) und (8), in denen nun die Algorithmen `AusrLUMat` bzw. `AusrROMat` eingesetzt werden. Diese beiden Unteralgorithmen sind direkte Anwendungen von Korollar 9.8 (Dichotomie-Rekursion matrizenweise). □

Der obenstehende Algorithmus beruht nun auf der Multiplikation bestimmter Teilmatrizen. Daher ist es günstig, über eine "gute" Matrizenmultiplikation zu verfügen.

Ist zur Multiplikation zweier quadratischer $n \times n$-Matrizen ein Algorithmus mit der Komplexität $O(n^{2+\beta})$ gegeben, wobei β reellwertig mit $0 < \beta \leq 1$ ist, so ergibt sich für die Multiplikation einer $l \times n$-Matrix mit einer $n \times m$-Matrix die Komplexität $O(ln^\beta m)$, für $l, m, n \in \mathbb{N}_+$. Diese Überlegung fließt ein, wenn nun die Komplexität des Dichotomie-Algorithmus ermittelt wird.

10.5 Bemerkungen (Komplexität von MalDichHM)
Sei nun die Matrix $A \in \mathrm{Mat}_{n\times m}(R)$ eine $n \times m$-Matrix wie in Satz 10.4, wobei zusätzlich

für die Zeilenanzahl $n = 2^p$ mit einem $p \in \mathbb{N}_+$ gelte. Ferner sei $s = 1$. (In diesem Fall ist zur Durchführung von `MalDichHM` die Wahl eines Algorithmus `Diag` $\in L$ nicht nötig.)

Für eine Teilmatrix der Größe $\frac{n}{2^{k-1}} \times m_l$ mit $k = 1, \ldots, p$ und $m_l := m - l \cdot \frac{n}{2^{k-1}}$ für $l = 0, \ldots, 2^{k-1} - 1$ werden in Schritt (6) bzw. Schritt (8) Teilmatrizen der Größe $\frac{n}{2^k} \times (m_l - \frac{n}{2^k})$ bzw. der Größe $\frac{n}{2^k} \times (m_l - \frac{n}{2^{k-1}})$ neu berechnet, vgl. Bemerkung 10.3 zur Komplexität von `MalDichHR` bei zeilenweiser statt matrizenweiser Vorgehensweise.
Damit ergeben sich bei der Durchführung von Schritt (6) jeweils

$r_{kl}^{(6)} := \frac{n}{2^k}(m_l - \frac{n}{2^k})$ Divisionen (genau wie in Bemerkung 10.3),
$\widetilde{s}_{kl}^{(6)} := r_{kl}^{(6)} + \widetilde{r}_{kl}^{(6)}$ Multiplikationen,

wobei $\widetilde{r}_{kl}^{(6)}$ die Anzahl der Multiplikationen bezeichnet, die nötig sind, um eine $\frac{n}{2^k} \times \frac{n}{2^k}$-Matrix mit einer $\frac{n}{2^k} \times (m_l - \frac{n}{2^k})$-Matrix zu multiplizieren, also $\widetilde{r}_{kl}^{(6)} \approx O((\frac{n}{2k})^{1+\beta}(m_l - \frac{n}{2^k}))$.
In Schritt (8) erfolgen jeweils

$r_{kl}^{(8)} := \frac{n}{2^k}(m_l - \frac{n}{2^{k-1}})$ Divisionen (genau wie in Bemerkung 10.3),
$s_{kl}^{(8)} := r_{kl}^{(8)} + \widetilde{r}_{kl}^{(8)}$ Multiplikationen,

wobei $\widetilde{r}_{kl}^{(8)}$ die Anzahl der Multiplikationen angibt, die nötig sind, um eine $\frac{n}{2^k} \times \frac{n}{2^k}$-Matrix mit einer $\frac{n}{2^k} \times (m_l - \frac{n}{2^{k-1}})$-Matrix zu multiplizieren, also $\widetilde{r}_{kl}^{(8)} \approx O((\frac{n}{2k})^{1+\beta}(m_l - \frac{n}{2^{k-1}}))$.
Bei der Diagonalisierung von ganz A wird der Diagonalisierungsschritt 10.4 (7) rekursiv mit dem Dichotomie-Algorithmus durchgeführt, vgl. zur Veranschaulichung Beispiel 9.3. Daher ergeben sich insgesamt (mit $n = 2^p \iff p = \log_2 n$)

$$\sum_{k=1}^{p} \sum_{l=0}^{2^{k-1}-1} (r_{kl}^{(6)} + r_{kl}^{(8)}) \approx pmn - 2mn + \tfrac{1}{2}n^2 - \tfrac{1}{2}n^2 p =: \varepsilon \quad \text{Divisionen, vgl. 10.3.}$$

$$\sum_{k=1}^{p} \sum_{l=0}^{2^{k-1}-1} (\widetilde{s}_{kl}^{(6)} + \widetilde{s}_{kl}^{(8)}) \approx \varepsilon + \eta \approx O(mn^{1+\beta}) \quad \text{Multiplikationen,}$$

wobei gilt

$$\eta := \sum_{k=1}^{p} \sum_{l=0}^{2^{k-1}-1} (\widetilde{r}_{kl}^{(6)} + \widetilde{r}_{kl}^{(8)}) \approx O(mn^{1+\beta}).$$

Für die gewöhnliche Matrizenmultiplikation mit der Komplexität $O(n^3)$, also mit $\beta = 1$, deckt sich die Komplexität des matrizenweisen Dichotomie-Algorithmus mit der Komplexität des zeilenweisen Dichotomie-Algorithmus, vgl. Bemerkung 10.3.

Verwendet man nun zur Partitionierung die Verdopplungsmethode (Bemerkung 10.1c) bzw. d)) und die zeilenweisen Rekursionsformeln (Lemma 9.5), so folgt

10.6 Satz (Algorithmus MalDichVR)
Sei A eine $n \times m$-Matrix über einem Integritätsring R mit $n \leq m$ und $\operatorname{rang} A = n$, *wobei außerdem noch $j_k = k$ für $k = 1, \ldots, n$ gelte und wobei ein Zeilentausch nicht nötig sei. Ferner seien $s \in \{1, \ldots, n\}$ und* `Diag` $\in L$ *beliebig. Die Algorithmen* `AusrLURow` *und* `AusrRORow` *seien wie bei dem Algorithmus* `MalDichHR` *aus Satz 10.2. Betrachte die folgenden Instruktionen:*

(0) Setze $t_u := 1$.
(1) (Blockende der oberen bzw. unteren Hälfte) Setze $k_o := \min\{s, n\}$, $k_u := \min\{2k_o, n\}$.
(2) (Diagonalisieren links oben) Falls $s > 1$ *gilt, dann ersetze die obere Hälfte* $\widetilde{T} := (a_{ij})_{\substack{i=1\dots k_o \\ j=1\dots m}}$ *von* A *durch das Ergebnis von* $\mathtt{Diag}(\widetilde{T}, t_u)$.
(3) (Ausräumen links unten) Falls $k_o < n$ *gilt, dann ersetze die Zeilen* $i = k_o + 1, \dots, k_u$ *jeweils durch das Ergebnis von* $\mathtt{AusrLURow}(A, i, k_o, t_u)$.
(4) (Rekursiv diagonalisieren rechts unten) Falls $k_o < n$ *gilt, dann setze* $t_o := a_{k_o,k_o}$ *und ersetze die Teilmatrix* $\widetilde{T} := (a_{ij})_{\substack{i=k_o+1\dots k_u \\ j=k_o+1\dots m}}$ *durch die Matrix, die man als Ergebnis erhält, wenn man die Schritte (1) bis (6) für die* $(k_u - k_o) \times (m - k_o)$*-Matrix* $\widetilde{T}$ *mit* $t_u := t_o$ *durchführt. (Diese Rekursion ist wieder so wie in Satz 10.2 (7) zu verstehen.)*
(5) (Ausräumen rechts oben) Im Fall $k_o < n$ *ersetze die Zeilen* $i = 1, \dots, k_o$ *jeweils durch das Ergebnis von* $\mathtt{AusrRORow}(A, i, k_o, k_u, t_o)$. *Setze* $t_u := 1$, *ersetze* k_o *durch* k_u, k_u *durch* $\min\{2k_o, n\}$ *und fahre fort bei Schritt (3).*
(6) Gib das Ergebnis A *aus.*

Dies stellt einen Algorithmus mit der Bezeichnung $\mathtt{MalDichVR}(A)$ *dar, der eine zu* A *äquivalente Matrix in Diagonalform, nämlich* $(\delta_{ij}^{(n)})$, *zum Ergebnis hat.*

Beweis:
Die Instruktionen sind genau wie beim Algorithmus `MalDichHR`, außer daß die Partitionierung wie in Bemerkung 10.1c) bzw. 10.1d) durch Verdopplung vorgenommen wird. Dadurch vereinfacht sich die Wahl von k_o und k_u in obigem Schritt (1) bzw. (5). □

10.7 Bemerkung (Komplexität von MalDichVR)
Sei die Matrix $A \in \mathrm{Mat}_{n\times m}(R)$ eine $n \times m$-Matrix wie in Satz 10.6, wobei zusätzlich für die Zeilenanzahl $n = 2^p$ mit einem $p \in \mathbb{N}_+$ gelte. Ferner sei $s = 1$. (Für s=1 wird ein Algorithmus $\mathtt{Diag} \in L$ im Algorithmus `MalDichVR` gar nicht benötigt.)

Genau wie in Bemerkung 10.3 zur Komplexität von `MalDichHR` ergeben sich insgesamt (mit $n = 2^p \iff p = \log_2 n$)

$$\sum_{k=1}^{p} \sum_{l=0}^{2^{k-1}-1} (s_{kl}^{(6)} + s_{kl}^{(8)}) \approx \tfrac{1}{3}mn^2 + mnp - \tfrac{5}{21}n^3 - \tfrac{1}{2}n^2p - 2mn + \tfrac{1}{2}n^2$$
$$\approx \tfrac{1}{3}mn^2 - \tfrac{5}{21}n^3 \quad \text{Additionen bzw. Multiplikationen}$$
$$\sum_{k=1}^{p} \sum_{l=0}^{2^{k-1}-1} (r_{kl}^{(6)} + r_{kl}^{(8)}) \approx pmn - 2mn + \tfrac{1}{2}n^2 - \tfrac{1}{2}n^2p \quad \text{Divisionen.}$$

Verwendet man zur Partitionierung die Verdopplungsmethode (Bemerkung 10.1c) bzw. d)) und verwendet man die matrizenweisen Rekursionsformeln (Korollar 9.8), so folgt

10.8 Satz (Algorithmus MalDichVM)
Sei A *eine* $n \times m$*-Matrix über einem Integritätsring* R *mit* $n \le m$ *und* $\operatorname{rang} A = n$, *wobei außerdem noch* $j_k = k$ *für* $k = 1, \dots, n$ *gelte und wobei ein Zeilentausch nicht nötig sei. Ferner seien* $s \in \{1, \dots, n\}$ *und* $\mathtt{Diag} \in L$ *beliebig. Die Algorithmen* `AusrLUMat` *und* `AusrROMat` *seien wie bei dem Algorithmus* `MalDichHM` *aus Satz 10.4. Betrachte die folgenden Instruktionen:*
(0) Setze $t_u := 1$.
(1) (Blockende der oberen bzw. unteren Hälfte) Setze $k_o := \min\{s, n\}$, $k_u := \min\{2k_o, n\}$.

(2) (Diagonalisieren links oben) Falls $s > 1$ gilt, dann ersetze die obere Hälfte $\widetilde{T} := (a_{ij})_{\substack{i=1\dots k_o \\ j=1\dots m}}$ von A durch das Ergebnis von $\texttt{Diag}(\widetilde{T}, t_u)$.

(3) (Ausräumen links unten) Falls $k_o < n$ gilt, dann ersetze die Teilmatrix $(a_{ij})_{\substack{i=k_o+1\dots k_u \\ j=1\dots m}}$ durch das Ergebnis von $\texttt{AusrLUMat}(A, t_u, k_o, k_u)$.

(4) (Rekursiv diagonalisieren rechts unten) Falls $k_o < n$ gilt, dann setze $t_o := a_{k_o,k_o}$ und ersetze die Teilmatrix $\widetilde{T} := (a_{ij})_{\substack{i=k_o+1\dots k_u \\ j=k_o+1\dots m}}$ durch die Matrix, die man als Ergebnis erhält, wenn man die Schritte (1) bis (6) für die $(k_u - k_o) \times (m - k_o)$-Matrix $\widetilde{T}$ mit $t_u := t_o$ durchführt. (Diese Rekursion ist wieder so wie in Satz 10.2 (7) zu verstehen.)

(5) (Ausräumen rechts oben) Im Fall $k_o < n$ ersetze die Teilmatrix $(a_{ij})_{\substack{i=1\dots k_o \\ j=1\dots m}}$ durch das Ergebnis von $\texttt{AusrROMat}(A, t_o, k_o, k_u)$. *Setze $t_u := 1$, ersetze k_o durch k_u, k_u durch $\min\{2k_o, n\}$ und fahre fort bei Schritt (3).*

(6) Gib das Ergebnis A aus.

Dies stellt einen Algorithmus namens $\texttt{MalDichVM}(A)$ *dar, der eine zu A äquivalente Matrix in Diagonalform, nämlich $(\delta_{ij}^{(n)})$, zum Ergebnis hat.*

Beweis:
Die Instruktionen sind genau wie beim Algorithmus `MalDichVR` in Satz 10.6, außer daß in Schritt (3) und (5) nun die Unteralgorithmen `AusrLUMat` bzw. `AusrROMat` aus Satz 10.4 verwendet werden. □

10.9 Bemerkung (Komplexität von MalDichVM)

Sei nun die Matrix $A \in \mathrm{Mat}_{n\times m}(R)$ eine $n \times m$-Matrix wie in Satz 10.8, wobei zusätzlich für die Zeilenanzahl $n = 2^p$ mit einem $p \in \mathbb{N}_+$ gelte. Ferner sei $s = 1$. (In diesem Fall ist zur Durchführung von `MalDichVM` die Wahl eines Algorithmus $\texttt{Diag} \in L$ nicht nötig.)

Wie in Bemerkung 10.5 zur Komplexität von `MalDichHM` ergeben sich auch hier insgesamt (mit $n = 2^p \iff p = \log_2 n$)

$\approx pmn - 2mn + \frac{1}{2}n^2 - \frac{1}{2}n^2 p =: \varepsilon$ Divisionen

$\approx O(mn^{1+\beta})$ Multiplikationen.

10.10 Bemerkungen zur Implementierung (MalDichHR, MalDichHM, MalDichVM, MalDichVR)

a) Partitioniert man gemäß Bemerkung 10.1b) bzw. 10.1d) nur bis zu einer gewissen Blockgröße und diagonalisiert man größere Teilmatrizen dann mit einem der bekannten Diagonalisierungsalgorithmen, so muß man diese leicht abändern.
Dazu sind alle Algorithmen `B1TMD`, `B2TMD`, `B2TMDMD`, `B1DMD`, `B2DMD`, `B2DMDMD`, `B1DMDFB`, `B2DMDFB`, `B2DMDMDFB`, `Mal1DMD` so programmiert, daß sie, falls sie nur mit dem Argument A aufgerufen werden, die Matrix A trigonalisieren bzw. diagonalisieren, wie in den Abschnitten 3 bis 8 beschrieben. Werden zwei Argumente A und t eingegeben, so wird A trigonalisiert bzw. diagonalisiert, indem für die erste Rekursion nicht der Teiler $t = 1$, sondern der Teiler $t = t_u$ verwendet wird, der sich aus dem Dichotomie-Algorithmus ergibt. (Dieser Teiler $t = t_u$ ist nämlich der Teiler t_k aus Lemma 9.5, der ungleich 1 sein kann, wenn die zu diagonalisierende Teilmatrix nicht ganz links oben, sondern "mitten" in A liegt, wenn also zu berücksichtigen ist, daß obendrüber bereits eine Teilmatrix diagonalisiert worden ist.)
Im einzelnen werden die Modifikationen wie folgt vorgenommen. Betrachtet man die sechs erstgenannten Bareiss-Algorithmen aus den Sätzen 3.7, 4.3, 4.6, 5.6, 6.3 und 6.6, so wird dort jeweils in Schritt (1) statt $t := 1$ das gegebene Argument t verwendet. Die Forward-Backup-Algorithmen aus Satz 7.5 werden so modifiziert, daß dort

Schritt (1) mit `B1TMD`(A,t) bzw. `B2TMD`(A,t) bzw. `B2TMDMD`(A,t) durchgeführt wird. Der Algorithmus `Mal1DMD` wird abgeändert, indem in der in Schritt (5) verwendete Unteralgorithmus `Mal1DMDRow`$(A,\{j_1,\ldots,j_{k+1}\},k,c)$ ersetzt wird durch den Algorithmus `Mal1DMDRow`$(A,\{j_1,\ldots,j_{k+1}\},k,c,t)$. Letzterer ist genau wie ersterer definiert, außer daß alle in Schritt (R1) ermittelten neuen Koeffizienten zusätzlich noch durch t geteilt werden.

b) Auch die Dichotomie-Algorithmen sind so programmiert, daß mehrere Argumente (maximal vier) übergeben werden können, nämlich die zu diagonalisierende Matrix A, optional die Schrittgröße s und ein Diagonalisierungs-Verfahren `Diag` und schließlich der Teiler t, der als Argument übergeben wird, wenn ein Diagonalisierungsschritt rekursiv mit dem Dichotomie-Algorithmus durchlaufen wird. Damit wird beispielsweise der Schritt (7) beim Algorithmus `MalDichHR` aus Satz 10.2 rekursiv durch den Aufruf $\widetilde{T}$:= `MalDichHR`$(\widetilde{T},s,$`Diag`$,t_o)$ erledigt. Analoges gilt für alle anderen Dichotomie-Algorithmen.
Möchte man eine Matrix A mit einem der Dichotomie-Algorithmen diagonalisieren, genügt es, als Argument nur A eingegeben; die drei übrigen Argumente s, `Diag` und t sind dann als Voreinstellungen vorhanden, nämlich z.B. $s=1$ und $t=1$.

c) Da das Dichotomie-Verfahren schon kompliziert genug ist, wird darauf verzichtet, es für den allgemeinstmöglichen Fall einer $n\times m$-Matrix A mit beliebigen $n,m\in N_+$ und mit beliebiger Zeilenstufenverteilung zu programmieren. In diesem Falle müßte man nämlich vor jedem Ausräumungsschritt nachprüfen, wo dann in der nächsten Blockmatrix die Zeilenstufen zu liegen kommen und ob eventuell Zeilen getauscht werden müssen. Gelten für eine $n\times m$-Matrix A die vier Voraussetzungen (V1) bis (V4) nicht und ist die Verteilung der Zeilenstufen bekannt, so können diese Bedingungen durch Zeilen- und Spaltentausch bzw. durch das Weglassen bestimmter Zeilen hergestellt werden.

In den vier vorgestellten Algorithmen wird eine Matrix stets partitioniert, indem die Matrix halbiert bzw. annähernd halbiert wird, falls die Zeilenanzahl keine 2-Potenz ist. Denkbar ist jedoch jede beliebige Partitionierung.

10.11 Bemerkungen (Forward-Backup-Verfahren)

Sei $A\in \mathrm{Mat}_{n\times m}(R)$ eine $n\times m$-Matrix über einem Integritätsring R mit den Voraussetzungen (V1) bis (V4). Als Partitionierung wählt man die Methode, stets die oberste Zeile abzuschneiden. Mit den Bezeichnungen von Abschnitt 9 bedeutet dies, daß man bei jeder Partition $\mu=1$ wählt.

a) Auf diese Art und Weise erhält man gerade das Forward-Backup-Verfahren `B1DMDFB`. Anschaulich für eine $6\times m$-Matrix dargestellt, werden bei dieser Partition die Blockmatrizen (die in diesem Falle Zeilen und Spalten sind), in der folgenden Reihenfolge ausgeräumt:

$$\left(\begin{array}{c|c|c|c|c|c|c}
* & & & & & 10 & * \\ \hline
 & * & & & & 9 & * \\ \hline
 & & * & & & 8 & * \\ \hline
 & & & * & & 7 & * \\ \hline
 & & & & * & 6 & * \\ \hline
1 & 2 & 3 & 4 & 5 & * & *
\end{array}\right)$$

Hierbei symbolisiert $*$ die verbleibenden Koeffizienten auf der Diagonalen bzw. die verbleibenden letzten $(m-6)$ Koeffizienten.

b) *(Formeln)* Aus Lemma 9.5(1) ergibt sich für $k = 1, 2, \ldots, n-1$ und $\mu = 1$ für die Zeilen $i = k+2, \ldots, n$ und die Spalten $j = 1, \ldots, m$ die Formel

$$h_{ij}^{(k\mu,2)} = \frac{1}{t_k} \cdot (t_\mu \cdot h_{ij} - h_{i,k+1} \cdot h_{k+1,j}) = \frac{1}{t_k} \cdot [k+1\ i][k+1\ j](H),$$

also die Bareiss-1-Schritt-Formel.
Aus Lemma 9.5(2) ergibt sich mit $k = 1, 2, \ldots, n-1$ und $\mu = 1$ für die Zeile $i = k+1$ und die Spalten $j = 1, \ldots, m$ die Formel

$$h_{ij}^{(k\mu,4)} = \frac{1}{t_\mu} \cdot \Big[t_n \cdot \widetilde{h}_{ij} - (\widetilde{h}_{i,k+2} \ldots \widetilde{h}_{in}) \cdot \begin{pmatrix} \widetilde{h}_{k+2,j} \\ \vdots \\ \widetilde{h}_{nj} \end{pmatrix} \Big],$$

also die Backup-Formel. Insgesamt erhält man somit die in Lemma 7.2 angegebenen Rekursionsformeln für das Backup-Verfahren.

c) Als zusätzliche Verbesserung kann man den Forward-Schritt mit einem Bareiss-2-Schritt-Verfahren statt mit dem Bareiss-1-Schritt-Verfahren durchführen, vgl. dazu Satz 7.5 (Algorithmen `B1DMDFB, B2DMDFB` und `B2DMDMDFB`).

d) Wie in Abschnitt 7 ausführlich dargestellt, kann A auch weniger strenge Voraussetzungen erfüllen, d.h. A kann ganz und gar beliebig sein.

10.12 Bemerkungen (Malashonok-1-Schritt-Verfahren)
Sei $A \in \mathrm{Mat}_{n\times m}(R)$ eine $n \times m$-Matrix über einem Integritätsring R mit den Voraussetzungen (V1) bis (V4). Man partitioniert A, indem man stets die unterste Zeile abschneidet. Mit den Bezeichnungen aus Abschnitt 9 heißt dies also, bei den Partitionen nacheinander $\mu = n-1$, $\mu = n-2, \ldots, \mu = 1$ zu wählen.

a) Mit dieser Partitionierungsmethode ergibt sich das Malashonok-1-Schritt-Verfahren. Anschaulich für eine $6 \times m$-Matrix dargestellt, räumt man bei dieser Partitionierung die Blockmatrizen (die in diesem Falle Zeilen und Spalten sind) in der folgenden Reihenfolge aus:

$$\left(\begin{array}{c|c|c|c|c|c|c} * & 2 & 4 & 6 & 8 & 10 & * \\ \hline 1 & * & & & & & * \\ \hline 3 & & * & & & & * \\ \hline 5 & & & * & & & * \\ \hline 7 & & & & * & & * \\ \hline 9 & & & & & * & * \end{array}\right)$$

Hierbei symbolisiert $*$ die verbleibenden Koeffizienten auf der Diagonalen bzw. die verbleibenden letzten $(m-6)$ Koeffizienten.

b) *(Formeln)* Aus Lemma 9.5 (1) ergibt sich mit $k = 0$ für die Zeile $i = \mu + 1$ und die Spalten $j = 1, \ldots, m$

$$h_{ij}^{(k\mu,2)} = t_\mu \cdot h_{ij} - (h_{i1} \ldots h_{i\mu}) \cdot \begin{pmatrix} h_{1j} \\ \vdots \\ h_{\mu j} \end{pmatrix},$$

und aus Lemma 9.5 (2) ergibt sich mit $k = 0$ für die Zeilen $i = 1, \ldots, \mu$ und die Spalten $j = 1, \ldots, m$

$$h_{ij}^{(k\mu,4)} = \frac{1}{t_\mu} \cdot (t_n \cdot \widetilde{h}_{ij} - \widetilde{h}_{i,\mu+1} \cdot \widetilde{h}_{\mu+1,j}) = \frac{1}{t_\mu} \cdot [\mu + 1\ i][\mu + 1\ j](\widetilde{H}),$$

also die Bareiss-1-Schritt-Formel. Insgesamt erhält man somit die in Lemma 8.2 angegebenen Formeln zur Malashonok-1-Schritt-Rekursion.

c) In Abschnitt 8 wird dieses Verfahren viel allgemeiner für vollkommen beliebige Matrizen durchgeführt.

10.13 Bemerkung zur Implementierung

Wie man an den beiden vorangegangenen Bemerkungen sehen kann, lassen sich das Forward-Backup- und das Malashonok-1-Schritt-Verfahren als Spezialfälle des Dichotomie-Verfahrens auffassen. Man könnte folglich die Zeilenalgorithmen `AusrLURow` bzw. `AusrRORow` aus Satz 10.2 so allgemein implementieren, daß man sie anstelle von `BackupRow` bzw. `Mal1DMDRow` in Satz 7.5 bzw. 8.4 verwenden kann. Umgekehrt könnte man auch letztere so allgemein formulieren, daß sie die Algorithmen `AusrLURow` bzw. `AusrRORow` ersetzen.

Die Kokosnüsse fallen nach und nach
in den Korb.

(Aus Afrika)

Im folgenden Kapitel werden die Verfahren aus den beiden vorangegangenen Kapiteln modifiziert, indem eine spezielle, mit $\otimes$ symbolisierte Multiplikation verwendet wird.

Soll eine gegebene $n \times m$-Matrix $A \in \mathrm{Mat}_{n\times m}(R)$ mit $n \leq m$ trigonalisiert bzw. diagonalisiert werden, so werden zunächst die Zeilenstufenkoeffizienten a_{kj_k}, $k = 1, \ldots, n$, durch unabhängige Variablen $X_1, \ldots, X_n$ substituiert. Dann wird bei jeder Rekursion statt der Division im Integritätsring R (bzw. im zugehörigen Quotientenkörper) eine spezielle Multiplikation $\otimes$ im Erweiterungsring $R[X_1, \ldots, X_n]$ verwendet. Bei dem damit erzielten Endergebnis wird schließlich die Rücksubstitution durchgeführt.

Der Vorteil dieser Multiplikation $\otimes$ besteht darin, bei den rekursiv ermittelten Koeffizienten nur solche Terme zu berechnen, die auch tatsächlich zum Endergebnis beitragen. Anders ausgedrückt, statt in den Rekursionsformeln am Produkt $f \cdot g$ zweier Polynome die Division mit Rest durchzuführen und damit den Quotienten q sowie den Rest r zu erhalten, kann man mittels $\otimes$ den Quotienten q ermitteln, ohne daß man das Produkt $f \cdot g$ und den Rest r berechnet. (Ist im folgenden von Division mit Rest die Rede, so meint man damit den eindeutigen Divisionsalgorithmus.)

Allgemeiner als von Sasaki und Murao dargestellt, läßt sich nicht nur der Algorithmus `B1TMD`, sondern darüberhinaus lassen sich auch andere Algorithmen aus den Kapiteln I und II modifizieren, vgl. [SM].

11 Grundlagen

In diesem Abschnitt wird zunächst eine spezielle, mit $\otimes$ symbolisierte Multiplikation eingeführt. Dazu sei wie üblich R ein Integritätsring, ferner sollen die Unbestimmten $X_1, \ldots, X_n$ mit $n \in \mathbb{N}_+$ unabhängig über R sein und nicht schon in R liegen.

Für ein Polynom f aus $R[X_1, \ldots, X_n]$ bezeichne im folgenden $\deg f$ stets den Grad von f in Bezug auf die Variablen $X_1, \ldots, X_n$. (Der Integritätsring R mag durchaus noch weitere Variablen enthalten, die für die Gradbestimmung aber nicht berücksichtigt werden sollen.)

11.1 Definition (Multiplikation $\otimes$)
Seien f und g zwei Polynome aus $R[X_1,\ldots,X_n]$, die linear in $X_1,\ldots,X_n$ sind und für ein $i \in \{1,\ldots,n\}$ zerlegt werden in

$$f = f_1 \cdot X_i + f_0 \qquad \text{und} \qquad g = g_1 \cdot X_i + g_0$$

mit eindeutig bestimmten Polynomen $f_0, f_1, g_0, g_1 \in R[X_1,\ldots,X_n]$, wobei f_0 und g_0 keine Terme enthalten, die Vielfaches von X_i sind. In dieser Situation definiert man für ein beliebiges $i \in \{1,\ldots,n\}$ und $k \in \{0,\ldots,n\}$ rekursiv die spezielle Multiplikation

$$f \overset{k}{\underset{i}{\otimes}} g := \begin{cases} f \cdot g & \text{falls } i > k \\ (f_1 \otimes_{i+1}^{k} g_1) \cdot X_i + f_1 \otimes_{i+1}^{k} g_0 + f_0 \otimes_{i+1}^{k} g_1 & \text{falls } i \le k. \end{cases}$$

11.2 Bemerkungen (Wohldefiniertheit und Assoziativität)
a) (*Wohldefiniertheit*) Die obige Multiplikation $\otimes$ ist wohldefiniert, denn einerseits existiert in Bezug auf X_i für alle $i = 1,\ldots,n$ eine Zerlegung der Polynome f und g in die Bestandteile f_0, f_1, g_0, g_1. Diese letztgenannten Polynome sind eindeutig bestimmt, sie sind ebenfalls linear in $X_1,\ldots,X_n$ und lassen sich, falls $i < n$ gilt, wiederum eindeutig zerlegen (in Bezug auf X_{i+1}). Andererseits endet die rekursiv definierte Multiplikation nach endlich vielen Schritten, da schließlich die Situation $i > k$ erreicht ist, und in diesem Falle ist $\otimes_i^k$ durch die normale Multiplikation $\cdot$ gegeben.

b) (*Assoziativität*) Wie sich durch Nachrechnen bestätigen läßt, ist die spezielle Multiplikation $\otimes$ assoziativ.

11.3 Satz (Algorithmus MultSM)
Seien $f, g \in R[X_1,\ldots,X_n]$ zwei Polynome, die linear in $X_1,\ldots,X_n$ sind. Ferner seien $k \in \{0,\ldots,n\}$ und $i \in \{1,\ldots,n\}$ beliebig. Die folgenden Instruktionen werden unter der Bezeichnung `MultSM`(f,g,i,k) *zusammengefaßt:*
(1) Falls $i > k$ gilt, dann gib das Ergebnis $f \cdot g$ aus.
(2) (Rekursion) Andernfalls berechne durch Einsetzen die vier Polynome
$f_0 = f(X_1,\ldots,X_{i-1},0,X_{i+1},\ldots,X_n)$, $f_1 = (f - f_0)(X_1,\ldots,X_{i-1},1,X_{i+1},\ldots,X_n)$,
$g_0 = g(X_1,\ldots,X_{i-1},0,X_{i+1},\ldots,X_n)$, $g_1 = (g - g_0)(X_1,\ldots,X_{i-1},1,X_{i+1},\ldots,X_n)$.
Erhöhe i um 1 und wiederhole die Schritte (1) und (2) jeweils für f_1, g_1, i, k, für f_1, g_0, i, k und für f_0, g_1, i, k, d.h. ermittle jeweils die mit a, b, c bezeichneten Ergebnisse von `MultSM`(f_1, g_1, i, k), `MultSM`(f_1, g_0, i, k) *bzw.* `MultSM`(f_0, g_1, i, k). *Gib das Ergebnis $a \cdot X_i + b + c$ aus.*
Dies ist ein Algorithmus, der als Ergebnis $f \otimes_1^k g$ ausgibt.
Beweis:
Daß die beiden Schritte nur endlich oft wiederholt werden, folgt aus der Bemerkung 11.2a), daß das Ergebnis wie behauptet ist, folgt sofort aus der Definition von $\otimes$ (11.1). Die Bestandteile f_0, g_0 bzw. f_1, g_1 lassen sich einfach ermitteln, indem man $X_i = 0$ in f und g bzw. $X_i = 1$ in $f - f_0$ und $g - g_0$ einsetzt. Damit erhält man die Zerlegungen $f = f_1 \cdot X_1 + f_0$ bzw. $g = g_1 \cdot X_i + g_0$. Alternativ läßt sich dies Zerlegung auch erhalten, indem man an f bzw. g die Division mit Rest durch X_i durchführt. □

11.4 Bemerkung zur Implementierung (MultSM)
Genau wie oben in Satz 11.3 beschrieben, wird bei der Implementierung von `MultSM` der Schritt (2) rekursiv ausgeführt, wobei die angegebenen Argumente f_1, g_0, i, k bzw. f_0, g_1, i, k verwendet werden. Die nötigen Zerlegungen der Polynome f und g lassen sich so gewinnen, wie in Schritt (2) angegeben.

Was bewirkt diese Multiplikation $\otimes$ nun eigentlich? Die Antwort liefern die folgende Bemerkung und das anschließende Beispiel.

11.5 Bemerkung (Division mit Rest)
Multipliziert man zwei Polynome f und g aus $R[X_1,\ldots,X_n]$ miteinander und führt man dann für beliebiges $k \in \{1,\ldots,n\}$ am Produkt $f\cdot g$ die Division durch das Monom $X_1\cdots X_k$ mit Rest durch, so erhält man

$$f\cdot g = q\cdot(X_1\cdots X_k)+r$$

mit eindeutig bestimmten Polynomen q und $r \in R[X_1,\ldots,X_n]$, wobei $\deg r \le \deg(f\cdot g)$ gilt. Hierbei werden unter r all die Terme von $f\cdot g$ zusammengefaßt, die nicht Vielfaches von $X_1\cdots X_k$ sind.

Ist man ausschließlich an dem Polynom q interessiert, genügt es, statt der normalen Multiplikation $f\cdot g$ und der anschließenden Division die eben eingeführte spezielle Multiplikation $f\otimes_1^k g$ auszuführen. Mit dieser Multiplikation erhält man das gewünschte Polynom q, nämlich $q = f\otimes_1^k g$. Für den Fall $k=1$ ergibt sich dies sofort aus der Definition von $\otimes_1^1$, denn mit den Zerlegungen von f und g aus Definition 11.1 gilt

$$\begin{aligned} f\cdot g &= f_1\cdot g_1\cdot X_1^2 + (f_1\cdot g_0 + g_1\cdot f_0)\cdot X_1 + f_0\cdot g_0 \quad \text{und}\\ f\overset{1}{\underset{1}{\otimes}} g &= f_1\cdot g_1\cdot X_1 + (f_1\cdot g_0 + g_1\cdot f_0). \end{aligned}$$

Für $k>1$ folgt induktiv, daß bei der rekursiv durchgeführten Multiplikation $\otimes_1^k$ all die Terme weggelassen werden, die zum Divisionsrest gehören. Damit gilt also

$$f\cdot g = f\overset{k}{\underset{1}{\otimes}} g\cdot(X_1\cdots X_k)+r.$$

Noch einmal deutlich in Worte gefaßt, mit der Multiplikation $f\otimes_1^k g$ ermittelt man die Terme von $f\cdot g$, die Vielfaches des Monoms $X_1\cdots X_k$ sind, ohne das Produkt $f\cdot g$ oder den Divisionsrest r zu berechnen.

Um an dieser Stelle die Multiplikation $\otimes_1^k$ verständlicher zu machen, werden nun mehrere Beispiele vorgeführt.

11.6 Beispiel (Multiplikation $\otimes$)
a) Die Multiplikation $\otimes_i^k$ wird zunächst für den einfachen Fall $i=k=1$ an den beiden folgenden Polynomen demonstriert:

$$\begin{aligned} f = X_1X_2+12 &= f_1X_1+f_0 \qquad \text{mit} \quad \begin{cases} f_1 = X_2\\ f_0 = 12 \end{cases}\\ g = X_1-2 &= g_1X_1+g_0 \qquad \text{mit} \quad \begin{cases} g_1 = 1\\ g_0 = -2. \end{cases} \end{aligned}$$

Definitionsgemäß ergibt sich

$$\begin{aligned} f\overset{1}{\underset{1}{\otimes}} g &= (f_1\overset{1}{\underset{2}{\otimes}} g_1)X_1 + f_1\overset{1}{\underset{2}{\otimes}} g_0 + f_0\overset{1}{\underset{2}{\otimes}} g_1 =\\ &= (f_1\cdot g_1)X_1 + f_1\cdot g_0 + f_0\cdot g_1 =\\ &= X_2\cdot 1\cdot X_1 + X_2\cdot(-2) + 12\cdot 1 = X_1X_2 - 2X_2 + 12 =: \widetilde{q}, \end{aligned}$$

wobei die Multiplikation $\otimes_2^1$ gleich der normalen Multiplikation $\cdot$ ist. Berechnet man andererseits zuerst das Produkt $f \cdot g$ und führt man daran die Division durch X_1 mit Rest durch, so folgt

$$\begin{aligned} f \cdot g &= f_1 g_1 X_1^2 + (f_1 g_0 + f_0 g_1) X_1 + f_0 g_0 = \\ &= q \cdot X_1 + r \end{aligned}$$

mit dem Quotienten $q = f_1 g_1 X_1 + (f_0 g_1 + f_1 g_0) = \widetilde{q}$ und dem Rest $r = f_0 g_0$. Durch die Multiplikation $\otimes$ erspart man sich in diesem Falle also die Berechnung des Terms $f_0 g_0 = 12(-2) = -24$.

b) Nun folgt ein Beispiel für den komplizierteren Fall $i = 1$ und $k = 2$ mit den Polynomen

$$f = 6X_1 - 2X_2 - 24 = f_1 X_1 + f_0,$$

$$\text{wobei} \quad \begin{cases} f_1 = 6 = \pi_1 X_2 + \pi_0 & \text{mit } \pi_1 = 0,\ \pi_0 = 6 \\ f_0 = -2X_2 - 24 = \rho_1 X_2 + \rho_0 & \text{mit } \rho_1 = -2,\ \rho_0 = -24, \end{cases}$$

$$g = -8X_2 - 12 = g_1 X_1 + g_0,$$

$$\text{wobei} \quad \begin{cases} g_1 = 0 = \sigma_1 X_2 + \sigma_0 & \text{mit } \sigma_1 = \sigma_0 = 0 \\ g_0 = -8X_2 - 12 = \tau_1 X_2 + \tau_0 & \text{mit } \tau_1 = -8,\ \tau_0 = -12. \end{cases}$$

Definitionsgemäß ergibt sich rekursiv

$$f \underset{1}{\overset{2}{\otimes}} g = \underbrace{(f_1 \underset{2}{\overset{2}{\otimes}} g_1)}_{=:a} X_1 + \underbrace{f_1 \underset{2}{\overset{2}{\otimes}} g_0}_{=:b} + \underbrace{f_0 \underset{2}{\overset{2}{\otimes}} g_1}_{=:c} = -48 =: \widetilde{q},$$

denn im einzelnen gilt (wobei der bei der rekursiven Berechnung von a, b, c auftretende Operator $\otimes_3^2$ gleich der normalen Multiplikation $\cdot$ ist)

$$a = f_1 \underset{2}{\overset{2}{\otimes}} g_1 = (\pi_1 \cdot \sigma_1) X_2 + \pi_1 \cdot \sigma_0 + \pi_0 \cdot \sigma_1 = 0 \cdot 0 \cdot X_2 + 0 \cdot 0 + 6 \cdot 0 = 0,$$

$$b = f_1 \underset{2}{\overset{2}{\otimes}} g_0 = (\pi_1 \cdot \tau_1) X_2 + \pi_1 \cdot \tau_0 + \pi_0 \cdot \tau_1 = 0 \cdot (-8) \cdot X_2 + 0 \cdot (-12) + 6 \cdot (-8) = -48,$$

$$c = f_0 \underset{2}{\overset{2}{\otimes}} g_1 = (\rho_1 \cdot \sigma_1) X_2 + \rho_1 \cdot \sigma_0 + \rho_0 \cdot \sigma_1 = (-2) \cdot 0 \cdot X_2 + (-2) \cdot 0 + (-24) \cdot 0 = 0.$$

Führt man die normale Multiplikation aus, so erhält man zunächst

$$\begin{aligned} f \cdot g &= (6X_1 - 2X_2 - 24)(-8X_2 - 12) = \\ &= -48X_1X_2 + \underline{16X_2^2} + \underline{192X_2} - \underline{72X_1} + \underline{24X_2} + \underline{288} = \\ &= -48X_1X_2 + 16X_2^2 - 72X_1 + 216X_2 + 288. \end{aligned}$$

Die Division mit Rest durch X_1X_2 ergibt

$$f \cdot g = q \cdot (X_1X_2) + r$$

mit $q = -48 = \widetilde{q}$ und dem Rest $r = 16X_2^2 - 72X_1 + 216X_2 + 288$. Die oben unterstrichenen Terme tragen zum Ergebnis $q = \widetilde{q}$ also nichts bei.

c) Nun wird der Fall $i = 1$ und $k = 3$ an einem Beispiel untersucht. Seien dazu

$$f = 2X_1X_2X_3 - 4X_1X_2 + 12X_3 - 36 \quad \text{und}$$
$$g = X_2X_3 - X_2 + 6X_3 - 6.$$

Damit erhält man

$$f \underset{1}{\overset{3}{\otimes}} g = \underbrace{\left((2X_2X_3 - 4X_2) \underset{2}{\overset{3}{\otimes}} 0\right)}_{=0} X_1 + (2X_2X_3 - 4X_2) \underset{2}{\overset{3}{\otimes}} g + \underbrace{(12X_3 - 36) \underset{2}{\overset{3}{\otimes}} 0}_{=0} =$$
$$= \underbrace{\left((2X_3 - 4) \underset{3}{\overset{3}{\otimes}} (X_3 - 1)\right)}_{=2X_3-6} X_2 + \underbrace{(2X_3 - 4) \underset{3}{\overset{3}{\otimes}} (6X_3 - 6)}_{12X_3-36} + \underbrace{0 \underset{3}{\overset{3}{\otimes}} (X_3 - 1)}_{=0} =$$
$$= 2X_2X_3 - 6X_2 + 12X_3 - 36.$$

Wie läßt sich nun die Multiplikation $\otimes$ gewinnbringend für die Verfahren aus den ersten beiden Kapiteln einsetzen, d.h. insbesondere wie müssen die jeweiligen Rekursionsformeln modifiziert werden, um darin die Division durch die Multiplikation $\otimes$ zu ersetzen? Dazu sind offensichtlich noch weitere Überlegungen nötig.

Sei im folgenden stets $A = (a_{ij})$ eine $n \times m$-Matrix über $R[X_1, \ldots, X_n]$ mit den beiden Eigenschaften

(V1) $n \leq m$,

(V2) $a_{kk} = X_k$ für $k = 1, \ldots, n$ und $a_{ij} \in R$ sonst.

Somit ist die Matrix A von der Gestalt

$$A = \begin{pmatrix} X_1 & a_{12} & \cdots & \cdots & a_{1n} & a_{1,n+1} & \cdots & a_{1m} \\ a_{21} & X_2 & \ddots & & \vdots & \vdots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & a_{n-1,n} & \vdots & & \vdots \\ a_{n1} & \cdots & \cdots & a_{n,n-1} & X_n & a_{n,n+1} & \cdots & a_{n,m} \end{pmatrix}$$

Auf der Hauptdiagonalen stehen die unabhängigen Variablen $X_1, \ldots, X_n$, die übrigen Koeffizienten von A enthalten diese Variablen nicht. Folglich liegen dann die Zeilenstufen der Matrix A alle auf der Hauptdiagonalen, also $j_k = k$ für $k = 1, \ldots, n$.

11.7 Proposition (Minoren und Division mit Rest)
In der obigen Matrix A gilt:

a) Für $k \in \{1, \ldots, n-1\}$, $i \in \{k+1, \ldots, n\}$ und $j \in \{k+1, \ldots, m\}$ erhält man mit einem Polynom r aus $R[X_1, \ldots, X_n]$

$$a_{ij}^{(k)} = a_{ij} \cdot (X_1 \cdots X_k) + r.$$

Das Polynom $a_{ij}^{(k)}$ liegt in $R[X_1, \ldots, X_k, a_{ij}]$, es ist linear in den Variablen $X_1, \ldots, X_n$, und für seinen Grad gilt $\deg a_{ij}^{(k)} \leq k + \deg a_{ij}$. Das Polynom r aus $R[X_1, \ldots, X_k, a_{ij}]$

ist linear in $X_1, \ldots, X_n$ *mit dem Grad* $\deg r \leq k-2+\deg a_{ij} < \deg a_{ij}^{(k)}$. *Insbesondere enthält* r *keinen Term, der Vielfaches von* $X_1 \cdots X_k$ *ist.*
Mit anderen Worten, die angegebene Formel stellt die Division von $a_{ij}^{(k)}$ *durch das Monom* $X_1 \cdots X_k$ *mit Rest* r *dar.*

b) *Für die Hauptminoren* k*-ter Ordnung ergibt sich für* $k = 1, \ldots, n$ *mit einem Polynom* r *aus* $R[X_1, \ldots, X_n]$ *das normierte Polynom*

$$a_{kk}^{(k-1)} = (X_1 \cdots X_k) + r.$$

Dieses Polynom liegt in $R[X_1, \ldots, X_k]$ *und ist linear in* $X_1, \ldots, X_k$ *mit dem Grad* $\deg a_{kk}^{(k-1)} = k$. *Das Polynom* r *liegt ebenfalls in* $R[X_1, \ldots, X_k]$ *und ist linear in* $X_1, \ldots, X_k$ *mit dem Grad* $\deg r \leq k-2 < \deg a_{kk}^{(k-1)}$. *Insbesondere enthält* r *keinen Term, der Vielfaches von* $X_1 \cdots X_k$ *ist.*
Mit anderen Worten, die angegebene Formel stellt die Division von $a_{kk}^{(k-1)}$ *durch das Monom* $X_1 \cdots X_k$ *mit Rest* r *dar.*

c) *Für* $k \in \{1, \ldots, n\}$, $i \in \{1, \ldots, k\}$ *und* $j \in \{k+1, \ldots, m\}$ *erhält man mit einem Polynom* r *aus* $R[X_1, \ldots, X_n]$

$$\delta_{ij}^{(k)} = a_{ij} \cdot (X_1 \cdots \widehat{X}_i \cdots X_k) + r.$$

Das Polynom $\delta_{ij}^{(k)}$ *aus* $R[X_1, \ldots, \widehat{X}_i, \ldots, X_k]$ *ist linear in den Variablen* $X_1, \ldots, X_k$, *für seinen Grad gilt* $\deg \delta_{ij}^{(k)} = k-1$. *Das Polynom* r *aus* $R[X_1, \ldots, \widehat{X}_i, \ldots, X_k]$ *ist linear in* $X_1, \ldots, X_k$ *mit dem Grad* $\deg r \leq k-2 < \deg \delta_{ij}^{(k)}$. *Insbesondere enthält* r *keinen Term, der Vielfaches von* $X_1 \cdots \widehat{X}_i \cdots X_k$ *ist.*
Mit anderen Worten, die angegebene Formel stellt die Division von $\delta_{ij}^{(k)}$ *durch das Monom* $X_1 \cdots \widehat{X}_i \cdots X_k$ *mit Rest* r *dar. (Hierbei bezeichne* $\widehat{\cdot}$ *die Auslassung.)*

Beweis:
Beachtet man, daß nach Voraussetzung in A auf der Diagonalen die unabhängigen Variablen $X_1, \ldots, X_n$ stehen und daß die übrigen Koeffizienten von A diese Variablen nicht enthalten, so folgt die Behauptung a) sofort aus der Entwicklung der Determinante

$$a_{ij}^{(k)} = [1 \ldots k\ i][1 \ldots k\ j](A) = \begin{pmatrix} X_1 & a_{12} & \cdots & \cdots & a_{1k} & a_{1j} \\ a_{21} & X_2 & \ddots & & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & & \ddots & \ddots & a_{k-1,k} & \vdots \\ a_{k1} & \cdots & \cdots & a_{k,k-1} & X_k & a_{kj} \\ a_{i1} & \cdots & \cdots & a_{i,k-1} & a_{ik} & a_{ij} \end{pmatrix}$$

wobei für den Koeffizienten a_{ij} rechts unten $a_{ij} = X_i$ gilt, falls $i = j$ ist.

