Universität Koblenz-Landau
Campus Koblenz
Fachbereich 4
Institut für Informatik

Berechnungsalgorithmen für Hermite-Normalformen und deren Anwendung zur Bestimmung ganzzahliger Lösungen linearer Gleichungssysteme

Studienarbeit

vorgelegt von

Kerstin Susewind

Zur Wegscheide 1 B
56070 Koblenz
susewind@uni-koblenz.de

18. Februar 2008

Betreuer: Professor Dr. rer. nat. Kurt Lautenbach, Universität Koblenz-Landau

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden. Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.

.....

(Ort und Datum) (Unterschrift)

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

1 Einleitung

In der mathematischen Literatur der vergangenen Jahrzehnte findet man eine Vielzahl verschiedener Algorithmen zu Berechnung der *Hermite-Normalform* (abgekürzt: HNF) ganzzahliger Matrizen, bei der es sich grundsätzlich um eine ganzzahlige Matrix in oberer oder unterer Dreiecksmatrixform handelt. Die Algorithmen unterscheiden sich in ihrer Vorgehensweise unter anderem dadurch, dass ihnen variante Definitionen der Hermite-Normalform zugrunde liegen. Einige Berechnungsverfahren akzeptieren als Eingabe nur quadratische, reguläre Matrizen [VZG99], andere nur Matrizen mit vollem Zeilenrang [Sch99, Nem88] und wiederum einige beliebige ganzzahlige Matrizen [The00, Jäg01, Kan79]. Auch die erzeugten Hermite-Normalformen können unterschiedliche Gestalten annehmen: Sie haben untere oder obere Dreiecksmatrixform und unter Umständen sogar negative Einträge.

In den meisten Fällen wird die Hermite-Normalform dazu verwendet, ganzzahlige Lösungen für lineare inhomogene und/oder homogene Gleichungssysteme zu gewinnen. Die ermittelten Ergebnisse können anschließend in weiteren Verfahren zur Berechnung nichtnegativer ganzzahliger Lösungen von linearen Gleichungssystemen herangezogen werden [Pas86, Han97, Krü87].

In dieser Arbeit werden vier Basisalgorithmen zur Berechnung der Hermite-Normalform einer ganzzahligen Matrix vorgestellt und erläutert, wie die HNF zum Auffinden ganzzahliger Lösungen linearer inhomogener Gleichungssysteme herangezogen werden kann. Das Augenmerk liegt hierbei auf der Arbeitsweise der verschiedenen Algorithmen und der Verwendbarkeit der Ergebnisse. In den letzten Jahren wurden Berechnungsverfahren entwickelt, die die HNF-Gewinnung für sehr große Matrizen optimieren sollen [Sto96, Mic01, The00], da die Basisalgorithmen bei Eingabe von Matrizen mit mehreren Tausend Zeilen und Spalten schnell an ihre Grenzen geraten.

Ein besonderes Anliegen dieser Arbeit ist es, die Basisalgorithmen im Detail vorzustellen. In Kapitel 2 wird zunächst eine ausführliche und vollständige Einführung in die dafür notwendigen mathematischen Grundlagen gegeben. Verschiedene Definitionen der Hermite-Normalform werden in Kapitel 3 diskutiert und darauf aufbauend in Kapitel 4 die entsprechenden Algorithmen detailliert und beispielorientiert vorgestellt. Alle Berechnungsschritte sind vollständig aufgeführt und können „von Hand“ und ohne

Zuhilfenahme weiterer Literatur durchgeführt werden.

Kapitel 5 beinhaltet die Verfahren zur Lösung linearer inhomogener Gleichungssysteme mittels der Hermite-Normalform. Die HNF-Algorithmen werden zusätzlich in einem Mathematica-Package bereitgestellt, auf dessen Besonderheiten in Kapitel 6 eingegangen wird. Abschließend folgt eine Zusammenfassung des Erreichten.

2 Grundlagen der linearen Algebra

In diesem Kapitel werden linear-algebraische Grundlagen wiederholt, die zum Verständnis für die in Kapitel 4 behandelten Algorithmen zur HNF-Berechnung notwendig sind. Zunächst werden Bezeichnungen und Notationen eingeführt, die in der gesamten Arbeit Verwendung finden. Anschließend folgt eine kurze Einführung in die Welt der Matrizen und Vektorräume. Darauf aufbauend beschäftigt sich Kapitel 2.3 mit der Lösbarkeit homogener und inhomogener linearer Gleichungssysteme, wobei hier zunächst ein allgemeines Lösungsverfahren beschrieben wird. Die Forderung der Ganzzahligkeit wird erst in Kapitel 5 einbezogen. Die Definitionen in diesem Kapitel entstammen hauptsächlich aus [Kna01] und [The00], wurden aber zum besseren Verständnis teilweise umformuliert.

2.1 Definitionen und Notationen

Dieser Abschnitt dient zur Vorstellung einiger numerischer, ganzzahliger und linearalgebraischer Definitionen und Notationen, die in den meisten Algorithmen verwendet werden. Zur Bezeichnung der Zahlenbereiche gelten in der gesamten Arbeit die folgenden Symbole:

- $\mathbb{N}$ - Menge der natürlichen Zahlen einschließlich Null,
- $\mathbb{Z}$ - Menge der ganzen Zahlen einschließlich Null.

2.1.1 Ganzzahlige und numerische Funktionen

Für $a, b \in \mathbb{Z}$ sei definiert:

Funktion	**Ausgabe**
$\mathrm{GCD}(a, b)$	größter gemeinsamer Teiler von a und b
$\textsc{extended}\mathrm{GCD}(a, b)$	liefert einen Tripel (d, u, v), wobei gilt: $\mathrm{GCD}(a, b) = d = a \cdot u + b \cdot v$
$\lfloor a \rfloor$	die größte ganze Zahl $\leq a$
$\lceil a \rceil$	die kleinste ganze Zahl $\geq a$
$\lvert a \rvert$	der Absolutwert („Betrag“) von a

Funktion	Ausgabe
$a \; mod \; b$	liefert den Rest der Division von a durch b (z.B. $5 \; mod \; 3 = 2$)

Tabelle 2.1: *Überblick über ganzzahlige und numerische Funktionen*

Eine zentrale Rolle in den in dieser Arbeit vorgestellten Algorithmen spielt die Berechnung des erweiterten größten gemeinsamen Teilers, die als Algorithmus 2.1 dargestellt ist. Mit Hilfe des EXTENDEDGCD werden Matrixeinträge annulliert oder wertmäßig reduziert. In der Literatur findet man häufig EXTENDEDGCD-Berechnungsverfahren, die als Eingabe nur positive ganze Zahlen akzeptieren. Algorithmus 2.1 akzeptiert zwei beliebige ganze Zahlen (vgl. [UT07]), wobei eine von Null verschieden sein muss[1]. Die Berechnung selbst erfolgt dann zwar mit den Absolutwerten der Eingabezahlen, jedoch werden vor der Ergebnisrückgabe die Vorzeichen nachträglich angepasst.

Algorithmus 2.1: EXTENDEDGCD

Eingabe: zwei ganze Zahlen a und b, wobei mindestens eine Zahl $\neq 0$
Ausgabe: Tripel (y, s, t) mit $ggT(a, b) = y = a \cdot s + b \cdot t$

```
1  if b = 0 then
2    if a > 0 then return (a, 1, 0)
3    else return (−a, −1, 0);
4    end_if
5  else_if a = 0 then
6       if b > 0 then return (b, 0, 1)
7       else return (−b, 0, −1);
8       end_if
9  end_if
10
11 if |a| ≥ |b| then
12   x := |a|;
13   y := |b|
14 else
15   x := |b|;
16   y := |a|;
17 end_if
18
19 s := 0;  s_1 := 1;  s_2 := 0;
20 t := 1;  t_1 := 0;  t_2 := 1;
21
```

[1] Die HNF-Algorithmen, die diese Berechnung verwenden, stellen dies sicher.

```
22 while x mod y ≠ 0 do
23    g := ⌊x/y⌋;  r := x mod y;
24    s := s_1 − g · s_2;  s_1 := s_2;  s_2 := s;
25    t := t_1 − g · t_2;  t_1 := t_2;  t_2 := t;
26    x := y;  y := r;
27 end_while
28
29 if |a| < |b| then
30    temp := s;  s := t;  t := temp;
31 end_if
32
33 if a < 0 then
34    s := −s;
35 end_if
36 if b < 0 then
37    t := −t;
38 end_if
39
40 return (y, s, t);
```

2.1.2 Vektoren und Matrizen

In der mathematischen Literatur gibt es viele variante Notationen für Matrizen und Vektoren. Folgende Definitionen gelten in dieser Arbeit:

Definition 2.1 (Matrix, Zeilenvektor, Spaltenvektor)

Sei

$$A = (a_{i,j})_{1 \le i \le m, 1 \le j \le n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

eine $(m \times n)$-Matrix *über* M*, also eine Matrix mit* m *Zeilen und* n *Spalten von Elementen aus einer beliebigen Menge* M*. Entsprechend nennt man eine* $(m \times 1)$*-Matrix*

$$a_{*,j} = \begin{pmatrix} a_{1,j} \\ a_{2,j} \\ \vdots \\ a_{m,j} \end{pmatrix}$$

den j-ten Spaltenvektor*, eine $(1 \times n)$-Matrix*

$$a_{i,*} = \begin{pmatrix} a_{i,1} & a_{i,2} & \dots & a_{i,n} \end{pmatrix}$$

den i-ten Zeilenvektor *von A. Die* transponierte $(n \times m)$*-Matrix*

$$A^T := (a_{j,i})_{1 \leq j \leq n, 1 \leq i \leq m}$$

zu einer Matrix A erhält man, indem man Zeilen und Spalten von A vertauscht.

Weitere Notationen:

- Die Menge aller $(m \times n)$-Matrizen über $\mathbb{Z}$ sei definiert als $\mathbf{Mat}_{m \times n}(\mathbb{Z})$.
- $I_n \in \mathbf{Mat}_{n \times n}(\mathbb{Z})$ bezeichnet die $(n \times n)$-Einheitsmatrix. I_n^M sei definiert als $I_n^M := M \cdot I_n$, $M \in \mathbb{N} \backslash \{0\}$.
- $\mathbf{0}_{m \times n}$ bezeichnet eine $(m \times n)$-Nullmatrix, d.h. eine $(m \times n)$-Matrix, die nur aus Nullen besteht.
- $A_{i..j,k..l}$ bezeichnet eine Submatrix von A, die sich aus den Zeilen i bis j und Spalten k bis l zusammensetzt.

Definition 2.2 (Determinante, Laplacescher Entwicklungssatz, Minor)

Die Determinante *einer quadratischen Matrix ist eine Funktion, die einer Matrix eine ganze Zahl zuordnet. Insbesondere gilt für*

(1×1)*-Matrizen A:*

$$\det A = \det(a_{1,1}) = a_{1,1}$$

(2×2)*-Matrizen A:*

$$\det A = \det \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} = a_{1,1} \cdot a_{2,2} - a_{2,1} \cdot a_{1,2}$$

(3×3)*-Matrizen A:*

$$\begin{aligned} \det A &= \det \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} \\ &= a_{1,1}a_{2,2}a_{3,3} + a_{1,2}a_{2,3}a_{3,1} + a_{1,3}a_{2,1}a_{3,2} \\ &- a_{1,3}a_{2,2}a_{3,1} - a_{1,1}a_{2,3}a_{3,2} - a_{1,2}a_{2,1}a_{3,3} \end{aligned}$$

$(n \times n)$-Matrizen A:
(Laplacescher Entwicklungssatz, Entwicklung nach der j-ten Spalte)

$$\det A = \sum_{i=1}^{n} (-1)^{i+j} \cdot a_{i,j} \cdot \det A_{i,j},$$

wobei $A_{i,j}$ die $((n-1)\times(n-1))$-Untermatrix von A ist, die man durch Streichung der i-ten Zeile und j-ten Spalte von A erhält.

Minore *sind so genannte Unterdeterminanten einer Matrix A. Sie werden auf quadratischen Untermatrizen berechnet, die man durch Streichen einer oder mehrerer Zeilen und Spalten von A erhält.*

Für eine $(n \times n)$-Matrix A seien $(k \times k)$-Matrizen A_k Teilmatrizen von A, die durch Streichung der $n - k$ letzten Spalten und untersten Zeilen enstehen. Die Determinanten dieser Teilmatrizen heißen Hauptminore*.*

Die Berechnung bestimmter Minore kommt in Algorithmus 4.2 zur Anwendung. Dort wird mit der Funktion

$$\textsc{Minors}(A, k)$$

eine Liste von Determinanten von $(k \times k)$-Submatrizen erzeugt. Jede Submatrix erhält man, indem man jede mögliche Menge von k Zeilen und k Spalten zu einer Matrix kombiniert. Für die Matrix

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

erhält man mit

$$\textsc{Minors}(A, 2) = (0, -4, 6, 6, -9, 10)$$

folglich die Determinanten aller möglichen (2×2)-Untermatrizen

$$(A_{1+2}, A_{1+3}, A_{1+4}, A_{2+3}, A_{2+4}, A_{3+4}),$$

wobei A_{x+y} die Kombination der Spalten x und y zu einer Submatrix darstellt. Verwendet man bei der Berechnung der Determinanten den Laplaceschen Entwicklungssatz, so ist es bei der Auswahl der Spalte j sinnvoll, eine Spalte mit vielen Nullen zu wählen. Damit gilt für viele Werte $a_{i,j}$: $a_{i,j} = 0$. In der Formel werden damit viele Summanden zu Null und der Rechenaufwand wird erheblich verkürzt.

Neben dem Rang einer Matrix, der später definiert wird, ist auch die Determinante einer quadratischen Matrix ein Kriterium für die Invertierbarkeit:

Definition 2.3 (Inverse einer Matrix)

Eine $(n \times n)$*-Matrix* A *ist genau dann* invertierbar *(oder:* regulär*,* nichtsingulär*), wenn* $\det A \neq 0$*.*

Die zu A inverse *Matrix bezeichnet man als* A^{-1}*.*

Es gilt: $A \cdot A^{-1} = A^{-1} \cdot A = I$*.*

Die inverse Matrix erhält man aus A, indem man zunächst eine neue Matrix $[A|I]$ erzeugt. Durch elementare Zeilenoperationen (vgl. Abschnitt 2.3) transformiert man diese neue Matrix um in $[I|A^{-1}]$, d.h. man erzeugt aus A eine Einheitsmatrix und wendet die gleichen Operationen auf I an[2]. A^{-1} ist danach direkt ablesbar.

Für Matrizen A, B, C gelten folgende in dieser Arbeit wichtige Rechenregeln:

- $(A \cdot B) \cdot C = A \cdot (B \cdot C)$
- $(A^T)^T = A$
- $(A \cdot B)^{-1} = B^{-1} \cdot A^{-1}$

2.2 Vektorräume

Die Definition des Vektorraumes bildet die Grundlage für dieses Kapitel. Alle folgenden Definitionen und Erläuterungen beziehen sich auf Elemente eines Vektorraumes. Exakte Definitionen für die Begriffe *Körper* und *Gruppe* findet man zum Beispiel in [Kna01]. Hier werden zum Verständnis und zur Vollständigkeit nur die wesentlichen Merkmale aufgeführt, damit anschließend der Begriff *Vektorraum* definiert werden kann:

Gruppe $(V, +)$**:**
V sei eine Menge von Vektoren. Das Paar $(V, +)$ heißt Gruppe, wenn V abgeschlossen gegenüber der Addition ist (die Summe zweier Vektoren ist wieder ein Element aus V) und folgendes gilt: Die Addition von Vektoren ist assoziativ, V besitzt ein neutrales Element (den Nullvektor $\vec{0}$) und jedes Element $\vec{v} \in V$ besitzt ein inverses Element $\vec{w} \in V$, sodass $\vec{v} + \vec{w} = \vec{0}$ gilt.

[2] Dieses Verfahren heißt Gauß-Jordan-Elimination (vgl. [Wei07a]).

Körper $(\mathbb{K}, +, \cdot)$**:**
Eine Menge $\mathbb{K}$ mit den Verknüpfungen Addition und Multiplikation ist ein Körper, wenn für die Verknüpfungen folgende drei Eigenschaften für alle Elemente aus $\mathbb{K}$ erfüllt sind:

1. Addition: Assoziativität, Kommutativität, Existenz des neutralen Elements **0**, Existenz eines inversen Elements zu jedem Element,

2. Multiplikation: Assoziativität, Kommutativität, Existenz des neutralen Elements **1**, Existenz eines inversen Elements zu jedem Element aus $\mathbb{K}\backslash\{0\}$,

3. Addition und Multiplikation: Für Elemente $a, b, c \in \mathbb{K}$ gilt: $a \cdot (b + c) = a \cdot b + a \cdot c$ (Distributivität).