Die Behauptung b) folgt analog sofort aus der Entwicklung der betreffenden Determinante

$$a_{kk}^{(k-1)} = [1 \ldots k][1 \ldots k](A) = \begin{pmatrix} X_1 & a_{12} & \cdots & \cdots & a_{1k} \\ a_{21} & X_2 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & a_{k-1,k} \\ a_{k1} & \cdots & \cdots & a_{k,k-1} & X_k \end{pmatrix}$$

Schließlich folgt die Behauptung c) aus der Entwicklung der Determinante

$$\delta_{ij}^{(k)} = [1\dots k][1\dots\widehat{i}\ j\ \dots k](A) = \begin{pmatrix} X_1 & a_{12} & \cdots & a_{1j} & \cdots & \cdots & a_{1k} \\ a_{21} & \ddots & \ddots & \vdots & & & \vdots \\ \vdots & \ddots & X_{i-1} & a_{i-1,j} & & & \vdots \\ \vdots & & \ddots & a_{ij} & \ddots & & \vdots \\ \vdots & & & a_{i+1,j} & X_{i+1} & \ddots & \vdots \\ \vdots & & & \vdots & \ddots & \ddots & a_{k-1,k} \\ a_{k1} & \cdots & \cdots & a_{kj} & \cdots & a_{k,k-1} & X_k \end{pmatrix}$$

□

Zur Erinnerung, für die obige Matrix A lautet die Bareiss-1-Schritt-Formel (Korollar 1.4a) für $k = 1, \dots, n-1$, $i = k+1, \dots, n$ und $j = 1, \dots, m$

$$a_{ij}^{(k)} = \frac{1}{a_{k-1,k-1}^{(k-2)}}[k\ i][k\ j](A^{(k-1)}) = \frac{a_{kk}^{(k-1)} \cdot a_{ij}^{(k-1)} - a_{ik}^{(k-1)} \cdot a_{kj}^{(k-1)}}{a_{k-1,k-1}^{(k-2)}}$$

Statt erst die beiden Produkte im Zähler zu berechnen und dann zu dividieren, sollen mit Hilfe der Multiplikation $\otimes$ nur diejenigen Terme berechnet werden, die auch tatsächlich zum Endergebnis $a_{ij}^{(k)}$ beitragen. In obiger Formel treten im Zähler Produkte zweier Determinanten vom Typ $a_{ij}^{(k-1)}$ auf, und der Divisor ist der Hauptminor $(k-1)$-ter Ordnung, nämlich $a_{k-1,k-1}^{(k-2)}$. Nach Proposition 11.7a) bzw. 11.7b) erhält man mit $x := X_1 \cdots X_{k-1}$ Zerlegungen

$$a_{ij}^{(k-1)} = a_{ij} \cdot x + r_{ij} \quad \text{bzw.} \quad a_{k-1,k-1}^{(k-2)} = x + r$$

mit gewissen Polynomen r_{ij} und r aus $R[X_1, \dots, X_n]$. Bevor man nun die obige 1-Schritt-Formel mittels $\otimes$ modifiziert, folgt zunächst eine Vorüberlegung für den vereinfachten Fall, daß der Zähler nur aus dem Produkt x^2 besteht.

11.8 Proposition (Division durch Hauptminoren mit Rest)
Sei die Matrix A wie oben, und sei $k \in \{2, \dots, n\}$ beliebig. Ferner bezeichne

$$x := X_1 \cdots X_{k-1}$$

das Produkt der Variablen $X_1, \dots, X_{k-1}$ und

$$t := a_{k-1,k-1}^{(k-2)} = x + r$$

den Hauptminor $(k-1)$-ter Ordnung mit $r := t - x \in R[X_1, \dots, X_{k-1}]$, vgl. Proposition 11.7b). Außerdem sei

$$x^2 = q \cdot t + \widetilde{r}$$

die eindeutige Zerlegung, die man bei der Division von x^2 durch t mit Rest erhält, mit Polynomen $q, \widetilde{r} \in R[X_1, \dots, X_n]$, wobei $\deg \widetilde{r} < \deg x^2 = 2(k-1)$ gilt.

a) Es gilt

$$x^2 = (x - r) \cdot t + r^2.$$

b) Die Polynome q und $\widetilde{r}$ liegen in $R[X_1,\dots,X_{k-1}]$, wobei der Rest $\widetilde{r}$ keine Terme enthält, die Vielfaches von x sind, und wobei q linear in $X_1,\dots,X_{k-1}$ ist mit

$$q = x - r + \; r \underset{1}{\overset{k-1}{\otimes}} r \; - \; r \underset{1}{\overset{k-1}{\otimes}} r \underset{1}{\overset{k-1}{\otimes}} r \; \pm \dots$$

Hierbei ist die Summe auf der rechten Seite endlich.

Beweis:

a) Es gilt

$$x^2 = (x+r)(x-r) + r^2 = (x-r)\cdot t + r^2,$$

also die Behauptung.

b) Daß die Polynome q und $\widetilde{r}$ in $R[X_1,\dots,X_{k-1}]$ liegen, daß $\widetilde{r}$ keine Terme enthält, die Vielfaches von x sind, und daß q linear in $X_1,\dots,X_{k-1}$ ist, folgt offensichtlich sofort aus der Division von x^2 durch t mit Rest. (Zu den Eigenschaften von t vergleiche Proposition 11.7b.) Nun ist noch die angegebene Darstellung von q zu beweisen. Nach a) und nach Voraussetzung gilt

$$(x-r)\cdot t + r^2 = x^2 = q\cdot t + \widetilde{r}. \qquad (*)$$

Möglicherweise kann man, wenn man die Division durch t mit Rest durchführt, von r^2 noch einen Bestandteil ungleich 0 abspalten, der Vielfaches von t ist — es gilt also i.a. nicht $x - r = q$.

Für zwei beliebige Polynome $f, g \in R[X_1,\dots,X_n]$ ergibt die Division von $f\cdot g$ durch das Monom $x = X_1\cdots X_{k-1}$ (vgl. Bemerkung 11.5)

$$f\cdot g = (f \underset{1}{\overset{k-1}{\otimes}} g)\cdot x + h = (f \underset{1}{\overset{k-1}{\otimes}} g)\cdot(x+r) - (f \underset{1}{\overset{k-1}{\otimes}} g)\cdot r + h, \qquad (**)$$

wobei der Rest h aus den Termen von $f\cdot g$ besteht, die kein Vielfaches von x sind.

Speziell für $f = g = r$ folgt aus $(**)$ mit $t = x + r$

$$r^2 = (r \underset{1}{\overset{k-1}{\otimes}} r)\cdot t - (r \underset{1}{\overset{k-1}{\otimes}} r)\cdot r + h_1$$

mit einem Rest $h_1 \in R[X_1,\dots,X_{k-1}]$, der die Terme von r^2 enthält, die nicht Vielfaches von x sind.

Mit $(r \otimes_1^{k-1} r)\cdot t$ ist also ein Bestandteil von r^2 gefunden, der Vielfaches von t ist. Möglicherweise enthält auch der mittlere Summand $(r \otimes_1^{k-1} r)\cdot r$ noch solche Bestandteile. Speziell für $f = (r \otimes_1^{k-1} r)$ und $g = r$ folgt aus $(**)$

$$(r \underset{1}{\overset{k-1}{\otimes}} r)\cdot r = (r \underset{1}{\overset{k-1}{\otimes}} r \underset{1}{\overset{k-1}{\otimes}} r)\cdot t - (r \underset{1}{\overset{k-1}{\otimes}} r \underset{1}{\overset{k-1}{\otimes}} r)\cdot r + h_2$$

mit einem Rest $h_2 \in R[X_1,\dots,X_{k-1}]$, der nur Terme enthält, die nicht Vielfaches sind von x.

Sukzessive folgt also, wenn man $(**)$ stets auf den mittleren Summanden des letzten Ergebnisses anwendet und wenn man von $(*)$ ausgeht,

$$q = x - r + r \underset{1}{\overset{k-1}{\otimes}} r - r \underset{1}{\overset{k-1}{\otimes}} r \underset{1}{\overset{k-1}{\otimes}} r \pm \dots$$

wie behauptet. Es bleibt noch die Endlichkeit dieser Summe zu zeigen. Da für den Grad von r nach Proposition 11.7b) $\deg r \le k-3 < \infty$ gilt, folgt bei wiederholter Anwendung von $(**)$, also bei wiederholter Division durch $x = X_1\cdots X_{k-1}$, schließlich

$$\underbrace{r \underset{1}{\overset{k-1}{\otimes}} \dots \underset{1}{\overset{k-1}{\otimes}} r}_{i-\text{mal}} = 0$$

für alle $i \ge i_0$ mit einem $i_0 \in \mathbb{N}_+$. □

Abschließend wird nun nach langer Vorbereitung am Beispiel der Bareiss-1-Schritt-Formel zur Trigonalisierung demonstriert, wie die Multiplikation $\otimes$ eingesetzt wird. Analog läßt sich dies auch auf die anderen Rekursionsformeln übertragen, die in den kommenden Abschnitten näher ausgeführt werden.

Zur Wiederholung, für die obige Matrix A lautet diese 1-Schritt-Formel (Korollar 1.4a) für $k = 1, \ldots, n-1$, $i = k+1, \ldots, n$ und $j = 1, \ldots, m$

$$a_{ij}^{(k)} = \frac{a_{kk}^{(k-1)} \cdot a_{ij}^{(k-1)} - a_{ik}^{(k-1)} \cdot a_{k,j}^{(k-1)}}{a_{k-1,k-1}^{(k-2)}}$$

11.9 Lemma (modifizierte Bareiss-1-Schritt-Formel zur Trigonalisierung)
Sei A wie oben, und seien $k \in \{1, \ldots, n-1\}$, $i \in \{k+1, \ldots, n\}$ und $j \in \{1, \ldots, m\}$ beliebig. Ferner seien wie in Proposition 11.8

$$x := X_1 \cdots X_{k-1} \qquad \textit{und} \qquad t := a_{k-1,k-1}^{(k-2)} = x + r \quad \textit{mit} \quad r := t - x,$$

wobei $x = t = 1$ und $r = 0$ für $k = 1$ gesetzt wird. Dann gilt

$$a_{ij}^{(k)} = \left(a_{kk}^{(k-1)} \overset{k-1}{\underset{1}{\otimes}} a_{ij}^{(k-1)} - a_{ik}^{(k-1)} \overset{k-1}{\underset{1}{\otimes}} a_{kj}^{(k-1)} \right) \overset{k-1}{\underset{1}{\otimes}} q$$

mit dem Polynom

$$q = x - r + r \overset{k-1}{\underset{1}{\otimes}} r - r \overset{k-1}{\underset{1}{\otimes}} r \overset{k-1}{\underset{1}{\otimes}} r \pm \ldots$$

Beweis:
Mit $t := a_{k-1,k-1}^{(k-2)}$ schreibt sich die Bareiss-1-Schritt-Formel als

$$a_{ij}^{(k)} = \frac{a_{kk}^{(k-1)} \cdot a_{ij}^{(k-1)} - a_{ik}^{(k-1)} \cdot a_{kj}^{(k-1)}}{t} =: \frac{u}{t}. \qquad (*)$$

Für $k = 1$ gilt $q = x = 1$, außerdem ist die Multiplikation $\otimes_1^{k-1}$ von der Form $\otimes_1^0$, also gleich der normalen Multiplikation. Für $k = 1$ ist die behauptete Formel also mit der herkömmlichen Bareiss-1-Schritt-Formel identisch.

Sei nun $k \in \{2, \ldots, n\}$ und $i \in \{k+1, \ldots, n\}$. Vorerst sei $j \in \{k+1, \ldots, m\}$. Aus Proposition 11.8 erhält man

$$x^2 = q \cdot t + \widetilde{r} \qquad (**)$$

wobei der Rest $\widetilde{r}$ keinen Term enthält, der Vielfaches von x ist, und wobei q linear in $X_1, \ldots, X_{k-1}$ ist. Multipliziert man die Bareiss-Formel $(*)$ und die Formel $(**)$ auf beiden Seiten miteinander, so folgt

$$a_{ij}^{(k)} \cdot x^2 = u \cdot q \; + \; a_{ij}^{(k)} \cdot \widetilde{r}. \qquad (***)$$

Da $a_{ij}^{(k)}$ nach Proposition 11.7a) linear in den Variablen $X_1, \ldots, X_n$ ist und da $\widetilde{r}$ kein Vielfaches von x enthält, kann der rechte Summand $a_{ij}^{(k)} \cdot \widetilde{r}$ aus $(***)$ kein Vielfaches von x^2 enthalten. Somit genügt es zur Berechnung von $a_{ij}^{(k)}$, nur diejenigen Terme des ersten Summanden $u \cdot q$ aus $(***)$ zu ermitteln, die Vielfaches von x^2 sind. Da der

Faktor q linear in $X_1, \ldots, X_{k-1}$ ist, tragen nur diejenigen Terme von u zu $a_{ij}^{(k)}$ bei, die Vielfache des Monoms $x = X_1 \cdots X_{k-1}$ sind. Damit folgt die behauptete Formel.

Die Behauptung ist nun noch für $k \in \{2, \ldots, n\}$, $i \in \{k+1, \ldots, n\}$ und $j \in \{1, \ldots, k\}$ zu zeigen. In diesem Fall gilt $0 = a_{ij}^{(k)} = \frac{u}{t}$, insbesondere also

$$u = a_{kk}^{(k-1)} \cdot a_{ij}^{(k-1)} - a_{ik}^{(k-1)} \cdot a_{kj}^{(k-1)} = 0$$

und dann erst recht

$$\left(a_{kk}^{(k-1)} \overset{k-1}{\underset{1}{\otimes}} a_{ij}^{(k-1)} - a_{ik}^{(k-1)} \overset{k-1}{\underset{1}{\otimes}} a_{kj}^{(k-1)}\right) \overset{k-1}{\underset{1}{\otimes}} q = 0 \overset{k-1}{\underset{1}{\otimes}} q = 0,$$

somit folgt die Behauptung. □

Wie läßt sich entsprechend die 1-Schritt-Formel zur Diagonalisierung modifizieren? Zur Erinnerung, für $k \in \{2, \ldots, n\}$, $i \in \{1, \ldots, k-1\}$ und $j \in \{1, \ldots, m\}$ lautet diese Formel

$$\delta_{ij}^{(k)} = \frac{a_{kk}^{(k-1)} \cdot \delta_{ij}^{(k-1)} - \delta_{ik}^{(k-1)} \cdot a_{kj}^{(k-1)}}{a_{k-,k-1}^{(k-2)}}.$$

11.10 Korollar (modifizierte Bareiss-1-Schritt-Formel zur Diagonalisierung)
Sei A wie oben, und seien $k \in \{2, \ldots, n\}$, $i \in \{1, \ldots, k-1\}$ und $j \in \{1, \ldots, m\}$ beliebig. Ferner seien wie in Lemma 11.9

$$x := X_1 \cdots X_{k-1} \qquad \textit{und} \qquad t := a_{k-1,k-1}^{(k-2)} = x + r \quad \textit{mit} \quad r := t - x.$$

Dann gilt

$$\delta_{ij}^{(k)} = \left(a_{kk}^{(k-1)} \overset{k-1}{\underset{1}{\otimes}} \delta_{ij}^{(k-1)} - \delta_{ik}^{(k-1)} \overset{k-1}{\underset{1}{\otimes}} a_{kj}^{(k-1)}\right) \overset{k-1}{\underset{1}{\otimes}} q$$

mit dem Polynom

$$q = x - r + r \overset{k-1}{\underset{1}{\otimes}} r - r \overset{k-1}{\underset{1}{\otimes}} r \overset{k-1}{\underset{1}{\otimes}} r \pm \ldots$$

Beweis:
Die Behauptung folgt analog zu Lemma 11.9, wenn man die in Proposition 11.7c) angegebenen Eigenschaften von Determinanten des Typs $\delta_{ij}^{(k)}$ berücksichtigt. □

Sei nun eine $n \times m$-Matrix A über einem Integritätsring R mit den vier Eigenschaften

(V1) $n \leq m$,
(V2) $\operatorname{rang} A = n$,
(V3) Zeilentausch ist nicht nötig und
(V4) für $k = 1, \ldots, n$ gilt $j_k = k$

gegeben. Ferner seien die Variablen $X_1, \ldots, X_n$, die nicht schon in R liegen sollen, unabhängig über R.

11.11 Bemerkung (Modifikation der Algorithmen und Merkregel für die Rekursionsformeln)
Genau wie oben sei A eine Matrix mit den Eigenschaften (V1) bis (V4). Die Modifikation der Algorithmen aus den Kapiteln I und II läuft stets in denselben drei Schritten ab.

(1) *(Substitution)* Die Koeffizienten auf der Hauptdiagonalen von A, also die Zeilenstufenkoeffizienten, werden durch die Variablen $X_1, \ldots, X_n$ ersetzt. Genaugenommen ist es dabei nicht nötig, alle Diagonalkoeffizienten zu ersetzen. Es müssen lediglich diejenigen ersetzt werden, die in den Rekursionsformeln als Teiler auftreten, also bei der Trigonalisierung die ersten $(n-2)$ und bei der Diagonalisierung die ersten $(n-1)$ Koeffizienten auf der Diagonalen.

(2) *(Verfahren mit Merkregel)* Die substituierte Matrix wird mit den modifizierten Algorithmen trigonalisiert bzw. diagonalisiert, wobei diese Algorithmen im wesentlichen wie die ursprünglichen Algorithmen sind, bis auf die Verwendung der Multiplikation $\otimes$. Als Merkregel für die modifizierten Formeln läßt sich im Hinblick auf Lemma 11.9 und Korollar 11.10 und als Vorgriff auf die folgenden Abschnitte feststellen, daß die Formeln mit kleinen Änderungen im wesentlichen erhalten bleiben. Es werden lediglich die normale Multiplikation $\cdot$ und die Division durch einen Hauptminor $(k-1)$-ter Ordnung jeweils ersetzt durch die spezielle Multiplikation $\otimes_1^{k-1}$, der Divisor wird stets ersetzt durch das Polynom q aus Lemma 11.9.

(3) *(Rücksubstitution)* Beim Ergebnis aus Schritt (2) wird die in Schritt (1) gemachte Substitution rückgängig gemacht. Damit erhält man wie gewünscht eine zu A äquivalente Matrix in Zeilenstufen- bzw. Diagonalform.

Für die modifizierten Algorithmen muß die Ausgangsmatrix A strengere Voraussetzungen als für die ursprünglichen Algorithmen erfüllen. Die obigen Voraussetzungen (V3) und (V4) bedeuten, daß von vornherein bekannt ist, wo die Zeilenstufen von A liegen. Dies ist nötig, da die Idee von Sasaki-Murao darauf beruht, die Zeilenstufenkoeffizienten zu substituieren, und zwar alle schon von Beginn an. Daher können die Zeilenstufen nicht erst, wie bei anderen Algorithmen, im Laufe des Verfahrens gefunden werden.

Erfüllt eine Matrix die Voraussetzungen (V1) bis (V4) nicht und ist die Zeilenstufenverteilung bekannt, so kann die Matrix durch Zeilen- bzw. Spaltentausch oder durch Weglassen überflüssiger Zeilen auf die gewünschte Form gebracht werden.

12 Trigonalisierung mit dem Bareiss-1-Schritt-Verfahren

In diesem Abschnitt wird nun die spezielle Multiplikation $\otimes$ aus dem vorigen Abschnitt auf das Bareiss-1-Schritt-Verfahren zur Trigonalisierung angewandt, vgl. Abschnitt 3.

Wie üblich bezeichne R einen Integritätsring, und die Variablen $X_1, \ldots, X_n$, die nicht schon in R liegen sollen, seien unabhängig über R. Ab sofort sei $A = (a_{ij})$ eine $n \times m$-Matrix über $R[X_1, \ldots, X_{n-2}]$ mit den Eigenschaften

(V1) $n \leq m$,
(V2) $\operatorname{rang} A = n$,
(V3) Zeilentausch ist nicht nötig,
(V4) für $k = 1, \ldots, n$ gilt $j_k = k$, d.h. die Zeilenstufen liegen auf der Hauptdiagonalen,
(V5) für $i = 1, \ldots, n-2$ sei $a_{ii} = X_i$ und sonst sei $a_{ij} \in R$, d.h. die übrigen Koeffizienten von A enthalten die Variablen $X_1, \ldots, X_{n-2}$ nicht.

Damit ergibt sich eine Matrix der Form

$$A = \begin{pmatrix} X_1 & a_{12} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & a_{1m} \\ a_{21} & X_2 & \ddots & & & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & & & \ddots & X_{n-2} & \ddots & & & \vdots \\ \vdots & & & & \ddots & a_{n-1,n-1} & a_{n-1,n} & \cdots & a_{n-1,m} \\ a_{n1} & \cdots & \cdots & \cdots & \cdots & a_{n,n-1} & a_{nn} & \cdots & a_{n,m} \end{pmatrix}$$

also eine Matrix, wie man sie erhält, wenn man bei einer Matrix aus $\mathrm{Mat}_{n\times m}(R)$ die ersten $(n-2)$ Koeffizienten auf der Hauptdiagonalen durch $X_1, \ldots, X_{n-2}$ ersetzt.

Genau wie in Abschnitt 3 (Trigonalisierung mit dem Bareiss-1-Schritt-Verfahren) wird nun ausgehend von der Matrix A die Folge der Matrizen $B^{(k)}$, $k = 0, \ldots, n-1$, definiert, nämlich

$$B^{(k)} = (b_{ij}^{(k)})_{\substack{i=1\ldots n \\ j=1\ldots m}} := \begin{pmatrix} (\alpha_{ij}^{(i-1)})_{\substack{i=1\ldots k \\ j=1\ldots m}} \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\ldots n \\ j=1\ldots m}} \end{pmatrix} = \begin{pmatrix} (a_{ij}^{(i-1)})_{\substack{i=1\ldots k \\ j=1\ldots m}} \\ (a_{ij}^{(k)})_{\substack{i=k+1\ldots n \\ j=1\ldots m}} \end{pmatrix}.$$

Außerdem setzt man

$$b_{00}^{(0)} := 1,$$

und für $k = 1$ setzt man für das leere Produkt

$$X_1 \cdots X_{k-1} := 1.$$

In der Matrix $B^{(k)}$ sind bekanntlich die ersten k Spalten unter der Diagonalen ausgeräumt. Die Folge dieser Matrizen führt die Matrix $A = B^{(0)}$ in die trigonalisierte Matrix $B^{(n-1)} = (a_{ij}^{(i-1)})$ über, indem sukzessive von links nach rechts eine Spalte nach der anderen unter der Diagonalen ausgeräumt wird. (Nach Voraussetzung liegen die Zeilenstufen von A auf der Diagonalen, also gilt $j_k = k$ für $k = 1, \ldots, n$, und insbesondere gilt stets $\alpha_{ij}^{(k)} = a_{ij}^{(k)}$.)

Statt mit der herkömmlichen Bareiss-1-Schritt-Formel (Lemma 3.5) sollen diese Matrizen $B^{(k)}$ nun aber mit der modifizierten 1-Schritt-Formel aus Lemma 11.9 rekursiv berechnet werden. Dazu betrachtet man für $k = 1, \ldots, n-1$ die Matrix

$$B^{(k-1)} = (b_{ij}^{(k-1)}) = \begin{pmatrix} (a_{ij}^{(i-1)})_{\substack{i=1\ldots k-1 \\ j=1\ldots m}} \\ (a_{ij}^{(k-1)})_{\substack{i=k\ldots n \\ j=1\ldots m}} \end{pmatrix},$$

in der die ersten $(k-1)$ Spalten unter der Diagonalen ausgeräumt sind, und den Übergang zur direkten Nachfolgerin

$$B^{(k)} = (b_{ij}^{(k)}) = \begin{pmatrix} (a_{ij}^{(i-1)})_{\substack{i=1\ldots k \\ j=1\ldots m}} \\ (a_{ij}^{(k)})_{\substack{i=k+1\ldots n \\ j=1\ldots m}} \end{pmatrix},$$

in der die ersten k Spalten unter der Diagonalen ausgeräumt sind.

Aus Lemma 3.5 (1-Schritt-Rekursion zur Trigonalisierung) folgt

12.1 Lemma (modifizierte 1-Schritt-Rekursion zur Trigonalisierung)
Die Matrizen A und $B^{(k)}$, $k = 0, \ldots, n-1$, seien wie oben. Betrachte für $k = 1, \ldots, n-1$ die Matrix $B := B^{(k-1)}$, wobei

$$t := b_{k-1,k-1} = x + r \qquad \text{mit} \quad x := X_1 \cdots X_{k-1} \qquad \text{und} \qquad r := t - x \qquad \text{sei.}$$

Ferner bezeichne q die Summe

$$q := x - r + r \overset{k-1}{\underset{1}{\otimes}} r - r \overset{k-1}{\underset{1}{\otimes}} r \overset{k-1}{\underset{1}{\otimes}} r \pm \ldots$$

Die Matrix $E = (e_{ij}) \in \mathrm{Mat}_{n\times m}(R[X_1, \ldots, X_{n-2}])$ werde wie folgt definiert:
(1) Die oberen Zeilen $i = 1, \ldots, k$ von E sind genau wie in B.
(2) Für die unteren Zeilen $i = k+1, \ldots, n$ gilt für $j = 1, \ldots, m$:

$$e_{ij} := \left(b_{kk} \overset{k-1}{\underset{1}{\otimes}} b_{ij} - b_{ik} \overset{k-1}{\underset{1}{\otimes}} b_{kj} \right) \overset{k-1}{\underset{1}{\otimes}} q.$$

Dann gilt $E = B^{(k)}$.
Beweis:
In den oberen Zeilen $i = 1, \ldots, k$ stimmen die beiden Matrizen $B = B^{(k-1)}$ und $B^{(k)}$ definitionsgemäß überein. Für die unteren Zeilen $i = k, \ldots, n$ gilt für die Spalten $j = 1, \ldots, m$ per definitionem

$$b_{ij} = a_{ij}^{(k-1)},$$

außerdem gilt

$$t = b_{k-1,k-1} = a_{k-1,k-1}^{(k-2)}.$$

Damit folgt für die Zeilen $i = k+1, \ldots, n$ und die Spalten $j = 1, \ldots, m$ die Behauptung sofort aus der in Lemma 11.9 angegebenen modifizierten Rekursionsformel:

$$b_{ij}^{(k)} = a_{ij}^{(k)} = \left(a_{kk}^{(k-1)} \overset{k-1}{\underset{1}{\otimes}} a_{ij}^{(k-1)} - a_{ik}^{(k-1)} \overset{k-1}{\underset{1}{\otimes}} a_{kj}^{(k-1)} \right) \overset{k-1}{\underset{1}{\otimes}} q =$$

$$= \left(b_{kk} \overset{k-1}{\underset{1}{\otimes}} b_{ij} - b_{ik} \overset{k-1}{\underset{1}{\otimes}} b_{kj} \right) \overset{k-1}{\underset{1}{\otimes}} q.$$

(Nach Proposition 11.8b) ist q eine endliche Summe.) □

Eine kurze Vorbemerkung zu den folgenden Algorithmen: an die Bezeichnung des ursprünglichen Algorithmus wird stets das Kürzel `SM` angehängt, um kenntlich zu machen, daß dies die nach Saski-Murao modifizierte Variante ist, vgl. zur Modifikation auch Bemerkung 11.11. Aus Satz 3.7 (`B1TMD`) folgt nun die Variante

12.2 Satz (Algorithmus B1TMDSM)
Sei A eine $n \times m$-Matrix über einem Integritätsring R mit den Eigenschaften (V1) bis (V4), d.h. daß $n \leq m$ und $\operatorname{rang} A = n$ sei, daß Zeilentausch nicht nötig sei in A und daß die Zeilenstufen auf der Diagonalen liegen, also $j_k = k$ für $k = 1, \ldots, n$. Der Unteralgorithmus `B1TMDSMRow` *sei wie anschließend definiert. Betrachte die folgenden Instruktionen:*

(1) (Substitution) Ersetze in A die ersten $(n-2)$ Diagonalkoeffizienten durch die Variablen $X_1, \ldots, X_{n-2}$ und merke die Substitutionen $X_1 = a_{11}, \ldots, X_{n-2} = a_{n-2,n-2}$.

(2) Setze $k := 1$ und $q := 1$.

(3) (Spalte k ausräumen) Falls $k \geq n$ ist, fahre fort bei Schritt (5). Andernfalls ersetze die unteren Zeilen $i = k+1, \ldots, n$ von A jeweils durch das Ergebnis von `B1TMDSMRow`(A, i, k, q)*. Setze dann, falls $k \leq n-2$ gilt, $x := X_1 \cdots X_k$ und $r := a_{kk} - x$ und berechne $q := x - r + r \otimes_1^k r \mp \ldots$.*

(4) Erhöhe k um 1 und wiederhole Schritt (3).

(5) (Rücksubstitution) Ersetze in A alle Variablen $X_1, \ldots, X_{n-2}$ durch die in Schritt (1) gemerkten Werte und gib das Ergebnis A aus.

Dies ist ein Algorithmus mit der Bezeichnung `B1TMDSM`(A)*, der als Ergebnis eine zu A äquivalente Matrix in Zeilenstufenform ausgibt.*

Hierbei berechnet der Unteralgorithmus `B1TMDSMRow`(A, i, k, q) *die i-te Zeile von A, d.h. räumt in Zeile i mit Hilfe von Zeile k die Spalte k aus:*

(R1) (Berechnen) Für $j = k+1, \ldots, m$ ersetze a_{ij} durch $a_{kk} \otimes_1^{k-1} a_{ij} - a_{ik} \otimes_1^{k-1} a_{kj}$. Ersetze dann den neuen Koeffizienten a_{ij} durch $a_{ij} \otimes_1^{k-1} q$.

(R2) (Ausräumen) Setze $a_{ik} := 0$.

(R3) Gib das Ergebnis $(a_{ij})_{j=1\ldots m}$ aus, d.h. die i-te Zeile von A.

Beweis:
Die Behauptung folgt sofort aus den vorangegangenen Überlegungen und den folgenden. Der Algorithmus `B1TMD` aus Satz 3.7 wird so abgeändert, daß auf den Unteralgorithmus `Pivot` und die Variable λ verzichtet wird, da nach Voraussetzung Zeilentausch ausgeschlossen ist und die Lage der Zeilenstufen von vornherein bekannt ist. An die Stelle des Teilers t aus Satz 3.7 tritt nun q. Schritt (3), d.h. insbesondere der Unteralgorithmus `B1TMDSMRow`, ist die Anwendung der 1-Schritt-Rekursion aus Lemma 12.1. Nach Proposition 11.8b) besteht die in Schritt (3) berechnete Variable q nur aus endlich vielen Summanden. □

12.3 Bemerkungen zur Implementierung (B1TMDSM)

Die folgenden Bemerkungen treffen auch auf die später noch kommenden modifizierten Algorithmen zu.

a) (*Substitution und Rücksubstitution*) In CoCoA ist es nicht ohne weiteres möglich, vom Grundring R zum Erweiterungsring $R[X_1, \ldots, X_{n-2}]$ überzugehen. Der modifizierte Algorithmus ist programmiert für den Fall, daß R gleich $\mathbb{Z}$, $\mathbb{Q}$ oder einem endlichen Körper $\mathbb{F}_p = \mathbb{Z}/(p)$ oder gleich einem Polynomring über den genannten Ringen ist. Um die Matrix A von R zu $R[X_1, \ldots, X_{n-2}]$ und umgekehrt zu transportieren, wird mit Hilfe der eingebauten Funktionen `RMap` und `Image` die Injektion bzw. Projektion durchgeführt. Die Rücksubstitution findet mit Hilfe des einfachen CoCoA-Programms `BackSubstTMD` statt.

b) (*Multiplikation* $\otimes$) Die in Satz 12.2, Schritt (3) bzw. (R3) verwendete Multiplikation $\otimes$ wird mit dem Algorithmus `MultSM` aus Satz 11.3 durchgeführt. Beispielsweise erhält man damit die in Schritt (R3) benötigte Multiplikation $a_{ij} \otimes_1^{k-1} q$ als Ergebnis von `MultSM`$(a_{ij}, q, 1, k-1)$. Die Berechnung von $q = x - r + r \otimes_1^k r \mp \ldots$ erfolgt einfachheitshalber stets mit der CoCoA-Funktion `KFSM`(A, K).

c) (*Modifikationen*) Dadurch, daß bei der Ausgangsmatrix A von vornherein bekannt sein muß, wo die Zeilenstufen liegen (und zwar auf der Diagonalen), vereinfachen sich die modifizierten Algorithmen gegenüber den ursprünglichen, da Zeilentauschprogramme (`Pivot` etc.) weggelassen werden können. Außerdem entfällt die Suche mit der Variablen λ nach der nächsten Zeilenstufe, und der Fall "langer" Zeilenstufen muß auch

nicht berücksichtigt werden. Ansonsten bleibt die Struktur der ursprünglichen Algorithmen aber erhalten, da bei den Rekursionsformeln lediglich die Multiplikation $\otimes$ statt der Multiplikation $\cdot$ bzw. der Division verwendet wird, und da der jeweilige Teiler durch das entsprechende Polynom q ersetzt wird, vgl. Bemerkung 11.11.

12.4 Bemerkung
Der Vorteil der Multiplikation $\otimes$ liegt sicherlich darin, daß nur solche Terme berechnet werden, die auch tatsächlich zum Endergebnis beitragen, daß also insgesamt weniger Rechenoperationen durchgeführt werden. Für einfachere Matrizen, deren Koeffizienten beispielsweise aus $\mathbb{Z}$ oder $\mathbb{Z}[X]$ stammen, ist die Modifikation mittels $\otimes$ aber im Grunde genommen unnötig kompliziert, da in diesem Falle bei den Rekursionsformeln die normale Multiplikation bzw. Division keine sonderlichen Schwierigkeiten bereitet. Sind die Koeffizienten dagegen multivariate Polynome, ist eher zu erwarten, daß die Multiplikation $\otimes$ einen Einsparungseffekt hat.

Als Nachteil ist anzusehen, daß bei der Trigonalisierung noch $(n-2)$ zusätzliche Variablen $X_1, \ldots, X_{n-2}$ ins Spiel kommen. Mit der Anzahl der Variablen steigt zugleich auch die Rechenzeit, die auf die Multiplikation multivariater Polynome verwandt werden muß. Zumindest treten bei einer Multiplikation der Form $f \otimes g$ aber nur Polynome auf, die linear in $X_1, \ldots, X_{n-2}$ sind und die lediglich Bestandteile von f bzw. g sind. Möglicherweise kann dann auch die anschließende Rücksubstitution noch einige Zeit in Anspruch nehmen. Ein weiterer Nachteil der modifizierten Trigonalisierung (bzw. Diagonalisierung) besteht darin, daß die Matrizen nicht gänzlich beliebig sein können, sondern daß von vornherein die Lage der Zeilenstufen bekannt sein muß.

13 Trigonalisierung mit dem Bareiss-2-Schritt-Verfahren

In diesem Abschnitt wird nun das Bareiss-2-Schritt-Verfahren zur Trigonalisierung mit der Multiplikation $\otimes$ modifiziert, vgl. Abschnitt 4. Für den Algorithmus `B2TMD` ist diese Methode jedoch nicht geeignet, da dort bei den Rekursionen die Division durch einen quadrierten Hauptminor, also einen i.a. nichtlinearen Teiler, durchgeführt wird. Beim Algorithmus `B2TMDMD` wird dagegen zweimal durch einen jeweils linearen Divisor geteilt, dieses Verfahren kann also sehr wohl nach Sasaki-Murao modifiziert werden.

Sei dazu wie üblich R ein Integritätsring, wobei die Variablen $X_1, \ldots, X_{n-2}$, die nicht schon in R liegen sollen, unabhängig über R seien. Ferner sei die Matrix A genau wie im vorangegangenen Abschnitt eine $n \times m$-Matrix über $R[X_1, \ldots, X_{n-2}]$ mit den Eigenschaften

(V1) $n \leq m$,
(V2) $\operatorname{rang} A = n$,
(V3) Zeilentausch ist nicht nötig,
(V4) für $k = 1, \ldots, n$ gilt $j_k = k$, d.h. die Zeilenstufen liegen auf der Hauptdiagonalen,
(V5) für $i = 1, \ldots, n-2$ sei $a_{ii} = X_i$ und sonst sei $a_{ij} \in R$, d.h. die übrigen Koeffizienten von A enthalten die Variablen $X_1, \ldots, X_{n-2}$ nicht.

Wieder betrachtet man die Folge der Matrizen $B^{(k)}$, $k = 0, \ldots, n-1$, definiert durch

$$B^{(k)} = (b_{ij}^{(k)})_{\substack{i=1\ldots n \\ j=1\ldots m}} := \begin{pmatrix} (\alpha_{ij}^{(i-1)})_{\substack{i=1\ldots k \\ j=1\ldots m}} \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\ldots n \\ j=1\ldots m}} \end{pmatrix} = \begin{pmatrix} (a_{ij}^{(i-1)})_{\substack{i=1\ldots k \\ j=1\ldots m}} \\ (a_{ij}^{(k)})_{\substack{i=k+1\ldots n \\ j=1\ldots m}} \end{pmatrix}.$$

Außerdem setzt man

$$b_{00}^{(0)} := 1,$$

und für $k = 1$ setzt man für das leere Produkt

$$X_1 \cdots X_{k-1} := 1.$$

Statt mit der normalen Bareiss-2-Schritt-Formel aus Abschnitt 4 sollen diese Matrizen $B^{(k)}$ nun aber mit einer modifizierten 2-Schritt-Formel rekursiv berechnet werden. Wie sieht die modifizierte 2-Schritt-Formel aus? Dazu untersucht man für gerades $k = 2, \ldots, n-1$ die Matrix

$$B^{(k-2)} = (b_{ij}^{(k-2)}) = \begin{pmatrix} (a_{ij}^{(i-1)})_{\substack{i=1\ldots k-2 \\ j=1\ldots m}} \\ (a_{ij}^{(k-2)})_{\substack{i=k-1\ldots n \\ j=1\ldots m}} \end{pmatrix},$$

in der die ersten $(k-2)$ Spalten unter der Diagonalen ausgeräumt sind, und den Übergang zur übernächsten Matrix

$$B^{(k)} = (b_{ij}^{(k)}) = \begin{pmatrix} (a_{ij}^{(i-1)})_{\substack{i=1\ldots k \\ j=1\ldots m}} \\ (a_{ij}^{(k)})_{\substack{i=k+1\ldots n \\ j=1\ldots m}} \end{pmatrix},$$

in der nun die ersten k Spalten unter der Diagonalen ausgeräumt sind.

Zur Erinnerung sei hier die bisherige 2-Schritt-Rekursionsformel angegeben, vgl. dazu Lemma 4.1 (2-Schritt-Rekursion) und Bemerkung 4.2 (Determinantenentwicklung), wobei bei der Determinantenentwicklung nun die Teilbarkeit der Kofaktoren ausgenutzt wird.

13.1 Bemerkung (Bareiss-2-Schritt-Rekursion zur Trigonalisierung)
Die Matrizen A und $B^{(k)}$, $k = 0, \ldots, n-1$, seien wie oben. Zur Vereinfachung setzt man $B := B^{(k-2)}$ und $t := b_{k-2,k-2}$ für ein beliebiges, gerades $k \in \{2, \ldots, n-1\}$.

a) Es gilt für $i = k+1, \ldots, n$ und $j = 1, \ldots, m$

$$b_{ij}^{(k)} = \frac{1}{t^2} \cdot [k-1\ k\ i][k-1\ k\ j](B) = \frac{c_0 \cdot b_{ij} - c_{i1} \cdot b_{k,j} + c_{i2} \cdot b_{k-1,j}}{t}$$

mit den Faktoren

$$c_0 := \frac{1}{t} \cdot [k-1\ k][k-1\ k](B) = \frac{b_{k-1,k-1} \cdot b_{k,k} - b_{k,k-1} \cdot b_{k-1,k}}{t}$$

$$c_{i1} := \frac{1}{t} \cdot [k-1\ i][k-1\ k](B) = \frac{b_{k-1,k-1} \cdot b_{i,k} - b_{i,k-1} \cdot b_{k-1,k}}{t}$$

$$c_{i2} := \frac{1}{t} \cdot [k\ i][k-1\ k](B) = \frac{b_{k,k-1} \cdot b_{i,k} - b_{i,k-1} \cdot b_{k,k}}{t}$$

b) Insbesondere für den Zeilenstufenkoeffizienten der k-ten Zeile gilt

$$b_{kk}^{(k)} = c_0.$$

Die obigen Formeln können analog zur modifizierten 1-Schritt-Rekursion (Lemma 12.1) mittels $\otimes$ modifiziert werden, da jeweils durch den Hauptminor $b_{k-2,k-2}$ geteilt wird, der linear in den Variablen $X_1, \ldots, X_{n-2}$ ist. Als Modifikation von Lemma 4.1 (2-Schritt-Rekursion) folgt somit

13.2 Lemma (modifizierte 2-Schritt-Rekursion zur Trigonalisierung)
Die Matrizen A und $B^{(k)}$, $k = 0, \ldots, n-1$, seien wie oben. Betrachte für ein gerades $k \in \{2, \ldots, n-1\}$ die Matrix $B := B^{(k-2)}$, wobei

$$t := b_{k-2,k-2} = x + r \qquad \textit{mit} \quad x := X_1 \cdots X_{k-2} \qquad \textit{und} \qquad r := t - x \qquad \textit{sei.}$$

Ferner bezeichne q die Summe

$$q := x - r + r \overset{k-2}{\underset{1}{\otimes}} r - r \overset{k-2}{\underset{1}{\otimes}} r \overset{k-2}{\underset{1}{\otimes}} r \pm \ldots$$

Die Matrix $E = (e_{ij}) \in \mathrm{Mat}_{n\times m}(R[X_1, \ldots, X_{n-2}])$ sei wie folgt definiert:

(1) Die oberen Zeilen $i = 1, \ldots, k-1$ von E sind genau wie in B.

(2) Für die unteren Zeilen $i = k+1, \ldots, n$ gilt für $j = 1, \ldots, m$:

$$e_{ij} := \left(c_0 \overset{k-2}{\underset{1}{\otimes}} b_{ij} - c_{i1} \overset{k-2}{\underset{1}{\otimes}} b_{k,j} + c_{i2} \overset{k-2}{\underset{1}{\otimes}} b_{k-1,j}\right) \overset{k-2}{\underset{1}{\otimes}} q$$

mit den Faktoren

$$c_0 := \left(b_{k-1,k-1} \overset{k-2}{\underset{1}{\otimes}} b_{k,k} - b_{k,k-1} \overset{k-2}{\underset{1}{\otimes}} b_{k-1,k}\right) \overset{k-2}{\underset{1}{\otimes}} q$$

$$c_{i1} := \left(b_{k-1,k-1} \overset{k-2}{\underset{1}{\otimes}} b_{i,k} - b_{i,k-1} \overset{k-2}{\underset{1}{\otimes}} b_{k-1,k}\right) \overset{k-2}{\underset{1}{\otimes}} q$$

$$c_{i2} := \left(b_{k,k-1} \overset{k-2}{\underset{1}{\otimes}} b_{i,k} - b_{i,k-1} \overset{k-2}{\underset{1}{\otimes}} b_{k,k}\right) \overset{k-2}{\underset{1}{\otimes}} q.$$

(3) Für die Zeile k gilt für $j = 1, \ldots, m$:

$$e_{kj} := \left(b_{k-1,k-1} \overset{k-2}{\underset{1}{\otimes}} b_{kj} - b_{k,k-1} \overset{k-2}{\underset{1}{\otimes}} b_{k-1,j}\right) \overset{k-2}{\underset{1}{\otimes}} q.$$

Dann gilt $E = B^{(k)}$.

Beweis:
In den oberen Zeilen $i = 1, \ldots, k-1$ stimmen die beiden Matrizen $B = B^{(k-2)}$ und $B^{(k)}$ definitionsgemäß überein. Für die Zeile k folgt die Behauptung aus der 1-Schritt-Rekursion (Lemma 12.1). Für die unteren Zeilen $i = k+1, \ldots, n$ folgt die Gültigkeit der modifizierten 2-Schritt-Formel analog zu Lemma 12.1 (modifizierte 1-Schritt-Formel).

(Statt durch den Hauptminor $t := b_{k-2,k-2} = a^{(k-1)}_{k-2,k-2} = X_1 \cdots X_{k-2} + r$, der linear in $X_1, \ldots, X_{k-2}$ ist, zu dividieren, tritt hier an Stelle der normalen Multiplikation $\cdot$ die Multiplikation $\otimes_1^{k-2}$ auf, und an Stelle von t erscheint q. Zu den modifizierten Formeln vergleiche die ursprünglichen in Lemma 4.1 (2-Schritt-Rekursion zur Trigonalisierung) und in der vorangegangenen Bemerkung 13.1, vergleiche dazu auch Bemerkung 11.11 zur Modifikation.) □

Damit folgt nun als Modifikation von Satz 4.6 (`B2TMDMD`)

13.3 Satz (Algorithmus B2TMDMDSM)
Sei A eine $n\times m$-Matrix über einem Integritätsring R mit den Eigenschaften (V1) bis (V4), d.h. daß $n \le m$ und $\operatorname{rang} A = n$ *sei, daß Zeilentausch nicht nötig sei in A und daß die Zeilenstufen auf der Diagonalen liegen, also $j_k = k$ für $k = 1,\dots,n$. Der Unteralgorithmus* `B1TMDSMRow` *sei genau wie in Satz 12.2,* `B2TMDMDSMRow` *sei wie anschließend definiert. Betrachte die folgenden Instruktionen:*

(1) (Substitution) Ersetze in A die ersten $(n-2)$ Diagonalkoeffizienten durch die Variablen $X_1,\dots,X_{n-2}$ und merke die Substitutionen $X_1 = a_{11},\dots,X_{n-2} = a_{n-2,n-2}$.

(2) Setze $k := 1$ und $q := 1$.

(3) (Spalten $k, k+1$ ausräumen) Falls $k \ge n$ ist, fahre fort bei Schritt (5). Andernfalls berechne $c_0 := \left(a_{kk} \otimes_1^{k-1} a_{k+1,k+1} - a_{k+1,k} \otimes_1^{k-1} a_{k,k+1}\right) \otimes_1^{k-1} q$. Ersetze die unteren Zeilen $i = k+2,\dots,n$ von A jeweils durch das Ergebnis von `B2TMDMDSMRow`(A,i,k,q,c_0). *Ersetze die Zeile $(k+1)$ von A durch das Ergebnis von* `B1TMDSMRow`$(A,k+1,k,q)$. *Setze dann, falls $k \le n-3$ gilt, $x := X_1 \cdots X_{k+1}$ und außerdem $r := a_{k+1,k+1} - x$ und berechne $q := x - r + r \otimes_1^{k+1} r \mp \dots$*

(4) Erhöhe k um 2 und fahre fort bei Schritt (3).

(5) (Rücksubstitution) Ersetze in A alle Variablen $X_1,\dots,X_{n-2}$ durch die in Schritt (1) gemerkten Werte und gib das Ergebnis A aus.

Dies ist ein Algorithmus mit der Bezeichnung `B2TMDMDSM`(A), *der als Ergebnis eine zu A äquivalente Matrix in Zeilenstufenform ausgibt.*

Dabei berechnet der Unteralgorithmus `B2TMDMDSMRow`(A,i,k,q,c_0) *die i-te Zeile von A, d.h. räumt in Zeile i mit Hilfe der Zeilen $k, k+1$ die Spalten $k, k+1$ aus:*

(R1) (Kofaktor berechnen) Berechne $c_1 := a_{kk} \otimes_1^{k-1} a_{i,k+1} - a_{ik} \otimes_1^{k-1} a_{k,k+1}$. Ersetze c_1 durch $c_1 \otimes_1^{k-1} q$.

(R2) (Kofaktor berechnen) Berechne $c_2 := a_{k+1,k} \otimes_1^{k-1} a_{i,k+1} - a_{ik} \otimes_1^{k-1} a_{k+1,k+1}$. Ersetze dann c_2 durch $c_2 \otimes_1^{k-1} q$.

(R3) (Determinante entwickeln) Für $j = k+2,\dots,m$ ersetze den Koeffizienten a_{ij} erst durch $c_0 \otimes_1^{k-1} a_{ij} - c_1 \otimes_1^{k-1} a_{k+1,j} + c_2 \otimes_1^{k-1} a_{kj}$. Ersetze dann den neuen Koeffizienten a_{ij} durch $a_{ij} \otimes_1^{k-1} q$.

(R4) (Ausräumen) Setze $a_{ik} := 0$ und $a_{i,k+1} := 0$.

(R5) Gib das Ergebnis $(a_{ij})_{j=1\dots m}$ aus, d.h. die i-te Zeile von A.