Definition 2.4 (Vektorraum)

Sei $(\mathbb{K}, +, \cdot)$ *ein Körper. Eine kommutative Gruppe* $(V, +)$ *heißt* $\mathbb{K}$-Vektorraum *(*Vektorraum $\mathbb{K}$*,* linearer Raum*), wenn eine äußere Verknüpfung*

$$\bullet : \mathbb{K} \times V \to V$$

definiert ist, sodass für alle $r, s \in \mathbb{K}$ *und alle* $\vec{v}, \vec{w} \in V$ *gilt:*

$$\begin{aligned} (rs) \bullet \vec{v} &= r \bullet (s \bullet \vec{v}) \\ (r + s) \bullet \vec{v} &= r \bullet \vec{v} + s \bullet \vec{v} \\ r \bullet (\vec{v} + \vec{w}) &= r \bullet \vec{v} + r \bullet \vec{w} \\ 1 \bullet \vec{v} &= \vec{v}. \end{aligned}$$

Für jeden Körper $(\mathbb{K}, +, \cdot)$ ist $\mathbb{K}^n$ mit $n \in \mathbb{N}$ ein so genannter *Standard-Vektorraum.* Die Vektoren $\vec{v} \in V$ sind in diesem Fall n-dimensional. Die folgenden Definitionen gelten nun für Körper $\mathbb{K} = \mathbb{R}$ und $\mathbb{R}$-Vektorräume V bzw. Vektorräume $\mathbb{R}^n$.

Definition 2.5 (Erzeugendensystem)

Die Vektoren

$$\vec{e}_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \vec{e}_2 = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}, ..., \vec{e}_n = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}$$

heißen Einheitsvektoren *und bilden eine* Standardbasis *oder* kanonische Basis *von* $\mathbb{K}^n$*. Jeder Vektor* $\vec{v}$ *aus* $\mathbb{K}^n$ *kann geschrieben werden als*

$$\vec{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} = \vec{e}_1 v_1 + ... + \vec{e}_n v_n.$$

$\vec{v}$ *heißt dann* Linearkombination *der Vektoren* $\vec{e}_1, ..., \vec{e}_n$.

Allgemeiner formuliert gelangt man zu einem so genannten Erzeugendensystem:

Eine Menge von Vektoren $\vec{v}_1, ..., \vec{v}_n$ *heißt* Erzeugendensystem *eines* $\mathbb{K}$*-Vektorraumes* V*, wenn jeder Vektor* $\vec{x} = (\ x_1 \quad ... \quad x_n\ )^T$ *aus* V *geschrieben werden kann als*

$$\vec{x} = \vec{v}_1 x_1 + ... + \vec{v}_n x_n.$$

Eine Basis ist ein spezielles Erzeugendensystem. Auch sie enthält eine endliche Anzahl von Vektoren, die jedoch linear unabhängig voneinander sind.

Definition 2.6 (Lineare Unabhängigkeit)

Eine Familie von Vektoren $(\vec{v}_1, ..., \vec{v}_k)$ *aus einem* $\mathbb{K}$*-Vektorraum* V *heißt* linear abhängig*, wenn* $\lambda_1, ..., \lambda_k \in \mathbb{K}$ *so existieren, dass*

$$\lambda_1 \vec{v}_1 + \lambda_2 \vec{v}_2 + ... + \lambda_k \vec{v}_k = \vec{0}$$

und mindestens eines der $\lambda_i \neq 0$ *ist.*

Sie heißt linear unabhängig*, wenn* $\lambda_1 = \lambda_2 = ... = \lambda_k = 0$ *die einzige Lösung für* $\lambda_1 \vec{v}_1 + \lambda_2 \vec{v}_2 + ... + \lambda_k \vec{v}_k = \vec{0}$ *ist.*

Definition 2.7 (Basis)

Die maximal linear unabhängige Menge von Vektoren eines Erzeugendensystems heißt minimales *Erzeugendensystem oder* Basis *eines* $\mathbb{K}$*-Vektorraumes* V.

Die Basis $B = (\vec{b}_1, \vec{b}_2, ..., \vec{b}_k)$ eines $\mathbb{K}$-Vektorraumes V ist nicht eindeutig bestimmt; V kann mehrere Basen haben, die jedoch immer gleich viele Elemente besitzen. Diese Elementenzahl heißt *Dimension* von V über $\mathbb{K}$. Zusammenfassend sind folgende Aussagen äquivalent:

1. B ist eine Basis von V.

2. B ist ein minimales Erzeugendensystem von V.
3. B ist ein linear unabhängiges Erzeugendensystem von V.
4. B ist maximal linear unabhängig in V.

2.3 Lineare Gleichungssysteme

Ein Erzeugendensystem für eine Menge von Vektoren zu finden, hängt eng mit dem Lösen linearer Gleichungssysteme (LGS) zusammen. Diese haben in der Regel die Form $A \cdot \vec{x} = \vec{b}$. Die *rechte Seite* $\vec{b}$ entscheidet über die Gestalt des Gleichungssystems:

Definition 2.8 (Homogenes/Inhomogenes Gleichungssystem)

Ein lineares Gleichungssystem $A \cdot \vec{x} = \vec{0}$ *nennt man* homogenes *Gleichungssystem.*

Ein lineares Gleichungssystem $A \cdot \vec{x} = \vec{b}$ *mit* $\vec{b} \neq \vec{0}$ *nennt man* inhomogenes *Gleichungssystem.*

Insgesamt ergeben sich für ein System der Form $A \cdot \vec{x} = \vec{b}$ folgende Fragen:

(1) Gibt es eine Lösung?

(2) Wie findet man diese Lösung?

(3) Gibt es noch weitere Lösungen?

(4) Wie findet man alle weiteren Lösungen?

Die Lösbarkeit des Gleichungssystems ist vom Rang der Matrix A abhängig:

Definition 2.9 (Rang einer Matrix)

Unter dem Spaltenrang *einer Matrix A versteht man die Anzahl linear unabhängiger Spalten von A, unter* Zeilenrang *die Anzahl linear unabhängiger Zeilen. Da Zeilenrang und Spaltenrang einer Matrix identisch sind, spricht man allgemein vom* Rang *einer Matrix A, geschrieben* $rg(A)$.

Die Fragen (1) und (2) beziehen sich auf die Lösbarkeit inhomogener Gleichungssysteme, während Fragen (3) und (4) homogene LGS betreffen. Ein hilfreiches Verfahren zum Lösen linearer Gleichungssysteme ist die so genannte *Gauß-Elimination* (vgl. [Wei07b]). Es werden dabei *elementare Zeilenoperationen* so ausgeführt, dass das System $A \cdot \vec{x} = \vec{b}$ in ein System $A' \cdot \vec{x} = \vec{b'}$ überführt wird, wobei Matrix A' eine obere

Dreiecksmatrix ist und sich der Rang nicht ändert. Diese elementaren Zeilenoperationen sind

- das Vertauschen von Zeilen,
- die Multiplikation einer Zeile mit einem Faktor $k \in \mathbb{Z}\backslash\{0\}$,
- die Addition eines ganzzahligen Vielfachen einer Zeile zu einer anderen Zeile und
- das Vertauschen von Spalten - hierbei müssen die Indizes der Variablen umbenannt werden.

Durch Gauß-Elimination gelangt man letztendlich zu der in Abbildung 2.1 dargestellten Form, wobei $a'_{1,1}, a'_{2,2}, ..., a'_{r,r} \neq 0$ gilt. In den folgenden zwei Abschnitten wird die

$$\begin{pmatrix} a'_{1,1} & \cdots & \cdots & a'_{1,r} & \cdots & \cdots & a'_{1,n} \\ 0 & \ddots & & \vdots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & a'_{r,r} & \cdots & \cdots & a'_{r,n} \\ 0 & \cdots & \cdots & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & & \vdots & \ddots & & \vdots \\ 0 & \cdots & \cdots & 0 & \cdots & \cdots & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b'_1 \\ \vdots \\ \vdots \\ b'_r \\ b'_{r+1} \\ \vdots \\ b'_m \end{pmatrix}$$

Abbildung 2.1: *Resultat des Gaußschen Eliminationsverfahrens*

Lösung linearer Gleichungssysteme auf Basis der Gauß-Elimination und in Anlehnung an [Mnu07] diskutiert.

2.3.1 Lösbarkeit homogener Gleichungssysteme

Gegeben sei ein lineares homogenes Gleichungssystem $A \cdot \vec{x} = \vec{0}$, wobei die Matrix A aus m Zeilen und n Spalten bestehe und $r := rg(A)$ gelte. Das System lässt sich auch schreiben als

$$a_{*,1}x_1 + a_{*,2}x_2 + ... + a_{*,n}x_n = \vec{0}.$$

Man unterscheidet nun zwei Fälle:

1. Fall: $r = n$
A hat vollen Spaltenrang, die Spaltenvektoren $a_{*,1}, ..., a_{*,n}$ sind also linear unabhängig. Für die Unbekannten $x_1, ..., x_n$ gilt gemäß der Definition der linearen Unabhängigkeit: $x_1 = x_2 = ... = x_n = 0$. Das Gleichungssystem hat damit nur die triviale Lösung $\vec{x} = \vec{0}$.

2. Fall: $r < n$
Das System hat die Form

$$a_{*,1}x_1 + ... + a_{*,r}x_r + a_{*,r+1}x_{r+1} + ... + a_{*,n}x_n = \vec{0},$$

wobei o.B.d.A. die ersten r Spaltenvektoren von A linear unabhängig sind. Mit dem Gaußschen Eliminationsverfahren gelangt man zu einer Form wie in Abbildung 2.1 mit $\vec{b'} = \vec{0}$, das heißt die letzten $m - r$ Zeilen bestehen nur aus Nullen und haben die Form $0 = 0$. Die Variablen $x_{r+1}, ..., x_n$ können darum frei gewählt werden. Man setze

$$(\ x_{r+1} \quad x_{r+2} \quad ... \quad x_n\ )^T = (\ 1 \quad 0 \quad ... \quad 0\ )^T$$

und bestimme $x_1, ..., x_r$ so, dass der Vektor

$$\vec{x}_1 = (\ x_1 \quad ... \quad x_r \quad 1 \quad 0 \quad ... \quad 0\ )^T$$

eine Lösung des Systems $A' \cdot \vec{x} = \vec{0}$ ist. Die Prozedur wird fortgeführt für

$$\begin{aligned} \vec{x}_2 &= (\ x'_1 \quad ... \quad x'_r \quad 0 \quad 1 \quad 0 \quad 0 \quad ... \quad 0\ )^T, \\ \vec{x}_3 &= (\ x''_1 \quad ... \quad x''_r \quad 0 \quad 0 \quad 1 \quad 0 \quad ... \quad 0\ )^T \end{aligned}$$

bis

$$\vec{x}_{n-r} = (\ x'''_1 \quad ... \quad x'''_r \quad 0 \quad 0 \quad ... \quad 1\ )^T.$$

Die errechneten $n - r$ Vektoren $\vec{x}_1, ..., \vec{x}_{n-r}$ bilden eine *Basis* des homogenen Gleichungssystems, da sie aufgrund der Festlegung ihrer letzten $n - r$ Komponenten linear unabhängig sind. Alle Lösungen $\vec{x}$ von $A \cdot \vec{x} = \vec{0}$ sind damit gegeben als

$$\vec{x} = \sum_{i=1}^{n-r} \lambda_i \vec{x}_i, \quad \lambda_i \in \mathbb{K}.$$

2.3.2 Lösbarkeit inhomogener Gleichungssysteme

Ein inhomogenes lineares Gleichungssystem $A \cdot \vec{x} = \vec{b}$ hat entweder

- *genau eine eindeutige* Lösung ($rg(A) = rg([A|\vec{b}]) = n$),
- *unendlich viele* Lösungen ($rg(A) = rg([A|\vec{b}]) = r < n$)
- oder *keine* Lösung ($rg(A) < rg([A|\vec{b}]$).

Der Rang wird zunächst mit der Anzahl der Zeilen in Beziehung gesetzt und es ergeben sich drei Fälle. In Abbildung 2.2 sind die Kriterien graphisch zusammengefasst.

1. Fall: $r = m = n$
A hat vollen Zeilen- und Spaltenrang, somit hat das System $A \cdot \vec{x} = \vec{0}$ nur die triviale Lösung. Durch Gauß-Elimination erhält man das System

$$\begin{pmatrix} a'_{1,1} & \cdots & \cdots & a'_{1,n} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a'_{m,n} \end{pmatrix} \cdot \vec{x} = \begin{pmatrix} b'_1 \\ \vdots \\ \vdots \\ b'_m \end{pmatrix}.$$

Die letzte Matrixzeile ist eine Gleichung der Form

$$a'_{m,n} x_n = b'_m, \ (a'_{m,n} \neq 0),$$

sodass

$$x_n = \frac{b'_m}{a'_{m,n}}$$

errechnet werden kann. Durch sukzessives Einsetzen der ermittelten Werte in darüber liegende Gleichungen kann die *eindeutige und einzige Lösung* des Gesamtsystems bestimmt werden.

2. Fall: $r < m$ und $(b'_{r+1}, ..., b'_m) = (0, ..., 0)$
Durch Gauß-Elimination gelangt man zu dem System aus Abbildung 2.1. Die letzten $m - r$ Zeilen haben die Form $0 = 0$. Damit existiert eine partielle Lösung des inhomogenen Gleichungssystems, in diesem Fall der Vektor

$$\vec{x}_0 = (\ x_1 \quad ... \quad x_r \quad 0 \quad ... \quad 0\ )^T,$$

für dessen Elemente $x_{r+1} = ... = x_n = 0$ gilt.

Die verbleibenden Unbekannten $x_1, ..., x_r$ können durch sukzessives Einsetzen wie in Fall 1 gewonnen werden. Nun gibt es noch ein weiteres Kriterium, das über die Existenz weiterer Lösungen Aufschluss gibt:

(a) $r = n$
A hat vollen Spaltenrang, sodass der errechnete Vektor $\vec{x}_0$ die *eindeutige und einzige Lösung* des Gleichungssystems ist.

(b) $r < n$

Es gibt $n - r$ Vektoren, die zusätzlich zur partiellen Lösung $\vec{x}_0$ eine Basis des zugehörigen homogenen Gleichungssystems bilden. Das gesamte System hat damit *unendlich viele Lösungen*. Die Berechnung der Basis erfolgt wie in Fall 2 aus Abschnitt 2.3.1 (mit $\vec{b}' = \vec{0}$). Die allgemeine Lösung des Gleichungssystems $A \cdot \vec{x} = \vec{b}$ ist somit gegeben als

$$\vec{x} = \vec{x}_0 + \sum_{i=1}^{n-r} \lambda_i \vec{x}_i, \quad \lambda_i \in \mathbb{K}.$$

3. Fall: $r < m$ und $(b'_{r+1}, ..., b'_m) \neq (0, ..., 0)$
Durch Gauß-Elimination gelangt man zu dem System aus Abbildung 2.1. Nun existiert mindestens ein b'_i, $r + 1 \leq i \leq m$, mit $b'_i \neq 0$. Folglich hat mindestens eine der letzten $m - r$ Gleichungen die Form

$$0 = b'_i,$$

was einen Widerspruch darstellt. Das Gleichungssystem hat *keine Lösung*.

Obige Lösungsverfahren für lineare Gleichungssysteme im $\mathbb{R}^n$ ergeben nicht zwangsläufig ganzzahlige Lösungen. Durch die Transformation einer Matrix A (bzw. der erweiterten Matrix $[A| - \vec{b}]$) in ihre Hermite-Normalform ist es möglich, Aussagen über die Existenz ganzzahliger Lösungen für inhomogene und homogene Gleichungssysteme zu treffen und diese zu berechnen bzw. direkt aus einer resultierenden Matrix abzulesen.

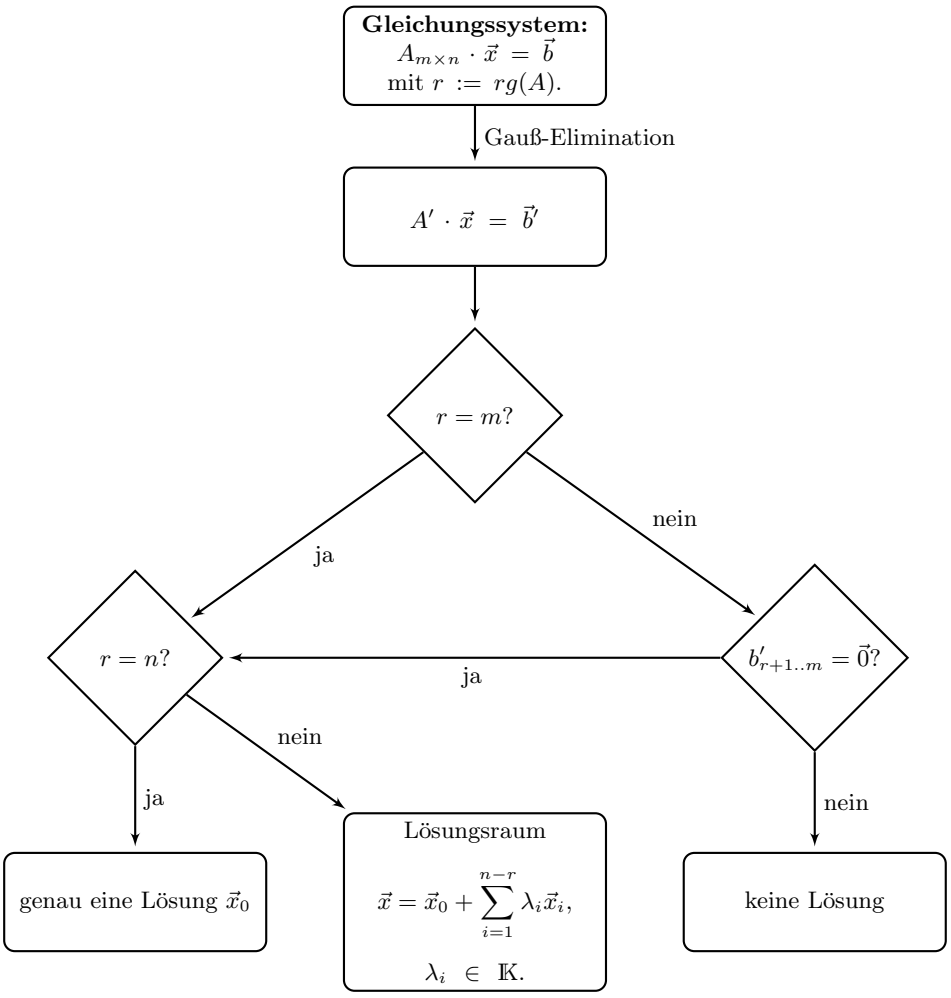

Abbildung 2.2: *Kriterien zur Lösung eines linearen inhomogenen Gleichungssystems, basierend auf der Gauß-Elimination*

3 Die Hermite-Normalform

Nachdem im vorherigen Kapitel erforderliche linear-algebraische Grundlagen gelegt wurden, wird nun das Kernelement dieser Arbeit vorgestellt: die Hermite-Normalform einer ganzzahligen Matrix. Sie wird später dazu verwendet, ganzzahlige Lösungen linearer Gleichungssysteme zu berechnen. In der mathematischen Literatur der letzten Jahrzehnte findet man eine Vielzahl varianter Definitionen der Hermite-Normalform (z.B. [Nem88, The00, Sch99]). Abschnitt 3.1 beinhaltet darum zunächst die ursprüngliche Definition von Charles Hermite selbst.

Die Hermite-Normalform wird in allen Fällen aus einer Ausgangsmatrix durch Anwendung unimodularer Zeilen- oder Spaltenoperationen erzeugt, welche man durch Matrizen beschreiben kann, und ist dadurch äquivalent zu ihrer Ausgangsmatrix. Entsprechende Definitionen und Erläuterungen sind in Kapitel 3.2 zu finden. Anschließend werden in Kapitel 3.3 variante Definitionen der HNF vorgestellt und durch Beispiele veranschaulicht, die jeweils einem der in Kapitel 4 präsentierten Algorithmen zugrunde liegen. Für Matrizen mit vollem Zeilenrang ist die Darstellung der HNF eindeutig; einen Beweis skizziert Kapitel 3.4.

3.1 Definition nach Charles Hermite

Bereits im Jahre 1851 veröffentliche Charles Hermite in seinem Artikel „*Sur l'introduction des variables continues dans la théorie des nombres*“ [Her51] die erste Definition der später nach ihm benannten Hermite-Normalform. Hermites Begriffsbestimmung bezieht sich auf invertierbare, quadratische Matrizen und beschreibt grundsätzlich nur eine diagonalisierte Matrix mit nichtnegativen Einträgen und ein paar weiteren Eigenschaften, auf die später genauer eingegangen wird. Nachfolgende Definitionen erweitern die Festlegungen auf rechteckige Matrizen, die nicht zwangsläufig vollen Zeilenrang besitzen und zum Teil nicht einmal vollständig nichtnegativ sind.

Definition 3.1 zeigt die Gestalt der Normalform, wie Hermite sie ursprünglich formulierte. Die geforderte Invertierbarkeit der Matrix ($\det \neq 0$) wird nicht explizit erwähnt, ist jedoch aus dem Zusammenhang des Artikels erkennbar und wurde in die Definition übernommen. Außerdem wurde der Begriff „Hermite-Normalform“ hinzugefügt.

Definition 3.1 (Hermite-Normalform)

Eine invertierbare Matrix $H \in \mathbf{Mat}_{n\times n}(\mathbb{Z})$ *mit*

$$H = \begin{pmatrix} \delta_0 & g & h & \cdots & l \\ 0 & \delta_1 & h' & \cdots & l' \\ 0 & 0 & \delta_2 & \cdots & l'' \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \delta_{n-1} \end{pmatrix}$$

ist in Hermite-Normalform*, wenn gilt:*

- *Die Diagonalelemente* δ_i*,* $0 \leq i \leq n-1$*, sind positiv.*
- *Die durch die Buchstaben* $g, h, ..., l$ *repräsentierten Elemente sind positiv und erfüllen die Bedingungen*

$$g < \delta_1,\ h < \delta_2,\ ...,\ l < \delta_{n-1}.$$

Um eine Matrix nach dieser Definition in Hermite-Normalform zu bringen, wird eine obere Dreiecksmatrix erzeugt, d.h. alle Elemente unterhalb der Diagonalen werden „annulliert“. Dieser Vorgang wird als *Elimination* bezeichnet. In einem weiteren Schritt muss die Nichtnegativität aller restlichen Elemente hergestellt werden; außerdem soll jedes Element einer Spalte kleiner sein als das Diagonalelement dieser Spalte. Man nennt diesen Vorgang *Normalisierung*.

Jeder der in Kapitel 4 vorgestellten Algorithmen führt Eliminations- und Normalisierungsschritte aus, um eine Hermite-Normalform in der Gestalt der jeweils zugrunde liegenden Definition zu erzeugen.

3.2 Matrixäquivalenz und unimodulare Matrizen

Erzeugt man die Hermite-Normalform aus einer Ausgangsmatrix, so können die dazu notwendigen Zeilen- oder Spaltenoperationen durch spezielle, so genannte *unimodulare* Matrizen ausgedrückt werden. Eine besondere Eigenschaft der Hermite-Normalform ist dadurch ihre *Äquivalenz* zu der Matrix, aus der sie erzeugt wurde. Diese Äquivalenzrelation ist von der Existenz der unimodularen Matrix abhängig.

Definition 3.2 (Unimodulare Matrix)

Eine Matrix $U \in \mathbf{Mat}_{n\times n}(\mathbb{Z})$ *heißt* unimodular*, wenn* $\det U = \pm 1$ *ist.*

$GL_n(\mathbb{Z})$ bezeichnet die Menge aller unimodularen Matrizen $U \in \boldsymbol{Mat}_{n\times n}(\mathbb{Z})$.

Definition 3.3 (Matrixäquivalenz)

Seien $A, B \in \boldsymbol{Mat}_{m\times n}(\mathbb{Z})$. B heißt rechtsäquivalent *zu A, wenn eine unimodulare Matrix $U \in GL_n(\mathbb{Z})$ existiert, sodass gilt: $B = A \cdot U$.*

Analog ist B linksäquivalent *zu A, wenn eine unimodulare Matrix $U \in GL_m(\mathbb{Z})$ existiert, sodass $B = U \cdot A$ gilt.*

In dieser Arbeit heißen diese unimodularen Matrizen von nun an *Transformationsmatrizen* (siehe auch Definition 3.8) oder *Permutationsmatrizen*. Unimodulare Zeilen- und Spaltenoperationen können bereits durch sehr einfache Permutationsmatrizen beschrieben werden, was im folgenden Abschnitt veranschaulicht wird.

3.2.1 Permutationsmatrizen

Grundsätzlich erzeugt man eine entsprechende unimodulare Matrix U zu einer Matrix $A_{m\times n}$ immer aus einer Einheitsmatrix I_n (Rechtsäquivalenz, für Spaltenoperationen) bzw. I_m (Linksäquivalenz, für Zeilenoperationen). Folgende Beschreibungen beziehen sich auf Permutationsmatrizen U basierend auf $U := I_n$ und beschreiben unimodulare Spaltenoperationen. Das Konstruktionsverfahren von U für Zeilenoperationen, ausgehend von $U := I_m$, ist identisch.

Anmerkung:
Unimodulare Operationen unterscheiden sich von den elementaren Zeilen- oder entsprechenden Spaltenoperationen aus Kapitel 2.3 dadurch, dass bei der Multiplikation mit einem Faktor $z \in \mathbb{Z}\backslash\{0\}$ nur $z = \pm 1$ erlaubt ist. Sonst ist die Bedingung $\det U = \pm 1$ verletzt.

(1) Vertauschen zweier Spalten

Zum Vertauschen zweier Spalten k und l einer Matrix $A_{m\times n}$ wird die Permutationsmatrix U_n wie folgt konstruiert:

1. Man setzt

$$\begin{aligned} u_{k,k} &:= 0 \\ u_{k,l} &:= 1 \\ u_{l,l} &:= 0 \end{aligned}$$

$$u_{l,k} := 1.$$

2. Nun multipliziert man A mit U und erhält das gewünschte Ergebnis.

Beispiel:

$$A_1 = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Es sollen die Spalten 2 und 3 vertauscht werden. U_3 wird wie oben beschrieben konstruiert und man erhält folgendes Ergebnis:

$$A_2 = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \mathbf{0} & \mathbf{1} \\ 0 & \mathbf{1} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} 1 & 3 & 2 \\ 4 & 6 & 5 \\ 7 & 9 & 8 \end{pmatrix}$$

(2) Multiplikation einer Spalte mit (-1)

Zum Multiplizieren einer Spalte k mit (-1) ändert man in U_n folgenden Eintrag:

$$u_{k,k} := -1.$$

Beispiel:

$$A_2 = \begin{pmatrix} 1 & 3 & 2 \\ 4 & 6 & 5 \\ 7 & 9 & 8 \end{pmatrix}$$

Spalte 2 soll mit (-1) multipliziert werden. U_3 wird wie beschrieben konstruiert und man erhält folgendes Ergebnis:

$$A_3 = \begin{pmatrix} 1 & 3 & 2 \\ 4 & 6 & 5 \\ 7 & 9 & 8 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \mathbf{-1} & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -3 & 2 \\ 4 & -6 & 5 \\ 7 & -9 & 8 \end{pmatrix}$$

(3) Addition eines Vielfachen einer Spalte zu einer anderen Spalte

Zur Addition eines Vielfachen $z \in \mathbb{Z}$ einer Spalte k zu einer anderen Spalte l setzt man in U_n

$$u_{k,l} := z.$$

Beispiel:

$$A_3 = \begin{pmatrix} 1 & -3 & 2 \\ 4 & -6 & 5 \\ 7 & -9 & 8 \end{pmatrix}$$

Das Zweifache ($z = 2$) von Spalte $k = 2$ soll zu Spalte $l = 3$ addiert werden. Konstruiert man U_3 wie beschrieben, so sieht A nach der Matrixmultiplikation wie folgt aus:

$$A_4 = \begin{pmatrix} 1 & -3 & 2 \\ 4 & -6 & 5 \\ 7 & -9 & 8 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & \mathbf{2} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -3 & -4 \\ 4 & -6 & -7 \\ 7 & -9 & -10 \end{pmatrix}$$

(4) Hintereinanderausführung aller 3 Operationen

Die Hintereinanderausführung aller drei Operationen kann ebenfalls durch eine unimodulare Matrix ausgedrückt werden, indem man die drei Permutationsmatrizen miteinander multipliziert. Man beachte hierbei, dass Matrixmultiplikation allgemein nicht kommutativ ist.

$$U_{gesamt} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \mathbf{0} & \mathbf{1} \\ 0 & \mathbf{1} & \mathbf{0} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \mathbf{-1} & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & \mathbf{2} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & -2 \end{pmatrix}$$

Damit erhält man:

$$A_1 \cdot U_{gesamt} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & -2 \end{pmatrix} = \begin{pmatrix} 1 & -3 & -4 \\ 4 & -6 & -7 \\ 7 & -9 & -10 \end{pmatrix} = A_4$$

3.3 Variante Definitionen der Hermite-Normalform

Nach der ursprünglichen Definition von Hermite folgen nun vier variante Definitionen der Hermite-Normalform. Zunächst beschäftigt sich Abschnitt 3.3.1 mit Hermite-Normalformen für Matrizen mit vollem Zeilenrang. Die Definitionen sind leicht zu verstehen. Komplizierter wird es bei beliebigen ganzzahligen Matrizen. Die Problematik wird in Abschnitt 3.3.2 erläutert. Insgesamt erweitern alle Definitionen die Beschreibung der Hermite-Normalform auf nicht quadratische Matrizen und haben drei wichtige Gemeinsamkeiten: Die HNF ist jeweils eine *untere* oder *obere* Dreiecksmatrix mit *positiven* Diagonalelementen und die Absolutwerte der zu normalisierenden Einträge sind *kleiner* als das zugehörige Diagonalelement.

3.3.1 Definitionen für ganzzahlige Matrizen mit vollem Zeilenrang

Die erste variante Definition, die nun vorgestellt wird, stammt aus [Nem88] und beschreibt eine untere Dreiecksmatrix. Es wird hier bereits auf die Existenz einer unimodularen Matrix U eingegangen, die eine Matrix A in ihre Hermite-Normalform transformiert.

Definition 3.4 (Hermite-Normalform – Variante 1)

Eine nichtsinguläre $(m \times m)$-Matrix H ist in Hermite-Normalform*, wenn*

1. *H eine untere Dreiecksmatrix ist, also $h_{i,j} = 0$ für $i < j$ gilt,*
2. *$h_{i,i} > 0$ für $i = 1, ..., m$ und*
3. *$h_{i,j} \leq 0$ und $|h_{i,j}| < h_{i,i}$ für $i > j$.*

Ist A eine ganzzahlige $(m \times n)$-Matrix mit $rg(A) = m$, dann existiert eine unimodulare $(n \times n)$-Matrix U, sodass

4. *$A \cdot U = (H, 0)$ und H ist in Hermite-Normalform;*
5. *$H^{-1} \cdot A$ ist eine ganzzahlige Matrix.*

$(H, 0)$ ist die Hermite-Normalform von A.

Neben der Tatsache, dass hier eine untere Dreiecksmatrix erzeugt wird, beschreibt Punkt 3 einen wichtigen Unterschied zur Definition von Hermite: Die Einträge sind zwar normalisiert und kleiner als das entsprechende Diagonalelement, in diesem Fall aber negativ. Die Punkte 4 und 5 dieser Definition erweitern die Beschreibung der Hermite-Normalform zusätzlich auf $(m \times n)$-Matrizen A mit $m \leq n$. In diesem Fall hat die HNF die Gestalt $(H, 0)$, d.h. die letzten $n - m$ (denn $rg(A) = m$) Spalten sind Nullspalten. Demnach kann der Rang der Matrix auch anhand der Anzahl der Nullspalten sofort abgelesen werden.

Die folgende Definition stammt aus [Sch99] und beschreibt eine fast identische HNF.

Definition 3.5 (Hermite-Normalform – Variante 2)

Eine $(m \times n)$-Matrix A mit $rg(A) = m$ ist in Hermite-Normalform*, wenn sie die Form*

$$\begin{bmatrix} B & 0 \end{bmatrix}$$

hat, wobei B eine nichtsinguläre, nichtnegative untere Dreiecksmatrix ist, in der jede Zeile einen eindeutigen maximalen Eintrag hat, der sich auf der Hauptdiagonalen von B befindet.

Der einzige Unterschied liegt in der Nichtnegativität der zu normalisierenden Matrixeinträge. Auch hier werden $n - m$ Nullspalten erzeugt, wenn $rg(A) = m < n$ gilt. Andernfalls gilt $HNF(A) = B$.