Beweis:
Schritt (3), insbesondere auch der Unteralgorithmus `B2TMDMDSMRow`, ergibt sich als Anwendung der modifizierten 2-Schritt-Rekursion aus Lemma 13.2. Dabei kann die Variable c_0 aus Schritt (3) übernommen werden und muß nicht noch einmal eigens berechnet werden. Im übrigen folgt die Behauptung analog zu Satz 12.2 (`B1TMDSM`), vgl. dazu auch den ursprünglichen Algorithmus `B2TMDMD` aus Satz 4.6. □

13.4 Bemerkung zur Implementierung (B2TMDMDSM)
(*Zuweisungen*) Bei dem Algorithmus `B2TMDMDSM` aus Satz 13.3 kann in Schritt (3) der Unteralgorithmus `B1TMDSMRow` so modifiziert werden, daß der $(k+1)$-te Diagonalkoeffizient nicht berechnet, sondern $a_{k+1,k+1} := c_0$ gesetzt wird, vgl. Bemerkung 13.1b).

Außerdem wird die Variable c_0, die nur von k abhängt, als Argument an `B2TMDMDSMRow` übergeben, um zu vermeiden, daß sie zur Berechnung der Zeilen $i = k+2,\dots,n$ in Schritt (3) jedesmal wieder unnötigerweise ermittelt wird.

14 Diagonalisierung mit dem Bareiss-1-Schritt-Verfahren

Auch das Bareiss-1-Schritt-Verfahren zur Diagonalisierung läßt sich mit der speziellen Mul-

tiplikation $\otimes$ modifizieren, vgl. Abschnitt 5.

Wie immer bezeichne R einen Integritätsring, wobei die Variablen $X_1, \ldots, X_{n-1}$, die nicht schon in R liegen sollen, unabhängig über R seien. Die $n \times m$-Matrix A über $R[X_1, \ldots, X_{n-1}]$ erfülle die fünf Voraussetzungen

(V1) $n \leq m$,
(V2) $\operatorname{rang} A = n$,
(V3) Zeilentausch ist nicht nötig,
(V4) für $k = 1, \ldots, n$ gilt $j_k = k$, d.h. die Zeilenstufen liegen auf der Hauptdiagonalen,
(V5) für $i = 1, \ldots, n-1$ sei $a_{ii} = X_i$ und sonst sei $a_{ij} \in R$, d.h. die übrigen Koeffizienten von A enthalten die Variablen $X_1, \ldots, X_{n-1}$ nicht.

Damit ergibt sich also eine Matrix der Form

$$A = \begin{pmatrix} X_1 & a_{12} & \cdots & \cdots & \cdots & \cdots & \cdots & a_{1m} \\ a_{21} & X_2 & \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & & \vdots \\ a_{n-1,1} & \cdots & \cdots & a_{n-1,n-2} & X_{n-1} & a_{n-1,n} & \cdots & a_{n-1,m} \\ a_{n1} & \cdots & \cdots & \cdots & a_{n,n-1} & a_{nn} & \cdots & a_{n,m} \end{pmatrix},$$

also eine Matrix, wie man sie erhält, wenn man bei einer Matrix aus $\operatorname{Mat}_{n\times m}(R)$ die ersten $(n-1)$ Koeffizienten auf der Hauptdiagonalen durch $X_1, \ldots, X_{n-1}$ ersetzt.

Genau wie in Abschnitt 5 (Diagonalisierung mit dem Bareiss-1-Schritt-Verfahren) wird nun ausgehend von der Matrix A die Folge der Matrizen $D^{(k)}$, $k = 0, \ldots, n$, definiert, nämlich durch

$$D^{(k)} = (d_{ij}^{(k)})_{\substack{i=1\ldots n \\ j=1\ldots m}} := \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\ldots k \\ j=1\ldots m}} \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\ldots n \\ j=1\ldots m}} \end{pmatrix} = \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\ldots k \\ j=1\ldots m}} \\ (a_{ij}^{(k)})_{\substack{i=k+1\ldots n \\ j=1\ldots m}} \end{pmatrix}.$$

Außerdem setzt man

$$d_{00}^{(0)} := 1.$$

In der Matrix $D^{(k)}$ sind bekanntlich die ersten k Spalten über und unter der Diagonalen ausgeräumt. Die Folge dieser Matrizen führt die Matrix $A = D^{(0)}$ in die diagonalisierte Matrix $D^{(n)} = (\delta_{ij}^{(n)})$ über, indem sukzessive von links nach rechts eine Spalte nach der anderen über und unter der Diagonalen ausgeräumt wird. Statt mit der herkömmlichen Bareiss-1-Schritt-Formel (Lemma 5.4) sollen diese Matrizen $D^{(k)}$ nun aber wieder mit einer modifizierten Formel rekursiv berechnet werden. Dazu betrachtet man für $k = 1, \ldots, n$ die Matrix

$$D^{(k-1)} = (d_{ij}^{(k-1)}) = \begin{pmatrix} (\delta_{ij}^{(k-1)})_{\substack{i=1\ldots k-1 \\ j=1\ldots m}} \\ (a_{ij}^{(k-1)})_{\substack{i=k\ldots n \\ j=1\ldots m}} \end{pmatrix},$$

in der ober- und unterhalb der Diagonalen die ersten $(k-1)$ Spalten ausgeräumt sind, und den Übergang zur direkten Nachfolgerin

$$D^{(k)} = (d_{ij}^{(k)}) = \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\ldots k \\ j=1\ldots m}} \\ (a_{ij}^{(k)})_{\substack{i=k+1\ldots n \\ j=1\ldots m}} \end{pmatrix},$$

in der nun ober- und unterhalb der Diagonalen die ersten k Spalten ausgeräumt sind.

Aus Lemma 5.4 (1-Schritt-Rekursion zur Diagonalisierung) folgt sofort

14.1 Lemma (modifizierte 1-Schritt-Rekursion zur Diagonalisierung)
Die Matrizen A und $D^{(k)}$, $k = 0, \ldots, n$, seien wie oben. Für $k = 1, \ldots, n$ betrachtet man die Matrix $D := D^{(k-1)}$, wobei

$$t := d_{k-1,k-1} = x + r \qquad mit \quad x := X_1 \cdots X_{k-1} \qquad und \qquad r := t - x \qquad sei.$$

Ferner bezeichne q die Summe

$$q := x - r + r \overset{k-1}{\underset{1}{\otimes}} r - r \overset{k-1}{\underset{1}{\otimes}} r \overset{k-1}{\underset{1}{\otimes}} r \pm \ldots$$

Die Matrix $E = (e_{ij}) \in \mathrm{Mat}_{n\times m}(R[X_1, \ldots, X_{n-1}])$ sei wie folgt definiert:
(1) Die Zeile k von E ist genau wie in D.
(2) Für alle anderen Zeilen $i \in \{1, \ldots, n\} \setminus \{k\}$ gilt für $j = 1, \ldots, m$:

$$e_{ij} := \left(d_{kk} \overset{k-1}{\underset{1}{\otimes}} d_{ij} - d_{ik} \overset{k-1}{\underset{1}{\otimes}} d_{kj} \right) \overset{k-1}{\underset{1}{\otimes}} q.$$

Dann gilt $E = D^{(k)}$. Insbesondere für die Zeilenstufenkoeffizienten der ersten k Zeilen gilt für $i = 1, \ldots, k$

$$d_{ii}^{(k)} = d_{kk}.$$

Beweis:
Die Behauptung folgt aus Lemma 5.4 (1-Schritt-Rekursion zur Diagonalisierung) ebenso, wie die modifizierte 1-Schritt-Rekursion zur Trigonalisierung (Lemma 12.2) aus der herkömmlichen 1-Schritt-Rekursion zur Trigonalisierung (Lemma 3.5) folgt. Unter Verwendung der Formeln aus Lemma 11.9 bzw. Korollar 11.10 modifiziert sich die in 5.4(2) gegebene Definition

$$e_{ij} := \frac{d_{kk} \cdot d_{ij} - d_{ik} \cdot d_{kj}}{t}$$

so wie oben angegeben. □

Als Modifikation von Satz 5.6 (`B1DMD`) folgt somit

14.2 Satz (Algorithmus B1DMDSM)
Sei A eine $n \times m$-Matrix über einem Integritätsring R mit den Eigenschaften (V1) bis (V4), d.h. daß $n \leq m$ und $\operatorname{rang} A = n$ sei, daß Zeilentausch nicht nötig sei in A und daß die Zeilenstufen auf der Diagonalen liegen, also $j_k = k$ für $k = 1, \ldots, n$. Der Unteralgorithmus `B1DMDSMRow` *sei wie anschließend definiert. Betrachte die folgenden Instruktionen:*

(1) (Substitution) Ersetze in A die ersten $(n-1)$ Diagonalkoeffizienten durch die Variablen $X_1, \ldots, X_{n-1}$ und merke die Substitutionen $X_1 = a_{11}, \ldots, X_{n-1} = a_{n-1,n-1}$.
(2) Setze $k := 1$ und $q := 1$.
(3) (Spalte k ausräumen) Falls $k > n$ ist, fahre fort bei Schritt (5). Andernfalls ersetze alle Zeilen i von A außer der k-ten jeweils durch das Ergebnis von `B1DMDSMRow`(A, i, k, q). *Setze dann, falls $k \leq n - 1$ gilt, $x := X_1 \cdots X_k$ und $r := a_{kk} - x$ und berechne $q := x - r + r \otimes_1^k r \mp \ldots$.*
(4) Erhöhe k um 1 und wiederhole Schritt (3).
(5) (Rücksubstitution) Ersetze in A alle Variablen $X_1, \ldots, X_{n-1}$ durch die in Schritt (1) gemerkten Werte und gib das Ergebnis A aus.

Dies ist ein Algorithmus namens `B1DMDSM`(A)*, der als Ergebnis eine zu* A *äquivalente Matrix in Diagonalform ausgibt.*

Dabei berechnet der Unteralgorithmus `B1DMDSMRow`(A, i, k, q) *die* i*-te Zeile von* A*, d.h. räumt in Zeile* i *mit Hilfe von Zeile* k *die Spalte* k *aus:*

(R1) (Berechnen) Für $j = k+1, \ldots, m$ *ersetze* a_{ij} *erst durch* $a_{kk} \otimes_1^{k-1} a_{ij} - a_{ik} \otimes_1^{k-1} a_{kj}$*. Ersetze dann den neuen Koeffizienten* a_{ij} *durch* $a_{ij} \otimes_1^{k-1} q$*.*

(R2) (Ausräumen) Setze $a_{ik} := 0$*.*

(R3) (Zeilenstufenkoeffizient) Falls $i < k$ *gilt, dann setze* $a_{ii} := a_{kk}$*.*

(R4) Gib das Ergebnis $(a_{ij})_{j=1\ldots m}$ *aus, d.h. die* i*-te Zeile von* A*.*

Beweis:
Der in Schritt (3) verwendete Unteralgorithmus `B1DMDSMRow` ist die direkte Anwendung der 1-Schritt-Rekursion aus Lemma 14.1. Ansonsten folgt die Behauptung analog zu Satz 12.2 (`B1TMDSM`), vgl. dazu auch Satz 5.6 (`B1DMD`). □

14.3 Bemerkung zur Implementierung (B1DMDSM)
(*Zuweisungen*) Beim Algorithmus `B1DMDSM` aus Satz 14.2 werden wie üblich in Schritt (R3) die oberen Zeilenstufenkoeffizienten nicht berechnet, sondern gleich a_{kk} gesetzt. Den Umstand, daß am Ende alle Diagonalkoeffizienten identisch sind, macht man sich auch bei der Rücksubstitution (mit Hilfe des einfachen CoCoA-Programms `BackSubstDMD`) zunutze. Dabei genügt es, nur einen Diagonalkoeffizienten zurückzusubstituieren, alle anderen Diagonalkoeffizienten werden diesem dann gleich gesetzt.

15 Diagonalisierung mit dem Bareiss-2-Schritt-Verfahren

Mittels der Multiplikation $\otimes$ kann man auch das 2-Schritt-Verfahren zur Diagonalisierung aus Abschnitt 6 modifizieren, und zwar den Algorithmus `B2DMDMDSM` aus Satz 6.6, analog zur Modifikation des Algorithmus `B2TMDMDSM` in Abschnitt 13.

Wie immer bezeichne R einen Integritätsring, wobei die Unbestimmten $X_1, \ldots, X_{n-1}$, die nicht schon in R liegen sollen, unabhängig über R seien. Die $n \times m$-Matrix A über $R[X_1, \ldots, X_{n-1}]$ erfülle genau wie im vorangegangenen Abschnitt die Voraussetzungen

(V1) $n \leq m$,
(V2) $\operatorname{rang} A = n$,
(V3) Zeilentausch ist nicht nötig,
(V4) für $k = 1, \ldots, n$ gilt $j_k = k$, d.h. die Zeilenstufen liegen auf der Hauptdiagonalen,
(V5) für $i = 1, \ldots, n-1$ sei $a_{ii} = X_i$ und sonst sei $a_{ij} \in R$, d.h. die übrigen Koeffizienten von A enthalten die Variablen $X_1, \ldots, X_{n-1}$ nicht.

Üblicherweise betrachtet man zur Diagonalisierung von A die Folge der Matrizen $D^{(k)}$, $k = 0, \ldots, n$, die gegeben sind durch

$$D^{(k)} = (d_{ij}^{(k)})_{\substack{i=1\ldots n \\ j=1\ldots m}} := \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\ldots k \\ j=1\ldots m}} \\ \\ (a_{ij}^{(k)})_{\substack{i=k+1\ldots n \\ j=1\ldots m}} \end{pmatrix}.$$

Außerdem setzt man

$$d_{00}^{(0)} := 1.$$

Um die modifizierte 2-Schritt-Formel zur rekursiven Berechnung dieser Matrizen zu finden, betrachtet man für gerades $k = 2, \ldots, n$ die Matrix

$$D^{(k-2)} = (d_{ij}^{(k-2)}) = \begin{pmatrix} (\delta_{ij}^{(k-2)})_{\substack{i=1\ldots k-2 \\ j=1\ldots m}} \\ \\ (a_{ij}^{(k-2)})_{\substack{i=k-1\ldots n \\ j=1\ldots m}} \end{pmatrix},$$

in der ober- und unterhalb der Diagonalen die ersten $(k-2)$ Spalten ausgeräumt sind, und die übernächste Matrix

$$D^{(k)} = (d_{ij}^{(k)}) = \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\dots k\\ j=1\dots m}} \\ \\ (a_{ij}^{(k)})_{\substack{i=k+1\dots n\\ j=1\dots m}} \end{pmatrix},$$

in der nun ober- und unterhalb der Diagonalen die ersten k Spalten ausgeräumt sind.

So wie sich in Abschnitt 13 die 2-Schritt-Rekursion zur Trigonalisierung modifiziert (vgl. Lemma 13.2), folgt nun aus der 2-Schritt-Rekursion zur Diagonalisierung (Lemma 6.1) entsprechend die Variante:

15.1 Lemma (modifizierte 2-Schritt-Rekursion zur Diagonalisierung)
Die Matrizen A und $D^{(k)}$, $k = 0, \dots, n$, seien genau wie oben. Für ein gerades $k \in \{2, \dots, n\}$ betrachtet man die Matrix $D := D^{(k-2)}$, wobei

$$t := d_{k-2,k-2} = t + r \qquad \textit{mit} \quad x := X_1 \cdots X_{k-2} \qquad \textit{und} \qquad r := t - x \qquad \textit{sei.}$$

Ferner bezeichne q die Summe

$$q := x - r + r \overset{k-2}{\underset{1}{\otimes}} r - r \overset{k-2}{\underset{1}{\otimes}} r \overset{k-2}{\underset{1}{\otimes}} r \pm \dots$$

Die Matrizen $E = (e_{ij})$ und $\widetilde{E} = (\widetilde{e}_{ij})$ aus $\mathrm{Mat}_{n\times m}(R[X_1, \dots, X_{n-1}])$ seien folgendermaßen definiert:

(1) In den oberen Zeilen $i = 1, \dots, k-2$ und den unteren Zeilen $i = k+1, \dots, n$ gilt für $j = 1, \dots, m$

$$e_{ij} := \widetilde{e}_{ij} := \Big(c_0 \overset{k-2}{\underset{1}{\otimes}} d_{ij} - c_{i1} \overset{k-2}{\underset{1}{\otimes}} d_{k,j} + c_{i2} \overset{k-2}{\underset{1}{\otimes}} d_{k-1,j}\Big) \overset{k-2}{\underset{1}{\otimes}} q$$

mit den Faktoren

$$c_0 := \Big(d_{k-1,k-1} \overset{k-2}{\underset{1}{\otimes}} d_{k,k} - d_{k,k-1} \overset{k-2}{\underset{1}{\otimes}} d_{k-1,k}\Big) \overset{k-2}{\underset{1}{\otimes}} q$$

$$c_{i1} := \Big(d_{k-1,k-1} \overset{k-2}{\underset{1}{\otimes}} d_{i,k} - d_{i,k-1} \overset{k-2}{\underset{1}{\otimes}} d_{k-1,k}\Big) \overset{k-2}{\underset{1}{\otimes}} q$$

$$c_{i2} := \Big(d_{k,k-1} \overset{k-2}{\underset{1}{\otimes}} d_{i,k} - d_{i,k-1} \overset{k-2}{\underset{1}{\otimes}} d_{k,k}\Big) \overset{k-2}{\underset{1}{\otimes}} q.$$

(2) Für die Zeile k gilt für $j = 1, \dots, m$

$$e_{kj} := \widetilde{e}_{kj} := \Big(d_{k-1,k-1} \overset{k-2}{\underset{1}{\otimes}} d_{kj} - d_{k,k-1} \overset{k-2}{\underset{1}{\otimes}} d_{k-1,j}\Big) \overset{k-2}{\underset{1}{\otimes}} q.$$

(3) Für die Zeile $(k-1)$ gilt für $j = 1, \dots, m$:

$$\widetilde{e}_{k-1,j} := d_{k-1,j}$$

$$e_{k-1,j} := \Big(\widetilde{e}_{k,k} \overset{k-1}{\underset{1}{\otimes}} \widetilde{e}_{k-1,j} - \widetilde{e}_{k-1,k} \overset{k-1}{\underset{1}{\otimes}} \widetilde{e}_{k,j}\Big) \overset{k-1}{\underset{1}{\otimes}} \widetilde{q}.$$

Hierbei bezeichnet $\widetilde{q}$ das Polynom

$$\widetilde{q} := \widetilde{x} - \widetilde{r} + \widetilde{r} \overset{k-1}{\underset{1}{\otimes}} \widetilde{r} - \widetilde{r} \overset{k-1}{\underset{1}{\otimes}} \widetilde{r} \overset{k-1}{\underset{1}{\otimes}} \widetilde{r} \pm \ldots$$

mit

$$\widetilde{x} := X_1 \cdots X_{k-1} \quad \textit{und} \quad \widetilde{r} := \widetilde{e}_{k-1,k-1} - \widetilde{x}.$$

Dann gilt $E = D^{(k)}$. Insbesondere für die Zeilenstufenkoeffizienten der ersten k Zeilen gilt für $i = 1, \ldots, k$

$$d_{ii}^{(k)} = c_0.$$

Beweis:
Die Behauptung folgt sofort aus Lemma 6.1 (2-Schritt-Rekursion zur Diagonalisierung) und Bemerkung 6.2 (Determinantenentwicklung), da die verwendeten Formeln analog zu früheren Beweisen modifiziert werden können, vgl. Lemma 12.1 (modifizierte 1-Schritt-Rekursion zur Trigonalisierung), Lemma 13.2 (modifizierte 2-Schritt-Rekursion zur Trigonalisierung) und Lemma 14.1 (modifizierte 1-Schritt-Rekursion zur Diagonalisierung).
□

Damit führt nun Satz 6.6 (`B2DMDMD`) zu der Variante

15.2 Satz (Algorithmus B2DMDMDSM)
Sei A eine $n \times m$-Matrix über einem Integritätsring R mit den Eigenschaften (V1) bis (V4), d.h. daß $n \leq m$ und $\operatorname{rang} A = n$ *sei, daß Zeilentausch nicht nötig sei in A und daß die Zeilenstufen auf der Diagonalen liegen, also $j_k = k$ für $k = 1, \ldots, n$. Der Unteralgorithmus* `B1DMDSMRow` *sei genau wie in Satz 14.2,* `B2DMDMDSMRow` *sei wie anschließend definiert. Betrachte die folgenden Instruktionen:*

(1) (Substitution) Ersetze in A die ersten $(n-1)$ Diagonalkoeffizienten durch die Variablen $X_1, \ldots, X_{n-1}$ und merke die Substitutionen $X_1 = a_{11}, \ldots, X_{n-1} = a_{n-1,n-1}$.

(2) Setze $k := 1$ und $q := 1$.

(3) (Spalte n ausräumen) Falls $k > n$ ist, fahre fort bei Schritt (6). Falls $k = n$ ist, dann ersetze alle Zeilen $i = 1, \ldots, n-1$ jeweils durch das Ergebnis von `B1DMDSMRow`(A, i, k, q). *Erhöhe k um 1 und fahre fort bei Schritt (6).*

(4) (Spalten $k, k+1$ ausräumen) Berechne $c_0 := \left(a_{k,k} \otimes_1^{k-1} a_{k+1,k+1} - a_{k+1,k} \otimes_1^{k-1} a_{k,k+1}\right) \otimes_1^{k-1} q$. Ersetze die Zeilen $i \in \{1, \ldots, n\} \setminus \{k, k+1\}$ von A jeweils durch das Ergebnis von `B2DMDMDSMRow`(A, i, k, q, c_0). *Ersetze die Zeile $(k+1)$ von A durch das Ergebnis von* `B1DMDSMRow`$(A, k+1, k, q)$. *Setze $x := X_1 \cdots X_k$ und $r := a_{kk} - x$. Berechne $q := x - r + r \otimes_1^k r \mp \ldots$ Ersetze die Zeile k von A durch das Ergebnis von* `B1DMDSMRow`$(A, k, k+1, q)$. *Setze dann, falls $k \leq n-2$ gilt, $x := X_1 \cdots X_{k+1}$ und $r := a_{k+1,k+1} - x$ und berechne $q := x - r + r \otimes_1^{k+1} r \mp \ldots$*

(5) Erhöhe k um 2 und fahre fort bei Schritt (3).

(6) (Rücksubstitution) Ersetze in A alle Variablen $X_1, \ldots, X_{n-1}$ durch die in Schritt (1) gemerkten Werte und gib das Ergebnis A aus.

Dies ist ein Algorithmus namens `B2DMDMDSM`(A), *der als Ergebnis eine zu A äquivalente Matrix in Diagonalform ausgibt.*

Dabei berechnet der Unteralgorithmus `B2DMDMDSMRow`(A, i, k, q, c_0) *die i-te Zeile von A, d.h. räumt in Zeile i mit Hilfe der Zeilen $k, k+1$ die Spalten $k, k+1$ aus:*

(R1) (Kofaktor berechnen) Berechne $c_1 := a_{kk} \otimes_1^{k-1} a_{i,k+1} - a_{ik} \otimes_1^{k-1} a_{k,k+1}$. Ersetze c_1 durch $c_1 \otimes_1^{k-1} q$.

(R2) (Kofaktor berechnen) Berechne dann $c_2 := a_{k+1,k} \otimes_1^{k-1} a_{i,k+1} - a_{ik} \otimes_1^{k-1} a_{k+1,k+1}$. *Ersetze* c_2 *durch* $c_2 \otimes_1^{k-1} q$.

(R3) (Determinante entwickeln) Für $j = k+2, \ldots, m$ *ersetze den Koeffizienten* a_{ij} *erst durch* $c_0 \otimes_1^{k-1} a_{ij} - c_1 \otimes_1^{k-1} a_{k+1,j} + c_2 \otimes_1^{k-1} a_{kj}$. *Ersetze dann den neuen Koeffizienten* a_{ij} *durch* $a_{ij} \otimes_1^{k-1} q$.

(R4) (Ausräumen) Setze $a_{ik} := 0$ *und* $a_{i,k+1} := 0$.

(R5) (Zeilenstufenkoeffizient) Falls $i < k$ *gilt, setze* $a_{ii} := c_0$.

(R6) Gib das Ergebnis $(a_{ij})_{j=1\ldots m}$ *aus, d.h. die i-te Zeile von A.*

Beweis:
Schritt (4), insbesondere auch der Unteralgorithmus `B2DMDMDSMRow`, ist die Anwendung der modifizierten 2-Schritt-Rekursion aus Lemma 15.1. Dabei kann die Variable c_0 aus Schritt (4) übernommen werden und muß daher nicht noch einmal eigens berechnet werden. Im übrigen folgt die Behauptung aus Satz 6.6 (`B2DMDMD`) und den vorangegangenen Überlegungen. □

15.3 Bemerkung zur Implementierung (B2DMDMDSM)

(*Zuweisungen*) Bei dem Algorithmus `B2DMDMDSM` aus Satz 15.2 kann in Schritt (4) der Unteralgorithmus `B1DMDSMRow` so modifiziert werden, daß der Diagonalkoeffizient nicht berechnet, sondern $a_{k+1,k+1} := c_0$ gesetzt wird. Im übrigen werden wie üblich bei jeder Rekursion in Schritt (R5) die oberen Zeilenstufenkoeffizienten $a_{ii} := c_0$ gesetzt, um die Berechnung zu sparen. Auch bei der folgenden Rücksubstitution genügt es wieder, nur einen der Zeilenstufenkoeffizienten zu berechnen und diesem die übrigen gleichzusetzen.

16 Diagonalisierung mit dem Forward-Backup-Verfahren

Wie die Bareiss-Verfahren lassen sich auch die Malashonok-Verfahren mittels $\otimes$ modifizieren. Zunächst untersucht man, wie dies für das Forward-Backup-Verfahren aus Abschnitt 7 abläuft. Satz 7.5 (`B1DMDFB,B2DMDMDFB`) führt zu

16.1 Satz (Algorithmen B1DMDSMFB, B2DMDMDSMFB)

Sei A eine $n \times m$*-Matrix über einem Integritätsring R mit* $n \leq m$ *und* $\operatorname{rang} A = n$, *ferner sei Zeilentausch nicht nötig in A, und die Zeilenstufen sollen auf der Diagonalen liegen, also* $j_k = k$ *für* $k = 1, \ldots, n$. *Die Algorithmen* `B1TMDSM` *und* `B2TMDMDSM` *seien wie in Satz 12.2 bzw. Satz 13.3,* `BackupRow` *sei wie in Satz 7.5.*

a) (Algorithmus `B1DMDSMFB`*) Betrachte die folgenden Instruktionen:*

(1) (Forward: trigonalisieren) Ersetze A durch das Ergebnis von `B1TMDSM`. *Setze* $k := n$.

(2) (Backup) Falls $k = 1$ *ist, gib das Ergebnis A aus. Andernfalls ersetze die Zeile* $(k-1)$ *durch das Ergebnis von* `BackupRow`$(A, \{k, \ldots, n\}, k, n)$. *Verkleinere k um 1 und wiederhole Schritt (2).*

Dies ist ein Algorithmus mit der Bezeichnung `B1DMDSMFB`(A), *der als Ergebnis eine diagonalisierte Matrix ausgibt, die zu A äquivalent ist.*

b) (Algorithmus `B2DMDMDFB`*) Ersetzt man in a) in Schritt (1)* `B1TMDSM` *durch* `B2TMDMDSM`, *so erhält man ebenfalls einen Algorithmus, der eine zu A äquivalente Matrix in Diagonalform zum Ergebnis hat. Dieser wird* `B2DMDMDSMFB`(A) *genannt.*

Beweis:
Die Behauptung folgt sofort aus Satz 7.5 (Algorithmen `B1DMDFB, B2DMDMDFB`), da es in Schritt (1) nur darauf ankommt, die trigonalisierte Matrix $(a_{ij}^{(i-1)})$ herzustellen. Dies wird auch von den modifizierten Algorithmen geleistet. Da nach Voraussetzung $\operatorname{rang} A = n$ und $j_k = k$ gilt, vereinfacht sich der Rest so wie angegeben, vgl. Satz 7.5. □

In den obigen Algorithmen wird nur der Forward-Schritt, also das Trigonalisieren, mit der Modifikation $\otimes$ durchgeführt. Auch der Backup-Schritt läßt sich modifizieren. Um die dafür nötige modifizierte Backup-Formel zu finden, seien wie üblich die Unbestimmten $X_1, \ldots, X_{n-1}$, die nicht schon in R liegen sollen, unabhängig über R. Ferner sei A eine $n \times m$-Matrix über $R[X_1, \ldots, X_{n-1}]$, die die Voraussetzungen erfülle

(V1) $n \leq m$,
(V2) $\operatorname{rang} A = n$,
(V3) Zeilentausch ist nicht nötig,
(V4) für $k = 1, \ldots, n$ gilt $j_k = k$, d.h. die Zeilenstufen liegen auf der Hauptdiagonalen,
(V5) für $i = 1, \ldots, n-1$ sei $a_{ii} = X_i$ und sonst sei $a_{ij} \in R$, d.h. die übrigen Koeffizienten von A enthalten die Variablen $X_1, \ldots, X_{n-1}$ nicht.

Dies ist also eine Matrix, wie man sie erhält, wenn man in einer Matrix aus $\mathrm{Mat}_{n\times m}(R)$ die ersten $(n-1)$ Koeffizienten auf der Diagonalen durch die Variablen $X_1, \ldots, X_{n-1}$ ersetzt.

Für den Backup-Schritt betrachtet man wie in Abschnitt 7 die Folge der Matrizen $F^{(k)}$, $k = n, n-1, \ldots, 0$, die wie folgt definiert sind. (Beachte, daß hier ausnahmsweise die Indexmenge absteigend von n bis 0 läuft.)

$$F^{(k)} := \begin{pmatrix} (\alpha_{ij}^{(i-1)})_{\substack{i=1\ldots k\\ j=1\ldots m}} \\ (\delta_{ij}^{(n)})_{\substack{i=k+1\ldots n\\ j=1\ldots m}} \end{pmatrix} = \begin{pmatrix} (a_{ij}^{(i-1)})_{\substack{i=1\ldots k\\ j=1\ldots m}} \\ (\delta_{ij}^{(n)})_{\substack{i=k+1\ldots n\\ j=1\ldots m}} \end{pmatrix}.$$

Die Matrix $F^{(k)}$ ist in den ersten k Zeilen schon trigonalisiert, aber noch nicht diagonalisiert, und in den unteren Zeilen $k+1, \ldots, n$ diagonalisiert. Statt mit der herkömmlichen Backup-Formel aus Abschnitt 7 sollen diese Matrizen nun wieder mit der modifizierten Variante rekursiv berechnet werden. Dazu betrachtet man für $k = n-1, n-2, \ldots, 1$ die Matrix

$$F^{(k)} = \begin{pmatrix} (a_{ij}^{(i-1)})_{\substack{i=1\ldots k\\ j=1\ldots m}} \\ (\delta_{ij}^{(n)})_{\substack{i=k+1\ldots n\\ j=1\ldots m}} \end{pmatrix},$$

in der die unteren Zeilen $k+1, \ldots, n$ ausgeräumt sind, und die direkt nachfolgende Matrix

$$F^{(k-1)} = \begin{pmatrix} (a_{ij}^{(i-1)})_{\substack{i=1\ldots k-1\\ j=1\ldots m}} \\ \\ (\delta_{ij}^{(n)})_{\substack{i=k\ldots n\\ j=1\ldots m}} \end{pmatrix},$$

in der nun die unteren Zeilen $k, \ldots, n$ ausgeräumt sind. Die herkömmliche Backup-Rekursion aus Lemma 7.2 ändert sich zu

16.2 Lemma (modifizierte Backup-Rekursion)
Sei A eine $n \times m$-Matrix mit den Eigenschaften (V1) bis (V5). Betrachte in der obigen Situation für $k = n-1, n-2, \ldots, 1$ die Matrix $F = (f_{ij}) := F^{(k)}$, in der der k-te Zeilenstufenkoeffizient bezeichnet werde mit

$$t := f_{kk} = a_{kk}^{(k-1)} = x + r \quad \textit{mit} \quad x := X_1, \ldots, X_k \quad \textit{und} \quad r := t - x.$$

Ferner sei

$$q := x - r + r \underset{1}{\overset{k}{\otimes}} r + r \underset{1}{\overset{k}{\otimes}} r \underset{1}{\overset{k}{\otimes}} \pm \ldots$$

Die Matrix $E = (e_{ij})$ aus $\mathrm{Mat}_{n\times m}(R[X_1, \ldots, X_{n-1}])$ sei wie folgt definiert:

(1) Die Zeile k ist für $j = 1, \ldots, m$ gegeben durch

$$e_{kj} := \Big(f_{nn} \overset{k}{\underset{1}{\otimes}} f_{kj} \; - \sum_{\lambda=k+1}^{n} f_{k\lambda} \overset{k}{\underset{1}{\otimes}} f_{\lambda j} \Big) \overset{k}{\underset{1}{\otimes}} q.$$

(2) Alle anderen Zeilen von E sind genau wie in F.
Dann gilt $E = F^{(k-1)}$. Insbesondere gilt für den k-ten Zeilenstufenkoeffizienten

$$f_{kk}^{(k-1)} = f_{k+1,k+1} = \ldots = f_{nn}.$$

Beweis:
Die Behauptung folgt aus Lemma 7.2, wobei die in 7.2(1) gegebene Formel

$$e_{kj} := \frac{f_{nn} \cdot f_{kj} \; - \; \sum_{\lambda=k+1}^{n} f_{k\lambda} \cdot f_{\lambda j}}{t}.$$

in der üblichen Art und Weise modifiziert werden kann. □

16.3 Satz (Algorithmen B1DMDFBSM, B2DMDMDFBSM)
Sei A eine $n \times m$-Matrix über einem Integritätsring R mit $n \leq m$ und $\operatorname{rang} A = n$*, ferner sei Zeilentausch nicht nötig in A, und die Zeilenstufen sollen auf der Diagonalen liegen, also $j_k = k$ für $k = 1, \ldots, n$. Die Algorithmen* `B1TMDSM` *und* `B2TMDMDSM` *seien wie in Satz 12.2 bzw. Satz 13.3,* `BackupSMRow` *sei wie anschließend definiert.*

a) (Algorithmus `B1DMDFBSM`*) Betrachte die folgenden Instruktionen:*

(1) (Substitution, forward: trigonalisieren) Ersetze A durch das Ergebnis, das man aus `B1TMDSM` *erhält, wenn man darin im ersten Schritt (Substitution) nicht nur die ersten $(n-2)$, sondern die ersten $(n-1)$ Diagonalkoeffizienten substituiert durch $X_1, \ldots, X_{n-1}$. Merke die Substitutionen $X_1 = a_{11}, \ldots, X_{n-1} = a_{n-1,n-1}$ und führe den letzten Schritt (Rücksubstitution) noch nicht durch. Setze dann $k := n$.*

(2) (Backup) Falls $k = 1$ ist, fahre fort bei Schritt (3). Andernfalls ersetze die Zeile $(k-1)$ durch das Ergebnis von `BackupSMRow`(A, k, n)*. Verkleinere k um 1 und wiederhole Schritt (2).*

(3) (Rücksubstitution) Ersetze in der Matrix A alle Variablen $X_1, \ldots, X_{n-1}$ durch die in Schritt (1) gemerkten Werte und gib das Ergebnis A aus.

Dies ist ein Algorithmus namens `B1DMDFBSM`(A)*, der als Ergebnis eine diagonalisierte Matrix ausgibt, die zu A äquivalent ist.*

Hierbei wird mit dem Unteralgorithmus `BackupSMRow`(A, k, n) *die Zeile $(k-1)$ neu berechnet, d.h. in ihr werden die Spalten $k, \ldots, n$ mit Hilfe der Zeilen $k, \ldots, n$ ausgeräumt.*

(R1) (Berechnen) Setze $t := X_1 \cdots X_{k-1}$, $r := a_{k-1,k-1} - t$ und berechne außerdem $q := t - r + r \otimes_1^{k-1} \mp \ldots$. Für $j \in \{n+1, \ldots, m\}$ ersetze zunächst die Koeffizienten $a_{k-1,j}$ durch $a_{nn} \otimes_1^{k-1} a_{k-1,j} \; - \sum_{\lambda=k}^{n} a_{k-1,\lambda} \otimes_1^{k-1} a_{\lambda j}$. Ersetze dann die neuen Koeffizienten $a_{k-1,j}$ durch $a_{k-1,j} \otimes_1^{k-1} q$. Setze den Zeilenstufenkoeffizienten $a_{k-1,k-1} := a_{nn}$.

(R2) (Ausräumen) Setze für $j = k, \ldots, n$ jeweils $a_{k-1,j} := 0$.

(R3) Gib das Ergebnis $(a_{k-1,j})_{j=1\ldots m}$, also die Zeile $(k-1)$ von A aus.

b) (Algorithmus `B2DMDMDFBSM`*) Ersetzt man in a) in Schritt (1) den Algorithmus* `B1TMDSM` *durch* `B2TMDMDSM`*, so erhält man ebenfalls einen Algorithmus, der eine zu A äquivalente Matrix in Diagonalform zum Ergebnis hat. Er wird* `B2DMDMDFBSM`(A) *genannt.*

Beweis:
Die Behauptung folgt aus Satz 7.5 (Algorithmen `B1DMDFB,B2DMDMDFB`) und aus den vorangegangenen Überlegungen, insbesondere aus der modifizierten Backup-Rekursion (Lemma 16.2). Da nun auch der $(n-1)$-te Diagonalkoeffizient als Teiler auftreten kann, werden in Schritt (1) zur Trigonalisierung die ersten $(n-1)$ und nicht wie bei der Trigonalisierung üblich die ersten $(n-2)$ Koeffizienten auf der Diagonalen substituiert. □

16.4 Bemerkung zur Implementierung (B1DMDFBSM, B2DMDMDFBSM)
Beim Algorithmus `B1DMDFBSM` aus Satz 16.3 wird in Schritt (1) der Algorithmus `B1TMDSM` in leicht abgewandelter Form verwendet. Um diesen Schritt zu implementieren, muß man bei `B1TMDSM` wie beschrieben die Substitution abändern und die Rücksubstitution weglassen. Ansonsten bleibt `B1TMDSM` unverändert. Analog gilt dies auch für die Algorithmen `B2DMDMDFBSM` bzw. `B2TMDMDSM`.

17 Diagonalisierung mit dem Malashonok-1-Schritt-Verfahren

Zum Abschluß dieses Kapitels wird nun noch die Modifikation des Malashonok-1-Schritt-Verfahrens mit der Multiplikation $\otimes$ untersucht.

Wie immer bezeichne R einen Integritätsring, und die Unbestimmten $X_1, \ldots, X_{n-1}$ sollen unabhängig über R sein, aber nicht schon in R liegen. Ferner sei A wieder eine $n \times m$-Matrix über $R[X_1, \ldots, X_{n-1}]$, die wie in den vorangegangenen Abschnitten die fünf Voraussetzungen erfülle

(V1) $n \leq m$,
(V2) $\operatorname{rang} A = n$,
(V3) Zeilentausch ist nicht nötig,
(V4) für $k = 1, \ldots, n$ gilt $j_k = k$, d.h. die Zeilenstufen liegen auf der Hauptdiagonalen,
(V5) für $i = 1, \ldots, n-1$ sei $a_{ii} = X_i$ und sonst sei $a_{ij} \in R$, d.h. die übrigen Koeffizienten von A enthalten die Variablen $X_1, \ldots, X_{n-1}$ nicht.

Wie in Abschnitt 8 betrachtet man die Matrizen $A =: G^{(0)} = G^{(1)}, G^{(2)}, \ldots, G^{(n)}$, die allgemein für $k = 1, \ldots, n$ von folgender Form sind:

$$G^{(k)} := \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\ldots k \\ j=1\ldots m}} \\ \\ (a_{ij})_{\substack{i=k+1\ldots n \\ j=1\ldots m}} \end{pmatrix}.$$

$G^{(k)}$ bezeichnet bekanntlich die Matrix, in der die $k \times k$-Blockmatrix links oben schon diagonalisiert ist und die unteren Zeilen $k+1, \ldots, n$ unverändert genau wie in A sind. Die Folge der so definierten Matrizen führt A in Diagonalform über, indem die bereits diagonalisierte Blockmatrix links oben sukzessive um die nächste Zeile und die nächste Spalte erweitert wird, d.h. indem abwechselnd die darunterliegende Zeile und die danebenstehende Spalte (in den oberen Zeilen) ausgeräumt werden.

Um für $k = 2, \ldots, n$ diese Matrizen rekursiv zu berechnen, betrachtet man die Matrix

$$G^{(k-1)} = \begin{pmatrix} (\delta_{ij}^{(k-1)})_{\substack{i=1\ldots k-1 \\ j=1\ldots m}} \\ \\ (a_{ij})_{\substack{i=k\ldots n \\ j=1\ldots m}} \end{pmatrix}$$

und die direkt nachfolgende Matrix

$$G^{(k)} = \begin{pmatrix} (\delta_{ij}^{(k)})_{\substack{i=1\ldots k\\ j=1\ldots m}} \\ \\ (a_{ij})_{\substack{i=k+1\ldots n\\ j=1\ldots m}} \end{pmatrix}$$

Aus der Malashonok-1-Schritt-Rekursion (Lemma 8.2) folgt sofort die Variante

17.1 Lemma (modifizierte Malashonok-1-Schritt-Rekursion)
Sei A eine $n \times m$-Matrix mit den Eigenschaften (V1) bis (V5). In obiger Situation betrachtet man für $k = 2, \ldots, n$ die Matrix $G := G^{(k-1)}$, in der der $(k-1)$-te Zeilenstufenkoeffizient bezeichnet werde mit

$$t := g_{k-1,k-1} = x + r \quad \textit{mit} \quad x := X_1 \cdots X_{k-1} \quad \textit{und} \quad r := t - x.$$

Ferner sei

$$q := x - r + r \overset{k-1}{\underset{1}{\otimes}} r - r \overset{k-1}{\underset{1}{\otimes}} r \overset{k-1}{\underset{1}{\otimes}} r \pm \ldots$$

Die Matrizen $E, \widetilde{E} \in \mathrm{Mat}_{n\times m}(R[X_1, \ldots, X_{n-1}])$ seien folgendermaßen definiert:

(1) Die Zeile k ist für $j = 1, \ldots, m$ gegeben durch

$$e_{kj} := \widetilde{e}_{kj} := t \cdot g_{kj} - \sum_{\lambda=1}^{k-1} g_{k\lambda} \cdot g_{\lambda j}.$$

(2) Die unteren Zeilen $i = k+1, \ldots, n$ von E und $\widetilde{E}$ sind genau wie in G.

(3) Für die oberen Zeilen $i = 1, \ldots, k-1$ gilt für $j = 1, \ldots, m$:
In $\widetilde{E}$ sind die Zeilen genau wie in G.
In E gilt

$$e_{ij} := (\widetilde{e}_{kk} \overset{k-1}{\underset{1}{\otimes}} \widetilde{e}_{ij} - \widetilde{e}_{ik} \overset{k-1}{\underset{1}{\otimes}} \widetilde{e}_{kj}) \overset{k-1}{\underset{1}{\otimes}} q.$$

Dann gilt $E = G^{(k)}$.

Beweis:
Die Behauptung folgt sofort aus der Malashonok-1-Schritt-Rekursion aus Lemma 8.2, wobei die in 8.2(1) angegebene Formel keine Division beinhaltet und daher nicht modifiziert wird. Die Bareiss-1-Schritt-Formel aus 8.2(3)

$$e_{ij} := \frac{\widetilde{e}_{kk} \cdot \widetilde{e}_{ij} - \widetilde{e}_{ik} \cdot \widetilde{e}_{kj}}{t}.$$

kann auf die übliche Art und Weise mit der Multiplikation $\otimes_1^{k-1}$ modifiziert werden, da durch den linearen Hauptminor $t := g_{k-1,k-1} = a_{k-1,k-1}^{(k-2)}$ geteilt wird. □

Damit folgt nun aus Satz 8.4 (`Mal1DMD`) sofort die Variante

17.2 Satz (Algorithmus Mal1DMDSM)
Sei A eine $n \times m$-Matrix über einem Integritätsring R mit $n \leq m$ und $\operatorname{rang} A = n$, *ferner sei Zeilentausch nicht nötig in A, und die Zeilenstufen sollen auf der Diagonalen liegen, also $j_k = k$ für $k = 1, \ldots, n$. Der Algorithmus* `B1DMDSMRow` *sei wie in Satz 14.2,* `Mal1DMDSMRow` *sei wie nachfolgend definiert. Betrachte die folgenden Instruktionen:*

(1) (Substitution) Ersetze in A die ersten $(n-1)$ Diagonalkoeffizienten durch die Variablen $X_1, \ldots, X_{n-1}$ und merke die Substitutionen $X_1 = a_{11}, \ldots, X_{n-1} = a_{n-1,n-1}$. Setze $k := 1$.

(2) (Zeile $k+1$, Spalte $k+1$ ausräumen) Falls $k \geq n$ ist, fahre fort bei Schritt (3). Andernfalls ersetze die Zeile $(k+1)$ durch das Ergebnis von `Mal1DMDSMRow`(A, k). *Setze $x := X_1 \cdots X_k$, $r := a_{kk} - x$ und berechne $q := x - r + r \otimes_1^k r \mp \ldots$ Ersetze dann alle Zeilen $i = 1, \ldots, k$ jeweils durch das Ergebnis von* `B1DMDSMRow`$(A, i, k+1, q)$. *Erhöhe k um 1 und wiederhole Schritt (2).*

(3) (Rücksubstitution) Ersetze in A alle Variablen $X_1, \ldots, X_{n-1}$ durch die in Schritt (1) gemerkten Werte und gib das Ergebnis A aus.

Dies ist ein Algorithmus, der mit `Mal1DMDSM`(A) *bezeichnet wird und als Ergebnis eine zu A äquivalente Matrix in Diagonalform ausgibt.*

Die Zeile $(k+1)$ wird dabei mit dem Algorithmus `Mal1DMDSMRow`(A, k) *berechnet, d.h. es werden mit Hilfe der oberen Zeilen $1, \ldots, k$ die Spalten $1, \ldots, k$ ausgeräumt:*

(R1) (Berechnen) Für $j = k+1, \ldots, m$ ersetze den alten Koeffizienten $a_{k+1,j}$ jeweils durch $a_{kk} \cdot a_{k+1,j} \; - \; \sum_{\lambda=1}^{k} a_{k+1,\lambda} \cdot a_{\lambda j}$.

(R2) (Ausräumen) Für $j = 1, \ldots, k$ setze $a_{k+1,j} := 0$.

(R3) Gib das Ergebnis $(a_{k+1,j})_{j=1\ldots m}$, also die Zeile $(k+1)$ von A aus.

Beweis:
Dies folgt sofort aus Satz 8.4 (`Mal1DMD`) und aus den vorangegangenen Überlegungen, insbesondere aus der modifizierten Malashonok-1-Schritt-Rekursion (Lemma 17.1). □

17.3 Bemerkung zur Implementierung (Mal1DMDSM)

Nach Lemma 17.1 ist eine Modifikation von `Mal1DMDRow` beim Algorithmus `Mal1DMDSM` (Satz 17.2) überhaupt nicht nötig — trotzdem wird dort aber eigens der Unteralgorithmus `Mal1DMDSMRow` definiert. Warum?

Im ursprünglichen Algorithmus `Mal1DMD` wird mit dem Zeilentauschalgorithmus `Mal1-Pivot` schon der $(k+1)$-te Zeilenstufenkoeffizient c berechnet und anschließend bei `Mal1-DMDRow` weiterverwendet. Im modifizierten Algorithmus `Mal1DMDSM` entfällt das Zeilentauschen, daher muß der Zeilenstufenkoeffizient c noch berechnet werden. Dies wird im Algorithmus `Mal1DMDSMRow` erledigt. Ansonsten ist `Mal1DMDSMRow`(A, k) exakt wie `Mal1DMDRow`$(A, \{j_1, \ldots, j_{k+1}\}, k, c)$, wobei hier nach Voraussetzung sogar stets $j_k = k$ gilt. Daher vereinfachen sich die Schritte (R1) bis (R3) etwas.

17.4 Bemerkung (Dichotomie-Verfahren)

Die zwei zeilenweise vorgehenden Dichotomie-Algorithmen `MalDichHR` bzw. `MalDichVR` aus Satz 10.2 bzw. Satz 10.6 lassen sich analog mit der Multiplikation $\otimes$ abändern, da dort bei den Rekursionen stets durch einen linearen Hauptminor dividiert wird. Diese Modifikationen werden hier jedoch nicht mehr weiter ausgeführt. Für die beiden matrizenweisen Dichotomie-Algorithmen `MalDichHM` bzw. `MalDichVM` aus Satz 10.4 bzw. 10.8 müßte man die Matrizenmultiplikation so implementieren, daß dazu die spezielle Multiplikation $\otimes$ verwendet wird.

Denn es ist eines ausgezeichneten Mannes nicht würdig, wertvolle Stunden wie ein Sklave im Keller der einfachen Rechnungen zu verbringen. Diese Aufgaben könnten ohne Besorgnis abgegeben werden, wenn wir Maschinen hätten.