3.3.2 Definitionen für beliebige ganzzahlige Matrizen

Bereits in Kapitel 2.3 wurde die Gestalt einer Matrix nach Anwendung eines Gaußschen Eliminationsverfahrens beschrieben (vgl. Abbildung 2.1). Für eine Matrix A mit $rg(A) < m$ sind die ersten $rg(A)$ Zeilen in oberer Dreiecksform, die restlichen Zeilen Nullzeilen. Wie die Gauß-Elimination ist auch die Konstruktion der Hermite-Normalform ein Diagonalisierungsverfahren. Was passiert nun bei $(m \times n)$-Matrizen A mit $rg(A) < m$, wenn sie in HNF gebracht werden? Hier können nur $rg(A)$ Zeilen ein Diagonalelement besitzen. Zusätzlich liegen diese nicht gezwungenermaßen auf den Positionen $a_{i,i}$; sie werden darum von nun an auch als *Pseudodiagonalelemente* bezeichnet.

In diesem Kapitel werden zwei HNF-Definitionen vorgestellt, die diese Problematik formal beschreiben. Da sie nicht auf Anhieb verständlich sein mögen, werden sie jeweils anhand eines kleinen Beispiels erläutert. Die folgende Definition stammt aus [The00] und beschreibt die Hermite-Normalform als obere Dreiecksmatrix.

Definition 3.6 (Hermite-Normalform – Variante 3)

Eine Matrix $A \in \boldsymbol{Mat}_{m\times n}(\mathbb{Z})$ ist in Hermite-Normalform*, wenn ein $r \leq n$ und eine streng monoton wachsende Funktion $f : [r+1, n] \to [1, m]$ existieren[1], die den folgenden Eigenschaften genügen:*

1. *Für $r < j \leq n$ gilt:*
   (a) $a_{f(j),j} \geq 1$,
   (b) $a_{i,j} = 0$ *für* $i > f(j)$ *(Elimination) und*
   (c) $0 \leq a_{f(k),j} < a_{f(k),k}$ *für* $r < k < j$ *(Normalisierung).*
2. *Die ersten r Spalten von A sind Nullspalten, d.h. sie bestehen nur aus Nullelementen.*

[1] Hinweis: r bezeichnet hier nicht den Rang von A, sondern es gilt: $r = n - rg(A)$.

Was bedeutet diese Definition nun für die Matrix

$$A = \begin{pmatrix} -2 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{pmatrix}$$

und deren zugehörige HNF

$$HNF(A) = H = \begin{pmatrix} 0 & -1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}?$$

Zunächst handelt es sich um eine Matrix mit $rg(A) = 2$, somit gilt $r = n - rg(A) = 3 - 2 = 1$. Die erste Spalte der HNF von A ist somit eine Nullspalte (Eigenschaft 2). Die Funktion f hat die Form

$$f : [2, 3] \to [1, 3]$$

und beschreibt die Position der Pseudodiagonalelemente in der HNF. Der Definitionsbereich $[2, 3]$ besagt, dass sie in diesem Fall in der zweiten und dritten Spalte liegen müssen. Die Funktion ist streng monoton wachsend, d.h. für zwei Diagonalelemente in den Spalten i und j mit $i < j$ gilt: $f(i) < f(j)$. Dadurch ist gewährleistet, dass eine obere Dreiecksmatrix erzeugt wird. Für obige HNF ergeben sich folgende Funktionswerte:

$$\begin{aligned} f(2) &= 2 \\ f(3) &= 3. \end{aligned}$$

Die zwei Pseudodiagonalelemente dieser HNF liegen damit auf den Positionen $h_{2,2}$ und $h_{3,3}$; die erste Zeile von H besitzt kein Diagonalelement. Für $1 < j \leq 3$ gilt damit (Eigenschaft 1):

- $j = 2$:
  (a) $h_{2,2} = 1 \geq 1$,
  (b) $h_{i,2} = 0$ für $i > f(2) = 2$, d.h. $h_{3,2} = 0$.
  (c) $0 \leq h_{f(k),2} < h_{f(k),k}$ für $k < 2$. Für $k = 1$ ist f nicht definiert, d.h. für die erste Zeile von H gilt diese Eigenschaft nicht.
- $j = 3$:
  (a) $h_{3,3} = 1 \geq 1$,

(b) $h_{i,3} = 0$ für $i > f(3) = 3$. Es gibt keine Zeile $i > 3$, somit muss diese Eigenschaft nicht weiter betrachtet werden.

(c) $0 \leq h_{f(k),3} < h_{f(k),k}$ für $k < 3$. Für den Fall $k = 2$ gilt: $h_{f(2),3} = h_{2,3} = 0 < h_{f(2),2} = h_{2,2}$.

Zwei weitere Definitionen, die die Hermite-Normalform für beliebige ganzzahlige Matrizen beschreiben, stammen aus [Jäg01] und hängen eng miteinander zusammen. Definition 3.7.1 beschreibt die HNF als untere Dreiecksmatrix, Definition 3.7.2 die so genannte Zeilen-Hermite-Normalform, bei der aus der transponierten Ausgangsmatrix eine obere Dreiecksmatrix erzeugt wird.

Definition 3.7 (Hermite-Normalform – Variante 4)

1. *Eine ganzzahlige Matrix $A_{m\times n}$ mit Rang r ist in* Hermite-Normalform*, wenn gilt:*
   (a) $\exists i_1, ..., i_r$ *mit* $1 \leq i_1 < ... < i_r \leq m$ *mit* $a_{i_j,j} > 0$ *für* $1 \leq j \leq r$.
   (b) $a_{i,j} = 0$ *für* $1 \leq i \leq i_j - 1$, $1 \leq j \leq r$.
   (c) Die Spalten $r + 1$ *bis* n *sind Nullvektoren.*
   (d) $a_{i_j,l} = a_{i_j,l} - \left\lfloor \frac{a_{i_j,l}}{a_{i_j,j}} \right\rfloor \cdot a_{i_j,j}$ *für* $1 \leq l < j \leq r$.
2. *Die Matrix A ist in* Zeilen-Hermite-Normalform *(abgekürzt: LHNF), wenn ihre transponierte Matrix A^T in Hermite-Normalform ist.*

Auch diese Definition mag nicht auf Anhieb verständlich sein. Angenommen, die Ausgangsmatrix sei

$$A_{3\times 3} = \begin{pmatrix} -2 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{pmatrix}$$

mit $r = rg(A) = 2$. Die zugehörige HNF von A und LHNF von A^T sehen wie folgt aus:

$$HNF(A) = H = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & -1 & 0 \end{pmatrix}, \quad LHNF(A^T) = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix}.$$

Bei den Elementen $a_{i_j,j}$ mit $1 \leq j \leq r$ (Def. 3.7.1(a)) handelt es sich wieder um die *Pseudodiagonalelemente*, da die Diagonalelemente bei Matrizen ohne vollen Zeilenrang nicht zwingend auf der Hauptdiagonalen liegen müssen. In der Matrix H sind es also die Elemente

$$h_{i_j,j},\ 1 \leq j \leq 2$$

und damit

$$h_{i_1,1} = h_{1,1},\ h_{i_2,2} = h_{2,2}.$$

Weiterhin gilt gemäß Definition 3.7.1(b), dass $h_{i,j} = 0$ ist für $1 \leq i \leq i_j - 1$ und $1 \leq j \leq 2$, d.h. die Einträge oberhalb eines Pseudodiagonalelements sind Null. Im Detail gilt also für

- $j = 1$: $h_{i,1} = 0$ für $1 \leq i \leq i_1 - 1 = 0$. Die Grenzen für i sind nicht erfüllbar, also muss dieser Fall nicht weiter betrachtet werden.
- $j = 2$: $h_{i,2} = 0$ für $1 \leq i \leq i_2 - 1 = 1$, also $h_{1,2} = 0$.

Da A den Rang 2 hat, sind gemäß Definition 3.7.1(c) Die Spalten $r + 1 = 3$ bis $n = 3$ Nullspalten. Definition 3.7.1(d) bezieht sich auf die Normalisierung der Matrix. Für $1 \leq l < j \leq 2$ (also für $l = 1$ und $j = 2$) gilt in diesem Fall

$$h_{i_2,1} = h_{2,1} := h_{2,1} - \left\lfloor \frac{h_{2,1}}{h_{2,2}} \right\rfloor \cdot h_{2,2},$$

d.h. die Einträge links von einem Pseudodiagonalelement sind normalisiert. Schlussendlich bleibt noch die dritte Zeile von H übrig, die kein Diagonalelement besitzt. Sie ergibt sich durch die Tatsache, dass $rg(A) = 2 < m$ ist.

Gemäß Definition 3.7.2 ist die Zeilen-Hermite-Normalform ($LHNF(A^T)$) die transponierte Matrix zu H. Die zugehörigen Transformationsmatrizen, die die Ausgangsmatrizen in HNF bzw. LHNF bringen, sind nach [Jäg01] wie folgt definiert:

Definition 3.8 (Transformationsmatrix)

Sei $A_{m\times n}$ eine ganzzahlige Matrix. Dann gibt es eine ganzzahlige unimodulare Matrix $U_{n\times n}$ derart, dass $H = A \cdot U$ in HNF ist.

Sei $A_{n\times m} := (A_{m\times n})^T$ eine ganzzahlige Matrix. Dann gibt es eine ganzzahlige unimodulare Matrix $U_{n\times n}$ derart, dass $H = U \cdot A$ in LHNF ist.

Bei der Berechnung der Zeilen-Hermite-Normalform wird also durch unimodulare Zeilenoperationen eine zu A linksäquivalente Matrix H erzeugt (vgl. Def. 3.3).

Bemerkung:
Die Gestalt der Zeilen-Hermite-Normalform entspricht grundsätzlich der ursprünglichen Definition von Hermite. Nicht quadratische Matrizen oder Matrizen ohne vollen Zeilenrang haben nur durch Nullspalten und $m - rg(A)$ zusätzliche Zeilen eine leicht

veränderte Gestalt. Es gilt allerdings zu beachten, dass eine aus einer $(m \times n)$-Matrix erzeugte Zeilen-Hermite-Normalform eine andere Bedeutung hat als die anderen hier vorgestellten HNF, da sie als einzige durch unimodulare *Zeilen*operationen gebildet wurde. Dies hat später erheblichen Einfluss auf deren Verwendbarkeit bei der Lösung linearer Gleichungssysteme: Durch die Linksäquivalenz wird kein $(m \times n)$-, sondern das transponierte $(n \times m)$-Gleichungssystem gelöst.

3.3.3 Reduktion beliebiger ganzzahliger Matrizen zu Matrizen mit vollem Zeilenrang

Zwei der in Kapitel 4 vorgestellten Algorithmen arbeiten nur auf Matrizen mit vollem Zeilenrang und erzeugen Hermite-Normalformen gemäß der Definitionen aus Abschnitt 3.3.1. Im Hinblick auf das Lösen linearer Gleichungssysteme mittels der Hermite-Normalform können jedoch $(m \times n)$-Matrizen, für die $r = rg(A) < m$ gilt, zu vollem Zeilenrang reduziert werden, indem man $m-r$ Zeilen löscht[2]. Dabei gilt es zu beachten, dass die „richtigen“ Zeilen eliminiert werden, sodass in jedem Fall genau r zueinander linear unabhängige Zeilen übrig bleiben. Soll beispielsweise die Matrix

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

mit $r = rg(A) = 2$ zu vollem Zeilenrang reduziert werden, so ist es unzulässig, die 2. und 4. Zeile zu löschen. Sowohl die verbleibenden Zeilen 1 und 3 sind Vielfache voneinander, als auch die Zeilen 2 und 4. Zulässige reduzierte Matrizen wären in diesem Fall

$$A_1 = \begin{pmatrix} 0 & 0 & 2 & -3 \\ 2 & -3 & 4 & -1 \end{pmatrix},\ A_2 = \begin{pmatrix} 0 & 0 & 2 & -3 \\ -4 & 6 & -8 & 2 \end{pmatrix},$$

$$A_3 = \begin{pmatrix} 2 & -3 & 4 & -1 \\ 0 & 0 & -4 & 6 \end{pmatrix},\ A_4 = \begin{pmatrix} 0 & 0 & -4 & 6 \\ -4 & 6 & -8 & 2 \end{pmatrix}.$$

Reduziert man eine beliebige Matrix A zu vollem Zeilenrang, können sämtliche in dieser Arbeit präsentierten Algorithmen auf A angewandt werden.

[2] Hier wird davon ausgegangen, dass das Gleichungssystem $A \cdot \vec{x} = \vec{b}$ mindestens eine Lösung hat, also gemäß Kapitel 2.3.2 gilt: $rg(A) = rg([A|\vec{b}])$.

3.4 Eindeutigkeit der Hermite-Normalform

In diesem Abschnitt wird die Eindeutigkeit der Hermite-Normalform für Matrizen mit vollem Zeilenrang bewiesen. Dazu müssen zunächst einige Begriffe erklärt werden. Die Definitionen und der Beweis stammen aus [Sch99] und beziehen sich daher auf Definition 3.5 der Hermite-Normalform. Für eine $(m \times n)$-Matrix A mit vollem Zeilenrang hat $B \in \mathbf{Mat}_{m\times m}(\mathbb{N})$ die Gestalt

$$B = \begin{pmatrix} b_{1,1} & 0 & \cdots & 0 \\ < b_{2,2} & b_{2,2} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ < b_{m,m} & < b_{m,m} & \cdots & b_{m,m} \end{pmatrix}.$$

Es handelt sich um eine untere Dreiecksmatrix mit positiven Diagonalelementen. Die Einträge einer Zeile sind positiv und kleiner als das Diagonalelement der Zeile.

Definition 3.9 (Gitter)

Eine Untermenge Δ *des* $\mathbb{R}^n$ *heißt* additive Gruppe, *wenn:*

(i) $\vec{0} \in \Delta$.

(ii) Wenn $\vec{x}, \vec{y} \in \Delta$, *dann auch* $\vec{x} + \vec{y} \in \Delta$ *und* $-\vec{x} \in \Delta$.

Diese Gruppe nennt man erzeugt von $\vec{a}_1, ..., \vec{a}_m$, *wenn*

$$\Delta = \{\lambda_1\vec{a}_1 + \ldots + \lambda_m\vec{a}_m | \lambda_1, ..., \lambda_m \in \mathbb{Z}\}.$$

Man nennt sie Gitter, *wenn sie durch linear unabhängige Vektoren erzeugt werden kann. Diese Vektoren nennt man dann* Basis *des Gitters.*

Für eine Matrix $A \in \mathbf{Mat}_{m\times n}(\mathbb{Q})$ *mit vollem Zeilenrang*[3] *und ihre Hermite-Normalform* $[\, B \;\; 0 \,]$ *gilt nun, dass m linear unabhängige Spalten von A die gleiche Gruppe erzeugen wie die Spalten von B ([Sch99, S. 47]). Sind* $\vec{a}_1, ..., \vec{a}_m$ *rationale Vektoren, so erzeugen sie ein Gitter.*

Theorem 3.1 (Eindeutigkeit der Hermite-Normalform)

Seien $A, A' \in \mathbf{Mat}_{m\times n}(\mathbb{Q})$ *Matrizen mit vollem Zeilenrang und zugehörigen Hermite-Normalformen* $[\, B \;\; 0 \,]$ *bzw.* $[\, B' \;\; 0 \,]$. *Dann erzeugen die Spalten von A das gleiche Gitter wie die Spalten von* A' *genau dann, wenn* $B = B'$.

[3] Es gilt $\mathbb{Z} \subset \mathbb{Q}$, dadurch ist eine ganzzahlige Matrix automatisch auch eine rationale Matrix.