(Gottfried Wilhelm Leibniz)

In der Theorie lassen sich die Verfahren aus den vorangegangenen drei Kapiteln anhand ihrer Komplexitäten vergleichen, die der besseren Übersicht wegen zu Beginn der Abschnitte 18 und 19 nochmals in Tabellen und Diagrammen zusammengefaßt sind. Dazu definiert man eine Ordnungsrelation $\geq_K$ auf der Menge der Trigonalisierungs- bzw. Diagonalisierungs-Algorithmen wie folgt: Es gilt für zwei beliebige, mit `Alg1` und `Alg2` bezeichnete Algorithmen `Alg1` $\geq_K$ `Alg2` genau dann, wenn beim Algorithmus `Alg1` sowohl die Anzahl der Multiplikationen als auch die der Divisionen größer oder gleich der entsprechenden Anzahl beim Algorithmus `Alg2` ist.

Nicht immer ist ein nach theoretischen Erkenntnissen gutes Verfahren gleichermaßen zur praktischen Durchführung geeignet. Die in Kapitel I und II angegebenen Komplexitäten können nur einen groben Anhaltspunkt für die Güte der Algorithmen liefern, denn es wird dazu die Anzahl an Rechenoperationen, etwa an Multiplikationen oder Divisionen, gezählt. Dabei bleibt aber völlig unberücksichtigt, daß beispielsweise zur Multiplikation zweier "komplizierter" Polynome relativ großen Grades mit großen Koeffizienten ein Vielfaches an Rechenzeit aufgewandt werden muß, verglichen mit der Multiplikation zweier sehr "einfacher" Polynome. Je weiter die Rekursion bei einem Algorithmus fortgeschritten ist, desto komplizierter sind i.a. die in den zugehörigen Matrizenfolgen auftretenden Koeffizienten.

Außerdem werden in Kapitel III die Komplexitäten der modifizierten Sasaki-Murao-Verfahren außer acht gelassen. Wie lassen sich diese ermitteln, d.h. wie ist die spezielle, rekursiv definierte Multiplikation $\otimes$ zu bewerten? Stellt man eine Multiplikation der Form $\otimes_1^k$ mit der normalen Multiplikation $\cdot$ bzw. der normalen Division gleich, ohne all die einzelnen Operationen bei der rekursiv durchgeführten Multiplikation $\otimes$ mitzuzählen, ergibt sich bei den modifizierten Verfahren die Anzahl der $\otimes$-Multiplikationen als Summe der jeweiligen Anzahl der normalen Multiplikationen bzw. Divisionen. Schwieriger ist es, die Anzahl der einzelnen Operationen bei der rekursiven $\otimes$-Multiplikation abzuschätzen, denn

hier geht unter anderem die Art und Länge der beteiligten Polynome ein. Wollte man die normale Multiplikation als Vergleichsmaßstab heranziehen, müßte man auch bei dieser die Art und Länge der beteiligten Polynome in geeigneter Weise berücksichtigen, was letztlich einer realistischeren Einschätzung der benötigten Rechenzeit gleichkäme, als in den angegebenen Komplexitäten abgebildet.

Nichtzuletzt können bestimmte Algorithmen für bestimmte Matrizen schlicht und einfach besser geeignet sein als andere.

Geht man beim praktischen Vergleich systematisch vor, eröffnet sich dabei ein weites Feld an Kombinationsmöglichkeiten, wenn man die folgenden drei Gesichtspunkte berücksichtigt:

(1) Verfahren zur Trigonalisierung bzw. Diagonalisierung

Sei $\mathrm{Mat}_{n\times m}(R)$ der Matrizenring über einem Integritätsring R mit $n, m \in \mathbb{N}_+$.
Für beliebige Matrizen A aus $\mathrm{Mat}_{n\times m}(R)$ stehen folgende Algorithmen zur Verfügung:

a) (*Trigonalisierung*) `B1TMD`, vgl. Satz 3.7,
`B1TOD`, vgl. Bemerkung 3.9,
`B2TMD`, vgl. Satz 4.3,
`B2TOD`, vgl. Bemerkung 4.5,
`B2TMDMD`, vgl. Satz 4.6.

b) (*Diagonalisierung*) `B1DMD`, vgl. Satz 5.6,
`B1DOD`, vgl. Bemerkung 5.8,
`B2DMD`, vgl. Satz 6.3,
`B2DOD`, vgl. Bemerkung 6.5,
`B2DMDMD`, vgl. Satz 6.6,
`B1DMDFB`, vgl. Satz 7.5,
`B2DMDFB`, vgl. Satz 7.5,
`B2DMDMDFB`, vgl. Satz 7.5,
`Mal1DMD`, vgl. Satz 8.4.

Für speziellere Matrizen A aus $\mathrm{Mat}_{n\times m}(R)$ mit den Voraussetzungen

(V1) $n \leq m$,
(V2) $\operatorname{rang} A = n$,
(V3) Zeilentausch ist unnötig und
(V4) $j_k = k$ für $k = 1, \ldots, n$

stehen außerdem noch folgende Algorithmen bereit:

c) (*Trigonalisierung*) `B1TMDSM`, vgl. Satz 12.2,
`B2TMDMDSM`, vgl. Satz 13.3.

d) (*Diagonalisierung*) `MalDichHR`, vgl. Satz 10.2,
`MalDichHM`, vgl. Satz 10.4,
`MalDichVR`, vgl. Satz 10.6,
`MalDichVM`, vgl. Satz 10.8,
`B1DMDSM`, vgl. Satz 14.2,
`B2DMDMDSM`, vgl. Satz 15.2,
`B1DMDSMFB`, vgl. Satz 16.1,
`B2DMDMDSMFB`, vgl. Satz 16.1,
`B1DMDFBSM`, vgl. Satz 16.3,
`B2DMDMDFBSM`, vgl. Satz 16.3,
`Mal1DMDSM`, vgl. Satz 17.2.

(2) Zugrunde liegender Integritätsring R

Die oben angeführten Algorithmen sind prinzipiell für beliebige Integritätsringe R durchführbar. CoCoA gestattet die folgenden Integritätsringe:

a) $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{F}_p = \mathbb{Z}/(p)$ mit einer Primzahl $p \in \mathbb{N}$, $p \leq 32003$.

b) Polynomringe (bzw. die entsprechenden rationalen Funktionenkörper) über einem der in a) genannten Integritätsringe in endlich vielen Unbestimmten $x_1, \ldots, x_r$ mit $r \in \mathbb{N}$.

c) Sonstige Restklassenringe $S[x_1, \ldots, x_r]/\mathcal{P}$, wobei $S = \mathbb{Q}$ oder $S = \mathbb{Z}/(n)$ mit einem $n \in \mathbb{N}$, $n \leq 32767$, ist, wobei r in $\mathbb{N}$ liegt und $\mathcal{P} \subseteq S[x_1, \ldots, x_r]$ ein Primideal ist.

(3) Matrizen

Schließlich spielen die zu trigonalisierenden bzw. zu diagonalisierenden Matrizen eine entscheidende Rolle, präziser ausgedrückt:

a) der Typ (symmetrisch, Hankelmatrix, zufällig, etc.)

b) die Art und Größe der Koeffizienten (univariate oder multivariate Polynome, deren Grad und Länge etc.)

c) die Größe der $n \times m$-Matrizen, also die Größe von n und m.

In der Originalarbeit werden Testreihen anhand verschiedener Beispielklassen (generische Matrizen, Zufallsmatrizen u.a.) durchgeführt. Dabei konzentriert man sich auf die Verfahren mit Division, durchgeführt an Matrizen über Polynomringen der Form $\mathbb{Z}[x_1, \ldots, x_r]$ und $\mathbb{F}_p[x_1, \ldots, x_r]$ mit dem endlichen Körper $\mathbb{F}_p = \mathbb{Z}/(32003)$, $r \in \mathbb{N}$.

18 Vergleich der Trigonalisierungsverfahren

Zur Erinnerung seien hier zunächst die Komplexitäten der einzelnen Trigonalisierungsverfahren tabellarisch zusammengefaßt, für $n \times m$-Matrizen über einem beliebigen Integritätsing R. Die Anzahl der Additionen kann dabei im Grunde genommen vernachlässigt werden, da vor allem die Multiplikationen und Divisionen kostspielig sind, was die Rechenzeit betrifft.

Trigonalisierung von n × m-Matrizen mit n ≤ m				
Quellen	Algorithmus	Additionen	Multiplikationen	Divisionen
3.9	`B1TOD`	$\frac{1}{2}mn^2 - \frac{1}{6}n^3$	$mn^2 - \frac{1}{3}n^3$	0
3.7, 3.8	`B1TMD`	"	"	$\frac{1}{2}mn^2 - \frac{1}{6}n^3$
4.5	`B2TOD`	"	$\frac{3}{4}mn^2 - \frac{1}{4}n^3$	0
4.3, 4.4	`B2TMD`	"	"	$\frac{1}{4}mn^2 - \frac{1}{12}n^3$
4.6, 4.7	`B2TMDMD`	"	"	"

Tabelle 18a

Trigonalisierung von n × m-Matrizen mit n > m				
Quellen	Algorithmus	Additionen	Multiplikationen	Divisionen
3.9	`B1TOD`	$\frac{1}{2}mn^2 - \frac{1}{6}m^3$	$mn^2 - \frac{1}{3}m^3$	0
3.7, 3.8	`B1TMD`	"	"	$\frac{1}{2}mn^2 - \frac{1}{6}m^3$
4.5	`B2TOD`	"	$\frac{3}{4}mn^2 - \frac{1}{4}m^3$	0
4.3, 4.4	`B2TMD`	"	"	$\frac{1}{4}mn^2 - \frac{1}{12}m^3$
4.6, 4.7	`B2TMDMD`	"	"	"

Tabelle 18b

Tabelle 18b gilt übrigens auch für den Fall $n = m$ und ist dann identisch mit Tabelle 18a. Für quadratische $n \times n$-Matrizen ergibt sich somit

Trigonalisierung von n × n-Matrizen				
Quellen	Algorithmus	Additionen	Multiplikationen	Divisionen
3.9	`B1TOD`	$\frac{1}{3}n^3$	$\frac{2}{3}n^3$	0
3.7, 3.8	`B1TMD`	"	"	$\frac{1}{3}n^3$
4.5	`B2TOD`	"	$\frac{1}{2}n^3$	0
4.3, 4.4	`B2TMD`	"	"	$\frac{1}{6}n^3$
4.6, 4.7	`B2TMDMD`	"	"	"

Tabelle 18c

Aus den obigen Tabellen folgt somit für beliebige $n \times m$-Matrizen mit $n, m \in \mathbb{N}_+$

$$\texttt{B1TMD} \geq_K \texttt{B2TMD} =_K \texttt{B2TMDMD}.$$

Was die Komplexitäten betrifft, liegen demnach die beiden 2-Schritt-Algorithmen `B2TMD` und `B2TMDMD` gleichauf, der 1-Schritt-Algorithmus `B1TMD` schneidet schlechter ab. Laut Tabellen benötigt man für ein 2-Schritt-Verfahren nur halb soviel Divisionen und dreiviertel soviel Multiplikationen wie für das 1-Schritt-Verfahren. Von den Sasaki-Murao-Algorithmen `B1TMDSM` bzw. `B2TMDMDSM` erhofft man sich eine Verbesserung der Rechengeschwindigkeit.

18.1 Bemerkungen (Optimierung der Trigonalisierungsalgorithmen)
Da fast die gesamte Rechenzeit auf die Durchführung von Multiplikationen oder Divisionen entfällt, empfiehlt es sich sehr, von vornherein auf unnötige Operationen zu verzichten und nur die unbedingt notwendigen Berechnungen möglichst geschickt auszuführen.

a) Ein Ansatzpunkt besteht darin, die eingegebene Matrix —soweit möglich — zu vereinfachen, beispielsweise indem überflüssige Zeilen weggelassen werden. Sonst entfällt viel Zeit darauf, die Koeffizienten überflüssiger Zeilen mitzuberechnen, die beim Endergebnis gleich 0 sein werden. Ratsam erscheint es auch, von vornherein die Koeffizienten möglichst klein zu halten, indem man ggf. die Zeilen mit dem größten gemeinsamen Teiler kürzt. Außerdem sollte man den zugrunde liegenden Polynomring minimal wählen, also etwa die Variablenanzahl auf das nötige beschränken; möglicherweise genügt als Koeffizientenring schon ein endlicher Körper.

b) (*Zuweisungen*) Was die Programmierung betrifft, ist es vorteilhaft, solche Spalten, die ausgeräumt werden, gleich 0 zu setzen. Außerdem werden mehrfach benötigte Parameter nur einmal berechnet und als Argument an die Unterprogramme übergeben, wie z.B. der Kofaktor c_0 etc. (Dies gilt auch für die Diagonalisierungsalgorithmen, bei denen man zusätzlich den oberen Zeilenstufenkoeffizienten den entsprechenden Wert zuweisen kann.)

c) (*Multiplikationen, Divisionen*) Ein weiterer Ansatzpunkt liegt in der Optimierung der Division bzw. Multiplikation, indem man beispielsweise die Division mit der eingebauten CoCoA-Funktion `DivAlg` statt mit "/" berechnet oder die spezielle $\otimes$-Multiplikation verwendet. Darüberhinaus wäre es denkbar, die CoCoA-interne Division oder Multiplikation von Polynomen noch weiter zu optimieren, was hier aber nicht Gegenstand dieser Arbeit ist.

d) (*Schrittgröße*) Grundsätzlich sind 2-Schritt-Verfahren günstiger als 1-Schritt-Verfahren, wobei das 2-Schritt-Verfahren `B2TMDMD` mit zweimaliger Division häufig dem Verfahren `B2TMD` mit einmaliger Division durch einen quadrierten Teiler vorzuziehen ist. Öfter durch kleinere Polynome zu teilen und dadurch für die folgenden Rechenschritte jeweils wieder kleinere Zwischenergebnisse zu erhalten, ist im Endeffekt meist günstiger, als erst am Schluß ein größeres Polynom durch einen größeren Teiler zu dividieren.
Verfahren ohne Division sind i.a. wenig vorteilhaft, da sie mehr Zeit und Speicherplatz beanspruchen und Ergebnisse mit viel größeren Koeffizienten liefern, als dies bei den Verfahren mit Division der Fall ist.

e) Bei CoCoA-Funktionen sind Listenkonstruktionen häufig günstiger als Schleifenkonstruktionen, was sich bei den vorliegenden Verfahren aber nicht bemerkbar macht. Möglicherweise sind die betrachteten Matrizen für diesen Effekt noch zu klein. Da Schleifen ein Programm i.a. leichter lesbar erscheinen lassen als Listen und da sie den als Sätzen dargestellten Algorithmen in ihrer Formulierung näherstehen, überwiegen sie daher im source code.

f) Man besorgt sich einen schnelleren Rechner.

g) Über die oben genannten allgemeinen Optimierungsansätze hinaus finden sich zahlreiche Publikationen dazu, wie man die Verfahren für ganz spezielle Matrizen weiter verbessern kann.

18.2 Zusammenfassung der Testergebnisse zur Trigonalisierung

a) (*Bareiss-Verfahren*) Für $n \times m$-Matrizen mit $n \leq m$ erweist sich unter den drei Bareiss-Verfahren `B1TMD`, `B2TMD` und `B2TMDMD` der Algorithmus `B2TMDMD` als der beste, für Matrizen mit $n > m$ der Algorithmus `B2TMD`.

b) (*Sasaki-Murao-Verfahren*) Das 2-Schritt-Verfahren `B2TMDMDSM` erscheint günstiger als das 1-Schritt-Verfahren `B1TMDSM`. Bei den modifizierten Verfahren gilt es außerdem

zu beachten, daß für die Rücksubstitution unter Umständen eine beträchtliche Zeit aufgewendet werden muß. Bei generischen Matrizen u.ä. ist die Rücksubstitution dagegen sehr einfacher Natur.

c) (*Testsieger*) Für $n \times m$-Matrizen, die eher klein sind, also höchstens 6 Zeilen enthalten, oder deren Einträge univariate Polynome oder nur Zahlen sind, laufen die modifizierten Verfahren — sofern sie definiert sind — langsamer ab als die herkömmlichen. In diesem Fall ist ein Bareiss-2-Schritt-Algorithmus empfehlenswert, vgl. a). Ansonsten kann man mit dem Algorithmus `B2TMDMDSM` meist sehr viel Zeit sparen, wenn die Rücksubstitution nicht allzu aufwendig ist, vgl. b).

d) (*Komplexitäten*) Im wesentlichen bestätigen die Testreihen die aufgrund der Komplexitäten zu erwartende Rangfolge der Algorithmen, vgl. die Tabellen 18.a–c.

19 Vergleich der Diagonalisierungsverfahren

In diesem Abschnitt werden nun die verschiedenen Diagonalisierungsverfahren bewertet. Eine tabellarische Zusammenfassung aller Komplexitäten zeigt zunächst, wie gut die Algorithmen theoretisch geeignet sind für beliebige $n \times m$-Matrizen über einem Integritätsring R.

Diagonalisierung von n × m-Matrizen mit n ≤ m				
Quellen	Algorithmus	Additionen	Multiplikationen	Divisionen
5.8	`B1DOD`	$mn^2 - \frac{1}{2}n^3$	$2mn^2 - n^3$	0
5.6, 5.7	`B1DMD`	"	"	$mn^2 - \frac{1}{2}n^3$
6.5	`B2DOD`	"	$\frac{3}{2}mn^2 - \frac{3}{4}n^3$	0
6.3, 6.4	`B2DMD`	"	"	$\frac{1}{2}mn^2 - \frac{1}{4}n^3$
6.6, 6.7	`B2DMDMD`	"	"	"
7.5, 7.6	`B1DMDFB`	$mn^2 - \frac{2}{3}n^3$	$\frac{3}{2}mn^2 - \frac{5}{6}n^3$	$\frac{1}{2}mn^2 - \frac{1}{6}n^3$
7.5, 7.6	`B2DMDFB`	"	$\frac{5}{4}mn^2 - \frac{3}{4}n^3$	$\frac{1}{4}mn^2 - \frac{1}{12}n^3$
7.5, 7.6	`B2DMDMDFB`	"	"	"
8.4, 8.5	`Mal1DMD`	"	$\frac{3}{2}mn^2 - n^3$	$\frac{1}{2}mn^2 - \frac{1}{3}n^3$
10.2, 10.3 10.6, 10.7	`MalDichHR` `MalDichVR`	$\frac{1}{3}mn^2 - \frac{5}{21}n^3$	$\frac{1}{3}mn^2 - \frac{5}{21}n^3$	$pmn - 2mn + \frac{1}{2}n^2 - \frac{1}{2}pn^2$
10.4, 10.5 10.8, 10.9	`MalDichHM` `MalDichVM`	$O(mn^{1+\beta})$	$O(mn^{1+\beta})$	"

Tabelle 19a

Dabei bezeichnet $O(mn^{1+\beta})$ die Komplexität der Matrizenmultiplikation mit einem $\beta \in \mathbb{R}$, $0 < \beta \leq 1$, und es gilt $p = \log_2 n$.

Ordnet man die oben angeführten Algorithmen bzgl. $\geq_K$ an, so ergibt sich für $n \times m$-Matrizen mit $n \leq m$ das folgende Diagramm:

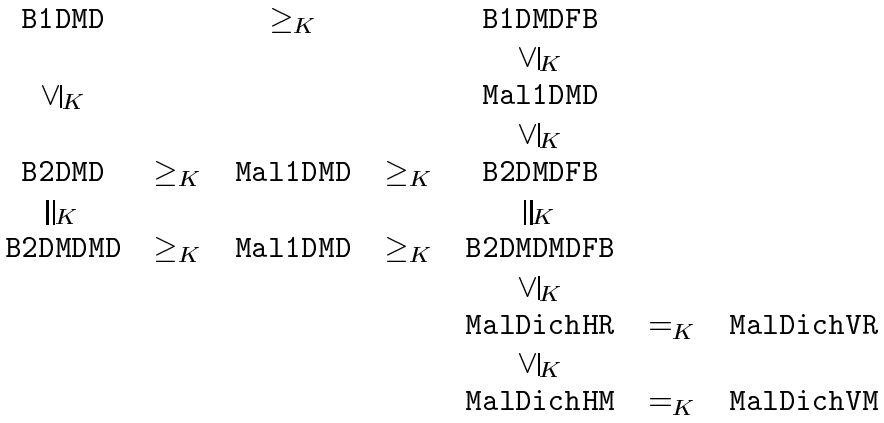


Diagramm 19a

In Bezug auf die Komplexität scheinen bei den $n \times m$-Matrizen mit $n \leq m$ die Dichotomie-Verfahren am besten geeignet zu sein, gefolgt von den 2-Schritt-Forward-Backup-Verfahren und dem Malashonok-1-Schritt-Verfahren. Am schlechtesten schneidet dabei das Bareiss-1-Schritt-Verfahren ab, dazwischen liegen die beiden Bareiss-2-Schritt-Verfahren sowie das 1-Schritt-Forward-Backup-Verfahren.

Die Algorithmen `B1DMDFB` und `B2DMD` bzw. `B2DMDMD` sind bzgl. $\geq_K$ nicht vergleichbar. Für den erstgenannten sind $\frac{1}{12}n^3$ Multiplikationen weniger bzw. $\frac{1}{12}n^3$ Divisionen mehr nötig als für die beiden letztgenannten Algorithmen.

Diagonalisierung von n × m-Matrizen mit n > m				
Quellen	Algorithmus	Additionen	Multiplikationen	Divisionen
5.8	`B1DOD`	$\frac{1}{2}m^2n$	m^2n	0
5.6, 5.7	`B1DMD`	"	"	$\frac{1}{2}m^2n$
6.5	`B2DOD`	"	$\frac{3}{4}m^2n$	0
6.3, 6.4	`B2DMD`	"	"	$\frac{1}{4}m^2n$
6.6, 6.7	`B2DMDMD`	"	"	"
7.5, 7.6	`B1DMDFB`	$mn^2 - \frac{2}{3}m^3$	$\frac{3}{2}mn^2 - \frac{5}{6}m^3$	$\frac{1}{2}mn^2 - \frac{1}{6}m^3$
7.5, 7.6	`B2DMDFB`	"	$\frac{5}{4}mn^2 - \frac{3}{4}m^3$	$\frac{1}{4}mn^2 - \frac{1}{12}m^3$
7.5, 7.6	`B2DMDMDFB`	"	"	"
8.4, 8.5	`Mal1DMD`	$\frac{1}{3}m^3$	$\frac{1}{2}m^3$	$\frac{1}{6}m^3$

Tabelle 19b

Ordnet man die oben angeführten Algorithmen bzgl. $\geq_K$ an, so ergibt sich für $n \times m$-

Matrizen mit $n > m$ das folgende Diagramm:

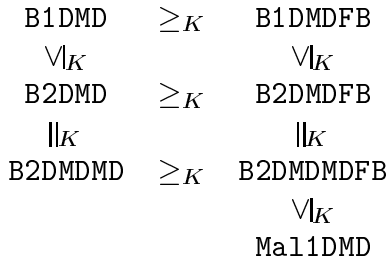


Diagramm 19b

In Bezug auf die Komplexität ist für $n \times m$-Matrizen mit $n > m$ das Malashonok-1-Schritt-Verfahren offensichtlich am günstigsten. Dies erscheint plausibel, da hier im Gegensatz zu den anderen Algorithmen die überflüssigen unteren Zeilen keinerlei Berechnungen unterworfen werden, es sei denn, es muß zwecks Zeilentausch vereinzelt ein Koeffizient berechnet werden. Am Ende kann man die überflüssigen Zeilen einfach gleich 0 setzen. Bei den übrigen Verfahren werden stets alle Zeilen neu berechnet, d.h. es wird unnötig viel Aufwand betrieben, um letztlich für die unteren Zeilen das Endergebnis 0 zu erhalten.

Hinsichtlich der Komplexität folgen auf das Malashonok-1-Schritt-Verfahren die beiden 2-Schritt-Forward-Backup-Verfahren, dann die beiden Bareiss-2-Schritt-Verfahren und das 1-Schritt-Forward-Backup-Verfahren. (Letzteres ist nicht direkt vergleichbar mit den Bareiss-2-Schritt-Verfahren.) Am schlechtesten schneidet wieder das Bareiss-1-Schritt-Verfahren ab.

Für quadratische $n \times n$-Matrizen vereinfacht sich Tabelle 19a, wie unten angegeben. (Tabelle 19b gilt übrigens auch für den Fall $n = m$ und ist dann identisch mit Tabelle 19a.)

Diagonalisierung von n × n-Matrizen				
Quellen	Algorithmus	Additionen	Multiplikationen	Divisionen
5.8	`B1DOD`	$\frac{1}{2}n^3$	n^3	0
5.6, 5.7	`B1DMD`	"	"	$\frac{1}{2}n^3$
6.5	`B2DOD`	"	$\frac{3}{4}n^3$	0
6.3, 6.4	`B2DMD`	"	"	$\frac{1}{4}n^3$
6.6, 6.7	`B2DMDMD`	"	"	"
7.5, 7.6	`B1DMDFB`	$\frac{1}{3}n^3$	$\frac{2}{3}n^3$	$\frac{1}{3}n^3$
7.5, 7.6	`B2DMDFB`	"	$\frac{1}{2}n^3$	$\frac{1}{6}n^3$
7.5, 7.6	`B2DMDMDFB`	"	"	"
8.4, 8.5	`Mal1DMD`	"	"	"
10.2, 10.3 10.6, 10.7	`MalDichHR` `MalDichVR`	$\frac{2}{21}n^3$	$\frac{2}{21}n^3$	$\frac{1}{2}pn^2 - \frac{3}{2}n^2$
10.4, 10.5 10.8, 10.9	`MalDichHM` `MalDichVM`	$O(n^{2+\beta})$	$O(n^{2+\beta})$	"

Tabelle 19c

Für den Fall quadratischer Matrizen ergibt sich somit das Diagramm

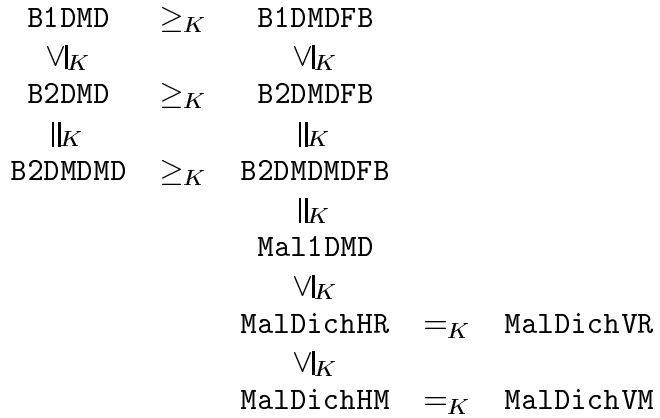


Diagramm 19c

In Bezug auf die Komplexität scheinen für quadratische Matrizen die Dichotomie-Verfahren am besten geeignet zu sein, gefolgt von den 2-Schritt-Forward-Backup-Verfahren und dem gleich guten Malashonok-1-Schritt-Verfahren. Am schlechtesten schneidet das Bareiss-1-Schritt-Verfahren ab, dazwischen liegen die beiden Bareiss-2-Schritt-Verfahren und das 1-Schritt-Forward-Backup-Verfahren. (Die Algorithmen `B1DMDFB` und `B2DMD` bzw. `B2DMDMD` sind bzgl. $\geq_K$ nicht vergleichbar.)

Zur Optimierung der Diagonalisierungsverfahren ist Bemerkung 18.1 (Optimierung der Trigonalisierungsalgorithmen) gleichermaßen gültig. Selbstverständlich kann man anhand einiger ausgewählter Beispielklassen den Vergleich der Verfahren nicht vollständig ausschöpfen, aber im wesentlichen findet die theoretisch zu erwartende Rangfolge der Algorithmen experimentelle Bestätigung, vgl. dazu die Tabellen und Diagramme 19.1a–c zu den Komplexitäten und die Zusammenfassung 18.2 zur Trigonalisierung.

19.1 Zusammenfassung der Testergebnisse zur Diagonalisierung

a) (*Bareiss- und Malashonok-Verfahren*) 2-Schritt-Verfahren sind in der Regel günstiger als 1-Schritt-Verfahren. Diagonalisiert man eine $n \times m$-Matrix mit $n > m$, so scheinen dafür die Bareiss-Verfahren i.a. weniger gut geeignet als das Malashonok-1-Schritt-Verfahren, am besten schneiden die Forward-Backup-Verfahren ab, insbesondere der Algorithmus `B2DMDMDFB`.
Für quadratische Matrizen mit höchstens 6 Zeilen erweist sich das Verfahren `Mal1DMD` als das beste, sogar besser als die Trigonalisierung, ansonsten für größere quadratische Matrizen das Verfahren `B2DMDMDFB`.
Das Malashonok-1-Schritt-Verfahren scheint gut geeignet für Matrizen, deren Rang kleiner als die Zeilenanzahl ist, da die überflüssigen Zeilen keinen Berechnungen unterliegen, außer es muß vereinzelt zum Zeilentausch ein Koeffizient berechnet werden. Ist von vornherein bekannt, welche Zeilen überflüssig sind, empfiehlt es sich allerdings, diese wegzulassen und ggf. ein besseres Verfahren zu wählen.

b) (*Dichotomie-Verfahren*) Die Dichotomie-Verhalten können die Bestzeiten der anderen Verfahren kaum unterbieten, wenn man sie auf Polynommatrizen anwendet. Bei solchen Matrizen kann man nämlich aufgrund der aufwendigen Polynommultiplikationen und -divisionen nur relativ kleine Matrizen betrachten, so daß sich der Vorteil des Dichotomieverfahrens, insbesondere einer günstigen Matrizenmultiplikation, noch nicht so recht bemerkbar machen kann. Betrachtet man dagegen große Matrizen mit ganzzahligen Koeffizienten, so sind die vier Dichotomie-Verfahren tatsächlich, wie von den Komplexitäten her zu erwarten, um ein Vielfaches schneller als alle anderen Bareiss- und Malashonok-Verfahren. Dabei ist es egal, ob man die Halbierungs- oder die Verdopplungsmethode wählt; die Verfahren mit Matrizenmultiplikation schneiden besser ab als die Verfahren, die zeilenweise vorgehen.

c) (*Sasaki-Murao-Verfahren*) Unter den modifizierten Verfahren scheint das Forward-Backup-Verfahren `B2DMDMDSMFB` am günstigsten für quadratische Matrizen; für nichtquadratische Matrizen ist es besser, auch die Backup-Prozedur zu modifizieren und den Algorithmus `B2DMDMDFBSM` zu verwenden.

d) (*Testsieger*) Für $n \times m$-Matrizen, die eher klein sind, also höchstens 5 oder 6 Zeilen enthalten, oder deren Einträge univariate Polynome oder nur Zahlen sind, laufen die modifizierten Verfahren — sofern sie überhaupt definiert sind — langsamer ab als die herkömmlichen. In diesem Fall ist der 2-Schritt-Forward-Backup-Algorithmus `B2DMDMDFB` empfehlenswert oder der Algorithmus `Mal1DMD`, vgl. a). Ansonsten kann man ab einer Größe von etwa 6 oder 7 Zeilen für quadratische Matrizen mit dem Algorithmus `B2DMDMDSMFB` (mit modifizierter Forward- und herkömmlicher Backupprozedur) sowie für nichtquadratische Matrizen mit dem Algorithmus `B2DMDMDFBSM` (mit modifizierter Forward- und modifizierter Backup-Prozedur) meist sehr viel Zeit sparen, wenn die Rücksubstitution einfach zu bewerkstelligen ist.

e) (*Komplexitäten*) Im wesentlichen bestätigen die Testreihen die aufgrund der Komplexitäten zu erwartende Rangfolge der Verfahren, s. die Tabellen und Diagramme 19.a–c.

Erst durch Lesen lernt man, wieviel man ungelesen lassen kann.
(Raabe, *Gedanken und Einfälle*)

Bekanntlich kann man lineare Abbildungen zwischen endlich-dimensionalen Vektorräumen vollständig durch Matrizen beschreiben, wobei letztere in der Praxis sicherlich die entscheidende Rolle spielen. Auch zahlreiche andere mathematische Probleme lassen sich letztendlich darauf reduzieren, bestimmte Matrizen zu betrachten.

Die Trigonalisierungs- und Diagonalisierungsverfahren aus den ersten drei Kapiteln erlauben es, den Rang, die Determinante, Adjunkte und Inverse einer Matrix zu bestimmen. Weitere wichtige Anwendungen aus der linearen Algebra bestehen darin, homogene und inhomogene lineare Gleichungssysteme zu lösen.

Fast alle der eben genannten Anwendungen beruhen darauf, daß die Verfahren mit Division Matrizen erzeugen, deren Einträge von der Form $\alpha_{ij}^{(k)} = [1 \dots k\ i][j_1 \dots j_k\ j](A)$ oder $\delta_{ij}^{(k)} = [1 \dots k][j_1 \dots \widehat{j_i}\ j \dots j_k](A)$ sind, daß also insbesondere ganz bestimmte Minoren berechnet werden. Es sei in diesem Zusammenhang daran erinnert, daß für eine $n \times m$-Matrix A aus $\mathrm{Mat}_{n\times m}(R)$, bei der Zeilentausch nicht nötig ist, die Trigonalisierung die Zeilenstufenmatrix

$$\begin{pmatrix} \left(\alpha_{ij}^{(i-1)}\right)_{\substack{i=1\dots r\\ j=1\dots m}} \\ \\ \left(0\right)_{\substack{i=r+1\dots n\\ j=1\dots m}} \end{pmatrix}$$

herstellt; die Diagonalisierung führt zur Matrix

$$\begin{pmatrix} \left(\delta_{ij}^{(r)}\right)_{\substack{i=1\dots r\\ j=1\dots m}} \\ \left(0\right)_{\substack{i=r+1\dots n\\ j=1\dots m}} \end{pmatrix}.$$

Hierbei bezeichne $r := \operatorname{rang} A$ den (beliebigen) Rang von A. Ist im Laufe eines Verfahrens Zeilentausch nötig, so ändern sich entsprechend die oben angegebenen Klammerausdrücke mit den Zeilenindizes. Der Vorzeichenwechsel der entsprechenden Determinanten ist in weiser Voraussicht bei den Zeilentauschalgorithmen `Pivot`, `C0Pivot` und `Mal1Pivot` berücksichtigt worden, indem eine der beiden getauschten Zeilen mit -1 multipliziert wird.

Eine ausführliche Beschreibung der Standardanwendungen aus der linearen Algebra ist in der Originalfassung nachzulesen. Ist A beispielsweise eine quadratische Matrix, so läßt sich in der dazu äquivalenten Zeilenstufen- bzw. Diagonalmatrix rechts unten an der Position (nn) die Determinante $|A|$ ablesen. Wie man die Inverse einer Matrix bestimmt bzw. wie man lineare Gleichungssysteme löst, dürfte ebenfalls aus der linearen Algebra bekannt sein.

Handelt es sich bei A um eine beliebige nichtsinguläre $n \times n$-Matrix und bei b um einen beliebigen Spaltenvektor, so enthält die zur erweiterten Koeffizientenmatrix $(A \mid b)$ äquivalente Diagonalmatrix $(\delta_{ij}^{(n)})$ genau die $n+1$ Determinanten n-ter Ordnung, die man laut *Cramerscher Regel* berechnen muß. Gewöhnlich bedient man sich dieser Regel eher zu theoretischen Untersuchungen denn zur tatsächlichen Berechnung des Lösungsvektors, da es vor allem für große $n \in \mathbb{N}$ unpraktikabel erscheint, $(n+1)$ Determinanten der Ordnung n zu berechnen. Die hier vorgestellten Diagonalisierungsverfahren ermöglichen es jedoch, alle benötigten Determinanten n-ter Ordnung geschickt, d.h. unter Verwendung gemeinsamer Zwischenergebnisse, zu berechnen. Die im Laufe eines solchen Verfahrens erzeugte Matrizenfolge enthält nämlich die besagten Zwischenergebnisse der Form $\alpha_{ij}^{(k)}$ oder $\delta_{ij}^{(k)}$. Dies macht man sich auch im folgenden Kapitel zunutze, wo es darum geht, alle maximalen Minoren einer Matrix oder sogar alle Minoren einer beliebigen Ordnung zu bestimmen.

A mathematician is a machine for turning coffee into theorems.

(Paul Erdös)

Über die klassischen Problemstellungen der linearen Algebra hinaus läßt sich die Frage stellen, wie man möglichst günstig alle maximalen Minoren einer Matrix oder sogar alle Minoren einer beliebigen Ordnung berechnen kann. Inwieweit sind dazu die im Rahmen dieser Diplomarbeit bereitgestellten Trigonalisierungs- und Diagonalisierungsverfahren von Nutzen? Für die folgenden Überlegungen sei A zunächst eine $n \times m$-Matrix über einem Integritätsring R mit der Eigenschaft

(V1) $n \leq m$.

In diesem Falle beträgt die Anzahl aller maximalen Minoren genau $\binom{m}{n}$, und die Anzahl aller Minoren k-ter Ordnung, $k \in \{1, \ldots, n\}$, beläuft sich auf $\binom{n}{k} \cdot \binom{m}{k}$. Für eine 5×10-Matrix ergeben sich damit immerhin $\binom{10}{5} = 252$ maximale Minoren und beispielsweise $\binom{5}{3} \cdot \binom{10}{3} = 1200$ Minoren dritter Ordnung. Eine 5×15-Matrix besitzt 2003 maximale Minoren, eine 10×20-Matrix 184.756 und eine 50×60-Matrix 75.394.027.566 maximale Minoren, um nur einige Beispiele zu nennen.

Bevor man sich in den kommenden Abschnitten der tatsächlichen Berechnung all dieser (unter Umständen sehr zahlreicher) Minoren zuwendet, sollte man sich zunächst über einige technische Probleme Gedanken machen, beispielsweise über die systematische Verwaltung all dieser Minoren. Da die maximalen Minoren von A als $[1 \ldots n][l_1 \ldots l_n](A)$ mit beliebigen Spaltenindizes $l_1, \ldots, l_n \in \{1, \ldots, m\}$, $l_1 < \ldots < l_n$, geschrieben werden können, liegt es nahe, diese Minoren lexikographisch nach dem zweiten Klammerausdruck zu ordnen. Betrachtet man beliebige Minoren k-ter Ordnung, so sind diese durch die Notation $[i_1 \ldots i_k][l_1 \ldots l_k](A)$ gekennzeichnet. Es sei daran erinnert, daß für einen Minor für die Zeilenindizes $i_1, \ldots, i_k \in \{1, \ldots, n\}$ definitionsgemäß $i_1 < \ldots < i_k$ und für die Spaltenindizes $l_1, \ldots, l_k \in \{1, \ldots, m\}$ definitionsgemäß $l_1 < \ldots < l_k$ gilt. Diese beiden Bedingungen werden im folgenden nicht mehr eigens erwähnt werden, wenn von Minoren die Rede ist. Auch die Minoren beliebiger Ordnung kann man lexikographisch ordnen.

20 Vorbereitungen

20.1 Schreibweisen

Seien $n, m, k \in \mathbb{N}_+$ beliebig mit $1 \leq k \leq n \leq m$.

a) Man bezeichnet mit

$$L_k^m := \{[1\ldots k],\ [1\ldots k-1,k+1],\ \ldots\ ,\ [m-k+1\ldots m]\}$$

die lexikographisch geordnete Menge aller Klammerausdrücke

$$[l_1\ldots l_k] \quad \text{mit} \quad l_1,\ldots l_k \in \{1,\ldots,m\},\ l_1 < \ldots < l_k,$$

und mit

$$L_k^{n,m} := \{[1\ldots k][1\ldots k],\ [1\ldots k][1\ldots k-1,k+1],\ \ldots\ ,\ [n-k+1\ldots n][m-k+1\ldots m]\}$$

die lexikographisch geordnete Menge aller Klammerausdrücke

$$[i_1\ldots i_k][l_1\ldots l_k] \quad \text{mit} \begin{cases} i_1,\ldots i_k \in \{1,\ldots,n\},\ i_1 < \ldots < i_k \\ l_1,\ldots l_k \in \{1,\ldots,m\},\ l_1 < \ldots < l_k. \end{cases}$$

b) Ferner verwendet man die Notationen

$$\operatorname{pos}^m_{[l_1\ldots l_k]} \qquad \text{bzw.} \qquad \operatorname{pos}^{n,m}_{[i_1\ldots i_k][l_1\ldots l_k]}$$

für die Position des Klammerausdrucks $[l_1\ldots l_k]$ in der geordneten Menge L_k^m bzw. für die Position von $[i_1\ldots i_k][l_1\ldots l_k]$ in der geordneten Menge $L_k^{n,m}$.

c) Sei $[l_1\ldots l_k]$ beliebig mit paarweise verschiedenen $l_1,\ldots,l_k \in \mathbb{N}_+$, ferner bezeichne $\pi : \{1,\ldots,k\} \to \{l_1,\ldots,l_k\}$ die eindeutig bestimmte Bijektion mit $\pi(1) < \ldots < \pi(k)$ und $\sigma : \{1,\ldots,k\} \to \{1,\ldots,k\}$ die eindeutig bestimmte Permutation, mit der man die Kompositionsabbildung

$$\begin{aligned} \pi \circ \sigma : \{1,\ldots,k\} &\to \{l_1,\ldots,l_k\} \\ i &\mapsto l_i \end{aligned}$$

erhält. Man nennt

$$\operatorname{sign}[l_1\ldots l_k] := \operatorname{sign}\sigma$$

das *Signum* von $[l_1\ldots l_k]$.
Ist A eine $n\times m$-Matrix mit $n \le m$ sowie $\operatorname{rang} A \ge k$ und sind außerdem $i \in \{1,\ldots,k\}$ und $j \in \{1,\ldots,m\} \setminus \{j_1, j_2, \ldots, j_{i-1}, j_{i+1}, \ldots j_k\}$ beliebig, so bezeichnet insbesondere

$$\operatorname{sign}\delta^{(k)}_{ij} := \operatorname{sign}[j_1\ldots\widehat{j_i}\ j\ldots j_k]$$

das Signum von $\delta^{(k)}_{ij} = [1\ldots k][j_1\ldots\widehat{j_i}\ j\ldots j_n](A)$.

20.2 Lemma (Positionsbestimmung)
Seien $n,m,k \in \mathbb{N}_+$ beliebig mit $1 \le k \le n \le m$.

a) Der Klammerausdruck $[l_1\ldots l_k] \in L_k^m$ befindet sich in der lexikographisch geordneten Menge L_k^m an der Position

$$\operatorname{pos}^m_{[l_1\ldots l_k]} = \sum_{\mu=1}^{l_1-1}\binom{m-\mu}{k-1} + \sum_{\mu=l_1+1}^{l_2-1}\binom{m-\mu}{k-2} + \ldots + \sum_{\mu=l_{k-1}+1}^{l_k-1}\binom{m-\mu}{0} \qquad +1 =$$

$$= 1 + \sum_{\lambda=1}^{k}\ \sum_{\mu=l_{\lambda-1}+1}^{l_\lambda-1}\binom{m-\mu}{k-\lambda},$$

wobei $\binom{m-\mu}{0} = 1$ *und* $l_0 = 0$ *gesetzt wird.*

b) Der Klammerausdruck $[i_1 \dots i_k][l_1 \dots l_k] \in L_k^{n,m}$ *steht in der Menge* $L_k^{n,m}$ *an der Position*

$$\mathrm{pos}^{n,m}_{[i_1\dots i_k][l_1\dots l_k]} = \left(\mathrm{pos}^{n}_{[i_1\dots i_k]} - 1\right) \cdot \binom{m}{k} + \mathrm{pos}^{m}_{[l_1\dots l_k]}.$$

Beweis:

a) Seien $l_1, \dots, l_k \in \{1, \dots, m\}$ mit $l_1 < \dots < l_k$ beliebig, aber fest. Gesucht ist zunächst die Anzahl p aller Vorgänger von $l := [l_1 \dots l_k]$ in L_k^m, d.h. aller Klammerausdrücke $\widetilde{l} := [\widetilde{l}_1 \dots \widetilde{l}_k]$ mit $\widetilde{l}_1 < \dots < \widetilde{l}_k$, wobei ferner $\widetilde{l}_\lambda \le l_\lambda$ für $\lambda = 1, \dots, k-1$ und $\widetilde{l}_k < l_k$ gilt.
Um dieses kombinatorische Problem zu lösen, geht man schrittweise vor. Die Anzahl aller Vorgänger $\widetilde{l}$ von l in L_k^m, für die $\widetilde{l}_1 < l_1$ gilt, beträgt genau $\sum_{\mu=1}^{l_1-1} \binom{m-\mu}{k-1}$.
Analog beläuft sich die Anzahl aller Vorgänger $\widetilde{l}$ von l in L_k^m, für die $\widetilde{l}_1 = l_1$ und $\widetilde{l}_2 < l_2$ gilt, genau auf $\sum_{\mu=l_1+1}^{l_2-1} \binom{m-\mu}{k-2}$ usw.
Ingesamt folgt also

$$p = \sum_{\mu=1}^{l_1-1} \binom{m-\mu}{k-1} + \sum_{\mu=l_1+1}^{l_2-1} \binom{m-\mu}{k-2} + \dots + \sum_{\mu=l_{k-1}+1}^{l_k-1} \binom{m-\mu}{0}.$$

Um die tatsächliche Position von l zu erhalten, muß man zu der Anzahl p der Vorgänger noch 1 addieren. Somit folgt die Behauptung.

b) Sei $k \in \{1, \dots, n\}$ beliebig, ferner sei $il := [i_1 \dots i_k][l_1 \dots l_k] \in L_k^{n,m}$ beliebig. Mit p_1 werde die Anzahl aller Vorgänger $\widetilde{il} := [\widetilde{i}_1 \dots \widetilde{i}_k][\widetilde{l}_1 \dots \widetilde{l}_k]$ von il in $L_k^{n,m}$ bezeichnet, deren erster Klammerausdruck $[\widetilde{i}_1 \dots \widetilde{i}_k]$ Vorgänger von $[i_1 \dots i_k]$ ist und für die der zweite Klammerausdruck $[\widetilde{l}_1 \dots \widetilde{l}_k]$ beliebig ist. Es folgt somit

$$p_1 := \left(\mathrm{pos}^{n}_{[i_1\dots i_k]} - 1\right) \cdot \binom{m}{k}.$$

Nach a) existieren ferner genau

$$p_2 := \mathrm{pos}^{m}_{[l_1\dots l_k]}$$

Klammerausdrücke $\widetilde{il}$, die Vorgänger von il oder gleich il sind und für die außerdem gilt $[\widetilde{i}_1 \dots \widetilde{i}_k] = [i_1 \dots i_k]$. Insgesamt folgt daher, daß sich il an der Position $p_1 + p_2$ befindet, also die Behauptung. □

Als Anwendung dieses Lemmas folgt sofort

20.3 Satz (Algorithmus PosMinor)
Seien $n, m, k \in \mathbb{N}_+$ *beliebig mit* $1 \le k \le n \le m$. *Ferner sei ein beliebiger Klammerausdruck* $[i_1 \dots i_k][l_1 \dots l_k] \in L_k^{n,m}$ *gegeben. Betrachte die folgenden Instruktionen:*

(1) Falls $k < n$ *gilt, berechne* $p_1 := \sum_{\lambda=1}^{k} \sum_{\mu=i_{\lambda-1}+1}^{i_\lambda-1} \binom{n-\mu}{k-\lambda}$. *Andernfalls setze* $p_1 := 0$.

(2) Berechne $p_2 := 1 + \sum_{\lambda=1}^{k} \sum_{\mu=l_{\lambda-1}+1}^{l_\lambda-1} \binom{m-\mu}{k-\lambda}$.

(3) Berechne $p := p_1 \cdot \binom{m}{k} + p_2$ *und gib als Ergebnis* p *aus.*

Dies ist ein Algorithmus namens `PosMinor`, *der als Ergebnis die Position* $\mathrm{pos}^{n,m}_{[i_1\dots i_k][l_1\dots l_k]}$ *des Klammerausdrucks* $[i_1 \dots i_k][l_1 \dots l_k]$ *in der lexikographisch geordneten Menge* $L_k^{n,m}$

ausgibt. Gilt insbesondere $n = k$, so erhält man als Ergebnis die Position $\mathrm{pos}^m_{[l_1 \ldots l_k]}$ *von* $[l_1 \ldots l_k]$ *in der geordneten Menge* L^m_k.

Beweis:
Die angegebenen Schritte werden offensichtlich nur endlich oft wiederholt. Daß das Ergebnis wie behauptet die Position von $[i_1 \ldots i_k][l_1 \ldots l_k] \in L^{n,m}_k$ bzw. von $[l_1 \ldots l_k] \in L^m_k$ angibt, folgt unmittelbar aus dem vorangegangenen Lemma 20.2. □

Nach dem derzeitigen Kenntnisstand kann man die Position eines beliebigen Klammerausdrucks der Form $[i_1 \ldots i_k][l_1 \ldots l_k]$ in der lexikographisch geordneten Menge $L^{n,m}_k$ aller Klammerausdrücke berechnen. Wie aber erhält man diese geordnete Menge aller Klammerausdrücke? Offensichtlich ergibt sich $L^{n,m}_k$ sofort aus dem kartesischen Produkt $L^n_k \times L^m_k$, indem man jedes Element $([i_1 \ldots i_k], [l_1 \ldots l_k])$ dieser Produktmenge zusammensetzt zu $[i_1 \ldots i_k][l_1 \ldots l_k]$. Es genügt somit, die Mengen L^n_k bzw. L^m_k zu bestimmen.

20.4 Satz (Algorithmen Subsets, AllSubsets)
Seien $n \in \mathbb{N}_+$ und $k \in \mathbb{N}$ beliebig mit $0 \leq k \leq n$.

a) (Algorithmus `Subsets`*) Es bezeichne M eine Menge von n-Tupeln. Betrachte die folgenden Instruktionen:*
(1) Falls $k = 0$ ist, setze $M := \{(0 \ldots 0)\}$. Gib als Ergebnis M aus.
(2) Falls $k = n$ ist, setze $M := \{(1 \ldots 1)\}$. Gib als Ergebnis M aus.
(3) Falls $0 < k < n$ gilt, führe `Subsets`$(n-1, k-1)$ *durch und nenne das Ergebnis M_1. Ergänze jedes Tupel aus M_1 an erster Position um die Komponente 1. Führe außerdem* `Subsets`$(n-1, k)$ *durch und nenne das Ergebnis M_0. Ergänze jedes Tupel aus M_0 an erster Position um die Komponente 0. Setze $M := M_1 \cup M_0$ und gib das Ergebnis M aus.*
Dies ist ein rekursiv definierter Algorithmus mit der Bezeichnung `Subsets`(n, k)*, der als Ergebnis die lexikographisch geordnete Menge $\{(1 \ldots 1\,0 \ldots 0), \ldots, (0 \ldots 0\,1 \ldots 1)\}$ aller n-Tupel ausgibt, in denen genau k-mal die Komponente 1 und genau $(n-k)$-mal die Komponente 0 vorkommt.*

b) (Algorithmus `AllSubsets`*) Sei die Menge $B = \{b_1, \ldots, b_n\} = \{1, \ldots, n\}$ geordnet, so daß $b_1 < \ldots < b_n$ gilt. Betrachte die folgenden Instruktionen:*
(1) Führe den Algorithmus `Subsets`(n, k) *durch und nenne sein Ergebnis M.*
(2) Ersetze jedes Tupel $(m_1, \ldots, m_n)$ aus der Menge M durch den Klammerausdruck $[b_\lambda : \lambda = 1, \ldots, n$ und $m_\lambda = 1]$. Gib als Ergebnis M aus.
Dies beschreibt einen Algorithmus namens `AllSubsets`(B, k)*, dessen Ergebnis die lexikographisch geordnete Menge L^n_k aller Klammerausdrücke $[i_1 \ldots i_k]$ mit $i_1, \ldots, i_k \in \{1, \ldots, n\}$ und $i_1 < \ldots < i_k$ ist.*

Beweis:
a) Da im rekursiv durchgeführten Schritt (3) jedesmal mindestens eine der beiden Variablen n und k um 1 verkleinert wird, bis schließlich $k = 0$ oder $k = n$ gilt, können die angegebenen Schritte nur endlich oft wiederholt werden. Nach Konstruktion wird offensichtlich sichergestellt, daß die 1 genau k-mal und die 0 genau $(n-k)$-mal in den n-Tupeln auftritt und daß ferner M lexikographisch geordnet ist.

b) Der in Schritt (1) eingesetzte Algorithmus `Subsets` besteht nur aus endlich vielen Schritten, sein Ergebnis ist die endliche Menge M. Offensichtlich besteht dann auch `AllSubsets` nur aus endlich vielen Schritten.
Interpretiert man die in Schritt (1) gewonnenen Tupel aus M dahingehend, daß ihre Komponenten Indikatoren dafür sind, welche Elemente von B für einen Klammerausdruck ausgewählt werden, so erhält man in Schritt (2) die Menge aller Klam-

merausdrücke $[i_1 \dots i_k]$ mit $i_1, \dots, i_k \in \{b_1, \dots, b_n\}$ und mit $i_1 < \dots < i_k$. Diese Menge ist überdies lexikographisch geordnet, also gleich L_k^n, weil die Menge M aus Schritt (1) schon lexikographisch geordnet ist und weil $b_1 < \dots < b_n$ gilt. □

20.5 Satz (Algorithmus Signum)
Sei $k \in \mathbb{N}_+$, und sei $l := [l_1 \dots l_k]$ mit paarweise verschiedenen $l_1, \dots, l_k \in \mathbb{N}_+$. Betrachte die folgenden Instruktionen:
(1) Setze $s := 1$.
(2) Falls $l \neq [\,]$ ist, suche die Position μ, an der das Minimum von l steht.
(3) Ersetze s durch $(-1)^{\mu-1} \cdot s$ und streiche aus l die Komponente an Position μ. Falls nun $l = [\,]$ gilt, gib das Ergebnis s aus, andernfalls fahre fort bei Schritt (2).
Dies ist ein Algorithmus mit der Bezeichnung `Signum`$([l_1 \dots l_k])$*, dessen Ergebnis das Signum von $[l_1 \dots l_k]$ ist.*
Beweis:
Da $l = [l_1 \dots l_k]$ nur aus endlich vielen Komponenten besteht, von denen in Schritt (3) jeweils die kleinste entfernt wird, können die angegebenen Schritte nur endlich oft wiederholt werden.

Die in Schritt (2) ermittelte Position μ des Minimums von l ist eindeutig bestimmt, denn nach Voraussetzung sind $l_1, \dots, l_k$ paarweise verschieden. Möchte man in l die Komponente $l_\mu = \min l$ von Position μ nach vorne an Position 1 tauschen, so kann man dies offenbar mit $(\mu - 1)$ Transpositionen bewerkstelligen. Diese Permutation trägt also das Signum $(-1)^{\mu-1}$. Entfernt man $\min l$ aus l und wiederholt man die eben beschriebene Permutation, d.h. wiederholt man die Schritte (2) und (3) solange, bis $l = [\,]$ ist, so erhält man insgesamt das Signum s der Permutation, die $l = [l_1 \dots l_k]$ in das aufsteigend geordnete $[\min l \dots \max l]$ verwandelt. Damit folgt $s = \operatorname{sign}[l_1 \dots l_k]$ wie behauptet. □

20.6 Bemerkungen zur Implementierung (PosMinor, Subsets, AllSubsets, Signum)

a) (`PosMinor`) Beim Algorithmus `PosMinor` aus Satz 20.3 kann man wahlweise zwei Argumente $[l_1 \dots l_k]$ und m eingeben, um $\operatorname{pos}^m_{[l_1 \dots l_k]}$ zu berechnen, oder vier Argumente in der Reihenfolge $[i_1 \dots i_k]$, $[l_1 \dots l_k]$, n und m, um $\operatorname{pos}^{n,m}_{[i_1 \dots i_k][l_1 \dots l_k]}$ zu erhalten.

b) (`Subsets`) Bei dem Algorithmus `Subsets` aus Satz 20.4a) läßt sich Schritt (3) rekursiv ausführen, indem man wie beschrieben `Subsets`$(n-1, k-1)$ bzw. `Subsets`$(n-1, k)$ aufruft.

c) (`AllSubsets`) Mit dem Algorithmus `AllSubsets`(B, k) aus Satz 20.4b) läßt sich nicht nur die Menge L_k^n, sondern sogar viel allgemeiner die Menge aller k-elementigen Teilmengen einer n-elementigen Menge B konstruieren. Im ersten Fall wählt man $B = \{1, \dots, n\}$, im zweiten Fall kann B eine beliebige n-elementige Liste sein.
Es ist außerdem für die Verwendung in späteren Algorithmen (z.B. `FindZeroMinors`) günstig, `AllSubsets`(B, k) auch für Argumente $k < 0$ zu definieren. Im Fall $k \leq 0$ wird als Ergebnis die Liste $[\,[\,]\,]$ ausgegeben.

d) (`Signum`) Der beschriebene Algorithmus `Signum`$([l_1 \dots l_k])$ stellt nur eine von vielen Möglichkeiten dar, wie man $\operatorname{sign}[l_1 \dots l_k]$ anhand bestimmter Transpositionen ermitteln kann. Die Permutation, die $l = [l_1 \dots l_k]$ auf $[\min l \dots \max l]$ abbildet, läßt sich nämlich auf vielfältige Art als Komposition von Transpositionen schreiben, die im Gegensatz zum Signum nicht eindeutig bestimmt sind.

20.7 Lemma (lineare Relationen)
Sei A eine beliebige $n \times m$-Matrix über einem Integritätsring R, und sei ferner $k \in \mathbb{N}_+$.

a) (Lineare Relationen) Sind $i_1, \ldots, i_k \in \{1, \ldots, n\}$ sowie $l_0, l_1, \ldots, l_k \in \{1, \ldots, m\}$ beliebige Zeilen- bzw. Spaltenindizes und ist $a_{i_\mu l_0} \neq 0$ für ein $i_\mu \in \{i_1, \ldots, i_k\}$, so gilt:

$$[i_1 \ldots i_k][l_1 \ldots l_k](A) = \frac{1}{a_{i_\mu l_0}} \cdot \sum_{\lambda=1}^{k} (-1)^{\lambda+1} \cdot a_{i_\mu l_\lambda} \cdot [i_1 \ldots i_k][l_0\ l_1 \ldots \widehat{l_\lambda} \ldots l_k](A) \quad \in R.$$

Mit anderen Worten, man erhält die Determinante $[i_1 \ldots i_k][l_1 \ldots l_k](A)$ aus solchen Determinanten in den Zeilen $i_1, \ldots, i_k$, die die Spalte l_0 und jeweils $(k-1)$ verschiedene Spalten aus $\{l_1, \ldots, l_k\}$ enthalten.

b) (Lineare Relationen) Sind $i_0, i_1, \ldots, i_k \in \{1, \ldots, n\}$ und $l_1, \ldots, l_k \in \{1, \ldots, m\}$ beliebige Zeilen- bzw. Spaltenindizes und ist $a_{i_0 l_\mu} \neq 0$ für ein $l_\mu \in \{l_1, \ldots, l_k\}$, so gilt:

$$[i_1 \ldots i_k][l_1 \ldots l_k](A) = \frac{1}{a_{i_0 l_\mu}} \cdot \sum_{\lambda=1}^{k} (-1)^{\lambda+1} \cdot a_{i_\lambda l_\mu} \cdot [i_0\ i_1 \ldots \widehat{i_\lambda} \ldots i_k][l_1 \ldots l_k](A) \quad \in R.$$

Mit anderen Worten, man erhält die Determinante $[i_1 \ldots i_k][l_1 \ldots l_k](A)$ aus solchen Determinanten in den Spalten $l_1, \ldots, l_k$, die die Zeile i_0 und jeweils $(k-1)$ verschiedene Zeilen aus $\{i_1, \ldots, i_k\}$ enthalten.

c) Ist A eine $n \times m$-Matrix mit $\operatorname{rang} A = n$ und mit den Zeilenstufen $j_1, j_2, \ldots, j_n$, wobei $a_{1j_1} \neq 0$ ist, so gilt in a) insbesondere mit $a_{i_\mu l_0} = a_{1j_1}$

$$\delta_{1j}^{(n)} = \frac{1}{a_{1j_1}} \cdot \Big(a_{ij} \cdot [1 \ldots n][j_1 \ldots j_n](A) + \sum_{\lambda=2}^{n} (-1)^{\lambda+1} \cdot a_{1j_\lambda} \cdot [1 \ldots n][j_1\ j\ j_2 \ldots \widehat{j_\lambda} \ldots j_n](A) \Big) =$$

$$= \frac{1}{a_{1j_1}} \cdot \Big(\delta^{(n)} \cdot a_{1j} - \sum_{\lambda=2}^{n} a_{1j_\lambda} \cdot \delta_{\lambda j}^{(n)} \Big).$$

Mit anderen Worten, in diesem Falle sind die lineare Relation aus a) und die Backup-Rekursionsformel aus Korollar 7.3 äquivalent.

Beweis:

a) Da die Determinante $[i_\mu\ i_1 \ldots i_k][l_0\ l_1 \ldots l_k](A)$ die Zeile i_μ doppelt enthält, gilt die folgende lineare Relation:

$$0 = [i_\mu\ i_1 \ldots i_k][l_0\ l_1 \ldots l_k](A) = \left| \begin{array}{c|ccc} a_{i_\mu l_0} & a_{i_\mu l_1} & \cdots & a_{i_\mu l_k} \\ \hline a_{i_1 l_0} & a_{i_1 l_1} & \cdots & a_{i_1 l_k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i_k l_0} & a_{i_k l_1} & \cdots & a_{i_k l_k} \end{array} \right| =$$

$$= a_{i_\mu l_0} \cdot [i_1 \ldots i_k][l_1 \ldots l_k](A) + \sum_{\lambda=1}^{k} (-1)^{\lambda+2} \cdot a_{i_\mu l_\lambda} \cdot [i_1 \ldots i_k][l_0\ l_1 \ldots \widehat{l_\lambda} \ldots l_k](A).$$

Letzteres ergibt sich, wenn man die Determinante nach der ersten Zeile entwickelt. Weil nach Voraussetzung $a_{i_\mu l_0} \neq 0$ ist, folgt daraus sofort wie behauptet

$$\frac{1}{a_{i_\mu l_0}} \cdot \sum_{\lambda=1}^{k} (-1)^{\lambda+1} \cdot a_{i_\mu l_\lambda} \cdot [i_1 \ldots i_k][l_0\ l_1 \ldots \widehat{l_\lambda} \ldots l_k](A) = [i_1 \ldots i_k][l_1 \ldots l_k](A) \quad \in R.$$

b) folgt analog zu a), wenn man die Determinante $[i_0\ i_1 \ldots i_k][l_\mu\ l_1 \ldots l_k](A) = 0$ nach der ersten Spalte entwickelt.

c) Nach a) gilt mit $[i_1 \ldots i_k] = [1 \ldots n]$, $[l_1 \ldots l_k] = [j\ j_2 \ldots j_n]$, $i_\mu = 1$ und $l_0 = j_1$

$$\begin{aligned}\delta_{1j}^{(n)} &= [1 \ldots n][j\ j_2 \ldots j_n](A) = \\ &= \frac{1}{a_{1j_1}} \cdot \sum_{\lambda=1}^{n} (-1)^{\lambda+1} \cdot a_{1l_\lambda} \cdot [1 \ldots n][j_1\ l_1 \ldots \widehat{l_\lambda} \ldots l_n](A).\end{aligned} \qquad (*)$$

Für $\lambda = 1$ erhält man den Summanden

$$(-1)^2 \cdot a_{1j} \cdot [1 \ldots n][j_1\ \widehat{j}\ j_2 \ldots j_n](A) = a_{1j} \cdot \delta^{(n)}.$$

Für $\lambda = 2, \ldots n$ ergibt sich mit

$$[1 \ldots n][j_1\ j\ j_2 \ldots \widehat{j_\lambda} \ldots j_n](A) = (-1)^{\lambda-2} \cdot [1 \ldots n][j_1\ j_2 \ldots \widehat{j_\lambda}\ j \ldots j_n](A) = (-1)^{\lambda-2} \cdot \delta_{\lambda j}^{(n)}$$

jeweils der Summand

$$(-1)^{\lambda+1} \cdot a_{1j_\lambda} \cdot [1 \ldots n][j_1\ j\ j_2 \ldots \widehat{j_\lambda} \ldots j_n](A) = -a_{1j_\lambda} \cdot \delta_{\lambda j}^{(n)}.$$

In $(*)$ eingesetzt, erhält man mit diesen Summanden aus der linearen Relation die Backup-Rekursionsformel aus Korollar 7.3 wie behauptet. □

21 Grundlagen zur Berechnung maximaler Minoren

Sei im folgenden A eine $n \times m$-Matrix über einem Integritätsring R mit der Eigenschaft

(V1) $n \leq m$.

Wie kann man alle $\binom{m}{n}$ maximalen Minoren dieser Matrix möglichst geschickt bestimmen, und wie verallgemeinert sich dies für beliebige $n \times m$-Matrizen? Die Minorenberechnung für den Fall $n > m$ läßt sich sehr leicht auf den Fall $n \leq m$ zurückführen, denn das Transponieren einer Matrix übt bekanntlich keinen Einfluß auf den Wert ihrer Determinante aus. Die Beantwortung der ersten Frage wird dagegen mehr Mühe kosten.

Die am wenigsten günstige — wenn auch korrekte — Methode, alle maximalen Minoren einer Matrix zu bestimmen, besteht wohl darin, diese einfach der Reihe nach zu berechnen, ohne auf gemeinsame Zwischenergebnisse zurückzugreifen. Besser ist es sicherlich, solche gemeinsamen Zwischenergebnisse festzuhalten, wobei das nächste Problem auftritt: dies kostet möglicherweise sehr viel Speicherplatz und macht eine gewisse Organisation erforderlich. Aus den ersten drei Kapiteln kennt man andererseits bereits diverse Verfahren, um Matrizenfolgen mit Einträgen der Form $\alpha_{ij}^{(k)}$ und $\delta_{ij}^{(k)}$ zu erzeugen. Diese Koeffizienten sind bis aufs Vorzeichen Minoren der Ordnung $k+1$ bzw. der Ordnung k und stellen somit nützliche Zwischenschritte auf dem Weg dahin dar, alle maximalen Minoren zu berechnen. Um aus den genannten Minoren wiederum andere zu berechnen, verfügt man bekanntlich über die folgenden Rekursionsformeln:

a) die Bareiss-Rekursion zur Trigonalisierung (1.4, 3.3)
b) die Bareiss-Rekursion zur Diagonalisierung (5.5)
c) die Backup-Rekursion (7.3)
d) die Malashonok-1-Schritt-Rekursion (8.1)
e) die Dichotomie-Rekursion (9.6)
f) die linearen Relationen (20.7)

Kann man mit Hilfe dieser Rekursionsformeln bzw. der zugehörigen Verfahren wirklich alle maximalen Minoren einer Matrix bestimmen? Für den Spezialfall einer $n \times (n+1)$-Matrix ist die Frage bereits beantwortet: durch Diagonalisieren erhält man daraus (bis auf Zeilentausch) die Matrix

$$\begin{pmatrix} \left(\delta_{ij}^{(r)}\right)_{\substack{i=1\ldots r\\ j=1\ldots m}} \\ \left(0\right)_{\substack{i=r+1\ldots n\\ j=1\ldots m}} \end{pmatrix},$$

wobei r den Rang der Matrix bezeichne. Ist $r < n$, so sind alle maximalen Minoren gleich 0. Andernfalls kann man sie bis aufs Vorzeichen sofort an der Diagonalmatrix ablesen.

Um auf den allgemeinen Fall beliebiger $n \times m$-Matrizen mit $n \leq m$ zurückzukommen, so sind die $\binom{m}{n}$ maximalen Minoren $[1 \ldots n][l_1 \ldots l_n](A)$ bestimmt durch die Auswahl der Spalten $l_1, \ldots, l_n \in \{1, \ldots, m\}$, wobei $l_1 < \ldots < l_n$ gilt. Offensichtlich erhält man die Gesamtheit dieser Minoren, indem der Klammerausdruck $[l_1 \ldots l_n]$ die lexikographisch geordnete Menge $L_n^m = \{[1 \ldots n], [1 \ldots n-1, n+1], \ldots, [m-n+1 \ldots m]\}$ durchläuft.

21.1 Definition
Ist A eine $n \times m$-Matrix mit $n \leq m$ und sind $l_1, \ldots, l_n \in \{1, \ldots, m\}$ beliebige Spaltenindizes, so schreibt man für die Determinante in den Zeilen $1, \ldots n$ und den Spalten $l_1, \ldots, l_n$ kurz

$$[l_1 \ldots l_n](A) := [1 \ldots n][l_1 \ldots l_n](A).$$

Da sich beim Vergleich der Diagonalisierungsverfahren in Kapitel IV das Forward-Backup-Verfahren als gut erwiesen hat, soll es im folgenden das Fundament der Minorenberechnung bilden. Für die nötigen Überlegungen erfülle die $n \times m$-Matrix A bis auf weiteres die Voraussetzungen

(V1) $3 \leq n \leq m$,
(V2) $\operatorname{rang} A = n$ und
(V3) Zeilentausch ist nicht nötig.

21.2 Bemerkungen (Voraussetzungen)

a) Die in (V1) getroffenen Vereinbarung $n \leq m$ stellt keine besondere Einschränkung dar, denn andernfalls betrachtet man eben die transponierte Matrix ${}^tA$. Auch die Voraussetzung $3 \leq n$ ist nicht weiter tragisch, weil im Fall $n = 1$ die $\binom{m}{n} = m$ maximalen Minoren der $1 \times m$-Matrix $A = (a_{11} \ldots a_{1m})$ trivialerweise die Koeffizienten $a_{11}, \ldots, a_{1m}$ sind. Für eine zweizeilige Matrix besteht keine sonderliche Notwendigkeit, ein günstiges Verfahren zur Minorenberechnung zu finden, denn die $\binom{m}{n} = \frac{m(m-1)}{2}$ maximalen Minoren der $2 \times m$-Matrix A kann man für alle $[l_1\ l_2] \in L_2^m$ noch einfach der Reihe nach berechnen:

$$[l_1\ l_2](A) = \begin{vmatrix} a_{1l_1} & a_{1l_2} \\ a_{2l_1} & a_{2l_2} \end{vmatrix} = a_{1l_1} \cdot a_{2l_2} - a_{2l_1} \cdot a_{1l_2}.$$

Dabei fallen insgesamt $m(m-1)$ Multiplikationen und $\frac{m(m-1)}{2}$ Additionen an. Abgesehen davon ergeben sich die Fälle $n = 1, 2$ ebensogut als Spezialfälle der folgenden Überlegungen.

b) Gilt (V2) nicht, d.h. ist $\operatorname{rang} A < n$, so nehmen sämtliche maximalen Minoren von A den Wert 0 an und weitere Überlegungen erübrigen sich.

c) Die Voraussetzung (V3) dient wie immer dazu, die folgenden Betrachtungen nicht dadurch unnötig zu komplizieren, daß man bei Zeilentausch statt der Koeffizienten

$\alpha_{ij}^{(k)} = [1 \dots k\ i][j_1 \dots j_k\ j](A)$ bzw. $\delta_{ij}^{(k)} = [1 \dots k][j_1 \dots \widehat{j_i}\ j \dots j_k](A)$ die Koeffizienten $\widetilde{\alpha}_{ij}^{(k)} = [i_1 \dots i_k\ i][j_1 \dots j_k\ j](A)$ bzw. $\widetilde{\delta}_{ij}^{(k)} = [i_1 \dots i_k][j_1 \dots \widehat{j_i}\ j \dots j_k](A)$ mit den eingetauschten Zeilen $i_1, \dots, i_k \in \{1, \dots, n\}$ erhält. Im Endeffekt spielt ein Zeilentausch keine Rolle, vergleiche die später folgende Bemerkung 21.8b).

Trigonalisiert man A mit einem Divisionsverfahren, so erzeugt man bekanntlich die Matrizenfolge $(B^{(k)})_{k=0\dots n-1}$, wobei

$$B^{(k)} = \begin{pmatrix} (\alpha_{ij}^{(i-1)})_{\substack{i=1\dots k \\ j=1\dots m}} \\ (\alpha_{ij}^{(k)})_{\substack{i=k+1\dots n \\ j=1\dots m}} \end{pmatrix}$$

gilt. Insbesondere erhält man für $k = 0$ die Matrix $B^{(0)} = A$ und für $k = n-1$ die Zeilenstufenmatrix

$$B^{(n-1)} = (\alpha_{ij}^{(i-1)})_{\substack{i=1\dots n \\ j=1\dots m}} =$$

$$= \begin{pmatrix} \alpha_{1j_1}^{(0)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{1m}^{(0)} \\ & & \alpha_{2j_2}^{(1)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{2m}^{(1)} \\ & & & \ddots & & & & & & & \vdots \\ & & & & \alpha_{n-2,j_{n-2}}^{(n-3)} & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{n-2,m}^{(n-3)} \\ & & \mathbf{0} & & & & \alpha_{n-1,j_{n-1}}^{(n-2)} & \cdots & \alpha_{n-1,j_n}^{(n-2)} & \cdots & \alpha_{n-1,m}^{(n-2)} \\ & & & & & & & & \alpha_{n,j_n}^{(n-1)} & \cdots & \alpha_{nm}^{(n-1)} \end{pmatrix}$$

Diese Matrix dient als Ausgangspunkt für die Backup-Prozedur, die gekennzeichnet ist durch die Matrizenfolge $(F^{(k)})_{k=n\dots 0}$, wobei

$$F^{(k)} = \begin{pmatrix} (\alpha_{ij}^{(i-1)})_{\substack{i=1\dots k \\ j=1\dots m}} \\ (\delta_{ij}^{(n)})_{\substack{i=k+1\dots n \\ j=1\dots m}} \end{pmatrix}$$

gilt. Insbesondere ergibt sich für $k = n$ die obige Zeilenstufenmatrix $F^{(n)} = B^{(n-1)}$ und für $k = 0$ die Diagonalmatrix

$$F^{(0)} = (\delta_{ij}^{(n)})_{\substack{i=1\dots n \\ j=1\dots m}} =$$

$$= \begin{pmatrix} \delta_{1j_1}^{(n)} & \cdots & * & 0 & * \cdots * & 0 & * \cdots * & 0 & \delta_{1,j_n+1}^{(n)} & \cdots & \delta_{1m}^{(n)} \\ & \ddots & \vdots & \vdots & \vdots\ \ \vdots & \vdots & \vdots\ \ \vdots & \vdots & \vdots & & \vdots \\ & & * & 0 & \vdots\ \ \vdots & \vdots & \vdots\ \ \vdots & \vdots & \vdots & \ddots & \vdots \\ & & & \delta_{n-2,j_{n-2}}^{(n)} & * \cdots * & 0 & \vdots\ \ \vdots & \vdots & \vdots & & \vdots \\ & & & & & \delta_{n-1,j_{n-1}}^{(n)} & * \cdots * & 0 & \delta_{n-1,j_n+1}^{(n)} & \cdots & \delta_{n-1,m}^{(n)} \\ & & \mathbf{0} & & & & & \delta_{n,j_n}^{(n)} & \delta_{n,j_n+1}^{(n)} & \cdots & \delta_{nm}^{(n)} \end{pmatrix}$$

Hierbei symbolisiert $*$ wie üblich Koeffizienten, die ungleich 0 sein können. Die Diagonalmatrix $F^{(0)}$ enthält bis aufs Vorzeichen bestimmte maximale Minoren, nämlich die Zeilenstufenkoeffizienten

$$\delta_{1j_1}^{(n)} = \delta_{2j_2}^{(n)} = \ldots = \delta_{nj_n}^{(n)} = \delta^{(n)} = [j_1 \ldots j_n](A) \neq 0$$

und die übrigen Determinanten

$$\delta_{ij}^{(n)} = [j_1 \ldots \widehat{j_i}\ j \ldots j_n](A),$$

bei denen jeweils $(n-1)$ paarweise verschiedene Spalten aus $\{j_1, \ldots, j_n\}$ festgehalten werden und der Spaltenindex j die Menge der Nichtzeilenstufenspalten, also die Menge $\{1, \ldots, m\} \setminus \{j_1, j_2, \ldots, j_n\}$ durchläuft. Da die Spaltenindizes in $[j_1 \ldots \widehat{j_i}\ j \ldots j_n]$ nicht unbedingt aufsteigend sortiert sind, kann der entsprechende maximale Minor von A in ebendiesen Spalten sich von $\delta_{ij}^{(n)}$ durch das Vorzeichen sign $\delta_{ij}^{(n)}$ unterscheiden.

21.3 Definitionen

Es bezeichne $B := B^{(n-1)}$ die obige Zeilenstufenmatrix.

a) Man sagt, die Spalte $j \in \{1, \ldots, m\}$ *liegt* in der Matrix B *auf der Höhe* h mit einem $h \in \{1, \ldots, n\}$, wenn die j-te Spalte von B die Gestalt

$$b^j = \begin{pmatrix} b_{1j} \\ \vdots \\ b_{hj} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

annimmt mit $b_{hj} \neq 0$. Ist die Spalte b^j gleich 0, so *liegt* sie *auf der Höhe* $h = 0$.

b) Mit $j_{n1}, \ldots, j_{n\rho} \in \{1, \ldots, m\}$ numeriert man alle Spalten von B, die auf der Höhe n liegen, also alle Spalten, deren letzter Koeffizient $b_{nj_\lambda} \neq 0$ ist, $\lambda = 1, \ldots, \rho$. Dabei sei $j_{n1} < \ldots < j_{n\rho}$ mit $\rho \in \mathbb{N}_+$.

Angenommen, in der obigen Zeilenstufenmatrix $B^{(n-1)}$ gilt $\rho \geq 2$, d.h. nach $a_{nj_n} = a_{nj_{n1}}$ folgt in der letzten Zeile von $B^{(n-1)}$ noch ein weiterer Koeffizient $a_{nj_{n2}} \neq 0$. Welche maximalen Minoren gewinnt man, wenn man in der Matrix $B^{(n-1)}$ die j_n-te Spalte durch 0 ersetzt und dann erneut die Backup-Prozedur durchführt? Zunächst erhält man dann aus $B^{(n-1)}$ die Zeilenstufenmatrix

$$\widetilde{B}^{(n-1)} = \begin{pmatrix} \alpha_{1j_1}^{(0)} & \cdots & \cdots & \cdots & \alpha_{1,j_{n_1}-1}^{(0)} & 0 & \alpha_{1,j_{n_1}+1}^{(0)} & \cdots & \alpha_{1j_{n2}}^{(0)} & \cdots & \alpha_{1m}^{(0)} \\ & \ddots & & & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ & & \alpha_{n-1,j_{n-1}}^{(n-2)} & \cdots & \alpha_{n-1,j_{n_1}-1}^{(n-2)} & 0 & \alpha_{n-1,j_{n_1}+1}^{(n-2)} & \cdots & \alpha_{n-1,j_{n2}}^{(n-2)} & \cdots & \alpha_{n-1,m}^{(n-2)} \\ & \mathbf{0} & & & & & & & \alpha_{n,j_{n2}}^{(n-1)} & \cdots & \alpha_{nm}^{(n-1)} \end{pmatrix}$$

deren n-te Zeilenstufe nicht mehr an der Position $j_n = j_{n1}$, sondern an der Position j_{n2} liegt. Schließt man nun die Backup-Prozedur an, so erhält man die Diagonalmatrix

$$\widetilde{F}^{(0)} = \left(\widetilde{\delta}_{ij}^{(n)}\right) = \Big([j_1 \ldots \widehat{j_i}\ j \ldots j_{n-1}\ j_{n2}](A)\Big) =$$

$$= \begin{pmatrix}
\widetilde{\delta}_{1j_1}^{(n)} & \cdots & * & 0 & * \cdots * & 0 & * \cdots * & 0 & \widetilde{\delta}_{1,j_{n2}+1}^{(n)} & \cdots & \widetilde{\delta}_{1m}^{(n)} \\
 & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\
 & & * & 0 & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
 & & & \widetilde{\delta}_{n-1,j_{n-1}}^{(n)} & * \cdots * & 0 & * \cdots * & 0 & \widetilde{\delta}_{n-1,j_{n2}+1}^{(n)} & \cdots & \widetilde{\delta}_{n-1,m}^{(n)} \\
 & \mathbf{0} & & & & & & \widetilde{\delta}_{n,j_{n2}}^{(n)} & \widetilde{\delta}_{n,j_{n2}+1}^{(n)} & \cdots & \widetilde{\delta}_{nm}^{(n)}
\end{pmatrix}$$

mit den Zeilenstufenkoeffizienten

$$\widetilde{\delta}_{1j_1}^{(n)} = \widetilde{\delta}_{2j_2}^{(n)} = \ldots = \widetilde{\delta}_{nj_{n2}}^{(n)} = [j_1 \ldots j_{n-1}\ j_{n2}](A)$$

und den übrigen Determinanten, die bis aufs Vorzeichen bestimmte maximale Minoren sind, nämlich den Determinanten

$$\widetilde{\delta}_{ij}^{(n)} = [j_1 \ldots \widehat{j_i}\ j \ldots j_{n-1}\ j_{n2}](A),$$

bei denen jeweils $(n-1)$ paarweise verschiedene Spalten aus $\{j_1, j_2 \ldots, j_{n-1}, j_{n2}\}$ festgehalten werden und der Spaltenindex j die Menge der Nichtzeilenstufenspalten, also die Menge $\{1, \ldots, m\} \setminus \{j_1, j_2, \ldots, j_{n1}, j_{n2}\}$ durchläuft.

Ähnlich wie zuvor kann man nun in der Matrix $\widetilde{B}^{(n-1)}$ die Spalte j_{n2} durch 0 ersetzen und dann erneut die Backup-Prozedur durchführen, bis man schließlich sukzessive alle Spalten $j_{n1}, j_{n2}, \ldots, j_{n\rho}$ erfaßt hat. Bei jeder Backup-Prozedur erhält man für $\lambda = 1, \ldots, \rho$ insbesondere die Determinanten

$$[j_1 \ldots j_{n-1}\ j_{n\lambda}](A) \quad \text{und}$$
$$[j_1 \ldots \widehat{j_i}\ j \ldots j_{n-1}\ j_{n\lambda}](A),$$

wobei letztere für $i \in \{1, \ldots, n\}$ bis auf das Vorzeichen maximale Minoren von A in den Spalten $j \in J := \{1, \ldots, m\} \setminus \{j_1, j_2, \ldots, j_{n-1}, j_{n_1}, j_{n2}, \ldots, j_{n\lambda}\}$ sind. Dieses besagte Vorzeichen ist das Signum der Permutation, die die Komponenten von $[j_1 \ldots \widehat{j_i}\ j \ldots j_{n-1}\ j_{n\lambda}]$ in aufsteigende Reihenfolge bringt.

Welche maximalen Minoren lassen sich mit dieser Vorgehensweise insgesamt gewinnen?

21.4 Lemma (wiederholte Backup-Prozedur)
Es liege die obige Situation vor, wobei man zur Abkürzung für die allererste Zeilenstufenmatrix $B := B^{(n-1)}$ setzt.

a) (Verschwindende Minoren) Seien die Spaltenindizes $l_1, \ldots, l_n \in \{1, \ldots, m\}$ paarweise verschieden und so durchnumeriert, daß $h_1 \le \ldots \le h_n$ gilt, wenn $h_\lambda \in \{0, \ldots, n\}$ jeweils die Höhe bezeichnet, auf der die Spalte l_λ in B liegt, $\lambda = 1, \ldots, n$.
Dann gilt $[l_1 \ldots l_n](A) = 0$, wenn $h_\lambda < \lambda$ für ein $\lambda \in \{1, \ldots, n\}$ ist. Bezeichnet $[\widetilde{l}_1 \ldots \widetilde{l}_n]$ eine Permutation von $[l_1 \ldots l_n]$, so verschwindet $[\widetilde{l}_1 \ldots \widetilde{l}_n](A)$ genau dann, wenn $[l_1 \ldots l_n](A)$ verschwindet.

Außerdem folgt $[l_1 \dots l_n](A) \neq 0$, *wenn* $h_\lambda = \lambda$ *für alle* $\lambda \in \{1, \dots, n\}$ *ist. Falls* $h_\lambda > \lambda$ *für ein* $\lambda \in \{1, \dots, n\}$ *gilt, ist keine Aussage über* $[l_1 \dots l_n](A)$ *möglich.*

b) Wiederholt man wie beschrieben die Backup-Prozedur solange, bis die n-te Zeile der zugrunde liegenden ersten Zeilenstufenmatrix $B^{(n-1)}$ *gleich 0 gesetzt ist, so kennt man aufgrund der zugehörigen Diagonalmatrizen bis aufs Vorzeichen insbesondere die Determinanten* $[l_1 \dots l_n](A)$ *mit paarweise verschiedenen Spaltenindizes* $l_1, \dots, l_n \in \{1, \dots, m\}$, *für die gilt:*

(1) Die ersten $(n-2)$ *Indizes* $l_1, \dots, l_{n-2}$ *sind Zeilenstufenindizes aus* $\{j_1, j_2, \dots, j_{n-1}\}$.

(2) Die beiden letzten Indizes l_{n-1} *und* l_n *sind beliebig. Bezeichnen insbesondere* l_{n-1} *und* l_n *Nichtzeilenstufenspalten aus* $\{1, \dots, m\} \backslash \{j_1, j_2, \dots, j_{n-1}, j_{n1}, j_{n2}, \dots, j_{n\rho}\}$, *so gilt* $[l_1 \dots l_n](A) = 0$.

Beweis:

a) Seien die Spaltenindizes $l_1, \dots, l_n \in \{1, \dots, m\}$ paarweise verschieden, wobei für die zugehörigen Höhen $h_1 \leq \dots \leq h_n$ gelte. Die quadratische Teilmatrix von B in den Zeilen $1, \dots, n$ und den Spalten $l_1, \dots, l_n$ nimmt dann die Gestalt

$$E = \begin{pmatrix} b_{1l_1} & b_{1l_2} & \cdots & \cdots & \cdots & b_{1l_n} \\ \vdots & \vdots & & & & \vdots \\ b_{h_1 l_1} & b_{h_1 l_2} & \cdots & \cdots & \cdots & b_{h_1 l_n} \\ 0 & \vdots & & & & \vdots \\ \vdots & b_{h_2 l_2} & & & & b_{h_2 l_n} \\ \vdots & 0 & & \ddots & & \vdots \\ \vdots & \vdots & & & & b_{h_n l_n} \\ \vdots & \vdots & & \mathbf{0} & & 0 \\ \vdots & \vdots & & & & \vdots \\ 0 & 0 & \cdots & \cdots & \cdots & 0 \end{pmatrix} \qquad (*)$$

an, wobei für $\lambda = 1, \dots, n$ definitionsgemäß $b_{h_\lambda l_\lambda} \neq 0$ gilt.
Ist $h_\lambda = \lambda$ für $\lambda = 1, \dots, n$, so folgt aus $(*)$

$$E = \begin{pmatrix} b_{1l_1} & \cdots & b_{1l_n} \\ & \ddots & \vdots \\ \mathbf{0} & & b_{nl_n} \end{pmatrix},$$

wobei die Koeffizienten auf der Diagonalen ungleich 0 sind und alle Koeffizienten unterhalb der Diagonalen gleich 0. Diese Teilmatrix E von B hat demnach den maximalen Rang n, somit auch die dazu äquivalente Teilmatrix $\widetilde{E}$ von A in ebendiesen Zeilen und Spalten, also

$$|\widetilde{E}| = [l_1 \dots l_n](A) \neq 0.$$

Gilt dagegen $h_\lambda < \lambda$ für ein $\lambda \in \{1, \dots, n\}$, so kann die quadratische Matrix E aus $(*)$ offensichtlich nicht maximalen Rang annehmen, ebensowenig kann dies die äquivalente Matrix $\widetilde{E}$, und es folgt

$$|\widetilde{E}| = [l_1 \dots l_n](A) = 0.$$

Da eine Permutation der Spalten sich bekanntlich nur auf das Vorzeichenverhalten der Determinante auswirkt, ist die Aussage über $[l_1 \dots l_n]$ und $[\widetilde{l_1} \dots \widetilde{l_n}]$ klar.

Im Falle, daß $h_\lambda > \lambda$ für ein $\lambda \in \{1,\dots,n\}$ gilt, ist offenbar keine Aussage über $\operatorname{rang} E = \operatorname{rang} \widetilde{E}$ möglich.

b) Den Vorüberlegungen zufolge erhält man bei Wiederholung der Backup-Prozedur für $\lambda = 1,\dots,\rho$ alle Determinanten n-ter Ordnung

$$[j_1 \dots \widehat{j_i}\ j \dots j_{n-1}\ j_{n\lambda}](A) \qquad (*)$$

mit $i \in \{1,\dots,n\}$, $j \in \{1,\dots,m\}\backslash\{j_1, j_2, \dots, j_{i-1}, j_{i+1}, \dots, j_{n-1}, j_{n1}, j_{n2}, \dots, j_{n,\lambda-1}\}$. Diese Determinanten erfüllen offensichtlich die Behauptung, wenn man die Spalten entsprechend tauscht. Es bleibt noch zu zeigen, daß für $i = 1,\dots,n-1$ neben j_i auch die Spalte $j_{n\lambda}$ in $(*)$ durch eine beliebige andere Nichtzeilenstufenspalte aus $\{1,\dots,m\} \setminus \{j_1, j_2, \dots, j_{n-1} j_{n1}, j_{n2} \dots, j_{n\rho}\}$ ersetzt werden kann und daß in diesem Fall die entsprechende Determinante den Wert 0 annimmt. Dies folgt aber sofort aus a), denn in diesem Fall verschwindet der zugehörige Minor, weil keine Spalte auf der Höhe n enthalten ist. □

Als erste Anwendung von Lemma 21.4 folgt

21.5 Satz (Algorithmus FindZeroMaxMinors)
Sei A eine beliebige $n \times m$-Matrix über einem Integritätsring R mit $n \le m$, und sei B eine dazu äquivalente Zeilenstufenmatrix, ferner sei $L \subseteq L_n^m$ eine beliebige Teilmenge. Die Menge $\Lambda \subseteq \{1,\dots,m\}$ bestehe ausschließlich aus Spaltenindizes $\lambda \in \{1,\dots,m\}$ mit $b^\lambda = 0$. Betrachte die folgenden Instruktionen:

(1) Ermittle für $i = 0,\dots,n$ die Menge H_i, die alle Spaltenindizes $j \in \{1,\dots,m\}$ enthält, für die die Spalte b^j auf der Höhe i liegt. Ersetze H_0 durch $H_0 \setminus \Lambda$.
(2) Falls H_0 nicht leer ist, streiche aus L alle Klammerausdrücke $[l_1 \dots l_n]$, die mindestens eine Spalte aus H_0 enthalten.
(3) Setze $i := 1$.
(4) Falls $i \ge n$ gilt, gib das Ergebnis L aus.
(5) Falls H_i höchstens ein Element enthält, erhöhe i um 1 und fahre fort bei Schritt (4).
(6) Setze $\mu := 2$.
(7) Falls H_i weniger als μ Elemente oder falls $\cup_{i \le \lambda \le n} H_\lambda$ weniger als $(n - i - 1 + \mu)$ Elemente besitzt, erhöhe i um 1 und fahre fort bei Schritt (4).
(8) Ermittle alle Klammerausdrücke $[l_1 \dots l_n]$ mit paarweise verschiedenen $l_1,\dots,l_n \in \{1,\dots,m\}$, für die gilt: $l_1,\dots,l_{i-\mu+1} \in \cup_{1 \le \lambda \le i-1} H_\lambda$, außerdem $l_{i-\mu+2},\dots,l_{i+1} \in H_i$ und $l_{i+2},\dots,l_n \in (\cup_{i \le \lambda \le n} H_\lambda) \setminus \{l_{i-\mu+2},\dots,l_{i+1}\}$. (Für $i = 1$ oder $i - \mu + 1 \le 0$ oder $i + 2 > n$ werden die erst- bzw. letztgenannten Komponenten nicht gewählt.)
(9) Permutiere alle diese $[l_1 \dots l_n]$ so, daß sie aufsteigend sortiert sind und streiche sie aus L, falls sie dort enthalten sind. Erhöhe μ um 1 und fahre fort bei Schritt (7).

Dies ist ein Algorithmus mit der Bezeichnung `FindZeroMaxMinors`(B, L, Λ)*, der aus der Menge L solche Klammerausdrücke $[l_1 \dots l_n]$ streicht, für die $l_1,\dots,l_n \notin \Lambda$ und für deren zugehörige maximale Minoren $[l_1 \dots l_n](A) = 0$ gilt.*

Beweis:
Die Variable i kann in den Schritten (5) und (7) nur endlich oft um 1 erhöht werden, bis vom Ausgangswert 1 der endliche Wert n erreicht ist. Auch μ wird in Schritt (9) vom Ausgangswert 2 offensichtlich nur endlich oft um 1 erhöht, bis in Schritt (7) eines der beiden Abbruchkriterien erreicht ist. Insgesamt sind die angegeben Instruktionen daher nur endlich oft zu durchlaufen.

Daß aus L wie behauptet ausschließlich Klammerausdrücke verschwindender Minoren gestrichen werden, folgt sofort aus Lemma 21.4a). In Schritt (8) werden nämlich alle

Klammerausdrücke $[l_1 \ldots l_n]$ konstruiert, die aus paarweise verschiedenen Spaltenindizes $l_1, \ldots, l_n \in \cup_{i=1\ldots n} H_i$ bestehen, wobei $1 \leq h_1 \leq \ldots \leq h_n$ und $h_{i+1} < i+1$ gilt für die zugehörigen Höhen $h_\lambda \in \{1, \ldots, n\}$, auf denen die Spalten b^λ, $\lambda = 1, \ldots, n$, liegen. Die beiden Abbruchkriterien in Schritt (7) stellen sicher, daß $l_{i-\mu+2}, \ldots, l_n$ auch wirklich gewählt werden können. Da i.a. nicht $l_1 < \ldots < l_n$ gilt, werden die $[l_1 \ldots l_n]$ aus Schritt (8) in Schritt (9) so permutiert, daß sie aufsteigend angeordnet sind und als Elemente aus L entfernt werden können. Weil in Schritt (1) aus der Indexmenge H_0 der Spaltenvektoren, die gleich 0 sind, die Elemente aus Λ ausgesondert werden, werden in Schritt (2) aus L außerdem nur solche $[l_1 \ldots l_n]$ mit $l_1, \ldots, l_n \notin \Lambda$ entfernt. □

21.6 Bemerkungen zur Implementierung (FindZeroMaxMinors)

a) Setzt man in der Matrix A bzw. B die Spalten $\lambda \in \Lambda$ nachträglich gleich 0, so muß man diese Spalten ausnehmen, wenn man die verschwindenden Minoren von A ermitteln will. Daher wird beim Algorithmus `FindZeroMaxMinors`(B, L, Λ) aus Satz 21.5 als drittes das Argument Λ übergeben, um nicht fälschlicherweise Elemente aus L zu entfernen.

b) Man kann mit `FindZeroMaxMinors` i.a. nicht alle verschwindenden Minoren von A aus L herausfiltern, sondern nur diejenigen, die man kombinatorisch aus der vorliegenden Zeilenstufenmatrix B gewinnen kann. Die Liste L mit den Klammerausdrücken wird dadurch angepaßt, daß bei der Definition von `FindZeroMaxMinors` das Argument L als `Var` L deklariert ist.

c) (*Algorithmus* `Height`) Die Bestimmung der Mengen H_i, $i = 0, \ldots, n$ aus Schritt (1) wird mit der einfachen Unterfunktion `Height`(B, i) erledigt, die auch an späterer Stelle noch nützlich sein wird. Die Konstruktion der Klammerausdrücke in Schritt (8) erfolgt mit Hilfe von `AllSubsets` aus Satz 20.4.

Um zu veranschaulichen, welche maximalen Minoren und insbesondere welche verschwindenden Minoren man mit der wiederholten Backup-Prozedur erhalten kann, wird diese im folgenden an diversen Matrizen nachvollzogen.