Beweis:
A und B bzw. A' und B' erzeugen nach Definition 3.9 jeweils das gleiche Gitter. Angenommen, A und A' erzeugen das gleiche Gitter Δ. Dadurch gilt das gleiche für B und B', da sie durch unimodulare Operationen aus A und A' entstanden sind. Sei nun

$$B =: (\beta_{i,j}),\ B' =: (\beta'_{i,j}).$$

Beweis durch Widerspruch: Gestartet wird mit der Annahme

$$B \neq B'.$$

Man wähle das erste Element

$$\beta_{i,j} \neq \beta'_{i,j}$$

mit möglichst kleinem i, d.h. die j-ten Spalten von B und B' unterscheiden sich in der i-ten Zeile, wohingegen sie in den Zeilen 1 bis $i-1$ identisch sind:

$$\left|\begin{array}{c|c|c} \beta_{1,j} & = & \beta'_{1,j} \\ \vdots & \vdots & \vdots \\ \beta_{i-1,j} & = & \beta'_{i-1,j} \\ \beta_{i,j} & \neq & \beta'_{i,j} \\ \beta_{i+1,j} & = \vee \neq & \beta'_{i+1,j} \\ \vdots & \vdots & \vdots \end{array}\right|$$

O.B.d.A. sei

$$\beta_{i,i} \geq \beta'_{i,i}$$

Seien nun $b_{*,j}$ und $b'_{*,j}$ die j-ten Spalten von B und B'. Da B und B' das gleiche Gitter Δ erzeugen, gilt

$$b_{*,j} \in \Delta,\ b'_{*,j} \in \Delta$$

und somit auch

$$b_{*,j} - b'_{*,j} \in \Delta.$$

Daraus folgt, dass $b_{*,j} - b'_{*,j}$ eine ganzzahlige Linearkombination von Spalten aus B ist, da die Spalten von B das Gitter Δ erzeugen. Der Vektor $b_{*,j} - b'_{*,j}$ hat Nullen in den

ersten $i-1$ Zeilen, denn die ersten $i-1$ Elemente sind in $b_{*,j}$ und $b'_{*,j}$ identisch. Da B eine untere Dreiecksmatrix ist, kann $b_{*,j} - b'_{*,j}$ nur eine ganzzahlige Linearkombination der Spalten $i, ..., m$ von B sein, denn eine Linearkombination mit den ersten $i-1$ Spalten würde für positive oder negative Einträge in den ersten $i-1$ Zeilen sorgen.

Dadurch kann $\beta_{i,j} - \beta'_{i,j}$ nur ein *ganzzahliges Vielfaches* x von $\beta_{i,i}$ sein, also

$$x \cdot \beta_{i,i} = \beta_{i,j} - \beta'_{i,j} \quad \Rightarrow \quad x = \frac{\beta_{i,j} - \beta'_{i,j}}{\beta_{i,i}},$$

da in den Spalten $i+1$ bis m von B Nullen in der i-ten Zeile stehen. Man unterscheidet nun zwei Fälle:

- Im Fall $j = i$ gilt $\beta'_{i,i} < \beta_{i,i}$, also (man beachte die Nichtnegativität von $\beta_{i,j}$ und $\beta'_{i,j}$)
$$\beta_{i,i} > \beta_{i,j} - \beta'_{i,j} = \beta_{i,i} - \beta'_{i,i} > 0.$$
Damit kann $\beta_{i,j} - \beta'_{i,j}$ kein ganzzahliges Vielfaches x von $\beta_{i,i}$ sein.

- Im Fall $j < i$ ist $0 \leq \beta_{i,j} < \beta_{i,i}$ sowie $0 < \beta'_{i,j} < \beta'_{i,i} \leq \beta_{i,i}$. Hier gilt also
$$0 < |\beta_{i,j} - \beta'_{i,j}| < \beta_{i,i},$$
die Differenz $\beta_{i,j} - \beta'_{i,j}$ kann betragsmäßig nur kleiner als $\beta_{i,i}$ und damit ebenfalls kein ganzzahliges Vielfaches sein.

Die Ganzzahligkeit von x ist insgesamt in *keinem* Fall erfüllt.

Die Hermite-Normalformen B und B' müssen also unter der Annahme, dass A und A' das gleiche Gitter erzeugen, identisch sein. Setzt man in Theorem 3.1 $A = A'$ und geht auf dieser Grundlage den Beweis erneut durch, so folgt daraus die Eindeutigkeit der Hermite-Normalform: Wenn A und A' identisch sind, müssen deren Hermite-Normalformen B und B' ebenfalls identisch sein. Folglich ist die HNF eindeutig bestimmt. □

Die Eindeutigkeit der Hermite-Normalform wurde *speziell* für Definition 3.5 bewiesen. Dies bedeutet gleichzeitig, dass alle Verfahren, die eine HNF dieser Gestalt aus einer Matrix A mit vollem Zeilenrang m erzeugen, dieselbe HNF erzeugen. Definition 3.7 beschreibt im Fall $rg(A) = m$ eine identische HNF: Es gibt in diesem Fall keine Zeilen ohne Diagonalelement, d.h. die gesamte Matrix ist nichtnegativ und entspricht in allen anderen Punkten der Definition von Schrijver.

Und noch eine weitere Eigenschaft der varianten HNF-Definitionen folgt aus diesem Theorem für Matrizen mit vollem Zeilenrang: Bei gegebener Matrix A (mit $rg(A) = m$) und deren HNF $(H, 0)$, welche z.B. die Form gemäß Definition 3.4 hat, erzeugen die Spalten von A und H das gleiche Gitter. Erzeugt man aus A bzw. $(H, 0)$ eine HNF $[B_A, 0]$ bzw. $[B_H, 0]$ gemäß Definition 3.5, so gilt $[B_A, 0] = [B_H, 0]$.

Für Matrizen mit vollem Zeilenrang lässt sich also jede in dieser Arbeit vorgestellte Variante der Hermite-Normalform durch unimodulare Spaltenoperationen aus einer anderen varianten HNF erzeugen. Die unterschiedlich definierten Hermite-Normalformen sind somit bei vollem Zeilenrang rechtsäquivalent zueinander.

4 Algorithmen zur Berechnung der Hermite-Normalform

In diesem Kapitel werden vier Algorithmen vorgestellt, die zu einer gegebenen Matrix A deren zugehörige Hermite-Normalform H berechnen. Es handelt sich hierbei um die HNF-Algorithmen, die Normalformen passend zu den in Kapitel 3.3 varianten Definitionen der Hermite-Normalform erzeugen. Keiner dieser Algorithmen zählt nach heutigem Standard zu den effizientesten, wenn Matrizen mit mehreren Tausend Zeilen und Spalten bearbeitet werden. Modernere Verfahren findet man z.B. unter [The00], [Mic01] oder [Sto96]. Da in dieser Arbeit jedoch die Berechnung ganzzahliger Lösungen linearer Gleichungssysteme mittels der Hermite-Normalform diskutiert wird, wurden relativ leicht verständliche und gut miteinander vergleichbare Algorithmen ausgewählt.

Es wird zwischen Algorithmen für ganzzahlige Matrizen mit vollem Zeilenrang (Abschnitt 4.1) und beliebige ganzzahlige Matrizen (Abschnitt 4.2) unterschieden. Jeder Algorithmus wird zunächst ausführlich vorgestellt und danach anhand eines vollständigen Beispielablaufs veranschaulicht. Abschließend werden die Ergebnisse verglichen.

4.1 HNF-Algorithmen für Matrizen mit vollem Zeilenrang

Die beiden in diesem Abschnitt präsentierten Algorithmen erzeugen Hermite-Normalformen gemäß der Definitionen 3.4 von Nemhauser/Wolsey [Nem88] und 3.5 von Schrijver [Sch99] und erwarten als Eingabe ganzzahlige Matrizen mit vollem Zeilenrang. Beide erzeugen eine Hermite-Normalform in unterer Dreiecksform. Im ersten Algorithmus wird die zugehörige unimodulare Transformationsmatrix automatisch mitberechnet; bei der Methode von Schrijver erfolgt die Berechnung nachträglich.

4.1.1 Der Algorithmus von Nemhauser/Wolsey

4.1.1.1 Vorstellung des Algorithmus

Beim Algorithmus von Nemhauser/Wolsey handelt es sich wahrscheinlich um das auf Anhieb verständlichste Berechnungsverfahren. Er lässt sich grundsätzlich in vier Schrit-

te aufgliedern (vgl. [Nem88, S. 193]), die in Tabelle 4.1 dargestellt sind.

Schritt	Beschreibung
1	Arbeite auf Zeile i und Spalten $j := i+1$ bis n. Wenn $i > m$ ist, stoppt der Algorithmus.
2	Elimination: Annulliere alle Elemente $a_{i,j}$, $j > i$, d.h. annulliere alle Elemente rechts vom Diagonalelement $a_{i,i}$.
3	Falls das Diagonalelement $a_{i,i}$ negativ ist, multipliziere dessen Spalte mit -1.
4	Normalisierung: Verändere die Einträge $a_{i,j}$, $j < i$, so, dass $a_{i,j} \leq 0$ und $\lvert a_{i,j}\rvert < a_{i,i}$, d.h. normalisiere alle Elemente links vom Diagonalelement in Zeile i. Wähle die nächste Zeile $i+1$ und gehe zu Schritt 1.

Tabelle 4.1: *Schritte des HNF-Algorithmus nach Nemhauser/Wolsey*

Die Hermite-Normalform wird bei diesem Algorithmus zeilenweise aufgebaut: In jeder Zeile erfolgen erst Eliminations-, dann Normalisierungsschritte. Eine Besonderheit, auf die bereits in Abschnitt 3.3.1 eingegangen wurde, ist der Wert der normalisierten Einträge: Die Elemente sind im Widerspruch zur Definition von Hermite zwar betragsmäßig kleiner als das Diagonalelement, jedoch nicht positiv.

Algorithmus 4.1 führt die Berechnungsschritte im Detail auf. Aus einer $(m \times n)$-Matrix A wird die Hermite-Normalform erzeugt, zusätzlich werden die gleichen Schritte auf eine Einheitsmatrix angewandt, sodass daraus die unimodulare Transformationsmatrix U erzeugt wird. H und A sind damit rechtsäquivalent. Alle Spaltenoperationen werden durch Multiplikation mit Permutationsmatrizen C ausgeführt.

Algorithmus 4.1: HNFNW

Eingabe: $A_{m\times n}$ mit $rg(A) = m$
Ausgabe: $(H_{m\times n}, U_{n\times n})$, wobei $H = HNF(A)$ und $H = A \cdot U$.

```
1  //Initialisierung
2  H := A; U := I_n; i := 1;
3
4  //Schritt 1: Spalte auswählen
5  j := i + 1;
6
7  //Schritt 2: Elimination
8  if j <= n then
9    if h_{i,j} ≠ 0 then
10     (r, p, q) := ExtendedGCD(h_{i,i}, h_{i,j});
```

```
11      C := I_n ;
12      c_{i,i} := p ;  c_{j,i} := q ;  c_{i,j} := -h_{i,j}/r ;  c_{j,j} := h_{i,i}/r ;
13      H := H · C ;  U := U · C ;
14    end_if
15  end_if
16  if j < n then
17    j := j + 1 ;  goto (8) ;
18  end_if
19
20  //Schritt 3: Spalte mit (-1) multiplizieren
21  if h_{i,i} < 0 then
22    C := I_n ;  c_{i,i} := -1 ;
23    H := H · C ;  U := U · C ;
24  end_if
25
26  //Schritt 4: Normalisierung
27  j := 1 ;
28  if j = i then
29   if i = m then return (H, U)
30   else
31     i := i + 1 ;  goto (5) ;
32   end_if
33  end_if
34  C := I_n ;  c_{i,j} := -⌈h_{i,j}/h_{i,i}⌉ ;
35  H := H · C ;  U := U · C ;
36  if j = i - 1 then
37    i := i + 1 ;
38    if i > m then return (H, U)
39    else
40      goto (5) ;
41    end_if
42  end_if
43  if j < i - 1 then
44    j := j + 1 ;  goto (28) ;
45  end_if
```

Bemerkung:
Schritt 4 erfolgt grundsätzlich nur für $j < i$ und startet mit $j = 1$. Die Abfrage $j = i = 1$ wurde Algorithmus 4.1 hinzugefügt; in [Nem88] wird nicht explizit darauf getestet. Im Fall $j = i = 1$ erfolgt keine Normalisierung.

4.1.1.2 Anwendungsbeispiel

Algorithmus 4.1 wird nun ausführlich anhand eines Beispiels präsentiert. Zur Übersichtlichkeit sind die einzelnen Schritte nach den in Tabelle 4.1 dargestellten Schritten gekennzeichnet. Bei jedem Eliminations- und Normalisierungsvorgang sind zusätzlich die Werte der Schleifenvariablen i und j angegeben. Es sei

$$A_{4\times 4} = \begin{pmatrix} 0 & 2 & 0 & -3 \\ 2 & 4 & -3 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Zunächst wird initialisiert:

$$H := \begin{pmatrix} 0 & 2 & 0 & -3 \\ 2 & 4 & -3 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},\ U := I_4,\ i := 1.$$

1. Spalte auswählen
$j := i + 1 = 2$ wird gesetzt.

2. Elimination
$[i = 1, j = 2]$:
Da $j \leq n = 4$ und $h_{1,2} = 2 \neq 0$ gilt, wird der EXTENDEDGCD für $h_{1,1} = 0$ und $h_{1,2} = 2$ berechnet:

$$(r, p, q) := \text{EXTENDEDGCD}(0, 2) = (2, 0, 1).$$

Anschließend wird eine Permutationsmatrix C konstruiert:

$$C = \begin{pmatrix} \mathbf{p} & -\mathbf{h_{1,2}/r} & 0 & 0 \\ \mathbf{q} & \mathbf{h_{1,1}/r} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{0} & \mathbf{-1} & 0 & 0 \\ \mathbf{1} & \mathbf{0} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Indem die Matrizen H und U mit C multipliziert werden, wird das Element $h_{1,1}$ positiv und Element $h_{1,2}$ annulliert:

$$H := H \cdot C = \begin{pmatrix} 2 & 0 & 0 & -3 \\ 4 & -2 & -3 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},\ U := U \cdot C = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

$[i = 1, j = 3]$:
Nun gilt immer noch $j = 3 \leq n = 4$. Da jedoch $h_{1,3} = 0$ ist, muss dieser Eintrag nicht annulliert werden. j wird folglich weiter hochgezählt, ohne dass weitere Schritte ausgeführt werden müssen.

$[i = 1, j = 4]$:
Nun wird das letzte Element in der ersten Zeile von H bearbeitet. Die Berechnung des EXTENDEDGCD ergibt

$$(r, p, q) := \text{EXTENDEDGCD}(2, -3) = (1, -1, -1).$$

Man erhält die folgende Permutationsmatrix C und durch Multiplikation mit C veränderte Matrizen H und U:

$$C = \begin{pmatrix} \mathbf{-1} & 0 & 0 & \mathbf{3} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \mathbf{-1} & 0 & 0 & \mathbf{2} \end{pmatrix}, H := \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & -2 & -3 & 10 \\ 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 2 \end{pmatrix}, U := \begin{pmatrix} 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 2 \end{pmatrix}.$$

Die Eliminationsschritte auf der ersten Zeile von H sind nun beendet.

3. Spalte mit (-1) multiplizieren
In diesem Fall ist $h_{1,1} = 1 > 0$, es wird nichts verändert und mit Schritt 4 fortgefahren.

4. Normalisierung
Zunächst wird $j := 1$ gesetzt. Normalisierungsschritte sind in diesem Fall nicht möglich, da nun $j = i = 1$ gilt und es in der ersten Zeile ohnehin keine Einträge links vom Diagonalelement $h_{1,1}$ gibt, die angepasst werden müssten. Die erste Zeile ist vollständig bearbeitet. $i := i + 1 = 2$ wird gesetzt und mit Schritt 1 fortgefahren.

1. Spalte auswählen
$j := i + 1 = 3$ wird gesetzt.

2. Elimination
$[i = 2, j = 3]$:
Es gilt $h_{2,3} = -3 \neq 0$, dieser Eintrag muss also annulliert werden. Zunächst wird

$$(r, p, q) := \text{EXTENDEDGCD}(-2, -3) = (1, 1, -1)$$

berechnet. Durch C werden H und U wie folgt aktualisiert:

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \mathbf{1} & \mathbf{3} & 0 \\ 0 & \mathbf{-1} & \mathbf{-2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, H := \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 10 \\ 0 & -1 & -2 & 0 \\ -1 & 0 & 0 & 2 \end{pmatrix}, U := \begin{pmatrix} 0 & -1 & -3 & 0 \\ -1 & 0 & 0 & 3 \\ 0 & -1 & -2 & 0 \\ -1 & 0 & 0 & 2 \end{pmatrix}.$$

$[i = 2, j = 4]$:
Nun wird das letzte Element in der zweiten Zeile von H bearbeitet. Die Berechnung des EXTENDEDGCD ergibt

$$(r, p, q) := \text{EXTENDEDGCD}(1, 10) = (1, 1, 0).$$

Damit erhält man Matrizen C, H und U:

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & \mathbf{-10} \\ 0 & 0 & 1 & 0 \\ 0 & \mathbf{0} & 0 & \mathbf{1} \end{pmatrix}, H := \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ 0 & -1 & -2 & 10 \\ -1 & 0 & 0 & 2 \end{pmatrix}, U := \begin{pmatrix} 0 & -1 & -3 & 10 \\ -1 & 0 & 0 & 3 \\ 0 & -1 & -2 & 10 \\ -1 & 0 & 0 & 2 \end{pmatrix}.$$

3. Spalte mit (-1) multiplizieren
$h_{2,2} > 0$, hier muss nichts getan werden.

4. Normalisierung
Zunächst wird $j := 1$ gesetzt. Zur Normalisierung wird eine Permutationsmatrix C erzeugt,

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -\lceil \mathbf{h_{2,1}} / \mathbf{h_{2,2}} \rceil & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \mathbf{3} & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

danach werden H und U wieder mit C multipliziert und man erhält

$$H := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -1 & -2 & 10 \\ -1 & 0 & 0 & 2 \end{pmatrix}, U := \begin{pmatrix} -3 & -1 & -3 & 10 \\ -1 & 0 & 0 & 3 \\ -3 & -1 & -2 & 10 \\ -1 & 0 & 0 & 2 \end{pmatrix}.$$

Weitere Normalisierungsschritte sind nicht notwendig, da es links von $h_{2,2}$ keine weiteren Einträge gibt. Es wird $i := i + 1 = 3$ gesetzt und mit Schritt 1 fortgefahren.