21.7 Beispiel (FindZeroMaxMinors, Backup-Prozedur, lineare Relationen)
Zunächst betrachtet man die 3×8-Matrix

$$A = \begin{pmatrix} 1 & 1 & 1 & 0 & 3 & 1 & 2 & 1 \\ 1 & 1 & 2 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 2 & 1 & 2 & 1 & 1 & 2 \end{pmatrix}.$$

(1) Mit `B2TMDMD` erhält man aus A die dazu äquivalente Zeilenstufenmatrix

$$B = \begin{pmatrix} 1 & 1 & 1 & 0 & 3 & 1 & 2 & 1 \\ 0 & 0 & 1 & 1 & -3 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 1 & 1 \end{pmatrix}.$$

Ihre Zeilenstufen liegen bei $j_1 = 1$, $j_2 = 3$ und $j_3 = 5$. Als potentielle Zeilenstufen in der dritten Zeile kommen $j_{31} = 5$, $j_{32} = 7$ und $j_{33} = 8$ in Frage. Außerdem liest man ab, daß die Spalten 1, 2 und 6 auf der Höhe 1 liegen sowie die Spalten 3 und 4 auf der Höhe 2. Daher verschwinden alle Minoren von A, die aus mindestens zweien der drei Spalten 1,2 und 6 bestehen oder die die beiden Spalten 3 und 4 beinhalten und eine der Spalten 1,2,6. Kombinatorisch ergeben sich insgesamt 19 verschwindende Minoren

mit den Spalten

$$\begin{aligned}&[\underline{12}3],[\underline{12}4],[\underline{12}5],[\underline{12}6],[\underline{12}7],[\underline{12}8],\\&[\underline{13}\underline{6}],[\underline{14}\underline{6}],[\underline{15}\underline{6}],[\underline{16}7],[\underline{16}8],\\&[\underline{23}\underline{6}],[\underline{24}\underline{6}],[\underline{25}\underline{6}],[\underline{26}7],[\underline{26}8],\\&[1\underline{34}],[2\underline{34}],[\underline{34}6].\end{aligned}$$

Die Bedingung, daß die Spalten 3 und 4 enthalten sind, ist alleine nicht hinreichend für das Verschwinden eines Minors, denn es gilt beispielsweise

$$[345](A)=\begin{vmatrix}1&0&3\\2&1&0\\2&1&2\end{vmatrix}=2\neq 0.$$

In der letzten Zeile von B lassen sich außerdem drei nichtverschwindende maximale Minoren ablesen, nämlich

$$\begin{aligned}b_{35}=\alpha_{35}^{(2)}&=[j_1\,j_2\,5](A)=[135](A)=2,\\b_{37}=\alpha_{37}^{(2)}&=[j_1\,j_2\,7](A)=[137](A)=1,\\b_{38}=\alpha_{38}^{(2)}&=[j_1\,j_2\,8](A)=[138](A)=1.\end{aligned}$$

(2) Für $j_3=j_{31}=5$ ergibt sich mit der Backup-Prozedur die Diagonalmatrix $D=(\delta_{ij}^{(3)})$

$$D=\begin{pmatrix}\delta_{11}^{(3)}&\delta_{12}^{(3)}&0&\delta_{14}^{(3)}&0&\delta_{16}^{(3)}&\delta_{17}^{(3)}&\delta_{18}^{(3)}\\0&0&\delta_{23}^{(3)}&\delta_{24}^{(3)}&0&\delta_{26}^{(3)}&\delta_{27}^{(3)}&\delta_{28}^{(3)}\\0&0&0&0&\delta_{35}^{(3)}&\delta_{36}^{(3)}&\delta_{37}^{(3)}&\delta_{38}^{(3)}\end{pmatrix}=$$

$$=\begin{pmatrix}2&2&0&-2&0&2&2&-4\\0&0&2&2&0&0&-1&3\\0&0&0&0&2&0&1&1\end{pmatrix}$$

Diese Diagonalmatrix enthält bis aufs Vorzeichen alle maximalen Minoren, die aus zweien der drei Spalten $j_1=1, j_2=3$ und $j_3=5$ bestehen. Man erhält somit weitere 7 maximale Minoren von A, die noch nicht aus (1) bekannt sind, nämlich in der

$$\begin{aligned}\text{zweiten Zeile}\quad&[\underline{14}\underline{5}],[\underline{15}7],[\underline{15}8],\\\text{ersten Zeile}\quad&[2\underline{35}],[\underline{34}\underline{5}],[\underline{35}6],[\underline{35}7],[\underline{35}8],\end{aligned}$$

Detailliert folgt beispielsweise

$$\begin{aligned}2=d_{16}=\delta_{16}^{(3)}&=[6\;j_2\;j_3](A)=[6\underline{35}](A)=[\underline{35}6](A),\\2=d_{24}=\delta_{24}^{(3)}&=[j_1\;4\;j_3](A)=[\underline{14}\underline{5}](A),\\-1=d_{27}=\delta_{27}^{(3)}&=[j_1\;7\;j_3](A)=[\underline{17}\underline{5}](A)=-[\underline{15}7](A).\end{aligned}$$

Einen Teil, aber nicht alle der verschwindenden Minoren aus (1) kann man auch in dieser Matrix finden, z.B. an der Position (2,2)

$$0 = d_{22} = \delta_{22}^{(3)} = [j_1 2 j_3](A) = [125](A)$$

oder an Position (3,6)

$$0 = d_{36} = \delta_{36}^{(3)} = [j_1 j_2 6](A) = [136](A).$$

Der Minor $[126](A) = 0$ hat dagegen keine Darstellung der Form $\delta_{ij}^{(k)}$, er befindet sich also nicht in der Diagonalmatrix D.

(3) Ersetzt man in der Matrix B aus Schritt (1) die Spalte $j_{31} = 5$ durch 0, so erhält man die Zeilenstufenmatrix

$$\widetilde{B} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Ihre Zeilenstufen liegen bei $j_1 = 1$, $j_2 = 3$ und $j_3 = j_{32} = 7$. An dieser Matrix kann man keine weiteren verschwindenden Minoren mehr ablesen, die man nicht schon aus (1) kennt.

(4) Mit der Backup-Prozedur ergibt sich die Diagonalmatrix $\widetilde{D} = (\widetilde{\delta}_{ij}^{(3)})$ mit

$$\widetilde{D} = \begin{pmatrix} \widetilde{\delta}_{11}^{(3)} & \widetilde{\delta}_{12}^{(3)} & 0 & \widetilde{\delta}_{14}^{(3)} & 0 & \widetilde{\delta}_{16}^{(3)} & 0 & \widetilde{\delta}_{18}^{(3)} \\ 0 & 0 & \widetilde{\delta}_{23}^{(3)} & \widetilde{\delta}_{24}^{(3)} & 0 & \widetilde{\delta}_{26}^{(3)} & 0 & \widetilde{\delta}_{28}^{(3)} \\ 0 & 0 & 0 & 0 & 0 & 0 & \widetilde{\delta}_{37}^{(3)} & \widetilde{\delta}_{38}^{(3)} \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & 1 & 0 & -1 & 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Sie enthält bis aufs Vorzeichen maximale Minoren von A, die sich aus mindestens zweien der drei Spalten $j_1 = 1, j_2 = 3, j_3 = 7$ ergeben, nicht aber die Spalte 5 enthalten. Noch nicht bekannt aus früheren Schritten sind die folgenden 6 Minoren, die man ablesen kann in der

$$\begin{aligned} \text{zweiten Zeile} \quad & [\underline{1}4\underline{7}], [\underline{1}7\underline{8}], \\ \text{ersten Zeile} \quad & [2\underline{37}], [\underline{3}4\underline{7}], [\underline{3}6\underline{7}], [\underline{37}8]. \end{aligned}$$

(5) Ersetzt man in der Matrix $\widetilde{B}$ aus Schritt (3) die Spalte $j_{32} = 7$ durch 0, so erhält man die Zeilenstufenmatrix

$$\ddot{B} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

mit den Zeilenstufen $j_1 = 1$, $j_2 = 3$ und $j_3 = j_{33} = 8$.

(6) Führt man die Backup-Prozedur durch, erhält man die Diagonalmatrix $\ddot{D} = (\ddot{\delta}_{ij}^{(3)})$ mit

$$\ddot{D} = \begin{pmatrix} \ddot{\delta}_{11}^{(3)} & \ddot{\delta}_{12}^{(3)} & 0 & \ddot{\delta}_{14}^{(3)} & 0 & \ddot{\delta}_{16}^{(3)} & 0 & 0 \\ 0 & 0 & \ddot{\delta}_{23}^{(3)} & \ddot{\delta}_{24}^{(3)} & 0 & \ddot{\delta}_{26}^{(3)} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ddot{\delta}_{38}^{(3)} \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & 1 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Diese Matrix besteht bis aufs Vorzeichen aus maximalen Minoren von A, die sich aus mindestens zweien der drei Spalten $j_1 = 1, j_2 = 3, j_3 = 8$ ergeben, ohne die Spalten 5 und 7 zu enthalten. Zusätzlich zu den bereits bekannten Minoren erhält man 4 weitere Minoren, nämlich aus der

$$\begin{array}{rl} \text{zweiten Zeile} & [\underline{1}4\underline{8}], \\ \text{ersten Zeile} & [2\underline{38}], [\underline{3}4\underline{8}], [\underline{36}\underline{8}]. \end{array}$$

(7) Mit `AllSubsets`$(\{1,\ldots,8\},3)$ erhält man die lexikographisch geordnete Menge aller Klammerausdrücke $[l_1\, l_2\, l_3]$ mit paarweise verschiedenen $l_1, l_2, l_3 \in \{1,\ldots,8\}$. Schreibt man unter jeden Klammerausdruck den Wert des zugehörigen Minors, so folgt aus den Schritten (1) bis (6)

[123],	[124],	[125],	[126],	[127],	[128],	[134],	[135],	[136],	[137],
0	0	0	0	0	0	0	2	0	1
[138],	[145],	[146],	[147],	[148],	[156],	[157],	[158],	[167],	[168],
1	2	0	1	1	0	1	−3	0	0
[178],	[234],	[235],	[236],	[237],	[238],	[245],	[246],	[247],	[248],
−2	0	2	0	1	1		0		
[256],	[257],	[258],	[267],	[268],	[278],	[345],	[346],	[347],	[348],
0			0	0		2	0	1	1
[356],	[357],	[358],	[367],	[368],	[378],	[456],	[457],	[458],	[467],
2	2	−4	−1	−1	−3				
[468],	[478],	[567],	[568],	[578],	[678].				

Bis jetzt hat man schon 40 der 56 maximalen Minoren von A berechnet. Es bestätigt sich Lemma 21.4b), d.h. man kennt alle Minoren, die die Spalte $j_1 = 1$ bzw. $j_2 = 3$ beinhalten, und außerdem kennt man eine beträchtliche Anzahl verschwindender Minoren.

Da alle Minoren $[1\ l_2\ l_3](A)$ mit paarweise verschiedenen $l_2, l_3 \in \{2,\ldots,m\}$ zur Verfügung stehen, kann man die 16 noch fehlenden Minoren mit Hilfe der linearen Relationen aus Lemma 20.7 berechnen, siehe Schritt (8a). Als zweite Möglichkeit wiederholt man einfach die Schritte (1) bis (6), also die Backup-Prozedur, solange an geeigneten Matrizen, bis alle Minoren ermittelt sind, siehe weiter unten die Schritte (8b) bis (12b).

(8a) (*Lineare Relationen*) Unter Verwendung der Formel

$$[l_1\, l_2\, l_3](A) = \frac{1}{a_{1j_1}} \cdot \sum_{\lambda=1}^{3} (-1)^{\lambda+1} \cdot a_{1l_\lambda} \cdot [j_1\ l_1 \ldots \widehat{l_\lambda} \ldots l_3](A)$$

aus Lemma 20.7a) ergibt sich mit $j_1 = 1$ und $a_{1j_1} = 1$ beispielsweise

$$[245](A) = a_{12} \cdot [145](A) - a_{14} \cdot [125](A) + a_{15} \cdot [124] = 1 \cdot 2 - 0 \cdot 0 + 3 \cdot 0 = 2.$$

Insgesamt erhält man damit alle maximalen Minoren:

[123],	[124],	[125],	[126],	[127],	[128],	[134],	[135],	[136],	[137],
0	0	0	0	0	0	0	2	0	1
[138],	[145],	[146],	[147],	[148],	[156],	[157],	[158],	[167],	[168],
1	2	0	1	1	0	1	−3	0	0
[178],	[234],	[235],	[236],	[237],	[238],	[245],	[246],	[247],	[248],
−2	0	2	0	1	1	2	0	1	1
[256],	[257],	[258],	[267],	[268],	[278],	[345],	[346],	[347],	[348],
0	1	−3	0	0	−2	2	0	1	1
[356],	[357],	[358],	[367],	[368],	[378],	[456],	[457],	[458],	[467],
2	2	−4	−1	−1	−3	2	1	−1	−1
[468],	[478],	[567],	[568],	[578],	[678].				
−1	−1	−1	3	1	−2				

(8b) Alternativ zu Schritt (8a) wiederholt man einfach die Schritte (1) bis (6), also die Backup-Prozedur, indem man oben in Schritt (7) den ersten noch nicht berechneten Klammerausdruck sucht, in diesem Falle [245], und dann in der Matrix A die nicht benötigten Spalten 1 und 3 gleich 0 setzt. Man erhält also aus A die Matrix

$$E = \begin{pmatrix} 0 & 1 & 0 & 0 & 3 & 1 & 2 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 2 & 1 & 1 & 2 \end{pmatrix}$$

und daraus wiederum die Zeilenstufenmatrix

$$F = \begin{pmatrix} 0 & 1 & 0 & 0 & 3 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & -3 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 1 & 1 \end{pmatrix}$$

und die Diagonalmatrix

$$G = \begin{pmatrix} 0 & 2 & 0 & 0 & 0 & 2 & 1 & -1 \\ 0 & 0 & 0 & 2 & 0 & 0 & -1 & 3 \\ 0 & 0 & 0 & 0 & 2 & 0 & 1 & 1 \end{pmatrix}.$$

(9b) Anhand der Zeilenstufenmatrix F kann man offenbar keine weiteren Minoren entlarven, die die Spalten 1 und 3 nicht enthalten und die verschwinden. Die Diagonalmatrix G enthält bis aufs Vorzeichen Minoren, die aus mindestens zweien der drei Zeilenstufenspalten $j_1 = 2$, $j_2 = 4$ und $j_3 = 5$ bestehen, nämlich in der

dritten Zeile	$[\underline{24}5], [\underline{24}7], [\underline{24}8],$
zweiten Zeile	$[\underline{25}7], [\underline{25}8]$
ersten Zeile	$[\underline{45}6], [\underline{45}7], [\underline{45}8].$

(10b) Ersetzt man in der Zeilenstufenmatrix F aus Schritt (9a) die fünfte Spalte durch 0, so erhält man zunächst die Matrix

$$\widetilde{F} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

mit den Zeilenstufen $j_1 = 2$, $j_2 = 4$ und $j_3 = 7$ und daraus die Diagonalmatrix

$$\widetilde{G} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix},$$

aus der man die folgenden Minoren erhalten kann aus der

$$\begin{array}{rl} \text{zweiten Zeile} & [\underline{2}7\underline{8}] \\ \text{ersten Zeile} & [\underline{4}6\underline{7}], [\underline{4}7\underline{8}]. \end{array}$$

(11b) Setzt man in der Zeilenstufenmatrix $\widetilde{F}$ aus Schritt (10a) die siebte Spalte gleich 0, so erhält man zunächst die Matrix

$$\ddot{F} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

mit den Zeilenstufen $j_1 = 2$, $j_2 = 4$ und $j_3 = 8$ und daraus die Diagonalmatrix

$$\ddot{G} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

aus der man aus der ersten Zeile den Minor $[\underline{4}6\underline{8}]$ gewinnt.

(12b) Es fehlen nur noch die Minoren in den Spalten $[567], [568], [578], [678]$. Dazu setzt man in A die ersten vier Spalten gleich 0 und erhält schließlich die zugehörige Diagonalmatrix

$$\begin{pmatrix} 0 & 0 & 0 & 0 & -1 & 0 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 3 \end{pmatrix},$$

die die fehlenden vier Minoren erhält. Insgesamt ergibt sich dasselbe Endergebnis, wie in (8a) angegeben ist.

Nimmt man die linearen Relationen in Schritt (8a) in Anspruch, muß man A einmal trigonalisieren in Schritt (1) und dreimal in den Schritten (2), (4) und (6) die Backup-Prozedur verwenden. Im Laufe der Backup-Prozedur hat man bereits einige Minoren ermittelt, die man in Schritt (8a) auch mit den linearen Relationen erhalten kann, beispielsweise [238]. Allgemein gesprochen betrifft dies all die Minoren, die man jeweils aus der ersten Zeile der Diagonalmatrizen abliest. Nach Lemma 20.7 (lineare Relationen) sind für diese Minoren die Backup-Formel und die Relationenformel äquivalent.

Verzichtet man auf die linearen Relationen, so muß man nochmals in Schritt (8b) eine Matrix trigonalisieren und in den Schritten (9b), (10b) und (11b) die Backup-Prozedur sukzessive durchführen. Anschließend wird in Schritt (12b) eine Trigonalisierung und eine Backup-Prozedur nötig.

Als Ergänzung zu dem eher einfachen Beispiel 21.7 einer dreizeiligen Matrix verschafft man sich nun noch einmal Klarheit darüber, welche maximalen Minoren sich mit den linearen Relationen (Lemma 20.7) und der wiederholten Backup-Prozedur (Lemma 21.4) gewinnen lassen, vor allem auch für größere Matrizen.

21.8 Bemerkungen (lineare Relationen, Backup-Prozedur)
Sei A eine $n \times m$-Matrix mit $n \leq m$ und $\operatorname{rang} A = n$. Zudem bezeichne B die zu A äquivalente Zeilenstufenmatrix, die man bei der Trigonalisierung mit Division erhält.

a) (*Lineare Relationen*) Wenn $a_{il_0} \neq 0$ für ein $i \in \{1, \ldots, n\}$ und ein $l_0 \in \{1, \ldots, m\}$ gilt und wenn man alle maximalen Minoren von A kennt, die die Spalte l_0 enthalten, so kann man nach Lemma 20.7a) (Lineare Relationen) alle anderen Minoren $[l_1 \ldots l_n](A)$ mit $l_1, \ldots, l_n \in \{1, \ldots, m\}$ und $l_1 < \ldots < l_n$ berechnen, und zwar mit der Formel

$$[l_1 \ldots l_n](A) = \frac{1}{a_{il_0}} \cdot \sum_{\lambda=1}^{n} (-1)^{\lambda+1} \cdot a_{il_\lambda} \cdot [l_0\ l_1 \ldots \widehat{l_\lambda} \ldots l_n](A). \qquad (*)$$

Unabhängig davon, welche Spalte aus $\{1, \ldots, m\}$ die Variable l_0 bezeichnet, existieren stets $\binom{m-1}{n-1}$ maximale Minoren, die die Spalte l_0 beinhalten. Beispielsweise kann man unter Verwendung der Zeilenstufenmatrix B

$$l_0 = j_1 \quad \text{und} \quad a_{il_\lambda} = b_{1l_\lambda},\ \lambda = 0, \ldots, n,$$

wählen. Dabei ist insbesondere sichergestellt, daß $a_{il_0} = b_{1j_1} \neq 0$ gilt, denn dies wird bei der Trigonalisierung durch eventuellen Zeilentausch erzwungen. Damit folgt aus $(*)$ die Formel

$$[l_1 \ldots l_n](A) = \frac{1}{b_{1j_1}} \cdot \sum_{\lambda=1}^{n} (-1)^{\lambda+1} \cdot b_{1l_\lambda} \cdot [j_1\ l_1 \ldots \widehat{l_\lambda} \ldots l_n](A). \qquad (**)$$

b) (*Backup-Prozedur*) Im Laufe der Trigonalisierung mit einem Bareiss-Verfahren wird bei jedem Zeilentausch der Vorzeichenwechsel der Determinante berücksichtigt, so daß am Ende die durch die Backup-Prozedur erzeugte Diagonalmatrix immer aus den Koeffizienten $\delta_{ij}^{(n)} = [1 \ldots n][j_1 \ldots j_i\ j \ldots j_n](A)$ besteht. Für die Koeffizienten $\delta_{1j}^{(n)}$, die sich in der Diagonalmatrix in der ersten Zeile befinden, ist die Backup-Rekursionsformel identisch mit der in a) angegebenen Relationenformel $(**)$, vergleiche dazu Lemma 20.7c).

21.9 Beispiele (wiederholte Backup-Prozedur)
Sei A eine $n \times m$-Matrix mit $n \leq m$ und $\operatorname{rang} A = n$. Zudem bezeichne B die zu A äquivalente Zeilenstufenmatrix, die man bei der Trigonalisierung mit Division erhält.

a) (*Dreizeilige Matrizen*) Sei A eine $3 \times m$-Matrix. Führt man an der Zeilenstufenmatrix B mit den potentiellen n-ten Zeilenstufen $j_{n1}, \ldots, j_{n\rho}$ die Backup-Prozedur sukzessive ρ-mal durch, so erhält man aus den jeweiligen Diagonalmatrizen einen Teil der folgenden Determinanten aus der

$$\begin{aligned} &\text{3. und 2. Zeile} \quad [j_1\ l_2\ l_3](A) \text{ mit } l_2, l_3 \in \{1, \ldots, m\} \setminus \{j_1\},\ l_2 \neq l_3, \\ &\text{1. Zeile} \quad [j_2\ l_2\ l_3](A) \text{ mit } l_2, l_3 \in \{1, \ldots, m\} \setminus \{j_1, j_2\},\ l_2 \neq l_3. \end{aligned}$$

Diejenigen der oben aufgeführten Determinanten, die man nicht an den Diagonalmatrizen ablesen kann, und weitere verschwindende Minoren erhält man aus der trigonalisierten Matrix B mit dem Algorithmus `FindZeroMaxMinors` aus Satz 21.5. So sind etwa alle Minoren $[l_1\ l_2\ l_3](A) = 0$, für die $l_1 < j_1$ oder $l_2 < j_2$ oder $l_3 < j_3$ gilt.
Ist beispielsweise $j_1 = 1$ und $j_2 = 2$, so kennt man alle maximalen Minoren mit den Spalten $[1**]$ und $[2**]$, wobei $*$ beliebige Spalten symbolisiere. Insbesondere stehen damit alle nötigen Minoren $[1**](A)$ zur Verfügung, um nach Bemerkung 21.8a) die im Fall $m \geq 5$ eventuell noch fehlenden Minoren mittels linearer Relationen berechnen zu können. Alternativ kann man diese auch dadurch gewinnen, daß man in A die Spalten 1 und 2 durch 0 ersetzt, die so erhaltene Matrix $\widetilde{A}$ dann trigonalisiert und, falls $\operatorname{rang} \widetilde{A} = 3$ gilt, mit der Backup-Prozedur diagonalisiert. Analog erhält man dann die Minoren mit den Spalten $[\widetilde{j}_1 **]$ und $[\widetilde{j}_2 **]$, wobei $\widetilde{j}_1$ und $\widetilde{j}_2$ die Zeilenstufen von $\widetilde{A}$ bezeichnen und $3 \leq \widetilde{j}_1 < \widetilde{j}_2$ gilt. Für den Fall $\widetilde{j}_1 = 3$ und $\widetilde{j}_2 = 4$ kennt man also alle Minoren mit den Spalten $[3**]$ und $[4**]$, für den Fall $\widetilde{j}_3 > 3$ verschwinden alle Minoren mit den Spalten $[3**], \ldots, [\widetilde{j}_3 - 1**]$, usw.

b) (*Vierzeilige Matrizen*) Ist A eine $4 \times m$-Matrix, so erhält man aus B mit der wiederholten Backup-Prozedur und mit `FindZeroMaxMinors` alle Minoren, die zwei der drei Spalten j_1, j_2 und j_3 enthalten, also alle unten angeführten Determinanten. In den entsprechenden Diagonalmatrizen kann man zumindest die nichtverschwindenden Minoren bis aufs Vorzeichen ablesen in der

$$\begin{aligned}
&\text{4. und 3. Zeile} && [j_1\ j_2\ l_3\ l_4](A) \text{ mit } l_3, l_4 \in \{1, \ldots, m\} \setminus \{j_1, j_2\},\ l_3 \neq l_4 \\
&\text{2. Zeile} && [j_1\ j_3\ l_3\ l_4](A) \text{ mit } l_3, l_4 \in \{1, \ldots, m\} \setminus \{j_1, j_3\},\ l_3 \neq l_4, \\
&\text{1. Zeile} && [j_2\ j_3\ l_3\ l_4](A) \text{ mit } l_3, l_4 \in \{1, \ldots, m\} \setminus \{j_2, j_3\},\ l_3 \neq l_4.
\end{aligned}$$

Angenommen A ist eine 4×7-Matrix, deren $\binom{7}{4} = 35$ maximale Minoren einfachheitshalber alle ungleich 0 sein sollen. Insbesondere gilt also $j_1 = 1$, $j_2 = 2$, $j_3 = 3$, $j_{41} = 4$, $j_{42} = 5$, $j_{43} = 6$ und $j_{44} = 7$.

(1) Aus A erhält man mit der wiederholten Backup-Prozedur zunächst alle Minoren, die aus zweien der drei Spalten 1, 2 und 3 bestehen, also alle maximalen Minoren mit den Spalten $[12**]$, $[13**]$ und $[23**]$, wobei $*$ wieder beliebige Spaltenindizes symbolisiere. Insgesamt kennt man nun die unterstrichenen Minoren:

$\underline{[1234]}$	$\underline{[1235]}$	$\underline{[1236]}$	$\underline{[1237]}$	$\underline{[1245]}$	$\underline{[1246]}$	$\underline{[1247]}$	$\underline{[1256]}$	$\underline{[1257]}$	$\underline{[1267]}$
$\underline{[1345]}$	$\underline{[1346]}$	$\underline{[1347]}$	$\underline{[1356]}$	$\underline{[1357]}$	$\underline{[1367]}$	[1456]	[1457]	[1467]	[1567]
$\underline{[2345]}$	$\underline{[2346]}$	$\underline{[2347]}$	$\underline{[2356]}$	$\underline{[2357]}$	$\underline{[2367]}$	[2456]	[2457]	[2467]	[2567]
[3456]	[3457]	[3467]	[3567]						
[4567]									

(2) Der nächste noch nicht berechnete Minor besteht aus den Spalten [1456]. Setzt man in A die beiden nicht benötigten Spalten 2 und 3 beide gleich 0, so erhält man schließlich mit der wiederholten Backup-Prozedur alle Minoren $[14**]$, $[15**]$ und $[45**]$, also

$\underline{[1234]}$	$\underline{[1235]}$	$\underline{[1236]}$	$\underline{[1237]}$	$\underline{[1245]}$	$\underline{[1246]}$	$\underline{[1247]}$	$\underline{[1256]}$	$\underline{[1257]}$	$\underline{[1267]}$
$\underline{[1345]}$	$\underline{[1346]}$	$\underline{[1347]}$	$\underline{[1356]}$	$\underline{[1357]}$	$\underline{[1367]}$	$\underline{[1456]}$	$\underline{[1457]}$	$\underline{[1467]}$	$\underline{[1567]}$
$\underline{[2345]}$	$\underline{[2346]}$	$\underline{[2347]}$	$\underline{[2356]}$	$\underline{[2357]}$	$\underline{[2367]}$	[2456]	[2457]	[2467]	[2567]
[3456]	[3457]	[3467]	[3567]						
$\underline{[4567]}$									

Soweit sind nun alle Minoren mit der Spalte 1 bekannt, d.h. aus ihnen könnte man die noch fehlenden mit geeigneten linearen Relationen berechnen. Man kann aber auch wie folgt fortfahren.

(3) Als nächstes fehlt der maximale Minor mit den Spalten [2456]. Ersetzt man in A die beiden Spalten 1 und 3 durch 0, so bekommt man mit der Backup-Prozedur alle Minoren $[24**], [25**]$ und $[45**]$, also

[1234]	[1235]	[1236]	[1237]	[1245]	[1246]	[1247]	[1256]	[1257]	[1267]
[1345]	[1346]	[1347]	[1356]	[1357]	[1367]	[1456]	[1457]	[1467]	[1567]
[2345]	[2346]	[2347]	[2356]	[2357]	[2367]	[2456]	[2457]	[2467]	[2567]
[3456]	[3457]	[3467]	[3567]						
[4567]									

Den Minor $[45**](A) = [4567](A)$ kennt man bereits aus Schritt (2). Diese Doppelberechnung kann vermieden werden, wenn man die Backup-Prozedur vor der obersten Zeile, in der dieser Minor bekanntlich abzulesen ist, stoppt.

(4) Zuletzt berechnet man den maximalen Minor mit den Spalten [3456]. Ersetzt man in A die beiden Spalten 1 und 2 durch 0, so bekommt man mit der Backup-Prozedur alle Minoren $[34**], [35**]$ und $[45**]$, d.h. jetzt sind alle maximalen Minoren erfaßt. Den Minor $[45**](A) = [4567](A)$ kennt man schon aus Schritt (2) bzw. Schritt (3).

c) (*Fünfzeilige Matrizen*) Sei A nun eine $5 \times m$-Matrix. Man erhält aus der Zeilenstufenmatrix B mit `FindZeroMaxMinors` und mit der wiederholten Backup-Prozedur aus den zugehörigen Diagonalmatrizen alle Minoren, die drei der vier Spalten $j_1,\ j_2,\ j_3,\ j_4$ enthalten, also die unten angegeben Minoren, die zum Teil in den Diagonalmatrizen in der

$$\begin{aligned}
\text{5. und 4. Zeile} \quad & [j_1\ j_2\ j_3\ l_4\ l_5](A) \text{ mit } l_4, l_5 \in \{1,\dots,m\} \setminus \{j_1, j_2, j_3\},\ l_4 \neq l_5,\\
\text{3. Zeile} \quad & [j_1\ j_2\ j_4\ l_4\ l_5](A) \text{ mit } l_4, l_5 \in \{1,\dots,m\} \setminus \{j_1, j_2, j_4\},\ l_4 \neq l_5,\\
\text{2. Zeile} \quad & [j_1\ j_3\ j_4\ l_4\ l_5](A) \text{ mit } l_4, l_5 \in \{1,\dots,m\} \setminus \{j_1, j_3, j_4\},\ l_4 \neq l_5,\\
\text{1. Zeile} \quad & [j_2\ j_3\ j_4\ l_4\ l_5](A) \text{ mit } l_4, l_5 \in \{1,\dots,m\} \setminus \{j_2, j_3, j_4\},\ l_4 \neq l_5
\end{aligned}$$

stehen. Angenommen A ist eine 5×8-Matrix, deren $\binom{8}{5} = 56$ maximale Minoren einfachheitshalber alle ungleich 0 sein sollen. Insbesondere gilt also $j_1 = 1$, $j_2 = 2$, $j_3 = 3$, $j_4 = 4$, $j_{51} = 5$, $j_{52} = 6$, $j_{53} = 7$ und $j_{54} = 8$.

(1) Aus A erhält man mit der 3-maligen Backup-Prozedur zunächst alle Minoren, die drei der vier Spalten 1, 2, 3, 4 enthalten, also alle unterstrichenen Minoren der Form $[123**]$, $[124**]$, $[134**]$, $[234**]$, wobei $*$ einen beliebigen Spaltenindex ersetze.

[12345],	[12346],	[12347],	[12348],	[12356],	[12357],	[12358],	[12367],
[12368],	[12378],	[12456],	[12457],	[12458],	[12467],	[12468],	[12478],
[12567],	[12568],	[12578],	[12678],	[13456],	[13457],	[13458],	[13467],
[13468],	[13478],	[13567],	[13568],	[13578],	[13678],	[14567],	[14568],
[14578],	[14678],	[15678],					
[23456],	[23457],	[23458],	[23467],	[23468],	[23478],	[23567],	[23568],
[23578],	[23678],	[24567],	[24568],	[24578],	[24678],	[25678],	
[34567],	[34568],	[34578],	[34678],	[35678],			
[45678]							

(2) Der erste noch nicht berechnete Minor besteht aus den Spalten [12567]. Setzt man in A die beiden nicht benötigten Spalten 3 und 4 gleich 0, so erhält man mit der wiederholten Backup-Prozedur alle Minoren $[125**]$, $[126**]$, $[156**]$, $[256**]$:

[12345], [12346], [12347], [12348], [12356], [12357], [12358], [12367],
[12368], [12378], [12456], [12457], [12458], [12467], [12468], [12478],
[12567], [12568], [12578], [12678], [13456], [13457], [13458], [13467],
[13468], [13478], [13567], [13568], [13578], [13678], [14567], [14568],
[14578], [14678], [15678],
[23456], [23457], [23458], [23467], [23468], [23478], [23567], [23568],
[23578], [23678], [24567], [24568], [24578], [24678], [25678],
[34567], [34568], [34578], [34678], [35678],
[45678]

(3) Der nächste noch nicht berechnete Minor besteht aus den Spalten [13567]. Setzt man in der ursprünglichen Matrix A die beiden Spalten 2 und 4 gleich 0, so erhält man die Minoren $[135**]$, $[136**]$, $[156**]$, $[356**]$:

[12345], [12346], [12347], [12348], [12356], [12357], [12358], [12367],
[12368], [12378], [12456], [12457], [12458], [12467], [12468], [12478],
[12567], [12568], [12578], [12678], [13456], [13457], [13458], [13467],
[13468], [13478], [13567], [13568], [13578], [13678], [14567], [14568],
[14578], [14678], [15678],
[23456], [23457], [23458], [23467], [23468], [23478], [23567], [23568],
[23578], [23678], [24567], [24568], [24578], [24678], [25678],
[34567], [34568], [34578], [34678], [35678],
[45678]

Hier tritt offensichtlich eine Überschneidung mit (2) auf, denn die maximalen Minoren $[156**]$ kennt man schon. Möchte man sie nicht doppelt berechnen, bleiben zwei Möglichkeiten. Entweder man stoppt die Backup-Prozedur rechtzeitig und verzichtet darauf, die beiden obersten Zeilen fertig zu diagonalisieren. Dann entfällt zwar die Berechnung von $[156**]$, aber man verzichtet auch darauf, $[356**]$ zu ermitteln. Letzteres kann man dann allerdings mit linearen Relationen erhalten, vgl. Bemerkung 21.8a). Oder man diagonalisiert mit der Backup-Prozedur ganz normal die drei untersten Zeilen, holt dann die bereits berechneten $[156**]$ mit dem richtigen Vorzeichen aus dem Speicher in die zweite Zeile und berechnet wie üblich die erste Zeile, also die Minoren $[356**]$, mit der Backup-Formel.

(4) Als nächstes benötigt man [14567]. Setzt man in der Matrix A die beiden Spalten 2 und 3 gleich 0, so erhält man die Minoren $[145**]$, $[146**]$, $[156**]$, $[456**]$:

[12345], [12346], [12347], [12348], [12356], [12357], [12358], [12367],
[12368], [12378], [12456], [12457], [12458], [12467], [12468], [12478],
[12567], [12568], [12578], [12678], [13456], [13457], [13458], [13467],
[13468], [13478], [13567], [13568], [13578], [13678], [14567], [14568],
[14578], [14678], [15678],
[23456], [23457], [23458], [23467], [23468], [23478], [23567], [23568],
[23578], [23678], [24567], [24568], [24578], [24678], [25678],
[34567], [34568], [34578], [34678], [35678],
[45678]

Wie schon in Schritt (3) ist $[156**]$ bereits bekannt. Insgesamt sind nun alle Minoren ermittelt, die die Spalte 1 beinhalten.

d) (*Sechszeilige Matrizen*) Es kann noch schlimmer kommen: sei A eine 6×9-Matrix, deren $\binom{9}{6} = 84$ maximale Minoren nicht verschwinden sollen.

(1) Aus A bzw. B erhält man mit der 3-maligen Backup-Prozedur zunächst einmal alle Minoren, die vier der fünf Spalten 1, 2, 3, 4, 5 enthalten, also die unterstrichenen Minoren der Form $[1234**]$, $[1235**]$, $[1245**]$, $[1345**]$, $[2345**]$:

[123456],	[123457],	[123458],	[123459],	[123467],	[123468],	[123469],
[123478],	[123479],	[123489],	[123567],	[123568],	[123569],	[123578],
[123579],	[123589],	[123678],	[123679],	[123689],	[123789],	[124567],
[124568],	[124569],	[124578],	[124579],	[124589],	[124678],	[124679],
[124689],	[124789],	[125678],	[125679],	[125689],	[125789],	[126789],
[134567],	[134568],	[134569],	[134578],	[134579],	[134589],	[134678],
[134679],	[134689],	[134789],	[135678],	[135679],	[135689],	[135789],
[136789],	[145678],	[145679],	[145689],	[145789],	[146789],	[156789],
[234567],	[234568],	[234569],	[234578],	[234579],	[234589],	[234678],
[234679],	[234689],	[234789],	[235678],	[235679],	[235689],	[235789],
[236789],	[245678],	[245679],	[245689],	[245789],	[246789],	[256789],
[345678],	[345679],	[345689],	[345789],	[346789],	[356789],	
[456789]						

(2) Um $[123678]$ zu berechnen, setzt man in A die Spalten 4 und 5 gleich 0 und erhält $[1236**]$, $[1237**]$, $[1267**]$, $[1367**]$, $[2367**]$. Übernimmt man aus Schritt (1) nur die noch nicht berechneten Minoren, so folgt also

[123678],	[123679],	[123689],	[123789],	[124678],	[124679],	[124689],
[124789],	[125678],	[125679],	[125689],	[125789],	[126789],	[134678],
[134679],	[134689],	[134789],	[135678],	[135679],	[135689],	[135789],
[136789],	[145678],	[145679],	[145689],	[145789],	[146789],	[156789],
[234678],	[234679],	[234689],	[234789],	[235678],	[235679],	[235689],
[235789],	[236789],	[245678],	[245679],	[245689],	[245789],	[246789],
[256789],						
[345678],	[345679],	[345689],	[345789],	[346789],	[356789],	
[456789]						

(3) Um $[124678]$ zu berechnen, setzt man in der ursprünglichen Matrix A die Spalten 3 und 5 gleich 0 und erhält $[1246**]$, $[1247**]$, $[1267**]$, $[1467**]$, $[2467**]$. Übernimmt man aus Schritt (2) nur die noch nicht berechneten Minoren, so folgt

[124678],	[124679],	[124689],	[124789],	[125678],	[125679],	[125689],
[125789],	[134678],	[134679],	[134689],	[134789],	[135678],	[135679],
[135689],	[135789],	[145678],	[145679],	[145689],	[145789],	[146789],
[156789],						
[234678],	[234679],	[234689],	[234789],	[235678],	[235679],	[235689],
[235789],	[245678],	[245679],	[245689],	[245789],	[246789],	[256789],
[345678],	[345679],	[345689],	[345789],	[346789],	[356789],	
[456789]						

Den Minor $[1267**] = [126789]$, der in der zugehörigen Diagonalmatrix in der dritten Zeile zu finden ist, hat man bereits in Schritt (2) ermittelt. Möchte man

diese Doppelberechnung vermeiden, kann man entweder die Backup-Prozedur an dieser Stelle abbrechen. In diesem Falle muß man $[1467**] = [146789]$ und $[2467**] = [246789]$ bei einer späteren Gelegenheit berechnen. Oder man holt sich den bereits berechneten Minor $[1267**] = [126789]$ mit dem passenden Vorzeichen aus dem Speicher.

(4) Als nächstes wird [125678] anvisiert. Dazu setzt man in der ursprünglichen Matrix A die beiden nicht benötigten Spalten 3 und 4 gleich 0 und erhält dann $[1256**], [1257**], [1267**], [1567**], [2567**]$:

[125678], [125679], [125689], [125789], [134678], [134679], [134689],
[134789], [135678], [135679], [135689], [135789], [145678], [145679],
[145689], [145789], [156789],
[234678], [234679], [234689], [234789], [235678], [235679], [235689],
[235789], [245678], [245679], [245689], [245789], [256789],
[345678], [345679], [345689], [345789], [346789], [356789],
[456789]

Den Minor $[1267**] = [126789]$, der sich in der zugehörigen Diagonalmatrix in der dritten Zeile befindet, hat man bereits in Schritt (2) bzw. (3) ermittelt.

(5) Nun soll [134678] berechnet werden, indem man in A die nicht benötigten Spalten 2 und 5 durch 0 ersetzt. Es ergeben sich dann die unterstrichenen Minoren $[1346**]$, $[1347**]$, $[1367**]$, $[1467**]$, $[3467**]$:

[134678], [134679], [134689], [134789], [135678], [135679], [135689],
[135789], [145678], [145679], [145689], [145789], [156789],
[234678], [234679], [234689], [234789], [235678], [235679], [235689],
[235789], [245678], [245679], [245689], [245789],
[345678], [345679], [345689], [345789], [346789], [356789],
[456789]

Der Minor $[1367**] = [136789]$, der in der zugehörigen Diagonalmatrix in der dritten Zeile steht, ist bereits in Schritt (2) ermittelt worden, und den Minor $[1467**] = [146789]$, der in der zugehörigen Diagonalmatrix in der zweiten Zeile zu finden ist, kennt man aus Schritt (3).

(6) Es wird als nächstes [135678] ermittelt. Zu diesem Zweck ersetzt man in der Matrix A die beiden Spalten 2 und 4 durch 0. Es ergeben sich dann die Minoren $[1356**]$, $[1357**]$, $[1367**]$, $[1567**]$, $[3567**]$:

[135678], [135679], [135689], [135789], [145678], [145679], [145689],
[145789],
[234678], [234679], [234689], [234789], [235678], [235679], [235689],
[235789], [245678], [245679], [245689], [245789],
[345678], [345679], [345689], [345789], [356789],
[456789]

Den Minor $[1367**] = [136789]$, der in der zugehörigen Diagonalmatrix in der dritten Zeile zu finden ist, hat man bereits in Schritt (2) bzw. (5) berechnet, und den Minor $[1567**] = [156789]$, der in der zugehörigen Diagonalmatrix in der zweiten Zeile steht, kennt man schon aus Schritt (4).

(7) Zuletzt wird [145678] gesucht. Daher ersetzt man in A die Spalten 2, 3 durch 0. Es ergeben sich dann die Minoren [1456∗∗], [1457∗∗], [1467∗∗], [1567∗∗], [4567∗∗]:

[145678], [145679], [145689], [145789],
[234678], [234679], [234689], [234789], [235678], [235679], [235689],
[235789], [245678], [245679], [245689], [245789],
[345678], [345679], [345689], [345789], [356789],
[456789]

Den Minor $[1467**] = [146789]$, der in der zugehörigen Diagonalmatrix in der dritten Zeile steht, hat man bereits in Schritt (3) bzw. (5) ermittelt, und den Minor $[1567**] = [156789]$, der in der zugehörigen Diagonalmatrix in der zweiten Zeile steht, kennt man schon aus Schritt (4) bzw. (6). Damit sind nun alle maximalen Minoren von A gefunden, die die erste Spalte enthalten. Entweder man ermittelt die noch fehlenden nicht unterstrichenen Minoren nun mit Hilfe linearer Relationen, oder man fährt an geeigneten Matrizen mit der wiederholten Backup-Prozedur fort. Während man in Schritt (1) allerdings noch sehr viele Minoren auf einen Schlag hat berechnen können, wird es zum Schluß hin zusehends mühsamer, die vereinzelt fehlenden Minoren zu berechnen.

21.10 Bemerkungen (Doppelberechnungen)

a) (*Backup-Prozedur*) Wie man im vorangegangenen Beispiel 21.9 sieht, kann bei $n \times m$-Matrizen mit $4 \leq n \leq m$ der Fall eintreten, daß bei der Backup-Prozedur mehrmals dieselben maximalen Minoren auftreten. Um eine Doppelberechnung zu vermeiden, kann man entweder die Backup-Prozedur rechtzeitig stoppen und die dadurch entgehenden Minoren bei einer späteren Gelegenheit nachholen. Dabei können keine Minoren unter den Tisch fallen, denn man hat stets im Auge, welche Minoren noch fehlen. Oder man holt sich für die Backup-Prozedur die bereits berechneten Minoren mit dem richtigen Vorzeichen in die Zeilen zurück.

b) (*Trigonalisierung*) Sei A eine $n \times m$-Matrix mit $n \leq m$ und $\operatorname{rang} A = n$. Trigonalisiert man A mit dem Bareiss-1-Schritt-Verfahren, so erhält man (bis auf Zeilentausch) bekanntlich die Matrizenfolge $(B^{(k)})_{k=1\ldots n-1}$, wobei

$$B^{(k)} = \begin{pmatrix} (\alpha_{ij}^{(i-1)})_{i=1\ldots k} \\ \\ (\alpha_{ij}^{(k)})_{i=k+1\ldots n} \end{pmatrix}$$

eine Matrix ist, in der unterhalb der Diagonalen die ersten k Zeilenstufenspalten $j_1, j_2, \ldots, j_k$ und somit die Spalten $1, \ldots, j_{k+1} - 1$ ausgeräumt sind. Es bezeichne $B := B^{(n-1)}$ die zu A äquivalente Zeilenstufenmatrix.

(1) Denkt man zurück an Beispiel 21.9d), so ist A dort eine 6×9-Matrix, die keine verschwindenden Minoren besitzt und zur Vereinfachung ohne Zeilentausch auskommen soll. In Schritt (1) wird zur Trigonalisierung die Matrizenfolge $(B^{(k)})_{k=1\ldots 5}$ erzeugt. In Schritt (2) werden dann in A die beiden Spalten 3 und 4 durch 0 ersetzt. Die entstehende Matrix werde mit $\widetilde{A}$ bezeichnet. Auch diese Matrix muß trigonalisiert werden, wobei man die Matrizenfolge $(\widetilde{B}^{(k)})_{k=1\ldots 5}$ konstruiert.

Ersetzt man in

$$B^{(1)} = \begin{pmatrix} \alpha_{11}^{(0)} & \alpha_{12}^{(0)} & \alpha_{13}^{(0)} & \alpha_{14}^{(0)} & \alpha_{15}^{(0)} & \cdots & \cdots & \cdots & \alpha_{19}^{(0)} \\ 0 & \alpha_{22}^{(1)} & \alpha_{23}^{(1)} & \alpha_{24}^{(1)} & \alpha_{25}^{(1)} & \cdots & \cdots & \cdots & \alpha_{29}^{(1)} \\ 0 & \alpha_{32}^{(1)} & \alpha_{33}^{(1)} & \alpha_{34}^{(1)} & \alpha_{35}^{(1)} & \cdots & \cdots & \cdots & \alpha_{39}^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \ddots & & \vdots \\ 0 & \alpha_{62}^{(1)} & \alpha_{63}^{(1)} & \alpha_{64}^{(1)} & \alpha_{65}^{(1)} & \cdots & \cdots & \cdots & \alpha_{69}^{(1)} \end{pmatrix}$$

und

$$B^{(2)} = \begin{pmatrix} \alpha_{11}^{(0)} & \alpha_{12}^{(0)} & \alpha_{13}^{(0)} & \alpha_{14}^{(0)} & \alpha_{15}^{(0)} & \cdots & \cdots & \cdots & \alpha_{19}^{(0)} \\ 0 & \alpha_{22}^{(1)} & \alpha_{23}^{(1)} & \alpha_{24}^{(1)} & \alpha_{25}^{(1)} & \cdots & \cdots & \cdots & \alpha_{29}^{(1)} \\ 0 & 0 & \alpha_{33}^{(2)} & \alpha_{34}^{(2)} & \alpha_{35}^{(2)} & \cdots & \cdots & \cdots & \alpha_{39}^{(2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \ddots & & \vdots \\ 0 & 0 & \alpha_{63}^{(2)} & \alpha_{64}^{(2)} & \alpha_{65}^{(2)} & \cdots & \cdots & \cdots & \alpha_{69}^{(2)} \end{pmatrix}$$

die dritte und vierte Spalte durch 0, erhält man offensichtlich die Matrizen

$$\widetilde{B}^{(1)} = \begin{pmatrix} \alpha_{11}^{(0)} & \alpha_{12}^{(0)} & 0 & 0 & \alpha_{15}^{(0)} & \cdots & \cdots & \cdots & \alpha_{19}^{(0)} \\ 0 & \alpha_{22}^{(1)} & 0 & 0 & \alpha_{25}^{(1)} & \cdots & \cdots & \cdots & \alpha_{29}^{(1)} \\ 0 & \alpha_{32}^{(1)} & 0 & 0 & \alpha_{35}^{(1)} & \cdots & \cdots & \cdots & \alpha_{39}^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \ddots & & \vdots \\ 0 & \alpha_{62}^{(1)} & 0 & 0 & \alpha_{65}^{(1)} & \cdots & \cdots & \cdots & \alpha_{69}^{(1)} \end{pmatrix}$$

und

$$\widetilde{B}^{(2)} = \begin{pmatrix} \alpha_{11}^{(0)} & \alpha_{12}^{(0)} & 0 & 0 & \alpha_{15}^{(0)} & \cdots & \cdots & \cdots & \alpha_{19}^{(0)} \\ 0 & \alpha_{22}^{(1)} & 0 & 0 & \alpha_{25}^{(1)} & \cdots & \cdots & \cdots & \alpha_{29}^{(1)} \\ 0 & 0 & 0 & 0 & \alpha_{35}^{(2)} & \cdots & \cdots & \cdots & \alpha_{39}^{(2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & 0 & \alpha_{65}^{(2)} & \cdots & \cdots & \cdots & \alpha_{69}^{(2)} \end{pmatrix},$$

da es bei diesen Matrizen nur darum geht, die ersten beiden Spalten auszuräumen. Für die Matrizen $B^{(k)}$ mit $3 \leq k \leq 5$ gilt diese Beziehung nicht mehr, denn hier sollen die Spalten 3, 4 und 5 ausgeräumt werden, während bei den Matrizen $\widetilde{B}^{(k)}$

mit $3 \leq k \leq 5$ die Spalten 5, 6 und 7 eliminiert werden müssen. Hätte man sich die Matrizenfolge $(B^{(k)})$ gemerkt, hätte man sich also für die Folge $(\widetilde{B}^{(k)})$ die Berechnung der ersten Glieder sparen können. Der Einsparungseffekt geht aber offenbar auf Kosten des Speicherplatzes.

(2) Allgemein liege nun die Matrizenfolge $(B^{(k)})_{k=1\ldots n-1}$ vor, wobei die Zeilenstufenindizes $j_1, j_2, \ldots, j_{n-1}$ zugrunde liegen und wobei mit der wiederholten Backup-Prozedur bereits eine Reihe maximaler Minoren berechnet worden ist. Möchte man nun den in der Menge L_n^m nächsten noch unbekannten maximalen Minor $[l_1\ l_2 \ldots l_n](A)$ berechnen, dann ersetzt man alle nicht benötigten Spalten aus $\{1, \ldots, l_n\} \setminus \{l_1, l_2, \ldots, l_n\}$ durch 0 und trigonalisiert die so entstandene Matrix $\widetilde{A}$, indem man entsprechend die Matrizenfolge $(\widetilde{B}^{(k)})_{k=1\ldots n-1}$ konstruiert. Sucht man das größte $\nu \in \{1, 2, \ldots, n\}$, so daß $j_i = l_i$ für $i = 1, \ldots \nu$ gilt und setzt man in den Matrizen $B^{(k)}$, $1 \leq k \leq \nu$, dieselben Spalten wie in $\widetilde{A}$ gleich 0, so erhält man offensichtlich die Matrizen $\widetilde{B}^{(k)}$, $1 \leq k \leq \nu$. Man muß folglich nur noch die Matrizen $\widetilde{B}^{(k)}$, $\nu+1 \leq k \leq n-1$, neu berechnen, die man dann wiederum in den Speicher legen kann. Die Matrizen $B^{(k)}$ mit $\nu + 1 \leq k \leq n-1$ darf man dafür aus dem Speicher werfen, denn da $[j_1\ l_2 \ldots l_n]$ aus der lexikographisch geordneten Menge L_n^m den ersten noch nicht berechneten Minor bezeichnet, sind schon alle Minoren berechnet worden, für die man eine dieser Matrizen gebrauchen könnte.

c) Bei einer Vorgehensweise wie in a) kann man zwischen der Belegung von Speicherplatz und der Vermeidung von Doppelberechnungen abwägen. So braucht man sich die Matrizen $B^{(k)}$ mit relativ kleinen k nicht unbedingt zu merken, denn sie enthalten nur Determinanten niedriger Ordnung, die man sich in geringfügiger Rechenzeit wieder beschaffen kann. Für sehr große Matrizen ist es dagegen durchaus vorteilhaft, sich bestimmte letzte Glieder solcher Folgen $(B^{(k)})$ zu merken.
Ist A beispielsweise eine 15×20-Matrix, wobei einfachheitshalber keiner ihrer maximalen Minoren verschwinden soll, so erzeugt man mit der ersten Trigonalisierung (bis auf Zeilentausch) die Matrizenfolge $(B^{(k)})_{k=0\ldots 14}$ und erhält dann aus $B^{(14)}$ mit der wiederholten Backup-Prozedur alle Minoren mit den Spalten $[1 \ldots \widehat{i} \ldots 14 * *]$, $i = 1, \ldots, 14$, wobei $*$ beliebige Spaltenindizes bezeichne. In lexikographischer Reihenfolge erhält man somit alle $[1 \ldots 13 * *]$, $[1 \ldots 12\ 14 * *]$, $[1 \ldots 11\ 13\ 14 * *]$ usw. Deprimierenderweise sind dies aber erst 231 von insgesamt 15.504 maximalen Minoren bzw. 216 von 11.628 maximalen Minoren mit der Spalte 1. Geht man wie üblich in lexikographischer Reihenfolge vor, so müssen als nächstes alle $[1 \ldots 12\ 15 * *]$ berechnet werden, wozu man die Spalten 13 und 14 in A durch 0 ersetzt und die entstehende Matrix $\widetilde{A}$ durch Konstruktion der Matrizenfolge $(\widetilde{B}^{(k)})_{k=0\ldots 14}$ trigonalisiert. Dabei kann man auf die Matrix $B^{(12)}$ zurückgreifen, in der die ersten 12 Spalten schon ausgeräumt sind. Ersetzt man in dieser Matrix die beiden Spalten 13 und 14 durch 0, so muß man anschließend nur noch die beiden Spalten 15 und 16 eliminieren, um die gewünschte Zeilenstufenmatrix $\widetilde{B}^{(14)}$ zu erhalten. Die Matrizen $B^{(13)}$ und $B^{(14)}$ werden nie mehr benötigt, denn es sind bereits alle $[1 \ldots 13 * *]$ bekannt.

Der Gedanke aus Bemerkung 21.8a), bereits berechnete Minoren für die Backup-Prozedur aus dem Speicher in die Matrix zurückzuholen, ist bei dem in Satz 22.2 beschriebenen Algorithmus `BackupMaxMinors` verwirklicht. Die anderen Vorschläge werden im folgenden nicht berücksichtigt.

22 Maximale Minoren

Sei A eine $n \times m$-Matrix mit $n \leq m$ über einem Integritätsring R. Um alle maximalen Mi-

noren von A zu berechnen, kann man dem vorangegangenen Abschnitt zufolge verschiedene Ansätze verfolgen:

a) Man wendet die wiederholte Backup-Prozedur solange auf geeignete Matrizen an, bis alle maximalen Minoren von A berechnet sind. Dies ist beim Algorithmus `MaxMinors` in Satz 22.2 verwirklicht.

b) Die Backup-Prozedur wird solange an geeigneten Matrizen wiederholt, bis alle maximalen Minoren von A berechnet sind, die die erste Zeilenstufenspalte j_1 enthalten. Die fehlenden Minoren ermittelt man anschließend mit linearen Relationen (Lemma 20.7a). Diese Vorgehensweise verfolgt der Algorithmus `MaxMinorsLR`, siehe Satz 22.5.

c) Speziell für $n \times m$-Matrizen mit $m = n + 1$ genügt es, diese ein einziges Mal mit einem (nahezu beliebigen) Divisionsverfahren zu diagonalisieren, weil man damit bereits sämtliche $(n + 1)$ maximalen Minoren gewinnt. Auf diese Art erhält man den Algorithmus `MaxMinors_2` aus Satz 22.9.

In den Fällen a) und b) müssen also zumindest die Minoren mit der Spalte j_1 möglichst geschickt berechnet werden, wobei einige technische Probleme zu bewältigen sind:

22.1 Bemerkungen (technische Probleme)

a) Die Verwaltung der Minoren wird so gehandhabt, daß eine Liste $L_1 = L_n^m$ die Klammerausdrücke in lexikographischer Ordnung enthält, und die gleich lange Liste L_2 die zugehörigen Minoren.

b) Die Liste $L_1 = L_n^m =$ `AllSubsets`$(\{1, \ldots, m\}, n)$ läßt sich mit dem Algorithmus `AllSubsets` aus Satz 20.4 bestimmen, und L_2 wird mit 0 vorbesetzt. Aus L_1 werden fortlaufend diejenigen Klammerausdrücke gestrichen, deren zugehörige Minoren ermittelt sind. Um diese in L_2 an die richtige Position zu bringen, steht der Algorithmus `PosMinor` aus Satz 20.3 zur Verfügung. Für die linearen Relationen ist es schließlich wichtig, daß die benötigten Minoren in L_2 genau lokalisierbar sind.

c) Verschwindende Minoren werden mit `FindZeroMaxMinors` aus Satz 21.5 aufgespürt, sonstige Minoren werden bei dem in Satz 22.2 beschriebenen Algorithmus `BackupMaxMinors` aus den jeweiligen Diagonalmatrizen mit dem richtigen Vorzeichen nach L_2 transportiert. Dieses Vorzeichen bestimmt man mit dem Algorithmus `Signum` aus Satz 20.5.

d) Bei der Ermittlung der Minoren ist zu beachten, daß man nicht fälschlich die durch 0 ersetzten Spalten miteinbezieht. Deren Spaltenindizes merkt man sich daher in der Menge Λ. Im übrigen könnte man, statt Spalten gleich 0 zu setzen, diese ebensogut streichen. Allerdings müßte man dann für `FindZeroMaxMinors` und `BackupMaxMinors` beachten, daß die vorhandenen Spalten in der Numerierung i.a. nicht mit denen der ursprünglichen Matrix übereinstimmen.

e) Der Vorzeichenwechsel bei einem Zeilentausch ist bekanntlich bereits in den Trigonalisierungsalgorithmen integriert, nämlich in den Zeilentauschalgorithmen `Pivot`, `COPivot` und `Mal1Pivot`.

f) Bemerkung 21.10 zeigt verschiedene Wege auf, wie man Doppelberechnungen vermeiden kann, wobei im folgenden nur einer dieser Vorschläge verwirklicht wird. Im Rahmen des Unteralgorithmus `BackupMaxMinors` werden dort bei der Diagonalisierung mit der Backup-Prozedur nur solche Koeffizienten berechnet, die noch nicht bekannt sind. Andernfalls werden sie mit dem richtigen Vorzeichen aus der Liste L_2 in die Matrix gebracht.

22.2 Satz (Algorithmus MaxMinors)
Sei A eine beliebige $n \times m$-Matrix über einem Integritätsring R. Der Algorithmus `AllSubsets` *sei wie in Satz 20.4,* `B2TMDMD` *wie in Satz 4.6,* `FindZeroMaxMinors` *wie in Satz 21.5 und* `BackupMaxMinors` *wie nachfolgend definiert, wobei die Algorithmen* `PosMinor` *und* `Signum` *wie in Satz 20.3 bzw. 20.5 seien.*

Betrachte die folgenden Instruktionen:

(1) Falls $n > m$ ist, ersetze A durch die transponierte Matrix ${}^tA$ und vertausche n und m.
(2) Erhalte mit `AllSubsets`$(\{1,\ldots,m\},n)$ *als Ergebnis die geordnete Menge $L_1 = L_n^m$. Setze $L_2 := \{0,\ldots,0\}$ für die geordnete Menge, die ebensoviele Elemente wie L_1 besitzt und nur aus 0 besteht. Setze $\Lambda := \emptyset$.*
(3) Falls L_1 leer ist, dann gib das Ergebnis L_2 aus. Andernfalls bezeichne $l = [l_1 \ldots l_n]$ das erste Element von L_1, und C sei die Matrix, die man aus A durch Nullsetzen aller Spalten $j \in \{1,\ldots,l_n\} \setminus \{l_1, l_2, \ldots, l_n\}$ erhält.
(4) Führe `B2TMDMD`(C) *durch und nenne das Ergebnis B.*
(5) Falls Λ leer ist und $\operatorname{rang} B < n$ gilt, ersetze L_1 durch die leere Menge und fahre fort bei Schritt (3).
(6) Führe `FindZeroMaxMinors`(B, L_1, Λ) *aus.*
(7) Bestimme die aufsteigend geordnete Indexmenge $H_n = \{j_{n1}, j_{n2}, \ldots, j_{n\rho}\}$ aller Spalten von B, die auf der Höhe n liegen. Falls H_n leer ist, fahre fort bei Schritt (3), andernfalls setze $\mu := 1$.
(8) Falls $\mu > \rho$ ist, fahre fort bei Schritt (3).
(9) Führe `BackupMaxMinors`(B, L_1, L_2, Λ) *durch, ersetze in B die $j_{n\mu}$-te Spalte durch 0, ersetze Λ durch $\Lambda \cup \{j_{n\mu}\}$, erhöhe μ um 1 und fahre fort bei Schritt (8).*

Dies ist ein mit `MaxMinors`(A) *bezeichneter Algorithmus, dessen Ergebnis die Menge der maximalen Minoren von A ist.*

Dabei werden unter der Bezeichnung `BackupMaxMinors`(B, L_1, L_2, Λ) *die folgenden Instruktionen zusammengefaßt:*

(B1) Setze $D := B$, ermittle die Zeilenstufen $j_1, j_2, \ldots, j_n$ von D, setze $z := [j_1\ j_2 \ldots j_n]$ und $k := n$.
(B2) Falls z in L_1 liegt, streiche es aus L_1 und ersetze in L_2 das Element an der Position `PosMinor`(z, m) *durch d_{nj_n}.*
(B3) Falls $k = 1$ gilt, gib als Ergebnis D, L_1 und L_2 aus.
(B4) Setze $J := \{1,\ldots,m\} \setminus (\Lambda \cup \{j_{k-1}, j_k, \ldots, j_n\})$.
(B5) Falls J leer ist, fahre fort bei Schritt (B10). Andernfalls wähle ein $j \in J$ und setze $l := [j_1 \ldots \widehat{j}_{k-1}\ j \ldots j_n]$.
(B6) Führe `Signum`(l) *durch und nenne das Ergebnis s. Ersetze l durch die aufsteigend geordnete Permutation von l. Führe* `PosMinor`(l, m) *durch und nenne das Ergebnis p.*
(B7) Liegt l in L_1, so ersetze den alten Koeffizienten $d_{k-1,j}$ zunächst durch $d_{nj_n} \cdot d_{k-1,j} - <(d_{k-1,j_k}, d_{k-1,j_{k+1}}, \ldots, d_{k-1,j_n}), (d_{kj}, d_{k+1,j}, \ldots, d_{nj})>$. Ersetze dann den neuen Koeffizienten $d_{k-1,j}$ durch $\frac{d_{k-1,j}}{d_{k-1,j_{k-1}}}$. Streiche l aus L_1, ersetze das p-te Element von L_2 durch $s \cdot d_{k-1,j}$ und fahre fort bei Schritt (B9).
(B8) Falls l nicht in L_1 liegt, dann ersetze $d_{k-1,j}$ durch das Produkt aus s und dem p-ten Element von L_2.
(B9) Streiche j aus J und fahre fort bei Schritt (B5).
(B10) Setze den Zeilenstufenkoeffizienten $d_{k-1,j_{k-1}} := d_{nj_n}$. Setze für $\lambda = k,\ldots,n$ jeweils $d_{k-1,j_\lambda} := 0$. Verkleinere k um 1 und fahre fort bei Schritt (B3).

Dies ist ein Algorithmus mit der Bezeichnung `BackupMaxMinors`(B, L_1, L_2, Λ), *der als Ergebnis die zu B äquivalente, mit der Backup-Prozedur hergestellte Diagonalmatrix D*

ausgibt und die Mengen L_1 und L_2 abändert. Einerseits werden aus L_1 solche Klammerausdrücke $[l_1 \dots l_n]$ gestrichen, deren zugehörige Minoren $[l_1 \dots l_n](A)$ in der Diagonalmatrix D als Zeilenstufenkoeffizient oder in den ersten $(n-1)$ Zeilen abgelesen werden können. Andererseits werden diese Minoren in L_2 an die entsprechende Position gebracht. Dabei enthält keiner dieser Minoren eine Spalte aus Λ.

Beweis:

Die für `MaxMinors` angegebenen Schritte können nur endlich oft durchlaufen werden, denn in den Schritten (5), (6) und (9) wird die endliche Menge L_1 solange verkleinert, bis sie leer ist. Außerdem läßt sich Schritt (9) nur endlich oft wiederholen, da die Menge H_n aus Schritt (7) endlich ist. Was `BackupMaxMinors` betrifft, so werden auch hier die angegebenen Schritte nur endlich oft repetiert, denn J kann in Schritt (B9) und k kann in Schritt (B10) nur endlich oft verkleinert werden, bis schließlich $J = \emptyset$ bzw. $k = 1$ gilt.

Bei dem Algorithmus `BackupMaxMinors` übernimmt man in Schritt (B8) für $k = n, \dots 2$ den Koeffizienten $d_{k-1,j}$ mit dem Vorzeichen s aus der Menge L_2, falls der entsprechende Minor bereits bekannt ist. Andernfalls wird der Koeffizient $d_{k-1,j}$ in Schritt (B7) mit der bekannten Backup-Rekursionsformel berechnet und mit dem Vorzeichen s an die richtige Position p in L_2 gebracht. Den zugehörigen Klammerausdruck entfernt man aus L_1. Somit werden L_1 und L_2 wie behauptet abgeändert. Mit Schritt (B4) wird dabei sichergestellt, daß Spalten aus Λ nicht berücksichtigt werden. (Zu den Schritten (B5) bis (B10) vergleiche auch Satz 7.5, insbesondere den Algorithmus `BackupRow`.)

Auch wenn bei `BackupMaxMinors` nur der Zeilenstufenkoeffizient und die geeigneten Koeffizienten der ersten $(n-1)$ Zeilen aus der Diagonalmatrix D in die Liste L_2 transportiert werden, so werden im Endeffekt trotzdem auch die für L_2 in Frage kommenden Koeffizienten der n-ten Zeile von D erfaßt. Diese Koeffizienten sind nämlich identisch mit den Koeffizienten aus der n-ten Zeile der Zeilenstufenmatrix B. Nimmt einer dieser Koeffizienten den Wert 0 an, ist er bereits mit `FindZeroMaxMinors` in Schritt (6) erwischt worden; andernfalls tritt er bei irgendeiner Wiederholung von Schritt (9) zwangsläufig als Zeilenstufenkoeffizient einer Diagonalmatrix auf, sobald alle seine Vorgänger durch 0 ersetzt sind.

Daß das Endergebnis L_2 von `MaxMinors` alle maximalen Minoren von A enthält, folgt für den Fall $3 \leq n, m$ sofort aus den vorangegangenen Betrachtungen (Abschnitt 20 und 21). Sind n oder m gleich 1 oder 2, so führen die angegebenen Instruktionen ebenfalls zum gewünschten Ergebnis, denn die wiederholte Backup-Prozedur ist auch dann wie in Abschnitt 31 beschrieben durchführbar mit dem Ergebnis, daß man nach der ersten (einzigen) Trigonalisierung mit anschließender wiederholter Diagonalisierung alle maximalen Minoren kennt. Ist speziell der Rang von A nicht maximal, so führt Schritt (5) sofort zum Endergebnis $L_2 = \{0, \dots, 0\}$. □

22.3 Bemerkungen (Komplexität von MaxMinors)

Sei A eine $n \times m$-Matrix mit $n \ll m$, deren maximale Minoren alle ungleich 0 seien.

a) (*Algorithmus* `B2TMDMD`) Nach Bemerkung 4.7 benötigt man mit `B2TMDMD` für eine $n \times m$-Matrix mit $n \leq m$

$\approx \frac{1}{2}mn^2 - \frac{1}{6}n^3$ Additionen

$\approx \frac{3}{4}mn^2 - \frac{1}{4}n^3$ Multiplikationen

$\approx \frac{1}{4}mn^2 - \frac{1}{12}n^3$ Divisionen.

b) (*wiederholte Backup-Prozedur*) Wiederholt man die Backup-Prozedur an einer $n \times m$-Matrix, so werden nacheinander die Spalten $n, n+1, \dots, m-1$ durch 0 ersetzt. Weil für die 0-Spalten keine Berechnungenen anfallen, ist dies gleichbedeutend damit, die

Backup-Prozedur der Reihe nach an $n \times m$-, $n \times (m-1)$-, ..., $n \times (n+1)$-Matrizen durchzuführen. Somit ergeben sich insgesamt für die wiederholte Backup-Prozedur

$\approx \frac{1}{4}m^2n^2 - \frac{1}{6}mn^3 - \frac{1}{12}n^4$ Additionen

$\approx \frac{1}{4}m^2n^2 - \frac{1}{6}mn^3 - \frac{1}{12}n^4$ Multiplikationen

$\approx \frac{1}{2}m^2n - \frac{1}{2}mn^2$ Divisionen,

vergleiche Bemerkung 7.6a) (Komplexität der Backup-Prozedur).

c) (*Algorithmus* `MaxMinors`) Wie oft muß man im Laufe des Algorithmus `MaxMinors` Matrizen welcher Größe trigonalisieren und anschließend mit der Backup-Prozedur wiederholt diagonalisieren? Sei $i = l_{n-2} \in \{n-2, \ldots, m-2\}$ beliebig und seien $l_1, \ldots, l_{n-3}$ beliebige Spaltenindizes aus $\{1, \ldots, m\}$ mit $l_1 < \ldots < l_{n-3} < l_{n-2}$. Um alle maximalen Minoren von A mit den Spalten $[l_1 \ldots l_{n-2} * *]$ zu erhalten, wobei $*$ beliebige Spaltenindizes symbolisiere, muß man in A $(i-n+2)$ Spalten durch 0 ersetzen, also eine $n \times (m+n-i-2)$-Matrix trigonalisieren und mit der Backup-Prozedur wiederholt diagonalisieren. Da für $i = n-2, \ldots, m-2$ jeweils $\binom{i-1}{n-3}$ verschiedene Vorgänger $l_1 < \ldots < l_{n-3}$ von $i = l_{n-2}$ existieren, wird daher schlimmstenfalls $\binom{i-1}{n-3}$-mal eine $n \times (m+n-i-2)$-Matrix trigonalisiert und mit der Backup-Prozedur wiederholt diagonalisiert, für $i = n-2, \ldots, m-2$. Mit den obigen Abschätzungen aus a) und b) und mit $\binom{m}{n} \approx m^n$ für große n, m mit $n \ll m$ ergeben sich folglich als Abschätzung für `MaxMinors`

$\approx O(m^n n^2)$ Additionen

$\approx O(m^n n^2)$ Multiplikationen

$\approx O(m^n n)$ Divisionen,

um alle $\binom{m}{n} \approx m^n$ maximale Minoren zu berechnen. Die bei den Unteralgorithmen (`PosMinor` etc.) anfallenden Rechenoperationen sind dabei nicht berücksichtigt.

22.4 Bemerkungen zur Implementierung (MaxMinors)

In Bemerkung 22.1 (technische Probleme) sind bereits einige Schwierigkeiten vorweggenommen, die man bei der Implementierung des Algorithmus `MaxMinors` aus Satz 22.2 beachten muß. Anzumerken ist noch:

a) Die Anpassung der Listen L_1 und L_2 erfolgt bei `FindZeroMaxMinors` und `BackupMaxMinors`, indem bei der Definition dieser beiden CoCoA-Funktionen die Argumente L_1 und L_2 mit `Var` deklariert und daher automatisch aktualisiert werden.

b) Dadurch, daß die Minorenliste L_2 mit 0 vorbesetzt ist, erspart man es sich bei dem Algorithmus `FindZeroMaxMinors`, in der Liste L_2 den verschwindenden Minoren eigens den Wert 0 zuzuweisen. Die Gefahr, daß beim Endergebnis L_2 irrtümlich Minoren den Wert 0 haben, besteht nicht, denn zur Kontrolle wird in der Liste L_1 Buch geführt, welche Minoren noch zu berechnen sind.

c) Der Algorithmus `BackupMaxMinors` ist so programmiert, daß analog zu früheren Algorithmen die Schritte (B4) bis (B10) zu einer eigenen Unterfunktion `BackupMaxMinorsRow`$(B, Z, k,$ `Var` $L_1,$ `Var` $L_2, \Lambda)$ zusammengefaßt sind, wobei $Z = \{j_1, j_2, \ldots, j_n\}$ gilt. Um Doppelberechnungen zu vermeiden, werden dabei nur solche Koeffizienten tatsächlich neu berechnet, die noch nicht aus L_2 bekannt sind, während die bekannten Koeffizienten aus L_2 übernommen werden (mit dem richtigen Vorzeichen.)

d) In Schritt (7) wird die Menge H_n mit der Funktion `Height`(B, n) berechnet, vgl. Bemerkung 21.6c (Implementierung von `FindZeroMaxMinors`).

e) Bei der wiederholten Backup-Prozedur trigonalisiert bzw. diagonalisiert man Matrizen, in denen eventuell bestimmte Spalten durch 0 ersetzt worden sind. Diese eigentlich überflüssigen Spalten haben aber keine Auswirkung auf die Rechenzeit. Ebensogut könnte man sie streichen und müßte sich dann aber merken, welche Spalten der ursprünglichen Matrix die verbleibenden Spalten tatsächlich bezeichnen.

22.5 Satz (Algorithmus MaxMinorsLR)
Sei A eine beliebige $n \times m$-Matrix über einem Integritätsring R. Der Algorithmus `AllSubsets` *sei wie in Satz 20.4,* `B2TMDMD` *wie in Satz 4.6,* `FindZeroMaxMinors` *wie in Satz 21.5,* `BackupMaxMinors` *wie in Satz 22.2 und* `LinRelMaxMinors` *wie nachfolgend definiert.*

Betrachte die folgenden Instruktionen:

(1) Falls $n > m$ ist, ersetze A durch die transponierte Matrix ${}^tA$ und vertausche n und m.
(2) Erhalte mit `AllSubsets`$(\{1,\ldots,m\},n)$ *als Ergebnis die geordnete Menge $L_1 = L_n^m$. Setze $L_2 := \{0,\ldots,0\}$ für die geordnete Menge, die ebensoviele Elemente wie L_1 besitzt und nur aus 0 besteht. Setze $\Lambda := \emptyset$ und $j_1 := 1$.*
(3) Falls L_1 leer ist, dann gib das Ergebnis L_2 aus. Andernfalls bezeichne $l = [l_1 \ldots l_n]$ das erste Element von L_1.
(4) Falls $l_1 > j_1$ gilt, fahre fort bei Schritt (12).
(5) Es sei C die Matrix, die man aus A durch Nullsetzen aller Spalten $j \in \{1,\ldots,l_n\} \setminus \{l_1, l_2, \ldots, l_n\}$ erhält. Führe `B2TMDMD`(C) *durch und nenne das Ergebnis B.*
(6) Falls Λ leer ist und $\operatorname{rang} B < n$ gilt, ersetze L_1 durch die leere Menge und fahre fort bei Schritt (3).
(7) Falls Λ leer ist, setze $P := b_1$ für die erste Zeile von B und ersetze j_1 durch den Spaltenindex, der den ersten Koeffizienten ungleich 0 in P anzeigt.
(8) Führe `FindZeroMaxMinors`(B, L_1, Λ) *aus.*
(9) Bestimme die aufsteigend geordnete Indexmenge $H_n = \{j_{n1}, j_{n2}, \ldots, j_{n\rho}\}$ aller Spalten von B, die auf der Höhe n liegen. Falls H_n leer ist, fahre fort bei Schritt (3), andernfalls setze $\mu := 1$.
(10) Falls $\mu > \rho$ ist, fahre fort bei Schritt (3).
(11) Führe `BackupMaxMinors`(B, L_1, L_2, Λ) *durch, ersetze in B die $j_{n\mu}$-te Spalte durch 0, ersetze Λ durch $\Lambda \cup \{j_{n\mu}\}$, erhöhe μ um 1 und fahre fort bei Schritt (10).*
(12) Führe für jedes $l \in L_1$ den Algorithmus `LinRelMaxMinors`(P, L_2, l) *durch und gib als Ergebnis L_2 aus.*

Dies ist ein mit `MaxMinorsLR`(A) *bezeichneter Algorithmus, dessen Ergebnis die Menge der maximalen Minoren von A ist.*

Dabei bedeute `LinRelMaxMinors`(P, L_2, l) *den folgenden Algorithmus, der den durch $l = [l_1 \ldots l_n]$ definierten maximalen Minor von A aus L_2 und aus $P = (p_1 \ldots p_m) = (0 \ldots 0\ p_{j_1} \ldots p_m)$ mit $p_{j_1} \neq 0$ berechnet und in der geordneten Minorenliste L_2 an die richtige Position schreibt. Hierfür sei der Algorithmus* `PosMinor` *wie in Satz 20.3.*

(P1) Für $\lambda = 1,\ldots,n$ setze $h_\lambda := [j_1\ l_1\ l_2 \ldots \widehat{l_\lambda} \ldots l_n]$.
(P2) Ersetze dann h_λ für $\lambda = 1,\ldots,n$ durch das Element, das in L_2 an der Position `PosMinor`(h_λ, m) *steht.*
(P3) Berechne zuerst $s := \sum_{\lambda=1}^{n} (-1)^{\lambda+1} \cdot p_{l_\lambda} \cdot h_\lambda$. Ersetze anschließend s durch s/p_{j_1} und gib als Ergebnis s, L_1 und L_2 aus.

Beweis:
Die angegebenen Schritten sind im wesentlichen mit denen von `MaxMinors` aus Satz 22.2 identisch. Zu beachten ist nur, daß Schritt (4) das Abbruchkriterium beinhaltet, um mit den linearen Relationen in Schritt (12) fortzufahren. Für diese verwendet man gemäß Bemerkung 21.8 die in Schritt (7) gefundene erste Zeile der allerersten Zeilenstufenmatrix B,

also $P = (p_1 \dots p_m) = (b_{11} \dots b_{1m})$, und alle Minoren, die die erste Zeilenstufenspalte j_1 enthalten. Somit ist der Algorithmus `LinRelMaxMinors` die direkte Anwendung der Formel

$$[l_1 \dots l_n](A) = \frac{1}{b_{1j_1}} \cdot \sum_{\lambda=1}^{n} (-1)^{\lambda+1} \cdot b_{1l_\lambda} \cdot [j_1\ l_1 \dots \widehat{l_\lambda} \dots l_n](A) =$$
$$= \frac{1}{p_{j_1}} \cdot \sum_{\lambda=1}^{n} (-1)^{\lambda+1} \cdot p_{l_\lambda} \cdot [j_1\ l_1 \dots \widehat{l_\lambda} \dots l_n](A).$$

□

22.6 Bemerkungen (Komplexität von MaxMinorsLR)
Sei A eine $n \times m$-Matrix mit $n \ll m$, deren maximale Minoren alle ungleich 0 seien.

a) (`B2TMDMD`, *wiederholte Backup-Prozedur*) Um mit Hilfe von `B2TMDMD` und der wiederholten Backup-Prozedur alle $\binom{m-1}{n-1}$ maximalen Minoren mit der Spalte j_1 zu ermitteln, benötigt man nach Bemerkung 22.3 (Komplexität von `MaxMinors`)

$\approx O(m^{n-1}n^2)$ Additionen

$\approx O(m^{n-1}n^2)$ Multiplikationen

$\approx O(m^{n-1}n)$ Divisionen.

b) (*Lineare Relationen*) Um einen maximalen Minor mittels einer geeigneten linearen Relation zu berechnen, benötigt man für den Algorithmus `LinRelMaxMinors`

$n-1$ Additionen

n Multiplikationen

1 Division.

Für $\binom{m-1}{n} \approx m^n$ maximale Minoren, die die Spalte j_1 nicht enthalten, ergeben sich somit insgesamt

$\approx O(m^n n)$ Additionen

$\approx O(m^n n)$ Multiplikationen

$\approx O(m^n)$ Divisionen.

c) (*Algorithmus* `MaxMinorsLR`) Mit den Abschätzungen aus a) und b) benötigt man folglich für `MaxMinorsLR`

$\approx O(m^n n)$ Additionen

$\approx O(m^n n)$ Multiplikationen

$\approx O(m^n)$ Divisionen,

um alle $\binom{m}{n} \approx m^n$ maximalen Minoren von A zu berechnen. Die bei den Unteralgorithmen (`PosMinor` etc.) anfallenden Rechenoperationen sind dabei nicht berücksichtigt.

22.7 Bemerkungen zur Implementierung (MaxMinorsLR)
In Ergänzung zu Bemerkung 22.1 (technische Probleme) und Bemerkung 22.4 (Implementierung von `MaxMinors`) bleibt zum Algorithmus `MaxMinorsLR` aus Satz 22.5 noch zu bemerken:

a) Die Anpassung der Liste L_2 erfolgt bei `LinRelMaxMinors` wieder dadurch, daß bei der Definition dieser CoCoA-Funktion das Argument L_2 mit `Var` deklariert ist und daher automatisch aktualisiert wird. Daneben gibt die Funktion `LinRelMaxMinors`(P, L_2, l) auch das Ergebnis s aus. Dies ist zwar nicht unbedingt nötig, da die Minorenliste L_2 diesen Wert schon an die richtige Position zugewiesen bekommen hat, aber auf diese

Art kann man den durch l definierten Minor auch explizit erhalten, wenn man dies wünscht.

b) Der Algorithmus `MaxMinorsLR` aus Satz 22.5 ist im wesentlichen genauso wie `MaxMinors` aus Satz 22.2 aufgebaut. Es muß nur mit Schritt (4) das Abbruchkriterium eingefügt werden, um dann in Schritt (12) mit der Funktion `LinRelMaxMinors`(P, L_2, l) die übrigen Minoren zu berechnen. Für die linearen Relationen muß man sich außerdem in Schritt (7) die erste Zeile der allerersten Zeilenstufenmatrix merken, also P, vgl. Bemerkung 21.8.

Insbesondere für $n \times m$-Matrizen $A \in \mathrm{Mat}_{n\times m}(R)$ mit $m = n+1$ genügt bereits eine einzige Diagonalisierung mit einem (nahezu beliebigen) Divisionsverfahren, um am Endergebnis $(\delta_{ij}^{(n)})$ alle maximalen Minoren von A ablesen zu können. Für diesen Spezialfall existiert ein eigener Algorithmus:

22.8 Satz (Algorithmus MaxMinors_2)
Sei A eine beliebige $n \times m$-Matrix aus $\mathrm{Mat}_{n\times m}(R)$ mit $m \in \{n+1, n-1\}$, ferner sei der Algorithmus `B2DMDMDFB` *wie in Satz 7.5. Der Algorithmus* `Update` *sei wie nachfolgend in a) definiert.*

a) Betrachte die folgenden Instruktionen:
(1) Falls $n > m$ ist, ersetze A durch die transponierte Matrix ${}^tA$ und vertausche n und m.
(2) Führe `B2DMDMDFB`(A) *aus und nenne das Ergebnis D. Setze $L := \{0, \ldots, 0\}$ für die geordnete $(n+1)$-elementige Menge, die nur aus 0 besteht.*
(3) Falls $\operatorname{rang} D = n$ gilt, führe `Update`(D, L) *durch.*
(4) Gib als Ergebnis L aus.
Dies ist ein Algorithmus mit der Bezeichnung `MaxMinors_2`(A)*, der als Ergebnis die Menge aller maximalen Minoren von A ausgibt.*

Dabei bezeichne `Update`(D, L) *den folgenden Algorithmus, der die Minorenliste L mit Koeffizienten aus der $n \times m$-Diagonalmatrix D aktualisiert. Die Algorithmen* `PosMinor` *und* `Signum` *seien wie in Satz 20.3 bzw. 20.5.*
(U1) Ermittle die Zeilenstufen $j_1, j_2, \ldots, j_n$ von D.
(U2) Ersetze für $l := [j_1\ j_2 \ldots j_n]$ das Element von L an der Position `PosMinor`(l, m) *durch den Koeffizienten d_{1j_1}. Setze $i := 1$.*
(U3) Falls $i > n$ gilt, gib als Ergebnis L aus.
(U4) Setze für alle Spaltenindizes $j \in \{1, \ldots, m\} \setminus \{j_1, j_2, \ldots, j_n\}$ mit $j > j_i$ jeweils $h_j := [j_1\ j_2 \ldots \widehat{j_i}\ j \ldots j_n]$ und $\widetilde{h}_j$ für die aufsteigend sortierte Permutation von h_j mit $\widetilde{h}_1 < \ldots < \widetilde{h}_n$. Ersetze in L das Element an Position `PosMinor`$(\widetilde{h}_j, m)$ *durch* `Signum`$(h_j) \cdot d_{ij}$.
(U5) Erhöhe i um 1 und fahre fort bei Schritt (U3).

b) Aussage a) gilt auch dann, wenn man in Schritt (2) den Algorithmus `B2DMDMDFB` *durch einen der Algorithmen* `B1DMD` *(Satz 5.6),* `B2DMD` *(Satz 6.3),* `B2DMDMD` *(Satz 6.6),* `B1DMDFB`*,* `B2DMDFB` *(Satz 7.5) oder* `Mal1DMD` *(Satz 8.4) ersetzt.*

c) Die $n \times m$-Matrix A erfülle nun die zusätzlichen Voraussetzungen, daß $m = n+1$ sei, daß ihr Rang gleich n und daß Zeilentausch nicht nötig sei. Die obige Aussage a) ist weiterhin gültig, wenn man dort in Schritt (2) statt des Algorithmus `B2DMDMDFB` *einen der Algorithmen* `MalDichHR` *(Satz 10.2),* `MalDichHM` *(Satz 10.4),* `MalDichVR` *(Satz 10.6),* `MalDichVM` *(Satz 10.8),* `B1DMDSM` *(Satz 14.2),* `B2DMDMDSM` *(Satz 15.2),* `B1DMDSMFB`*,* `B2DMDMDSMFB` *(Satz 16.1),* `B1DMDFBSM`*,* `B2DMDMDFBSM` *(Satz 16.3) oder* `Mal1DMDSM` *(Satz 17.2) verwendet.*

Beweis:
Offensichtlich können die für `MaxMinors_2` angegebenen Instruktionen nur endlich oft durchlaufen werden, da `Update` und da die zugrundeliegenden Diagonalisierungsverfahren nur aus endlich vielen Schritten bestehen.

Sei o.E. $m = n+1$. Diagonalisiert man A mit einem der in a), b) oder c) angegebenen Verfahren, so erhält man die Diagonalmatrix $D = (\delta_{ij}^{(n)})$. Gilt $\operatorname{rang} A = \operatorname{rang} D < n$, so ist $L = \{0, \ldots, 0\}$ die Menge aller maximalen Minoren von A, andernfalls enthält die Diagonalmatrix D alle maximalen Minoren von A, nämlich die Minoren

$$(\operatorname{sign}[j_1 \ldots \widehat{j_i}\ j \ldots j_n]) \cdot [j_1 \ldots \widehat{j_i}\ j \ldots j_n](A) = \operatorname{sign} \delta_{ij}^{(n)} \cdot \delta_{ij}^{(n)} = \operatorname{sign} \delta_{ij}^{(n)} \cdot d_{ij}^{(n)}$$

für $i \in \{1, \ldots, n\}$ und $j \in \{1, \ldots, m\}$ mit $j \geq j_i$ und $j \neq j_{i+1}, j_{i+2}, \ldots, j_n$. Diese Minoren werden mit `Update` aus der Diagonalmatrix nach L gebracht. □

22.9 Bemerkungen zur Implementierung (MaxMinors_2)

a) (*Optionen*) Der Algorithmus `MaxMinors_2` aus Satz 22.8 ist so programmiert, daß als Voreinstellung, wie in Satz 22.8a) beschrieben, der Algorithmus `B2DMDMDFB` verwendet wird. Optional läßt sich stattdessen ein beliebiger anderer Algorithmus auswählen, der als zusätzliches Argument eingegeben wird, siehe 22.8b) und c). Eine gute Alternative bietet oft der Algorithmus `B2DMDMDFBSM`. In speziellen Fällen kann auch ein anderes Diagonalisierungsverfahren die schnellsten Ergebnisse erzielen, z.B. `MalDichHM`. (Dichotomie- oder Sasaki-Murao-Verfahren dürfen aber nur unter bestimmten Voraussetzungen verwendet werden, vgl. 22.8c.)

b) `MaxMinors_2` gilt ausdrücklich nur für $n \times (n+1)$- oder $n \times (n-1)$-Matrizen. Verwendet man dabei den Algorithmus `B2DMDMDFB`, so ist `MaxMinors_2` praktisch identisch mit den Algorithmen `MaxMinors` und `MaxMinorsLR`, denn es muß in jedem Fall nur eine einzige Diagonalisierung (mit `B2DMDMDFB`) vorgenommen werden, um sämtliche maximalen Minoren zu erhalten.

c) (*Algorithmus* `Update`) Die Anpassung der Liste L erfolgt bei `Update`(D, L) wie üblich dadurch, daß das Argument L mit `Var` deklariert wird.

Neben den bisher vorgeschlagenen Methoden zur Berechnung maximaler Minoren gibt es sicherlich noch andere, die hier aber nicht mehr weiter ausgeführt werden sollen, beispielsweise die folgende:

22.10 Bemerkung (Entwicklung der maximalen Minoren)
Sei A eine $n \times m$-Matrix mit $n \leq m$ über einem Integritätsring R.

a) Kennt man alle Minoren $(n-1)$-ter Ordnung in den Zeilen $1, \ldots, n-1$, also alle Minoren $[1 \ldots n-1][l_1 \ldots l_{n-1}](A)$ mit beliebigen $l_1, \ldots, l_{n-1} \in \{1, \ldots, m\}$, so kann man daraus sofort alle Minoren n-ter Ordnung berechnen, indem man sie nach der letzten Zeile entwickelt.
Wie erhält man all diese Minoren $[1 \ldots n-1][l_1 \ldots l_{n-1}](A)$? Entweder man berechnet sie rekursiv aus den Minoren $(n-2)$-ter Ordnung in den Zeilen $1, \ldots, n-2$, oder mit einem anderen Verfahren...

b) Entwickelt man einen maximalen Minor nach der n-ten Zeile, so fallen dabei $(n-1)$ Additionen und n Multiplikationen an, wobei der erste Faktor ein Koeffizient von A und der zweite Faktor eine Determinante $(n-1)$-ter Ordnung ist. Berechnet man denselben Minor mit Hilfe einer geeigneten linearen Relation, so benötigt man dazu ebenfalls $(n-1)$ Additionen und n Multiplikationen, darüberhinaus aber auch eine Division. Bei den Multiplikationen ist stets ein Faktor ein Koeffizient von A und der andere eine Determinante n-ter Ordnung.

23 Vergleich der Verfahren (maximale Minoren)

Den Komplexitäten zufolge sollte unter den selbstgeschriebenen Funktionen der Algorithmus `MaxMinorsLR` (Satz 22.5) mit den linearen Relationen besser abschneiden als `MaxMinors` (Satz 22.2), siehe die Tabelle unten. In speziellen Fällen von $n \times (n+1)$-bzw. $n \times (n-1)$-Matrizen kann aber `MaxMinors_2` (Satz 22.8) in Verbindung mit einem geeigneten Diagonalisierungsverfahren günstiger sein, vergleiche Kapitel IV.

Maximale Minoren von n × m-Matrizen mit n ≤ m				
Quellen	Algorithmus	Additionen	Multiplikationen	Divisionen
22.2, 22.3	`MaxMinors`	$O(m^n n^2)$	$O(m^n n^2)$	$O(m^n n)$
22.5, 22.6	`MaxMinorsLR`	$O(m^n n)$	$O(m^n n)$	$O(m^n)$

Tabelle 23

23.1 Zusammenfassung (Berechnung maximaler Minoren)

a) (*Testsieger*) Unter den selbstgeschriebenen Funktionen ist `MaxMinorsLR` umso günstiger als `MaxMinors`, je größer der Unterschied zwischen n und m ist ($n \leq m$, ansonsten betrachtet man die transponierte Matrix). In speziellen Fällen ist `MaxMinors_2` in Verbindung mit einem geeigneten Diagonalisierungsverfahren empfehlenswert, vergleiche Kapitel IV.

b) (*Optimierungen*) Die selbstgeschriebenen Funktionen ließen sich noch weiter optimieren, wenn man mehr Zwischenergebnisse speichern würde, siehe Bemerkung 21.10b) und c) (Doppelberechnungen). Dazu müßte man sich die Matrizenfolgen, die bei den jeweils nötigen Trigonalisierungen anfallen, bzw. bestimmte Matrizen davon merken.

24 Beliebige Minoren

Sei im folgenden $k \in \mathbb{N}_+$, und A bezeichne eine beliebige $n \times m$-Matrix über einem Integritätsring R. Wie kann man für diese Matrix möglichst effektiv alle $\binom{m}{k} \cdot \binom{n}{k}$ Minoren k-ter Ordnung mit $1 \leq k \leq n, m$ bestimmen?

Eine Möglichkeit besteht offensichtlich darin, jeweils k paarweise verschiedene Zeilen $i_1 < \ldots < i_k$ aus A auszuwählen und dann mit den bereits bekannten Verfahren aus Abschnitt 32 alle maximalen Minoren in diesen k Zeilen zu bestimmen, also alle Minoren $[i_1 \ldots i_k][l_1 \ldots l_k](A)$. Führt man dies für alle $[i_1 \ldots i_k] \in L_k^n$ durch, so erhält man schließlich sämtliche Minoren k-ter Ordnung.

24.1 Satz (Algorithmen MyMinors, MyMinorsLR)
Sei A eine beliebige $n \times m$-Matrix über R, und sei $k \in \mathbb{N}_+$ mit $1 \leq k \leq n, m$. Der Algorithmus `AllSubsets` *sei wie in Satz 20.4, die Algorithmen* `MaxMinors` *und* `MaxMinorsLR` *wie in Satz 22.2 bzw. 22.5.*

a) (Algorithmus `MyMinors`*) Betrachte die folgenden Instruktionen:*
(1) Führe `AllSubsets`$(\{1, \ldots, n\}, k)$ *durch und nenne das Ergebnis L_1. Setze $L_2 := \emptyset$.*
(2) Falls $L_1 = \emptyset$ gilt, dann gib das Ergebnis L_2 aus. Andernfalls bestimme für das erste Element $[i_1 \ldots i_k] \in L_1$ die Teilmatrix B von A in den Zeilen $i_1, \ldots, i_k$ und den Spalten $1, \ldots, m$.

(3) Erhalte mit `MaxMinors`(B) *das Ergebnis* M *und ersetze* L_2 *durch* $L_2 \cup M$. *Streiche* $[i_1 \dots i_k]$ *aus* L_1 *und fahre fort bei Schritt (2).*

Dies ist ein Algorithmus mit der Bezeichnung `MyMinors`(k, A), *der alle Minoren* k*-ter Ordnung von* A *berechnet.*

b) (Algorithmus `MyMinorsLR`*) Behauptung a) gilt auch, wenn man dort in Schritt (3) den Algorithmus* `MaxMinors` *durch* `MaxMinorsLR` *ersetzt. In diesem Falle bezeichnet man den entsprechenden Algorithmus als* `MyMinorsLR`(k, A).

Beweis:
Es ist klar, daß man nach endlich vielen Schritten alle Minoren k-ter Ordnung von A erhält, wenn man für jede Zeilenauswahl $[i_1 \dots i_k] \in L_k^n =$ `AllSubsets`$(\{1, \dots, n\}, k)$ jeweils alle maximalen Minoren in diesen Zeilen berechnet. □

24.2 Bemerkungen (Komplexität von MyMinors, MyMinorsLR)
Sei $k \in \mathbb{N}_+$ und sei A eine $n \times m$-Matrix mit $1 \leq k \ll n, m$. Ferner soll kein Minor der Ordnung k von A verschwinden.

a) (*Algorithmus* `MyMinors`) Um mit dem Verfahren `MyMinors` aus Satz 24.1a) sämtliche $\binom{n}{k} \cdot \binom{m}{k} \approx n^k m^k$ Minoren k-ter Ordnung zu berechnen, muß man $\binom{n}{k}$-mal für eine $k \times m$-Matrix den Algorithmus `MaxMinors` durchführen. Insgesamt ergeben sich daher für `MyMinors`

$\approx O(k^2 n^k m^k)$ Additionen

$\approx O(k^2 n^k m^k)$ Multiplikationen

$\approx O(k n^k m^k)$ Divisionen,

vergleiche Bemerkung 22.3 (Komplexität von `MaxMinors`). Die bei den Unteralgorithmen (`AllSubsets` etc.) anfallenden Operationen sind dabei nicht berücksichtigt.

b) (*Algorithmus* `MyMinorsLR`) Analog zu a) benötigt man insgesamt für den Algorithmus `MyMinorsLR` aus Satz 24.1b)

$\approx O(k n^k m^k)$ Additionen

$\approx O(k n^k m^k)$ Multiplikationen

$\approx O(n^k m^k)$ Divisionen,

vergleiche Bemerkung 22.6 (Komplexität von `MaxMinorsLR`).

24.3 Bemerkung zur Implementierung (MyMinors, MyMinorsLR)
Arbeitet man bei den Algorithmen `MyMinors` bzw. `MyMinorsLR` (Satz 24.1) wie beschrieben die Liste $L_1 = L_k^n$ ab, indem man zu jedem Element $[i_1 \dots i_k] \in L_1$ die maximalen Minoren in diesen Zeilen mit `MaxMinors` bzw. `MaxMinorsLR` bestimmt, so enthält das Endergebnis L_2 automatisch die Minoren k-ter Ordnung in der Reihenfolge der lexikographisch geordneten Menge $L_k^{n,m} = \{[1 \dots k][1 \dots k], \dots, [n-k+1 \dots n][m-k+1 \dots m]\}$.