1. Spalte auswählen
$j := i + 1 = 4$ wird gesetzt.

2. Elimination
$[i = 3, j = 4]$:
Hier muss das Element $h_{3,4}$ annulliert werden. Der EXTENDEDGCD wird ermittelt:

$$(r, p, q) := \text{EXTENDEDGCD}(-2, 10) = (2, -1, 0).$$

C wird erzeugt und H und U werden aktualisiert:

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \mathbf{-1} & \mathbf{-5} \\ 0 & 0 & \mathbf{0} & \mathbf{-1} \end{pmatrix}, H := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -1 & 2 & 0 \\ -1 & 0 & 0 & -2 \end{pmatrix}, U := \begin{pmatrix} -3 & -1 & 3 & 5 \\ -1 & 0 & 0 & -3 \\ -3 & -1 & 2 & 0 \\ -1 & 0 & 0 & -2 \end{pmatrix}.$$

3. Spalte mit (-1) multiplizieren
$h_{3,3} > 0$, hier muss nichts getan werden.

4. Normalisierung
$j := 1$ wird gesetzt. Insgesamt müssen die zwei Einträge links von $h_{3,3}$ normalisiert werden.

$[i = 3, j = 1]$:
Die resultierenden Matrizen C, H und U sind:

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \mathbf{1} & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, H := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & -1 & 2 & 0 \\ -1 & 0 & 0 & -2 \end{pmatrix}, U := \begin{pmatrix} 0 & -1 & 3 & 5 \\ -1 & 0 & 0 & -3 \\ -1 & -1 & 2 & 0 \\ -1 & 0 & 0 & -2 \end{pmatrix}.$$

$[i = 3, j = 2]$:
Es wird $-\lceil h_{3,2}/h_{3,3} \rceil = 0$ berechnet, somit bleibt die Permutationsmatrix C eine Einheitsmatrix und H und U werden durch Multiplikation mit C nicht verändert. Es gilt nun $j = i - 1 = 2$, darum wird $i := i + 1 = 4$ gesetzt und mit Schritt 1 fortgefahren.

1. Spalte auswählen
$j := i + 1 = 5$ wird gesetzt.

2. Elimination
$[i = 4, j = 5]$:
Die Bedingung $j <= n$ ist nicht mehr erfüllt, es finden keine Eliminationsschritte statt.

3. Spalte mit (-1) multiplizieren
$h_{4,4} = -2 < 0$, die letzte Spalte von H muss folglich mit (-1) multipliziert werden. Dies folgt durch die abgebildete Permutationsmatrix C. H und U werden wie folgt verändert:

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \mathbf{-1} \end{pmatrix}, \; H := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & -1 & 2 & 0 \\ -1 & 0 & 0 & 2 \end{pmatrix}, \; U := \begin{pmatrix} 0 & -1 & 3 & -5 \\ -1 & 0 & 0 & 3 \\ -1 & -1 & 2 & 0 \\ -1 & 0 & 0 & 2 \end{pmatrix}.$$

4. Normalisierung
$j := 1$ wird gesetzt. Insgesamt müssen drei Einträge links von $h_{4,4}$ normalisiert werden.

$[i = 4, j = 1]$:
Es ist $-\lceil h_{4,1}/h_{4,4} \rceil = 0$, somit bleibt die Permutationsmatrix C eine Einheitsmatrix und H und U werden durch Multiplikation mit C nicht verändert.

$[i = 4, j = 2]$:
Es gilt wieder $-\lceil h_{4,2}/h_{4,4} \rceil = 0$. H und U werden nicht verändert.

$[i = 4, j = 3]$:
Auch hier werden H und U nicht verändert.

Nun gilt $j = i - 1 = 3$. Es wird $i := i + 1 = 5$ gesetzt. Damit ist $i > m = 4$ und der Algorithmus stoppt mit der Rückgabe der Matrizen H und U.

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & -1 & 2 & 0 \\ -1 & 0 & 0 & 2 \end{pmatrix}, \quad U = \begin{pmatrix} 0 & -1 & 3 & -5 \\ -1 & 0 & 0 & 3 \\ -1 & -1 & 2 & 0 \\ -1 & 0 & 0 & 2 \end{pmatrix}$$

Abbildung 4.1: *Ergebnismatrizen nach Ablauf von Algorithmus 4.1*

4.1.2 Der Algorithmus von Schrijver

4.1.2.1 Vorstellung des Algorithmus

Im Gegensatz zu den anderen in dieser Arbeit präsentierten Berechnungsverfahren wird bei dem folgenden Algorithmus die Hermite-Normalform nicht Zeile für Zeile bzw. Spalte für Spalte aufgebaut, sondern bei jedem k-ten Durchlauf wird der $(k+1)$-te Hauptminor diagonalisiert. Zusätzlich verwendet dieser Algorithmus eine Determinante M einer geeigneten Submatrix der Ausgangsmatrix A, um die Matrixeinträge gering (maximal M) zu halten. Die Normalisierungsschritte erfolgen erst nach Ablauf des gesamten Diagonalisierungsverfahrens.

Insgesamt ist der Algorithmus von Schrijver nicht so einfach wie der vorherige. Zunächst wird mit der in Kapitel 2.1.2 vorgestellten Funktion $\textsc{Minors}(A, m)$ eine Liste aller Determinanten von $(m \times m)$-Untermatrizen von A ermittelt. Aus dieser wird der Absolutwert M der ersten Determinante $\neq 0$ verwendet. Anschließend wird eine neue $(m \times (n+m))$-Matrix H erzeugt:

$$H = \left[A | I_m^M\right].$$

Bevor weitere Schritte ausgeführt werden, addiert man ganzzahlige Vielfache der letzten m Spalten von H zu den ersten n Spalten, sodass alle Einträge zwischen 0 und M liegen.

Der Eliminationsvorgang erfolgt nun schrittweise auf einer Untermatrix D von H. Begonnen wird mit Schritt $k = 0$ und gestoppt bei $k = m$. Bei jedem Schritt k hat H die Form

$$H = \left[\begin{array}{c|c|c} B & \mathbf{0}_{k\times(n+1)} & \mathbf{0}_{(k)\times(m-k-1)} \\ \hline C & D & \begin{array}{c}\mathbf{0}_{(1)\times(m-k-1)} \\ I_{m-k-1}^M\end{array} \end{array}\right],$$

wobei für die Matrizen B, C und D gilt:

- B ist eine $(k \times k)$-Matrix, die nach genau m Schritten die HNF darstellt,
- C ist eine $((m-k) \times k)$-Matrix und
- D ist eine $((m-k) \times (n+1))$-Matrix.

Im Fall $k = m - 1$ hat H also die Form

$$H = \left[\begin{array}{c|c} B_{(m-1)\times(m-1)} & \mathbf{0}_{(m-1)\times(n+1)} \\ \hline C_{1\times(m-1)} & D_{1\times(n+1)} \end{array}\right],$$

d.h. es wird auf einer einzeiligen Matrix D eliminiert. Tabelle 4.2 fasst die einzelnen Schritte des HNF-Algorithmus zusammen.

Schritt	**Beschreibung**
1	Berechne den Betrag M einer Determinante einer Submatrix von A vom Rang m. Erzeuge $H = \left[A|I_m^M\right]$.
2	Addiere ganzzahlige Vielfache der letzten m Spalten zu den ersten n Spalten, sodass alle Einträge zwischen 0 und M liegen.
3	Elimination für Schritt k: Arbeite auf der $((m-k)\times(n+1))$-Submatrix D.
3.1	Führe Eliminationsschritte aus, bis die erste Zeile von D nur noch ein positives Element enthält.
3.2	Nach jedem Eliminationsschritt: Addiere ganzzahlige Vielfache der letzten $m-k-1$ Spalten von H zu den Spalten von D, sodass alle Einträge in D zwischen 0 und M liegen.
3.3	Falls notwendig, tausche Spalten so, dass das einzige positive Element der 1. Zeile von D in dessen 1. Spalte steht. Es bildet dann das Diagonalelement dieser Zeile von H.
4	Ist $k < m$, setze $k := k + 1$ und fahre fort mit Schritt 3.
5	Normalisierung: Führe elementare Spaltenoperationen auf den ersten m Spalten von H so aus, dass $0 \leq h_{i,j} < h_{i,i}$, $j < i$.

Tabelle 4.2: *Schritte des HNF-Algorithmus nach Schrijver*

Nach Ablauf löscht man die letzten m Spalten von H (diese wurden am Anfang angehängt) und erhält somit die Hermite-Normalform von A. Wie beim Nemhauser/-Wolsey-Algorithmus hat H die Form $[\, B \;\; 0\,]$, falls $rg(A) < n$, d.h. die letzten $n - rg(A)$ Spalten sind Nullspalten. Die vollständige Berechnung wird durch die Algorithmen 4.2 und 4.3 (für die Eliminationsschritte auf Matrix D) dargestellt. Hierzu einige Anmerkungen:

- Die durch MINORS(A, m) ermittelten Minoren werden in einer Liste *minorList* gespeichert. Zugriff auf das i-te Element der Liste erhält man mit *minorList*$[i]$, die Länge der Liste kann durch LENGTH(*minorList*) ermittelt werden.

- Die Anweisung „$Y := X_{i..j,k..l}$“ extrahiert eine Submatrix Y aus X, die sich aus den Zeilen i bis j und Spalten k bis l von X zusammensetzt.

- *elemFound* ist eine boolesche Variable, in der gespeichert wird, ob in der ersten Zeile von Matrix D mindestens zwei Elemente > 0 existieren.

Algorithmus 4.2: HNFschrijver

Eingabe: $A_{m\times n}$ mit $rg(A) = m$
Ausgabe: $H_{m\times n}$ mit $H = HNF(A)$

```
1  //Schritt 1: Determinante wählen und Matrix erweitern
2  minorList := MINORS(A, m);
3  for i := 1 to LENGTH(minorList) do
4     if minorList[i] ≠ 0 then
5        M := |minorList[i]|;
6        break;
7     end_if
8  end_for
9  H := [A|I_m^M];
10
11 //Schritt 2: Matrixeinträge anpassen
12 for i := 1 to n do
13    for j := 1 to m do
14       if (h_{j,i} > M) ∨ (h_{j,i} < 0) then
15          h_{j,i} := h_{j,i} − ⌊h_{j,i}/M⌋ · M;
16       end_if
17    end_for
18 end_for
19
20 //Schritte 3 & 4: Elimination
21 for k := 0 to m − 1 do
22    D := H_{k+1..m,k+1..n+k+1};
23    D := ELIMINATE(D, k, M);
24    H_{k+1..m,k+1..n+k+1} := D;
25 end_for
26
27 //Schritt 5: Normalisierung
28 for i := 1 to m do
29    for j := 1 to i − 1 do
30       if (h_{i,j} < 0) ∨ (h_{i,j} ≥ h_{i,i}) then
31          h_{∗,j} := h_{∗,j} − ⌊h_{i,j}/h_{i,i}⌋ · h_{∗,i};
32       end_if
33    end_for
34 end_for
35
36 return H_{∗,1..n};
```

Der Algorithmus ELIMINATE wird von Algorithmus 4.2 aufgerufen und führt die ei-

gentlichen Eliminationsschritte (Schritt 3.1) auf der ersten Zeile von Matrix D aus, bis diese Zeile noch genau ein positives Element enthält. Werden dabei (hier wird das Vielfache einer Spalte zu einer anderen addiert) Matrixeinträge negativ, so erfolgt eine Normalisierung (Schritt 3.2).

Algorithmus 4.3: ELIMINATE

Eingabe: $D_{m\times n}, k, M$
Ausgabe: $D_{m\times n}$

```
1  elemFound := False;
2  //Finde 2 Elemente verschieden von Null
3  for i := 1 to n do
4    for j := 1 to n do
5      if (d_{1,i} > 0) ∧ (d_{1,j} > 0) ∧ (j ≠ i) then
6        if d_{1,i} ≥ d_{1,j} then
7          di := i; dj := j; elemFound := True; break
8        else
9          di := j; dj := i; elemFound := True; break;
10       end_if
11     end_if
12   end_for
13   if elemFound = True then
14     break;
15   end_if
16 end_for
17
18 //Schritt 3.1:
19 if elemFound = True then
20   f := ⌊d_{1,di}/d_{1,dj}⌋;
21   d_{*,di} := d_{*,di} − f · d_{*,dj};
22   //Schritt 3.2
23   if m > 1 then
24     for i := 1 to n do
25       for j := m downto 2 do
26         if (d_{j,i} < 0) ∨ (d_{j,i} > M) then
27           d_{j,i} := d_{j,i} − ⌊d_{j,i}/M⌋ · M;
28         end_if
29       end_for
30     end_for
31   end_if
32 end_if
```

```
33
34 if elemFound = True then
35   goto (1);
36 end_if
37
38 //Schritt 3.3: (falls notwendig) Spalten tauschen
39 if d_{1,1} = 0 then
40   for i := 2 to n do
41     if d_{1,i} > 0 then
42       pos := i; break;
43     end_if
44   end_for
45   temp := d_{*,pos}; d_{*,pos} := d_{*,1}; d_{*,1} := temp;
46 end_if
47
48 return D;
```

4.1.2.2 Anwendungsbeispiel

Zu berechnen sei erneut die Hermite-Normalform der Matrix

$$A_{4\times 4} = \begin{pmatrix} 0 & 2 & 0 & -3 \\ 2 & 4 & -3 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

1. Determinante wählen und Matrix erweitern
Zunächst wird mit

$$minorList := \textsc{Minors}(A, 4)$$

eine Liste der Determinanten berechnet. In diesem Fall gibt es nur eine (4×4)-Submatrix, nämlich A selbst. Unter Verwendung des Laplaceschen Entwicklungssatzes und Entwicklung nach Spalte $j = 1$ erhält man

$$\begin{aligned} \det A &= \sum_{i=1}^{4} (-1)^{i+1} \cdot a_{i,1} \cdot \det A_{i,1} \\ &= (-1)^2 \cdot \underbrace{a_{1,1}}_{=0} \cdot \det A_{1,1} + (-1)^3 \cdot a_{2,1} \cdot \det A_{2,1} \\ &+ (-1)^4 \cdot \underbrace{a_{3,1}}_{=0} \cdot \det A_{3,1} + (-1)^5 \cdot \underbrace{a_{4,1}}_{=0} \cdot \det A_{4,1} \end{aligned}$$

$$
\begin{aligned}
&= (-1)^3 \cdot a_{2,1} \cdot \det \begin{pmatrix} 2 & 0 & -3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= -2 \cdot (2 + 0 + 0 - 0 - 0 - 0) \\
&= -4
\end{aligned}
$$

und somit $minorList = \{-4\}$. Die einzige existierende Unterdeterminante ist damit der Eintrag in $minorList$. Man erhält $M = 4$ und die Matrix H kann aus A und $M \cdot I_4$ erzeugt werden:

$$
H = \left[A | I_4^4\right] = \left[\begin{array}{cccc|cccc} 0 & 2 & 0 & -3 & 4 & 0 & 0 & 0 \\ 2 & 4 & -3 & -1 & 0 & 4 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 4 \end{array}\right].
$$