Lineare Relationen sind nicht nur innerhalb des Verfahrens `MaxMinorsLR` von Nutzen, indem dort aus allen maximalen Minoren mit einer bestimmten Spalte die übrigen berechnet werden. Man kann schließlich auch aus den Minoren k-ter Ordnung, die eine bestimmte Zeile enthalten, andere Minoren k-ter Ordnung bestimmen. Zur Erinnerung:

24.4 Bemerkungen (lineare Relationen)
Sei A eine beliebige $n \times m$-Matrix aus $\mathrm{Mat}_{n \times m}(R)$, und sei $k \in \mathbb{N}_+$ mit $1 \leq k \leq n, m$.

a) (*Lineare Relationen*) Wenn $a_{i_0 l_\mu} \neq 0$ für ein $i_0 \in \{1, \dots, n\}$ und ein $l_\mu \in \{1, \dots, m\}$ gilt und wenn man alle Minoren k-ter Ordnung von A kennt, die die Zeile i_0 enthalten, so kann man daraus nach Lemma 20.7b) (lineare Relationen) alle anderen Minoren

$[i_1 \dots i_k][l_1 \dots l_k](A)$ mit beliebigen $i_1, \dots, i_k \in \{1, \dots, n\}$ und $l_1, \dots, l_k \in \{1, \dots, m\}$ berechnen, die die Spalte l_μ enthalten, und zwar mit der Formel

$$[i_1 \dots i_k][l_1 \dots l_k](A) = \frac{1}{a_{i_0 l_\mu}} \cdot \sum_{\lambda=1}^{k} (-1)^{\lambda+1} \cdot a_{i_\lambda l_\mu} \cdot [i_0\ i_1 \dots \widehat{i}_\lambda \dots i_k][l_1 \dots l_k](A). \quad (*)$$

Gilt $a_{i_0 l} = 0$ für höchstens $(k-1)$ Spalten $l \in \{1, \dots, m\}$, so kann man offensichtlich aus den Minoren mit der Zeile i_0 alle übrigen Minoren gewinnen. Unabhängig davon, welche Zeile aus $\{1, \dots, n\}$ die Variable i_0 bezeichnet, existieren stets $\binom{n-1}{k-1} \cdot \binom{m}{k}$ Minoren k-ter Ordnung, die die Zeile i_0 beinhalten. Gilt dabei $i_0 > i_1$, so muß man in $(*)$ auf der rechten Seite zusätzlich beachten, daß die Determinante $[i_0\ i_1 \dots \widehat{i}_\lambda \dots i_k][l_1 \dots l_k](A)$ sich durch das Vorzeichen $\operatorname{sign}[i_0\ i_1 \dots \widehat{i}_\lambda \dots i_k]$ von dem entsprechenden Minor in den Zeilen $i_0, i_1, \dots, \widehat{i}_\lambda, \dots, i_k$ und den Spalten $l_1, \dots, l_k$ unterscheiden kann. Ist speziell $i_0 = 1$ und $a_{1 l_\mu} \neq 0$ für ein $l_\mu \in \{l_1, \dots, l_k\}$, so folgt aus $(*)$ die Formel

$$[i_1 \dots i_k][l_1 \dots l_k](A) = \frac{1}{a_{1 l_\mu}} \cdot \sum_{\lambda=1}^{k} (-1)^{\lambda+1} \cdot a_{i_\lambda l_\mu} \cdot [1\ i_1 \dots \widehat{i}_\lambda \dots i_k][l_1 \dots l_k](A), \quad (**)$$

um aus den Minoren mit der Zeile 1 alle anderen zu erhalten.

b) Gesetzt den Fall, jede Zeile i von A enthält mindestens k-mal 0 als Koeffizienten. Sei o.E. $i = 1$, und seien $l_1, \dots, l_k \in \{1, \dots, m\}$ paarweise verschiedene Spaltenindizes mit $a_{1 l_1} = \dots = a_{1 l_k} = 0$. Wie kann man nun mittels geeigneter linearer Relationen aus den Minoren mit der Zeile 1 die Minoren $[i_1 \dots i_k][l_1 \dots l_k](A)$ mit beliebigen $i_1, \dots, i_k \in \{2, \dots, n\}$ berechnen? In diesem Fall lassen sie sich nicht direkt mit der Formel $(**)$ aus den Minoren mit der Zeile 1 erhalten, aber möglicherweise über einen Umweg. Berechnet man zunächst alle Minoren in den Zeilen $i_1, \dots, i_k$, die man mit der Formel $(**)$ schaffen kann, so kann man aus diesen Minoren eventuell den gewünschten Minor $[i_1 \dots i_k][l_1 \dots l_k](A)$ erhalten, indem man wie bei den maximalen Minoren die lineare Relation aus Lemma 20.7a) anwendet. Gelingt dies nicht, so muß man diesen Minor wohl oder übel ohne lineare Relationen anderweitig ermitteln.

Als Anwendung von Bemerkung 24.4a) folgt nun eine weitere Methode, um alle Minoren einer Matrix zu berechnen.

24.5 Satz (Algorithmen MyMinors_2, MyMinorsLR_2)
Sei A eine beliebige $n \times m$-Matrix über R, ferner sei $k \in \mathbb{N}_+$ mit $1 \leq k \leq n, m$, und die erste Zeile von A enthalte höchstens $(k-1)$-mal 0 als Koeffizienten. Der Algorithmus `AllSubsets` *sei wie in Satz 20.4, die Algorithmen* `MaxMinors` *und* `MaxMinorsLR` *seien wie in Satz 22.2 bzw. 22.5, und* `LinRelMinors` *sei wie nachfolgend definiert.*

a) (Algorithmus `MyMinors_2`*) Betrachte die folgenden Instruktionen:*
   (1) Führe `AllSubsets`$(\{1, \dots, n\}, k)$ *bzw.* `AllSubsets`$(\{1, \dots, m\}, k)$ *durch und nenne das Ergebnis L_1 bzw. L_2. Setze $L_3 := \emptyset$.*
   (2) Falls $L_1 = \emptyset$ gilt, dann gib das Ergebnis L_3 aus.
   (3) Falls das erste Element $[i_1 \dots i_k] \in L_1$ mit einem Index $i_1 > 1$ beginnt, fahre fort bei Schritt (5), andernfalls bestimme für das erste Element $[i_1 \dots i_k] \in L_1$ die Teilmatrix B von A in den Zeilen $i_1, \dots, i_k$ und den Spalten $1, \dots, m$.
   (4) Erhalte mit `MaxMinors`(B) *das Ergebnis M und ersetze L_3 durch $L_3 \cup M$. Streiche $[i_1 \dots i_k]$ aus L_1 und fahre fort bei Schritt (2).*

(5) Führe der Reihe nach für jede Spaltenauswahl $[l_1 \dots l_k] \in L_2$ *den Algorithmus* `LinRelMinors`$(A, L_3, [i_1 \dots i_k], [l_1 \dots l_k])$ *durch und ergänze* L_3 *mit dem jeweiligen Ergebnis. Streiche* $[i_1 \dots i_k]$ *aus* L_1 *und fahre fort bei Schritt (2).*

Dies ist ein mit `MyMinors_2`(k, A) *bezeichneter Algorithmus, dessen Ergebnis die Menge der Minoren k-ter Ordnung von A ist.*

Dabei bedeute `LinRelMinors`$(A, L_2, [i_1 \dots i_k], [l_1 \dots l_k])$ *den nachfolgenden Algorithmus, der den durch* $[i_1 \dots i_k][l_1 \dots l_k]$ *definierten Minor von A aus den Minoren in* L_3 *und aus der ersten Zeile von A berechnet. Hierfür sei der Algorithmus* `PosMinor` *wie in Satz 20.3.*

(P1) Für $\lambda = 1, \dots, k$ *setze* $h_\lambda := [1\ i_1\ i_2 \dots \widehat{i_\lambda} \dots i_k]$.

(P2) Ersetze dann h_λ *für* $\lambda = 1, \dots, k$ *durch das Element, das in* L_3 *an der Position* `PosMinor`$(h_\lambda, [l_1 \dots l_k], n, m)$ *steht.*

(P3) Finde den ersten Spaltenindex l_μ *aus* $\{l_1, \dots, l_k\}$*, der ungleich 0 ist.*

(P4) Berechne zuerst $s := \sum_{\lambda=1}^{k} (-1)^{\lambda+1} \cdot a_{i_\lambda, l_\mu} \cdot h_\lambda$*. Ersetze anschließend s durch* $s/a_{1,l_\mu}$ *und gib als Ergebnis s aus.*

b) *(Algorithmus* `MyMinorsLR_2`*) Behauptung a) gilt auch, wenn man dort in Schritt (4) den Algorithmus* `MaxMinors` *durch* `MaxMinorsLR` *ersetzt. In diesem Falle bezeichnet man den entsprechenden Algorithmus als* `MyMinorsLR_2`(k, A).

Beweis:
Es ist klar nach der vorangegangenen Bemerkung 24.4 (lineare Relationen), daß man nach endlich vielen Schritten alle Minoren k-ter Ordnung von A erhält, wenn man für jede Zeilenauswahl $[1\ i_2 \dots i_k] \in L_k^n =$ `AllSubsets`$(\{1, \dots, n\}, k)$ jeweils alle maximalen Minoren in diesen Zeilen berechnet und anschließend in Schritt (5) mit `LinRelMinors` die übrigen Minoren von A. Hierbei ist der Unteralgorithmus `LinRelMinors` die direkte Anwendung der Formel $(**)$ aus der obigen Bemerkung 24.4a). □

24.6 Bemerkungen (Komplexität von MyMinors_2, MyMinorsLR_2)
Sei $k \in \mathbb{N}_+$, und sei A eine $n \times m$-Matrix mit $1 \le k \ll n, m$. Ferner soll kein Minor der Ordnung k von A verschwinden, und die erste Zeile von A soll 0 höchstens $(k-1)$-mal enthalten.

a) Um alle $\binom{n-1}{k-1}\binom{m}{k} \approx n^{k-1}m^k$ Minoren k-ter Ordnung mit der Zeile 1 zu berechnen, benötigt man nach Bemerkung 22.3 (Komplexität von `MaxMinors`)

$\approx O(k^2 n^{k-1} m^k)$ Additionen

$\approx O(k^2 n^{k-1} m^k)$ Multiplikationen

$\approx O(k n^{k-1} m^k)$ Divisionen

bzw. nach Bemerkung 22.6 (Komplexität von `MaxMinorsLR`)

$\approx O(k n^{k-1} m^k)$ Additionen

$\approx O(k n^{k-1} m^k)$ Multiplikationen

$\approx O(n^{k-1} m^k)$ Divisionen.

b) Um einen Minor k-ter Ordnung mittels einer geeigneten linearen Relation zu berechnen, benötigt man genau

$k-1$ Additionen

k Multiplikationen

1 Division.

Für die $\binom{n}{k}\binom{m}{k} - \binom{n-1}{k-1}\binom{m}{k} \approx n^k m^k$ Minoren, die die Zeile 1 nicht enthalten, ergeben sich somit insgesamt

$\approx O(kn^k m^k)$ Additionen

$\approx O(kn^k m^k)$ Multiplikationen

$\approx O(n^k m^k)$ Divisionen.

c) (*Algorithmus* `MyMinors_2`) Mit den Abschätzungen aus a) und b) folgt, daß man für den Algorithmus `MyMinors_2` aus Satz 24.5a) insgesamt

$\approx O(kn^k m^k)$ Additionen

$\approx O(kn^k m^k)$ Multiplikationen

$\approx O(n^k m^k)$ Divisionen

benötigt. Die bei den Unteralgorithmen (`AllSubsets` etc.) anfallenden Operationen sind dabei nicht berücksichtigt.

d) (*Algorithmus* `MyMinorsLR_2`) Analog zu c) benötigt man insgesamt für den Algorithmus `MyMinorsLR_2` aus Satz 24.5b)

$\approx O(kn^k m^k)$ Additionen

$\approx O(kn^k m^k)$ Multiplikationen

$\approx O(n^k m^k)$ Divisionen.

24.7 Bemerkungen zur Implementierung (MyMinors_2, MyMinorsLR_2)

a) Die Algorithmen `MyMinors_2` und `MyMinorsLR_2` aus Satz 24.5 sind nur für Matrizen programmiert, deren erste Zeile 0 höchstens $(k-1)$-mal enthält, denn sonst könnte man nicht die Relationenformel $(**)$ aus Bemerkung 24.4a) anwenden. Erfüllt eine Matrix diese Voraussetzung nicht, so kann man möglicherweise eine entsprechende Zeile nach oben tauschen. Gelingt auch dies nicht, zeigt Bemerkung 24.4b) weitere Alternativen auf, die hier bei der Implementierung jedoch nicht berücksichtigt sind.

b) Fügt man an die Minorenliste L_3 immer die jeweiligen Ergebnisse an, die man in Schritt (4) bzw. (5) mit `MaxMinors`, `MaxMinorsLR` oder `LinRelMinors` bekommt, so enthält L_3 am Ende alle Minoren in der Reihenfolge der lexikographisch geordneten Menge $L_k^{n,m}$.

25 Weitere Überlegungen zur Berechnung beliebiger Minoren

Sei im folgenden $k \in \mathbb{N}_+$, und A bezeichne eine $n \times m$-Matrix über einem Integritätsring R. Wie kann man für diese Matrix möglichst effektiv alle $\binom{m}{k} \cdot \binom{n}{k}$ Minoren k-ter Ordnung mit $1 \leq k \leq n, m$ bestimmen? Im vorangegangenen Abschnitt hat man bereits einige Methoden kennengelernt. Dabei ermittelt man alle Minoren oder zumindest diejenigen mit der ersten Zeile sukzessive, indem man für jede in Frage kommende Zeilenauswahl die maximalen Minoren in diesen Zeilen bestimmt. Bei dieser Vorgehensweise nimmt man allerdings unnötige Berechnungen in Kauf, zum einen bei der Ermittlung der maximalen Minoren, vergleiche Bemerkung 21.10 (Doppelberechnungen), zum anderen dadurch, daß Zwischenergebnisse einer Zeilenauswahl sich auch für eine andere Zeilenauswahl verwerten ließen. Wie kann man letzteres vermeiden? Für die folgenden Überlegungen gelte zunächst

(V1) $3 \leq k \leq n \leq m$,

(V2) $\operatorname{rang} A \geq k$,

(V3) Zeilentausch sei nicht nötig.

Diese Voraussetzungen wählt man aus ähnlichen Gründen wie bei der Berechnung maximaler Minoren, die sich hier als Spezialfall für $k = n$ ergibt, vgl. Bemerkung 21.2 (Voraussetzungen). Als Fundament der Minorenberechnung dient im folgenden wieder das Forward-Backup-Verfahren.

Trigonalisiert man A mit einem Divisionsverfahren, so erhält man die bekannte Matrizenfolge $(B^{(\lambda)})_{\lambda=0\ldots n-1}$, wobei sich insbesondere für $\lambda = k-1$ die Matrix

$$B := B^{(k-1)} = \begin{pmatrix} (\alpha_{ij}^{(i-1)})_{\substack{i=1\ldots k-1\\ j=1\ldots m}} \\ (\alpha_{ij}^{(k-1)})_{\substack{i=k\ldots n\\ j=1\ldots m}} \end{pmatrix} =$$

$$= \begin{pmatrix} \alpha_{1j_1}^{(0)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{1m}^{(0)} \\ & & \alpha_{2j_2}^{(1)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{2m}^{(1)} \\ & & & \ddots & & & & & & & \vdots \\ & & & & \alpha_{k-1,j_{k-1}}^{(k-2)} & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{k-1,m}^{(k-2)} \\ & & & & & & \alpha_{kj_k}^{(k-1)} & \cdots & \cdots & \cdots & \alpha_{k,m}^{(k-1)} \\ & & \mathbf{0} & & & & \alpha_{k+1,j_k}^{(k-1)} & \cdots & \cdots & \cdots & \alpha_{k+1,m}^{(k-1)} \\ & & & & & & \vdots & & & & \vdots \\ & & & & & & \alpha_{n,j_k}^{(k-1)} & \cdots & \cdots & \cdots & \alpha_{nm}^{(k-1)} \end{pmatrix}$$

ergibt. In dieser Matrix sind die Zeilenstufenspalten $j_1, j_2, \ldots, j_{k-1}$, somit also die Spalten $1, \ldots, j_k - 1$ unter der Diagonalen ausgeräumt. Insbesondere enthalten die unteren Zeilen $k, \ldots, n$ die Koeffizienten $\alpha_{ij}^{(k-1)} = [1 \ldots k-1\ i][j_1 \ldots j_{k-1}\ j](A)$, welche Determinanten k-ter Ordnung sind. Da man an den Minoren k-ter Ordnung von A interessiert ist, genügt es offenbar, bei der Trigonalisierung nur die vordere Teilfolge $(B^{(\lambda)})_{\lambda=0\ldots k-1}$ zu erzeugen.

Die partiell trigonalisierte Matrix $B = B^{(k-1)}$ dient nun als Augangspunkt für die Backup-Prozedur. Dazu betrachtet man zunächst nur die Teilmatrix $C^{(k)}$ von B, die aus den obersten k Zeilen $b_1, \ldots, b_k$ besteht, also die Matrix

$$C^{(k)} := \begin{pmatrix} b_1 \\ \vdots \\ b_k \end{pmatrix} = \begin{pmatrix} \alpha_{1j_1}^{(0)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{1m}^{(0)} \\ & & \alpha_{2j_2}^{(1)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{2m}^{(1)} \\ & & & \ddots & & & & & & \vdots \\ & \mathbf{0} & & & \alpha_{k-1,j_{k-1}}^{(k-2)} & \cdots & \cdots & \cdots & \cdots & \alpha_{k-1,m}^{(k-2)} \\ & & & & & & \alpha_{kj_k}^{(k-1)} & \cdots & \cdots & \alpha_{k,m}^{(k-1)} \end{pmatrix}.$$

Diagonalisiert man diese Zeilenstufenmatrix $C^{(k)}$ mit der Backup-Prozedur, so erhält man in gewohnter Weise die Diagonalmatrix $(\delta_{ij}^{(k)})_{\substack{i=1\ldots k\\ j=1\ldots m}}$, deren Koeffizienten Determinanten k-ter Ordnung sind. Mit der wiederholten Backup-Prozedur aus Abschnitt 21 erhält man aus $C^{(k)}$ weitere Determinanten k-ter Ordnung, vgl. Lemma 21.4. Außerdem lassen sich anhand von $C^{(k)}$ gewisse verschwindende Minoren enttarnen.

Betrachtet man statt der Matrix $C^{(k)}$ allgemein für $\nu = k, \ldots, n$ die Matrizen

$$C^{(\nu)} := \begin{pmatrix} b_1 \\ \vdots \\ b_{k-1} \\ b_\nu \end{pmatrix} = \begin{pmatrix} \alpha_{1j_1}^{(0)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{1m}^{(0)} \\ & & \alpha_{2j_2}^{(1)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_{2m}^{(1)} \\ & & & \ddots & & & & & & \vdots \\ & \mathbf{0} & & & \alpha_{k-1,j_{k-1}}^{(k-2)} & \cdots & \cdots & \cdots & \cdots & \alpha_{k-1,m}^{(k-2)} \\ & & & & & & \alpha_{\nu j_k}^{(k-1)} & \cdots & \cdots & \alpha_{\nu,m}^{(k-1)} \end{pmatrix},$$

die jeweils aus den ersten $(k-1)$ Zeilen $b_1, \ldots, b_{k-1}$ und der ν-ten Zeile b_ν von B bestehen, so sind dies ebenfalls Zeilenstufenmatrizen. Gilt dabei $\operatorname{rang} C^{(\nu)} = k-1$, so verschwinden offensichtlich alle Minoren k-ter Ordnung in den Zeilen $1, \ldots, k-1, \nu$. Ist dagegen $\operatorname{rang} C^{(\nu)} = k$, so bestimmt man alle Indizes $j_{\nu 1} < j_{\nu 2} < \ldots < j_{\nu\rho}$ der Spalten, die in der Matrix $C^{(\nu)}$ auf der Höhe ν liegen, $\rho \in \mathbb{N}_+$. Damit läßt sich die Matrix $C^{(\nu)}$ genau wie zuvor die Matrix $C^{(k)}$ mit der wiederholten Backup-Prozedur aus Abschnitt 21 diagonalisieren, und außerdem lassen sich an $C^{(\nu)}$ bestimmte verschwindende Minoren erkennen.

Insgesamt wiederholt man die Forward-Backup-Prozedur nun also nicht nur, indem man wie bei den maximalen Minoren spaltenweise nach rechts fortschreitet, sondern mit den Matrizen $C^{(\nu)}$, $\nu = k, \ldots, n$, auch zusätzlich zeilenweise nach unten. Welche Minoren kann man damit alles in allem gewinnen?

25.1 Lemma (zeilen- und spaltenweise wiederholte Backup-Prozedur)
Es liege die obige Situation vor, wobei $\operatorname{rang} C^{(\nu)} = k$ *für alle* $\nu \in \{k, \ldots, n\}$ *sei. Wiederholt man wie beschrieben die Backup-Prozedur solange spalten- und zeilenweise, bis in allen Zeilenstufenmatrizen* $C^{(\nu)}$, $\nu = k, \ldots, n$, *die letzte Zeile gleich 0 gesetzt ist, so kennt man aufgrund der zugehörigen Diagonalmatrizen bis aufs Vorzeichen insbesondere die Determinanten* $[i_1 \ldots i_k][l_1 \ldots l_k](A)$ *mit paarweise verschiedenen Zeilenindizes* $i_1, \ldots, i_k \in \{1, \ldots, n\}$ *und paarweise verschiedenen Spaltenindizes* $l_1, \ldots, l_k \in \{1, \ldots, m\}$, *für die gilt:*
(1) Es ist $[i_1 \ldots i_k] = [1 \ldots k-1\ \nu]$ *mit* $\nu \in \{k, \ldots, n\}$.
(2) Die ersten $(k-2)$ *Indizes* $l_1, \ldots, l_{k-2}$ *sind Zeilenstufenindizes aus* $\{j_1, j_2, \ldots, j_{k-1}\}$.
(3) Die beiden letzten Indizes l_{k-1} *und* l_k *sind beliebig. Bezeichnen insbesondere* l_{k-1} *und* l_k *Nichtzeilenstufenspalten aus* $\{1, \ldots, m\} \setminus \{j_1, j_2, \ldots, j_{k-1}, j_{k1}, j_{k2}, \ldots, j_{k\rho}\}$, *so gilt* $[i_1 \ldots i_k][l_1 \ldots l_k](A) = 0$.

Beweis:
Die Behauptung folgt sofort, wenn man Lemma 21.4 (wiederholte Backup-Prozedur) für $\nu = k, \ldots, n$ auf die Matrizen $C^{(\nu)}$ anwendet. □

Die Matrix A erfülle nach wie vor die beiden Voraussetzungen
(V1) $3 \le k \le n \le m$,
(V2) $\operatorname{rang} A \ge k$,
jedoch wird nun die Voraussetzung (V3) fallengelassen, d.h. ein Zeilentausch ist jetzt zulässig. Welche Auswirkungen hat dies auf die bisherigen Überlegungen?

Bei der Bestimmung aller maximalen Minoren braucht man nicht darauf zu achten, ob ein Zeilentausch stattfindet oder nicht. Denn da bei der Trigonalisierung jeder Zeilentausch und der damit verbundene Vorzeichenwechsel der betreffenden Determinanten berücksichtigt wird, erhält man mit dem Backup-Verfahren am Ende stets die Determinanten n-ter Ordnung $\delta_{ij}^{(n)}$. Wie verhält es sich aber bei den Minoren beliebiger Ordnung k? Hier wird es in der Tat unangenehm, weil man sich nun im Fall $k < n$ die Permutation der

Zeilen merken muß, um hinterher zu wissen, welche Minoren $[i_1 \dots i_k][l_1 \dots l_k](A)$ sich in den jeweiligen Zeilenstufen- bzw. Diagonalmatrizen an welcher Position befinden.

25.2 Bemerkung (Zeilentausch)
Bezeichnet in der obigen Situation $\widetilde{B}$ die zu A äquivalente Matrix, die man erhält, wenn man in A mit einem Trigonalisierungsverfahren mit Division die ersten $(k-1)$ Zeilenstufenspalten $j_1, j_2, \dots, j_{k-1}$ ausräumt, und stellt $\pi = [i_1 \dots i_n]$ die Permutation der Zeilen $[1 \dots n]$ mit $\pi(1) = i_1$, $\pi(2) = i_2, \dots, \pi(n) = i_n$ dar, die dazu nötig ist, so gilt

$$\widetilde{B} = \widetilde{B}^{(k-1)} = \begin{pmatrix} (\widetilde{\alpha}_{ij}^{(i-1)})_{\substack{i=1\dots k-1\\ j=1\dots m}} \\ (\widetilde{\alpha}_{ij}^{(k-1)})_{\substack{i=k\dots n\\ j=1\dots m}} \end{pmatrix} =$$

$$= \begin{pmatrix} \widetilde{\alpha}_{1j_1}^{(0)} & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \widetilde{\alpha}_{1m}^{(0)} \\ & \widetilde{\alpha}_{2j_2}^{(1)} & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \widetilde{\alpha}_{2m}^{(1)} \\ & & \ddots & & & & & & & \vdots \\ & & & \widetilde{\alpha}_{k-1,j_{k-1}}^{(k-2)} & \dots & \dots & \dots & \dots & \dots & \widetilde{\alpha}_{k-1,m}^{(k-2)} \\ & & & & & \widetilde{\alpha}_{kj_k}^{(k-1)} & \dots & \dots & \dots & \widetilde{\alpha}_{k,m}^{(k-1)} \\ & & \mathbf{0} & & & \widetilde{\alpha}_{k+1,j_k}^{(k-1)} & \dots & \dots & \dots & \widetilde{\alpha}_{k+1,m}^{(k-1)} \\ & & & & & \vdots & & & & \vdots \\ & & & & & \widetilde{\alpha}_{n,j_k}^{(k-1)} & \dots & \dots & \dots & \widetilde{\alpha}_{nm}^{(k-1)} \end{pmatrix}$$

mit den Koeffizienten

$$\widetilde{\alpha}_{ij}^{(i-1)} = \operatorname{sign}[\pi(1) \dots \pi(i-1)\ \pi(i)] \ \cdot\ [\pi(1) \dots \pi(i-1)\ \pi(i)][j_1 \dots j_{i-1}\ j](A) \quad \text{und}$$

$$\widetilde{\alpha}_{ij}^{(k-1)} = \operatorname{sign}[\pi(1) \dots \pi(k-1)\ \pi(i)] \ \cdot\ [\pi(1) \dots \pi(k-1)\ \pi(i)][j_1 \dots j_{k-1}\ j](A).$$

Der Faktor $\operatorname{sign}[\dots]$ rührt daher, da bei den Zeilentauschalgorithmen `Pivot` und `COPivot` jeder Vorzeichenwechsel registriert wird.

Benennt man ähnlich wie zuvor für $\nu = k, \dots, n$ die Matrizen, die aus den ersten $(k-1)$ Zeilen und der ν-ten Zeile von $\widetilde{B}$ bestehen, mit

$$\widetilde{C}^{(\nu)} := \begin{pmatrix} \widetilde{b}_1 \\ \vdots \\ \widetilde{b}_{k-1} \\ \widetilde{b}_\nu \end{pmatrix},$$

so kann man einerseits anhand ihrer Zeilenstufenverteilung genau wie beim Algorithmus `FindZeroMaxMinors` aus Satz 21.5 bestimmte verschwindende Minoren aussieben. Führt man andererseits an diesen Matrizen wie gehabt die wiederholte Backup-Prozedur durch, so folgt aus Lemma 25.1 (Backup-Prozedur) sofort

25.3 Korollar (Zeilentausch und wiederholte Backup-Prozedur)
Es liege die obige Situation vor, wobei $\operatorname{rang}\widetilde{C}^{(\nu)} = k$ *für alle* $\nu \in \{k, \dots, n\}$ *sei. Wiederholt man wie beschrieben die Backup-Prozedur solange spalten- und zeilenweise, bis in allen Zeilenstufenmatrizen* $\widetilde{C}^{(\nu)}$, $\nu = k, \dots, n$, *die letzte Zeile gleich 0 gesetzt ist, so kennt man aufgrund der zugehörigen Diagonalmatrizen bis aufs Vorzeichen insbesondere die Determinanten* $[i_1 \dots i_k][l_1 \dots l_k](A)$ *mit paarweise verschiedenen Zeilenindizes* $i_1, \dots, i_k \in \{1, \dots, n\}$ *und paarweise verschiedenen Spaltenindizes* $l_1, \dots, l_k \in \{1, \dots, m\}$, *für die gilt:*
(1) Es ist $[i_1 \dots i_k] = [\pi(1) \dots \pi(k-1)\ \pi(\nu)]$ *mit* $\nu \in \{k, \dots, n\}$.
(2) Die ersten $(k-2)$ *Indizes* $l_1, \dots, l_{k-2}$ *sind Zeilenstufenindizes aus* $\{j_1, j_2, \dots, j_{k-1}\}$.
(3) Die beiden letzten Indizes l_{k-1} *und* l_k *sind beliebig. Bezeichnen insbesondere* l_{k-1} *und* l_k *Nichtzeilenstufenspalten aus* $\{1, \dots, m\} \setminus \{j_1, j_2, \dots, j_{k-1}, j_{k1}, j_{k2}, \dots, j_{k\rho}\}$, *so gilt* $[i_1 \dots i_k][l_1 \dots l_k](A) = 0$.

Beweis:
Die Behauptung folgt sofort aus Lemma 25.1 (zeilen- und spaltenweise wiederholte Backup-Prozedur), wenn man die Zeilenpermutation π miteinbezieht. □

Da man die Trigonalisierung nur bis zur partiellen Zeilenstufenmatrix $\widetilde{B}^{(k-1)}$ aus Bemerkung 25.2 durchzuführen braucht und da man sich dabei auch die Zeilenpermutation $\pi = [\pi(1) \dots \pi(n)]$ merken muß, folgt nun eine Version des Algorithmus `B2TMDMD` aus Satz 4.6, in der dies berücksichtigt ist. Um der Variablenbezeichnung aus Satz 4.6 treu zu bleiben, wird $\widetilde{B}^{(k-1)}$ vorübergehend durch $\widetilde{B}^{(p-1)}$ ersetzt. Gilt insbesondere $\operatorname{rang} A < p$, so kommt die partielle Trigonalisierung der vollständigen Trigonalisierung gleich.

25.4 Satz (Algorithmus PartialB2TMDMD)
Sei A *eine beliebige* $n \times m$*-Matrix über einem Integritätsring* R *mit* $n \leq m$, *und sei* $p \in \mathbb{N}_+$ *mit* $1 \leq p \leq n$. *Die Unteralgorithmen* `PartialPivot` *und* `PartialC0Pivot` *seien wie anschließend definiert,* `B1TMDRow` *und* `B2TMDMDRow` *seien wie in Satz 3.7 bzw. 4.6. Betrachte die folgenden Instruktionen:*
(1) Setze $k := 1$, $\lambda := 1$, $t := 1$, $\pi := [1 \dots n]$ *und* $Z := \emptyset$.
(2) (Evtl. Zeilentausch) Falls $\lambda > m$ *oder* $k > p$ *ist, gib als Ergebnis* A, π *und* Z *aus. Andernfalls ersetze* A *und* π *durch das Ergebnis von* `PartialPivot`(A, λ, k, π).
(3) (Lange Zeilenstufe) Ist $a_{k\lambda} = 0$, *dann erhöhe* λ *um 1 und fahre fort bei Schritt (2).*
(4) Falls $k = p$ *gilt, dann ersetze* Z *durch* $Z \cup \{\lambda\}$, *erhöhe* k *um 1 und fahre fort bei Schritt (2).*
(5) Setze $j_k := \lambda$.
(6) (Letzte Spalte ausräumen) Falls $j_k = m$ *oder* $k = p - 1$ *ist, dann ersetze die unteren Zeilen* $i = k+1, \dots, n$ *von* A *jeweils durch das Ergebnis von* `B1TMDRow`(A, i, j_k, k, t). *Ersetze dann* Z *durch* $Z \cup \{j_k\}$, *erhöhe* k *und* λ *jeweils um 1 und fahre fort bei Schritt (2).*
(7) (Evtl. Zeilentausch) Führe `PartialC0Pivot`(A, j_k, k, π) *aus und erhalte daraus als Ergebnis* A, c_0 *und* π.
(8) (Lange Zeilenstufe, Spalten $j_k, j_k + 1$ *ausräumen) Falls* $c_0 = 0$ *ist, dann ersetze die unteren Zeilen* $i = k+1, \dots, n$ *von* A *jeweils durch das Ergebnis von* `B1TMDRow`(A, i, j_k, k, t). *Ersetze* Z *durch* $Z \cup \{j_k\}$, *setze dann* $t := a_{kj_k}$, *erhöhe* k *um 1,* λ *um 2 und fahre bei Schritt (2) fort.*
(9) (Spalten $j_k, j_k + 1$ *ausräumen) Setze* $j_{k+1} := j_k + 1$ *und* $t_2 := t^2$. *Ersetze ferner die unteren Zeilen* $i = k+2, \dots, n$ *von* A *durch das jeweilige Ergebnis von* `B2TMDMDRow`(A, i, j_k, k, c_0, t_2). *Ersetze außerdem die Zeile* $(k+1)$ *durch das Ergebnis von* `B1TMDRow`$(A, k+1, j_k, k, t)$. *Ersetze* Z *durch* $Z \cup \{j_k, j_{k+1}\}$, *setze* $t := a_{k+1,j_k+1}$, *erhöhe* k *und* λ *jeweils um 2 und fahre bei Schritt (2) fort.*

Dies ist ein mit `PartialB2TMDMD`(A,p) *bezeichneter Algorithmus, der als Ergebnis die Menge der Zeilenstufenspalten* $Z = \{j_k : 1 \le k \le p \text{ und } k \le \operatorname{rang} A\}$ *ausgibt und ferner eine zu* A *äquivalente partielle Zeilenstufenmatrix Matrix, in der die Zeilenstufenspalten aus* $\{j_k : 1 \le k < p \text{ und } k \le \operatorname{rang} A\}$ *ausgeräumt sind. Außerdem wird als Ergebnis die Permutation* π *ausgegeben, die anzeigt, welche Zeilen im Laufe der partiellen Trigonalisierung vertauscht worden sind.*

Hierbei bedeute `PartialPivot`(A,λ,k,π) *den folgenden Zeilentauschalgorithmus:*

(P1) Falls $a_{k\lambda} = 0$ *ist, suche die nächste Zeile* l *mit* $k < l \le n$*, für die* $a_{l\lambda} \neq 0$ *gilt. Falls eine solche Zeile* l *existiert, dann multipliziere sie mit* -1 *und vertausche die Zeilen* l *und* k*. Vertausche ebenfalls in* π *die Komponenten an den Position* l *und* k*.*

(P2) Gib das Ergebnis A *aus.*

Der zweite Zeilentauschalgorithmus `PartialC0Pivot`(A,j_k,k,π) *ist gegeben mit:*

(P1) Berechne $c_0 := [k\ k+1][j_k\ j_k+1](A) = a_{kj_k} \cdot a_{k+1,j_k+1} - a_{k+1,j_k} \cdot a_{k,j_k+1}$*. Falls* $c_0 = 0$ *ist, suche die nächste Zeile* l *mit* $k+1 < l \le n$*, für die* $c_0 := [k\ l][j_k\ j_k+1](A) \neq 0$ *gilt. Falls eine solche Zeile* l *existiert, dann multipliziere sie mit* -1 *und vertausche die Zeilen* l *und* $(k+1)$*. Vertausche dann ebenfalls in* π *die Komponenten an den Positionen* l *und* $(k+1)$*.*

(P2) Gib als Ergebnis A*,* c_0 *und* π *aus.*

Beweis:
Die Instruktionen (P1) und (P2) sind jeweils genau wie bei bei den Algorithmen `Pivot` (Satz 3.7) und `C0Pivot` (Satz 4.3), bis auf den Umstand, daß nun die Zeilenvertauschung bei der Matrix A auch an π nachvollzogen wird. Somit erhält man als ein Endergebnis von `PartialB2TMDMD` explizit die zur Trigonalisierung nötige Permutation π der Zeilen $[1 \ldots n]$.

Die angegeben Schritte (1) bis (9) sind fast genauso wie bei `B2TMDMD` (Satz 4.7), wobei nun die Trigonalisierung abgebrochen wird, sobald $k > p$ gilt, d.h. sobald wie behauptet die Zeilenstufenspalten aus $\{j_k : 1 \le k < p \text{ und } k \le \operatorname{rang} A\}$ ausgeräumt sind. Da bei jeder Wiederholung von Schritt (4), (6), (8) oder (9) Zeilenstufenspalten zu Z hinzugefügt werden, gilt am Ende offensichtlich wie behauptet $Z = \{j_k : 1 \le k \le p \text{ und } k \le \operatorname{rang} A\}$.

□

25.5 Bemerkungen zur Implementierung (PartialB2TMDMD)

Wie im Beweis von Satz 25.4 (`PartialB2TMDMD`) bereits ausgeführt worden ist, genügt es, die Funktionen `Pivot`, `C0Pivot` und `B2TMDMD` leicht abzuändern.

a) (*Zeilentausch*) Die Funktionen `PartialPivot` und `PartialC0Pivot` erhalten neben den üblichen Argumenten A, j und k als viertes Argument π, welches zusätzlich als `Var` deklariert ist und daher automatisch aktualisiert wird. Daneben wird wie sonst auch eine der beiden vertauschten Zeilen mit -1 multipliziert, um den Vorzeichenwechsel der Determinante zu berücksichtigen.

b) (`PartialB2TMDMD`) Bei der Umänderung von `B2TMDMD` ist im Grunde nur zu beachten, daß sich bei `PartialB2TMDMD`(A,p) die Abbruchkriterien nun auf p und nicht mehr auf n beziehen, vgl. Satz 25.4, Schritt (2), (4), (6). Falls $\operatorname{rang} A < p$ gilt, wird A mit `PartialB2TMDMD`(A,p) nicht nur partiell, sondern sogar vollständig trigonalisiert.

Ein (mehrfacher) Zeilentausch und das zahlreiche Vorkommen verschwindender Minoren kann sehr viel Verwirrung stiften, vor allem bei größeren Matrizen und Minoren höherer Ordnung, die eine häufigere Trigonalisierung erfordern. In der Hinsicht ist es also sehr viel einfacher, wie in Abschnitt 24 die Minoren einfach in der Reihenfolge der Zeilenauswahl mit Hilfe der Algorithmen für maximale Minoren zu berechnen. Aus diesem Grund wird

der Leser (und die Autorin) damit verschont, daß auf der Grundlage der zeilen- und spaltenweise wiederholte Backup-Prozedur ein weiterer Algorithmus zur Minorenberechnung formuliert wird.

Neben den bis dato vorgeschlagenen Methoden zur Berechnung aller Minoren einer bestimmten Ordnung gibt es sicherlich noch andere, beispielsweise die folgende (vgl. dazu auch Bemerkung 22.10 zur Entwicklung maximaler Minoren).

25.6 Bemerkung (Entwicklung der Minoren)
Sei $k \in \mathbb{N}_+$, und sei A eine $n \times m$-Matrix über einem Integritätsring R mit $k \leq n \leq m$. Kennt man alle Minoren $(k-1)$-ter Ordnung von A, die nur Zeilen aus $\{1, \ldots, n-1\}$ enthalten, so kann man daraus sofort alle Minoren k-ter Ordnung berechnen, indem man sie nach der letzten Zeile entwickelt.

Wie erhält man all diese Minoren $(k-1)$-ter Ordnung, die die Zeile n nicht beinhalten? Entweder man berechnet sie rekursiv aus den Minoren $(k-2)$-ter Ordnung mit Zeilen aus $\{1, \ldots, n-2\}$, oder mit einem anderen Verfahren...

25.7 Bemerkung (Fittingideale)
Sei $k \in \mathbb{N}_+$, und sei ferner A eine beliebige $n \times m$-Matrix über einem Integritätsring R mit $k \leq n, m$. Wenn man sich damit zufriedengibt, statt aller (unter Umständen sehr zahlreicher) Minoren k-ter Ordnung nur das davon erzeugte Ideal (*Fittingideal*, vgl. [K]) zu kennen, so kann man die vorgestellten Algorithmen sicher noch weiter optimieren. Dies wird hier aber nicht mehr weiter ausgeführt.

26 Vergleich der Verfahren (beliebige Minoren)

Aufgrund der Komplexitäten sollte der Algorithmus `MyMinors` (Satz 24.1) schlechter abschneiden als die gleich guten Verfahren `MyMinorsLR` (Satz 24.1), `MyMinors_2` (Satz 24.5) und `MyMinorsLR_2` (Satz 24.5), siehe die folgende Tabelle. Um noch einmal Klarheit zu schaffen: die beiden erstgenannten Verfahren berechnen alle Minoren in der Reihenfolge der Zeilenauswahl mit `MaxMinors` bzw. `MaxMinorsLR`, die beiden nächstgenannten ermitteln dagegen nur die Minoren mit der Zeile 1 mit `MaxMinors` bzw. `MaxMinorsLR` und die übrigen mittels geeigneter linearer Relationen.

Minoren k-ter Ordnung von n $\times$ m-Matrizen				
Quellen	Algorithmus	Additionen	Multiplikationen	Divisionen
24.1a, 24.2a	`MyMinors`	$O(k^2m^kn^k)$	$O(k^2m^kn^k)$	$O(km^kn^k)$
24.1b, 24.2b	`MyMinorsLR`	$O(km^kn^k)$	$O(km^kn^k)$	$O(m^kn^k)$
24.5a, 24.6c	`MyMinors_2`	"	"	"
24.5b, 24.6c	`MyMinorsLR_2`	"	"	"

Tabelle 26

26.1 Zusammenfassung (Berechnung beliebiger Minoren)

a) (*Testsieger*) Unter den selbstgeschriebenen Funktionen sind für generische Matrizen die Verfahren `MyMinors` oder `MyMinorsLR` günstig. Bei anderen Matrizen erweist sich

oft das Verfahren `MyMinorsLR_2` als das beste, welches im Vergleich zu den anderen Verfahren die meisten Minoren mit linearen Relationen berechnet. Es mag oft auch vorteilhaft sein, statt einer $n \times m$-Matrix mit $n \leq m$ die transponierte Matrix zu behandeln, siehe auch Kapitel IV (Vergleiche der Trigonalisierungs- und Diagonalisierungsverfahren) oder Abschnitt 23 (Vergleich der Verfahren zur Berechnung maximaler Minoren).

b) (*Optimierungen*) Die selbstgeschriebenen Funktionen ließen sich noch weiter optimieren, wenn man mehr Zwischenergebnisse speichern würde, siehe Bemerkung 21.10b) und c) (Doppelberechnungen) und die Ausführungen in Abschnitt 25. Dazu müßte man sich die Matrizenfolgen, die bei den jeweils nötigen Trigonalisierungen anfallen, bzw. bestimmte Matrizen davon merken.

[B] E. H. Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Math. Comp.* **22** (1968), S. 65–578.

[CW] D. Coppersmith, S. Winograd. in *Proc. 19th Annu ACM Symp. on Theory of Comp.*, 1987, S. 1–6.

[F] G. Fischer. *Lineare Algebra.* 9., durchgesehene Auflage, Vieweg, Braunschweig–Wiesbaden, 1989.

[G] H. Gold. A Markovian single server with upstream job and downstream demand arrival stream. *Queueing systems* **30** (1998), S. 435–455.

[Ge] A. Geramita. *The Curves Seminar at Queen's, Volume X.* Queen's Papers in Pure and Appl. Math., **102**. Queen's University, Kingston (Ont.), 1996, S. 92ff.

[HH] G. Hämmerlin, K.-H. Hoffmann. *Numerische Mathematik.* 4. Auflage, Springer, Berlin–Heidelberg–New York, 1994.

[I] I. S. Iohvidov. *Hankel and Toeplitz matrices and forms: algebraic theory.* Birkhäuser, Boston–Basel–Stuttgart, 1982.

[KR] M. Kreuzer, L. Robbiano. *Computational Commutative Algebra 1.* Springer, Berlin–Heidelberg–New York, 2000.

[K] E. Kunz. *Kähler Differentials.* Vieweg, Braunschweig–Wiesbaden, 1986, S. 331ff.

[Mal1] G. I. Malashonok. Algorithms for the solution of systems of linear equations in commutative rings. *Effective Methods in Algebraic Geometry*, Edited by T. Mora and C. Traverso, Progress in Mathematics **94**. Birkhäuser, Boston–Basel–Berlin, 1991, S. 89–298.

[Mal2] G. I. Malashonok. *On the Solution of Systems of Linear Equations.* Preprint, 1995.

[M] G. Mangiapan. *Algoritmi fraction-free in algebra lineare.* Laurea tesi, Università di Genova, 1994.

[R] K. Rottmann. *Mathematische Formelsammlung.* 3., durchges. Auflage, Bibliographisches Institut, Mannheim–Wien–Zürich, 1984, S. 115–116.

[SM] T. Sasaki, H. Murao. Efficient Gaussian elimination method for symbolic determinants and linear systems. (1982) *ACM TOMS* **8**(3), S. 277–289.

[SchSt1] G. Scheja, U. Storch. *Lehrbuch der Algebra, Teil 1.* 2. Auflage, Teubner, Stuttgart, 1994.

[SchSt2] G. Scheja, U. Storch. *Lehrbuch der Algebra, Teil 2.* Teubner, Stuttgart, 1988.

[SchSt3] G. Scheja, U. Storch. *Lehrbuch der Algebra, Teil 3.* Teubner, Stuttgart, 1981.

[Str] V. Strassen. Gaussian Elimination is not optimal. *Numerische Mathematik* **13** (1968), S. 354–356.

REGENSBURGER MATHEMATISCHE SCHRIFTEN

der Fakultät für Mathematik der Universität Regensburg, 93040 Regensburg.
Preis: 8 EURO pro Exemplar (ohne Gewähr).

1. G. Wassermann. Classification of singularities with compact abelian symmetrie (1977)
2. P. Slodowy. Einfache Singularitäten und einfache algebraische Gruppen (1978), vergriffen, jetzt: Simple Singularities and Simple Algebraic Groups, Springer Lecture Notes 815 (1980)
3. J. Rung. Mengentheoretische Durchschnitte und Zusammenhang (1978), vergriffen
4. R. Waldi. Äquivariante Deformation monomialer Kurven (1980)
5. J. Koch. Über die Torsion des Differentialmoduls von Kurvensingularitäten (1983), vergriffen.
6. G. Meixner. Ein algebraischer Modulraum für die kompakten Riemannschen Flächen vom Geschlecht g (1983)
7. Ch. Meier. Über Familien von C^∞-Funktionen mit vorgegebenen Singularitäten auf D^2 (1983)
8. K. Jänich. Linienfelder mit Verzweigungsdefekten (1984)
9. S. Hellebrand. Deformation dicker Punkte und Netze von Quadriken (1986)
10. G. Bauer, R. Mennicken. Störungstheorie für diskrete Spektraloperatoren (1986)
11. E. Kunz. Über die Klassifikation numerischer Halbgruppen (1987)
12. M. Rost. Abbildungsdefekte in 4-Mannigfaltigkeiten (1987)
13. W. Krimmer, R. Mennicken. Entwicklungen analytischer Funktionen nach Eigenlösungen im Parameter nichtlinearer Differentialgleichungen (1987)
14. S. Kosarew. Grothendiecks existence theorem in analytic geometry and related results (1987)
15. N. Schwartz. The basic theory of real closed spaces (1987) (Doppelband 20,– DM)
16. H. Wagner. Charakterisierung von kubischen Galoiserweiterungen (1987)
17. M. Möller, C. Uschold. Über Randeigenwertprobleme für Differentialgleichungen mit der charakteristischen Gleichung $\lambda^p(\gamma^l - 1) = 0$ (1988)
18. J. Karl. Nichtnormale Umlaufseigenwertprobleme bei Differentialgleichungen im Komplexen (1988)
19. M. Seppälä. Topics on Teichmüller Spaces (1988)
20. R. Bieber. Der Satz von Emmy Noether über invariante Variationsprobleme (1988)
21. M. Kreuzer. Vektorbündel und der Satz von Cayley-Bacharach (1989)
22. A. Thalmaier. Asymptotik Brownscher Bewegungen (1989)
23. R. Huber. Bewertungsspektrum und rigide Geometrie (1992)
24. U. Helmke. The Cohomology of Moduli Spaces of Linear Dynamical Systems (1992)
25. M. Kraus. BRS-Transformationen und Slavnov-Taylor-Identitäten in einem endlichdimensionalen Modell (1996)
26. M. Kreuzer. Beiträge zur Theorie der nulldimensionalen Unterschemata projektiver Räume (1998)
27. R. Kaiser. Das sphärische Spektrum eines graduierten Ringes (1998)
28. M. Bockes. Dualitätstheorie projektiver Morphismen (1999)
29. P. Ullrich. Uniforme Zerlegung α-algebraischer Mengen zu bewerteten Körpern (1999)
30. M. Hien. Die Cluster-Eigenschaft der S-Matrix (2000)
31. T. Kaiser. Dirichletregularität in polynomial beschränkten o-minimalen Strukturen auf $\mathbb{R}$ (2001)
32. B. Krammer. Algorithmische lineare Algebra für Polynommatrizen (2002)