2. Matrixeinträge anpassen

Die Einträge der Matrix werden so verändert, dass sie zwischen 0 und 4 liegen:

$$
H := \left[\begin{array}{cccc|cccc} 0 & 2 & 0 & \mathbf{-3}\mathit{+4} & 4 & 0 & 0 & 0 \\ 2 & 4 & \mathbf{-3}\mathit{+4} & \mathbf{-1}\mathit{+4} & 0 & 4 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 4 \end{array}\right] = \left[\begin{array}{cccc|cccc} 0 & 2 & 0 & \mathbf{1} & 4 & 0 & 0 & 0 \\ 2 & 4 & \mathbf{1} & \mathbf{3} & 0 & 4 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 4 \end{array}\right].
$$

3. Elimination

$[k = 0]$:

Nun wird Matrix D erzeugt,

$$
D := H_{0+1..4,0+1..4+0+1} = H_{1..4,1..5} = \begin{pmatrix} 0 & 2 & 0 & 1 & 4 \\ 2 & 4 & 1 & 3 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix},
$$

und Algorithmus ELIMINATE(D,0,4) aufgerufen. Es werden zwei positive Elemente in der ersten Zeile von D gefunden ($elemFound$=$True$) mit $di = 2$ und $dj = 4$. Man erhält $f := \lfloor d_{1,2}/d_{1,4} \rfloor = 2$. Somit wird 2-mal Spalte 4 von Spalte 2 subtrahiert. Anschließend werden die Einträge normalisiert:

$$
D := \begin{pmatrix} 0 & \mathbf{0} & 0 & 1 & 4 \\ 2 & \mathbf{-2}\mathit{+4} & 1 & 3 & 0 \\ 0 & \mathbf{0} & 1 & 0 & 0 \\ 0 & \mathbf{-2}\mathit{+4} & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & \mathbf{0} & 0 & 1 & 4 \\ 2 & \mathbf{2} & 1 & 3 & 0 \\ 0 & \mathbf{0} & 1 & 0 & 0 \\ 0 & \mathbf{2} & 0 & 1 & 0 \end{pmatrix}.
$$

Da *elemFound=True* gilt, wird die erste Zeile von D erneut auf positive Elemente durchsucht. Gefunden werden Elemente in den Spalten $di = 5$ und $dj = 4$. Somit wird Spalte 4 4-mal von Spalte 5 subtrahiert und es erfolgt die gewohnte Normalisierung:

$$D := \begin{pmatrix} 0 & 0 & 0 & 1 & \mathbf{0} \\ 2 & 2 & 1 & 3 & \mathbf{-12}\textit{+3}\cdot\textit{4} \\ 0 & 0 & 1 & 0 & \mathbf{0} \\ 0 & 2 & 0 & 1 & \mathbf{-4}\textit{+4} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 & \mathbf{0} \\ 2 & 2 & 1 & 3 & \mathbf{0} \\ 0 & 0 & 1 & 0 & \mathbf{0} \\ 0 & 2 & 0 & 1 & \mathbf{0} \end{pmatrix}.$$

Das einzige positive Element in der ersten Zeile von D befindet sich nun in Spalte 4. Spalten 1 und 4 werden getauscht und $H_{1..4,1..5}$ durch D ersetzt:

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 3 & 2 & 1 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 \end{pmatrix} \quad \Rightarrow \quad H = \left[\begin{array}{ccccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 2 & 1 & 2 & 0 & 4 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 4 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 & 0 & 4 \end{array}\right].$$

4. Elimination – Fortsetzung
Da $k < m - 1 = 3$ gilt, wird k hochgezählt und mit Schritt 3 fortgefahren.

3. Elimination
$[k = 1]$:

$$D := H_{1+1..4,1+1..4+1+1} = H_{2..4,2..6} = \begin{pmatrix} 2 & 1 & 2 & 0 & 4 \\ 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \end{pmatrix}$$

ELIMINATE(D,1,4) wird aufgerufen. Die ersten beiden positiven Elemente in der ersten Zeile von D befinden sich in den Spalten $di = 1$ und $dj = 2$. Spalte 2 wird 2-mal von Spalte 1 subtrahiert, anschließend werden die Einträge normalisiert:

$$D := \begin{pmatrix} \mathbf{0} & 1 & 2 & 0 & 4 \\ \mathbf{-2}\textit{+4} & 1 & 0 & 0 & 0 \\ \mathbf{2} & 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} \mathbf{0} & 1 & 2 & 0 & 4 \\ \mathbf{2} & 1 & 0 & 0 & 0 \\ \mathbf{2} & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Im nächsten Durchlauf gilt $di = 3$ und $dj = 2$. Es wird wieder eliminiert (2-mal Spalte 2 von Spalte 3 subtrahiert) und normalisiert:

$$D := \begin{pmatrix} 0 & 1 & \mathbf{0} & 0 & 4 \\ 2 & 1 & \mathbf{-2}\textit{+4} & 0 & 0 \\ 2 & 0 & \mathbf{0} & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & \mathbf{0} & 0 & 4 \\ 2 & 1 & \mathbf{2} & 0 & 0 \\ 2 & 0 & \mathbf{0} & 0 & 0 \end{pmatrix}.$$

Im letzten Durchlauf ist $di = 5$ und $dj = 2$. Es wird 4-mal Spalte 2 von Spalte 5 subtrahiert und normalisiert:

$$D := \begin{pmatrix} 0 & 1 & 0 & 0 & \mathbf{0} \\ 2 & 1 & 2 & 0 & \mathbf{-4+4} \\ 2 & 0 & 0 & 0 & \mathbf{0} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & \mathbf{0} \\ 2 & 1 & 2 & 0 & \mathbf{0} \\ 2 & 0 & 0 & 0 & \mathbf{0} \end{pmatrix}.$$

In Schritt 3.3 müssen schließlich die ersten beiden Spalten von D getauscht werden. Danach wird $H_{2..4,2..6}$ durch D ersetzt:

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 2 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \end{pmatrix} \quad \Rightarrow \quad H = \left[\begin{array}{c|cccccc|cc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 2 & 0 & 0 & 4 & 0 \\ 1 & 0 & 2 & 0 & 0 & 0 & 0 & 4 \end{array}\right].$$

4. Elimination – Fortsetzung
Da $k < m - 1 = 3$ gilt, wird k hochgezählt und mit Schritt 3 fortgefahren.

3. Elimination
$[k = 2]$:
Im nächsten Durchlauf ist

$$D := H_{2+1..4,2+1..4+2+1} = H_{3..4,3..7} = \begin{pmatrix} 2 & 2 & 0 & 0 & 4 \\ 2 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

ELIMINATE(D,2,4) wird aufgerufen. Die ersten beiden positiven Elemente sind $di = 1$ und $dj = 2$. Somit wird 1-mal Spalte 2 von Spalte 1 subtrahiert:

$$D := \begin{pmatrix} \mathbf{0} & 2 & 0 & 0 & 4 \\ \mathbf{2} & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Im nächsten Schritt erhält man Werte $di = 5$, $dj = 2$. Spalte 2 wird 2-mal von Spalte 5 subtrahiert:

$$D := \begin{pmatrix} 0 & 2 & 0 & 0 & \mathbf{0} \\ 2 & 0 & 0 & 0 & \mathbf{0} \end{pmatrix}.$$

In Schritt 3.3 müssen die ersten beiden Spalten getauscht werden,

$$D = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \end{pmatrix},$$

danach wird $H_{3..4,3..7}$ durch D ersetzt:

$$H = \left[\begin{array}{cc|ccccc|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 & 0 & 0 & 0 & 4 \end{array}\right].$$

4. Elimination – Fortsetzung
Da $k < m - 1 = 3$ gilt, wird k hochgezählt und mit Schritt 3 fortgefahren.

3. Elimination
$[k = 3]$:
Hierbei handelt es sich um den letzten Schritt der k-Schleife. Es gilt $k = m - 1 = 3$, damit hat H also die Form

$$H = \left[\begin{array}{c|c|c} B_{3\times 3} & \mathbf{0}_{3\times 5} & \mathbf{0}_{3\times 0} \\ \hline C_{1\times 3} & D_{1\times 5} & \begin{array}{c}\mathbf{0}_{1\times 0} \\ I_0^4\end{array} \end{array}\right] = \left[\begin{array}{ccc|ccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 2 & 0 & 0 & 0 & 4 \end{array}\right]$$

und somit

$$D = (\ 2 \quad 0 \quad 0 \quad 0 \quad 4\ ).$$

In D gibt es zwei positive Einträge in den Spalten $di = 5$ und $dj = 1$. Man subtrahiere 2-mal Spalte 1 von Spalte 5. Dies ergibt

$$D = (\ 2 \quad 0 \quad 0 \quad 0 \quad \mathbf{0}\ ).$$

Eine Normalisierung ist nicht notwendig.

4. Elimination – Fortsetzung
Es gilt nun $k = m - 1 = 3$. Alle Eliminationsschritte sind beendet. H hat die Form:

$$H = \left[\begin{array}{cccccccc} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathit{3} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \mathbf{2} & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & \mathbf{2} & 0 & 0 & 0 & 0 \end{array}\right].$$

5. Normalisierung
Neben der Normalisierung während der Eliminationsschritte, die die Matrixeinträge gering halten sollte, erfolgt nun der eigentliche, zum HNF-Algorithmus gehörende Nor-

malisierungsvorgang auf der (4×4)-Matrix B. Das einzige anzupassende Element ist zunächst $h_{2,1} = 3$. Es wird $\lfloor h_{2,1}/h_{2,2} \rfloor = 3$-mal Spalte 2 von Spalte 1 subtrahiert:

$$H = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{0} & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathit{-3} & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 2 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Dadurch ist jetzt der Eintrag $h_{3,1}$ nicht normalisiert. Es muss (-2)-mal Spalte 3 von Spalte 1 subtrahiert werden und man erhält:

$$H = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{0} & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 2 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Weitere Normalisierungsschritte sind nicht notwendig. Die Hermite-Normalform der Matrix A ist somit die (4×4)-Matrix B, also die ersten vier Spalten von H.

$$\mathit{HNF}(A) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix}$$

Abbildung 4.2: *Ergebnismatrix nach Ablauf von Algorithmus 4.2*

4.1.2.3 Berechnung der Transformationsmatrix

Die Berechnung der Transformationsmatrix erfolgt im Schrijver-Algorithmus nicht parallel zur HNF-Berechnung. Unter Verwendung der Hermite-Normalform und der inversen Ausgangsmatrix kann die Transformationsmatrix jedoch erzeugt werden (vgl. [Sch99, S. 55]). Dazu wird die Ausgangsmatrix A zunächst in die Form

$$A_{neu} = \begin{bmatrix} A' & A'' \end{bmatrix}$$

gebracht[1], wobei A' eine $(m \times m)$-Matrix ist und invertierbar sein muss. A_{neu} hat gemäß der Definition vollen Zeilenrang. Zur besseren Übersichtlichkeit unterscheidet man nun zwei Fälle:

1. $rg(A_{neu}) = m = n$, d.h. die Matrix ist quadratisch und $\mathit{HNF}(A_{neu}) = B$,

[1] Dabei müssen selbstverständlich die Spaltenindizes angepasst werden.

2. $rg(A_{neu}) = m < n$, d.h. die Matrix hat mehr Spalten als Zeilen und $HNF(A_{neu}) = [\, B \;\; 0 \,]$.

Für Fall 1 lässt sich die Transformationsmatrix berechnen als

$$U \;\; := \;\; \left[\begin{array}{cc} A' & A'' \end{array} \right]^{-1} \cdot B.$$

Unter der Voraussetzung, dass eine Matrix A invertierbar ist, lässt sich die Berechnung von U auch direkt aus Rechenregeln für Matrizen ableiten:

$$\begin{aligned} A \cdot U &= B \\ (A^{-1} \cdot A) \cdot U &= A^{-1} \cdot B \\ I \cdot U &= A^{-1} \cdot B \\ U &= A^{-1} \cdot B. \end{aligned}$$

Im zweiten Fall wird A_{neu} zu einer quadratischen Matrix ergänzt, d.h. es müssen $n-m$ Zeilen angefügt werden, sodass A_{neu} die Form

$$A_{neu} := \left[\begin{array}{cc} A' & A'' \\ \mathbf{0}_{(n-m)\times(m)} & I_{n-m} \end{array} \right]$$

hat und die zugehörige Hermite-Normalform definiert ist als

$$H = \left[\begin{array}{cc} B & 0 \\ B' & B'' \end{array} \right],$$

B' und B'' sind hierbei ganzzahlige Matrizen. Damit ist die zugehörige Transformationsmatrix U_{neu} definiert als

$$U_{neu} \;\; := \;\; \left[\begin{array}{cc} A' & A'' \\ \mathbf{0} & I \end{array} \right]^{-1} \cdot \left[\begin{array}{cc} B & 0 \\ B' & B'' \end{array} \right]$$

und allgemein gilt

$$\left[\begin{array}{cc} A' & A'' \end{array} \right] \cdot U_{neu} = \left[\begin{array}{cc} B & 0 \end{array} \right].$$

Beispiel:
Die Transformationsmatrix U wird für die Matrix

$$A = \left(\begin{array}{cccc} 0 & 0 & 2 & -3 \\ 2 & -3 & 4 & -1 \end{array} \right)$$

berechnet. A hat den Rang 2, jedoch müssen die Spalten 2 und 3 von A getauscht werden, damit

$$A' = \begin{pmatrix} 0 & 2 \\ 2 & 4 \end{pmatrix}$$

invertierbar ist. Dies kann durch Multiplikation mit der Permutationsmatrix

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

ausgedrückt werden. Anschließend wird A zu einer quadratischen Matrix A_{neu} ergänzt:

$$A_{neu} := \begin{pmatrix} A' & A'' \\ \mathbf{0}_{2\times 2} & I_2 \end{pmatrix} = \begin{pmatrix} 0 & 2 & 0 & -3 \\ 2 & 4 & -3 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

In Abschnitt 4.1.2.2 wurde für diese Matrix A_{neu} bereits deren zugehörige Hermite-Normalform H (Abb. 4.2) berechnet. A_{neu} hat vollen Zeilen- und Spaltenrang und ist somit invertierbar. Zunächst muss die inverse Matrix berechnet werden:

$$[A_{neu}|I] = \left[\begin{array}{cccc|cccc} 0 & 2 & 0 & -3 & 1 & 0 & 0 & 0 \\ 2 & 4 & -3 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\xrightarrow[1 \text{ und } 2]{\text{tausche Zeilen}} \left[\begin{array}{cccc|cccc} 2 & 4 & -3 & -1 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & -3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\xrightarrow{1. -2 \cdot 2. \text{ Zeile}} \left[\begin{array}{cccc|cccc} 2 & 0 & -3 & 5 & -2 & 1 & 0 & 0 \\ 0 & 2 & 0 & -3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\xrightarrow{1. +3 \cdot 3. \text{ Zeile}} \left[\begin{array}{cccc|cccc} 2 & 0 & 0 & 5 & -2 & 1 & 3 & 0 \\ 0 & 2 & 0 & -3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\xrightarrow[\text{1. } -5 \cdot \text{4. Zeile}]{\text{2. } +3 \cdot \text{4. Zeile}} \left[\begin{array}{cccc|cccc} 2 & 0 & 0 & 0 & -2 & 1 & 3 & -5 \\ 0 & 2 & 0 & 0 & 1 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\xrightarrow[\text{2. Zeile durch 2 teilen}]{\text{1. Zeile durch 2 teilen}} \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & -1 & \frac{1}{2} & \frac{3}{2} & -\frac{5}{2} \\ 0 & 1 & 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{3}{2} \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

Die zu A_{neu} inverse Matrix A_{neu}^{-1} ist somit

$$A_{neu}^{-1} = \begin{pmatrix} -1 & \frac{1}{2} & \frac{3}{2} & -\frac{5}{2} \\ \frac{1}{2} & 0 & 0 & \frac{3}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

und $U_{neu} := A_{neu}^{-1} \cdot H$ kann berechnet werden:

$$U_{neu} = \begin{pmatrix} -1 & \frac{1}{2} & \frac{3}{2} & -\frac{5}{2} \\ \frac{1}{2} & 0 & 0 & \frac{3}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix} = \begin{pmatrix} -2 & 2 & 3 & -5 \\ 2 & 0 & 0 & 3 \\ 1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix}.$$

Die zugehörige Transformationsmatrix zur Hermite-Normalform aus Abbildung 4.2 ist damit die in Abbildung 4.3 dargestellte Matrix U_{neu}.

$$U_{neu} = \begin{pmatrix} -2 & 2 & 3 & -5 \\ 2 & 0 & 0 & 3 \\ 1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix}$$

Abbildung 4.3: *Transformationsmatrix zur HNF aus Abbildung 4.2*

Wie sieht aber nun die Transformationsmatrix U zu

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

aus? Vertauscht man in A Spalten, wie es oben geschehen ist, spannt die neue Matrix $[\,A'\ \ A''\,]$ selbstverständlich immer noch das gleiche Gitter auf wie A. Aus Theorem 3.1 folgt damit, dass aus beiden Matrizen dieselbe Hermite-Normalform erzeugt wird. Es

gilt also

$$[\,B\;\;0\,] = A \cdot U = [\,A'\;\;A''\,] \cdot U_{neu} = \underbrace{A \cdot T}_{[\,A'\;\;A''\,]} \cdot U_{neu}.$$

Matrizenmultiplikation ist assoziativ, d.h. es gilt auch

$$[\,B\;\;0\,] = A \cdot (T \cdot U_{neu}).$$

Die Multiplikation $T \cdot U_{neu}$ stellt eine unimodulare Zeilenoperation gemäß Abschnitt 3.2.1 auf U_{neu} dar: In U_{neu} wird die 2. mit der 3. Zeile vertauscht. Dies ergibt

$$U = T \cdot U_{neu}.$$

Folglich ist die zu A gesuchte Transformationsmatrix U die Matrix

$$U := T \cdot U_{neu} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} -2 & 2 & 3 & -5 \\ 2 & 0 & 0 & 3 \\ 1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix} = \begin{pmatrix} -2 & 2 & 3 & -5 \\ 1 & 1 & 2 & 0 \\ 2 & 0 & 0 & 3 \\ 1 & 0 & 0 & 2 \end{pmatrix}.$$

Dieses Verfahren zur Berechnung der Transformationsmatrix wird in einen weiteren Algorithmus HNFschrijverComplete übernommen, der als Algorithmus 4.4 dargestellt ist.

Algorithmus 4.4: HNFschrijverComplete

Eingabe: $A_{m\times n}$ mit $rg(A) = m$
Ausgabe: $(H_{m\times n}, U_{n\times n})$, wobei $H = HNF(A)$ und $H = A \cdot U$.

```
1  //Matrix A' auf Invertierbarkeit testen
2  d := det(A_{1..m,1..m});
3  reg := True;
4  T_prod := I_n;
5  if d = 0 then
6    reg := False;
7    //tausche Spalten, bis A' invertierbar
8    for i := 1 to m do
9      for j := m + 1 to n do
10       T := I_n;
11       //tausche Spalten i und j
12       t_{i,i} := 0; t_{i,j} := 1;
```

```
13          t_{j,j} := 0;  t_{j,i} := 1;
14          A := A · T;
15          T_prod := T_prod · T;
16          d := det(A_{1..m,1..m});
17          if d ≠ 0 then
18            break;
19          end_if
20        end_for
21        if d ≠ 0 then
22          break;
23        end_if
24      end_for
25    end_if
26
27    //quadratische Matrix erzeugen
28    A^neu := I_n;
29    for i := 1 to m do
30      a^neu_{i,*} := a_{i,*};
31    end_for
32
33    //HNFschrijver aufrufen
34    H := HNFschrijver(A^neu);
35
36    //U berechnen
37    U := (A^neu)^{-1} · H;
38
39    //falls Spalten getauscht wurden -> Zeilen in U tauschen
40    if reg = False then
41      U := T_prod · U;
42    end_if
43
44    return (H_{1..m,1..n}, U);
```

Zunächst wird also aus A, falls $A_{1..m,1..m}$ singulär ist, die Matrix $A = [\ A' \quad A''\ ]$ erzeugt, wobei A' eine invertierbare $(m \times m)$-Matrix ist. Die dazu notwendigen Spaltenvertauschungen werden in T_{prod} gespeichert. Dann wird aus A und einer Einheitsmatrix eine reguläre quadratische Matrix A^{neu} erzeugt, die Algorithmus 4.2 übergeben wird. Mit der resultierenden HNF H kann die Matrix U erzeugt werden.

Abschließend werden Zeilen von U vertauscht, wenn $A_{1..m,1..m}$ singulär war, indem eine Linksmultiplikation mit T_{prod} erfolgt. Dann werden $H_{1..m,1..n}$ und U zurückgegeben.

4.2 HNF-Algorithmen für beliebige Matrizen

In diesem Abschnitt werden zwei Algorithmen vorgestellt, die aus beliebigen ganzzahligen Matrizen deren Hermite-Normalform gemäß der Definitionen 3.6 und 3.7 erzeugen. Bereits in Kapitel 3.3.2 wurde darauf hingewiesen, dass für $(m \times n)$-Matrizen A mit $rg(A) = r < m$ nur r Zeilen ein Diagonalelement besitzen. Auf den verbleibenden Zeilen werden keine Eliminations- oder Normalsierungsschritte ausgeführt.

4.2.1 Der Algorithmus von Patrick Theobald

4.2.1.1 Vorstellung des Algorithmus

Im Gegensatz zu den beiden vorherigen Algorithmen wird bei diesem aus [The00] stammenden Berechnungsverfahren eine obere Dreiecksmatrix erzeugt. Die einzelnen Schritte sind in Tabelle 4.3 abgebildet.

Schritt	Beschreibung
0	Initialisierung: $i := m$, $j := n$.
1	Arbeite auf Zeile i und Spalte j.
2	Sind in einer Zeile die ersten $1, ..., j$ Elemente Null, so wird diese Zeile übersprungen und sie besitzt kein Diagonalelement. Der Spaltenindex für das Pseudodiagonalelement bleibt damit j.
3	Sonst wird in Zeile i beginnend ab Spalte $l = 1$ so eliminiert, dass das Diagonalelement in Spalte j steht und die ersten $j-1$ Einträge Null sind. Dabei werden immer zwei Spalten l und $l+1$ unter Verwendung von Permutationsmatrizen und der EXTENDEDGCD-Funktion bearbeitet. Es gilt drei Fälle zu unterscheiden:
3.1	Ist der Eintrag in Spalte l Null, muss nicht weiter eliminiert werden.
3.2	Ist der Eintrag l verschieden von Null und Eintrag $l+1$ Null, so werden die Spalten getauscht. Damit wird das potentielle Diagonalelement aus Spalte l in der Matrix weiter nach rechts verschoben.
3.3	Sind beide Einträge verschieden von Null, so wird das Element in Spalte l eliminiert.
4	Normalisierung: In Zeile i werden die Elemente rechts vom Diagonalelement (also $j+1$ bis n) normalisiert.
5	Der Algorithmus wird fortgesetzt mit Zeile $i-1$ und Spalte $j-1$. Gehe zu 1.

Tabelle 4.3: *Schritte des HNF-Algorithmus nach Theobald*

Der vollständige Algorithmus ist als Algorithmus 4.5 dargestellt. Gegenüber der Bezugsquelle ([The00, S. 8]) wurden einige Korrekturen vorgenommen:

- In der Variable v wird eine Matrixzeile gespeichert. Durch Eliminationsoperationen werden jedoch die Werte dieser Zeile verändert, sodass v aktualisiert werden muss. Entsprechende Befehle wurden Algorithmus 4.5 an den notwendigen Stellen hinzugefügt.
- Im Original-Algorithmus existiert kein Test, ob das Pseudodiagonalelement positiv ist. Vor der Normalisierung wurde daher eine Abfrage eingefügt, die bei Bedarf eine Spalte mit negativem Pseudodiagonalelement mit (-1) multipliziert.
- Die Zeilen- und Spaltenindizes starten im Original-Algorithmus bei 0, hier bei 1.

Anmerkung:
Schleifenaufrufe der Form „`for i:=1 to 0 do...`“[2] bedeuten, dass diese Schleife nicht durchlaufen wird. Mathematica bearbeitet solche Befehle entsprechend.

Algorithmus 4.5: HNFTHEO

Eingabe: $A_{m\times n}$
Ausgabe: $(H_{m\times n}, U_{n\times n})$, wobei $H = HNF(A)$ und $H = A \cdot U$.

```
1  //Initialisierung
2  H := A;
3  U := I_n;
4  i := m;  j := n;
5
6  //Schritt 1: Arbeite auf Zeile i und Spalte j
7  while (i > 0) ∧ (j > 0) do
8    //Schritt 2: Überspringe Zeile
9    if (h_{i,1}, ..., h_{i,j}) = (0, ..., 0) then
10     i := i − 1
11   else
12     //Schritt 3: Elimination
13     for l := 1 to j − 1 do
14       v := h_{i,*};
15       if v_l ≠ 0 then
16         if v_{l+1} ≠ 0 then
17           //Schritt 3.3: Annullieren
```

[2] Nicht zu verwechseln mit *„for i:=1 downto...“*. In diesem Fall wird die Schleife selbstverständlich durchlaufen.

```
18              (g, a, b) := ExtendedGCD(v_l, v_{l+1});
19              T := I_n;
20              t_{l,l} := -v_{l+1}/g;  t_{l,l+1} := a;
21              t_{l+1,l} := v_l/g;  t_{l+1,l+1} := b;
22              H := H · T;
23              U := U · T
24            else
25              //Schritt 3.2: Spalten tauschen
26              T := I_n;
27              t_{l,l} := 0;  t_{l,l+1} := 1;
28              t_{l+1,l} := 1;  t_{l+1,l+1} := 0;
29              H := H · T;
30              U := U · T;
31            end_if
32          end_if
33        end_for
34        //Schritt 4: Normalisierung
35        if h_{i,j} < 0 then
36          h_{*,j} := -h_{*,j};
37          u_{*,j} := -u_{*,j};
38        end_if
39        for l := j + 1 to n do
40          v := h_{i,*};
41          q := ⌊v_l/v_j⌋;
42          h_{*,l} := h_{*,l} - q · h_{*,j};
43          u_{*,l} := u_{*,l} - q · u_{*,j};
44        end_for
45        //Schritt 5: Wähle neue Zeile und Spalte
46        i := i - 1;
47        j := j - 1;
48      end_if
49    end_while
50
51    return (H, U);
```

4.2.1.2 Anwendungsbeispiel

Mit Algorithmus 4.5 werden die Hermite-Normalform H der Matrix

$$A_{4\times 4} = \begin{pmatrix} 0 & 2 & 0 & -3 \\ 2 & 4 & -3 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

und deren zugehöre Transformationsmatrix U berechnet. Zunächst wird initialisiert:

$$H := A,\ U := I_4,\ i := 4,\ j := 4.$$

1. $[i = 4, j = 4]$:

3. Elimination
Es werden nun Eliminationsschritte von $l = 1$ bis $l = j - 1 = 3$ durchgeführt.

$$v := \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}$$

wird betrachtet. Sofort erkennt man, dass $v_1 = v_2 = v_3 = 0$ gilt, damit erfolgen keine Eliminationsschritte.

4. Normalisierung
Das Diagonalelement $h_{4,4}$ ist positiv, es wird nicht bearbeitet. Ebenso ist keine Normalisierung notwendig, da die Schleife $l = j + 1 = 5$ bis $l = 4$ nicht durchlaufen wird.

5. Neue Zeile und Spalte wählen
Man setze $i := i - 1 = 3$ und $j := j - 1 = 3$ und fahre mit Schritt 1 fort.

1. $[i = 3, j = 3]$:

3. Elimination
Es sollen Eliminationsschritte von $l = 1$ bis $l = j - 1 = 2$ erfolgen. $v := \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}$ wird betrachtet. Es ist $v_1 = v_2 = 0$ und es müssen keine Elemente annulliert werden.

4. Normalisierung
Das Diagonalelement $h_{3,3}$ ist positiv, es muss nicht bearbeitet werden. Weitere Normalisierungsschritte sind nicht notwendig, da das Element $v_{j+1} = v_l = v_4 = 0$ ist. Damit

gilt dann auch $q := 0$ und die Spalte $l = 4$ von H wird nicht verändert.

5. Neue Zeile und Spalte wählen
Man setze $i := i - 1 = 2$ und $j := j - 1 = 2$ und fahre mit Schritt 1 fort. Bisher wurden sowohl an H als auch an U noch keine Änderungen vorgenommen.

1. $[i = 2, j = 2]$:

3. Elimination
Es werden nun Eliminationsschritte von $l = 1$ bis $l = j - 1 = 1$ durchgeführt. Betrachtet wird $v := (\ 2 \quad 4 \quad -3 \quad -1\ )$.

$l = 1$:
Gearbeitet wird auf Elementen $v_l = v_1 = 2$ und $v_{l+1} = v_2 = 4$. Beide Elemente sind positiv, sodass der EXTENDEDGCD ermittelt wird:

$$(g, a, b) = \text{EXTENDEDGCD}(2, 4) = (2, 1, 0).$$

Anschließend wird eine Permutationsmatrix T erzeugt,

$$T := \begin{pmatrix} -\mathbf{v_{l+1}/g} & \mathbf{a} & 0 & 0 \\ \mathbf{v_l/g} & \mathbf{b} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{-2} & \mathbf{1} & 0 & 0 \\ \mathbf{1} & \mathbf{0} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

mit der H und U multipliziert werden:

$$H := \begin{pmatrix} 2 & 0 & 0 & -3 \\ 0 & 2 & -3 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},\ U := \begin{pmatrix} -2 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

4. Normalisierung
Das Diagonalelement $h_{2,2}$ ist positiv, es muss nicht bearbeitet werden. Normalisierungsschritte erfolgen nun für $l = j + 1 = 3$ und $l = n = 4$.

$l = 3$:
Es wird noch auf der 2. Zeile gearbeitet, also auf $v := (\ 0 \quad 2 \quad -3 \quad -1\ )$. Man ermittelt $q := \lfloor v_3/v_2 \rfloor = -2$.

In H und U wird danach (-2)-mal die 2. von der 3. Spalte subtrahiert und man erhält aktualisierte Matrizen:

$$H := \begin{pmatrix} 2 & 0 & 0 & -3 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \; U := \begin{pmatrix} -2 & 1 & 2 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

$l = 4$:
Zunächst wird $v := \begin{pmatrix} 0 & 2 & 1 & -1 \end{pmatrix}$ aktualisiert. Danach erhält man $q := -1$ und in H und U wird jeweils (-1)-mal die 2. von der 4. Spalte subtrahiert:

$$H := \begin{pmatrix} 2 & 0 & 0 & -3 \\ 0 & 2 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \; U := \begin{pmatrix} -2 & 1 & 2 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Die Normalisierungsschritte für Zeile $i = 2$ sind damit abgeschlossen.

5. Neue Zeile und Spalte wählen
Man setze $i := i - 1 = 1$ und $j := j - 1 = 1$ und fahre mit Schritt 1 fort.

1. $[i = 1, j = 1]$:

3. Elimination
Es müssten Eliminationsschritte von $l = 1$ bis $l = j - 1 = 0$ durchgeführt werden. Diese Schleife wird nicht durchlaufen.

4. Normalisierung
Das Diagonalelement $h_{1,1}$ ist positiv, es wird nicht bearbeitet. Normalisierungsschritte erfolgen nun für $l = j + 1 = 2$ bis $l = 4$.

$l = 2$:
Man erhält $v := \begin{pmatrix} 2 & 0 & 0 & -3 \end{pmatrix}$ und damit $q := 0$. Damit werden H und U nicht verändert.

$l = 3$:
Hier ist ebenfalls $v := \begin{pmatrix} 2 & 0 & 0 & -3 \end{pmatrix}$ und $q := 0$. H und U werden nicht aktualisiert.

$l = 4$:
Es gilt $v := (\ 2 \quad 0 \quad 0 \quad -3\ )$ und damit $q := \lfloor -3/2 \rfloor = -2$. In H und U wird (-2)-mal die 1. von der 4. Spalte subtrahiert. Die aktualisierten Matrizen sind:

$$H := \begin{pmatrix} 2 & 0 & 0 & 1 \\ 0 & 2 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},\ U := \begin{pmatrix} -2 & 1 & 2 & -3 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

5. Neue Zeile und Spalte wählen
Man setze $i := i - 1 = 0$ und $j := j - 1 = 0$ und fahre mit Schritt 1 fort.

1. $[i = 0, j = 0]$:
Alle Zeilen und Spalten der Matrix wurden durchlaufen und der Algorithmus stoppt. Die Hermite-Normalform H und zugehörige Transformationsmatrix U sind damit die in Abbildung 4.4 dargestellten Matrizen.

$$H = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 0 & 2 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} -2 & 1 & 2 & -3 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Abbildung 4.4: *Ergebnismatrizen nach Ablauf von Algorithmus 4.5*

4.2.2 Die Algorithmen von Gerold Jäger

4.2.2.1 Vorstellung der Algorithmen

In diesem Abschnitt werden die zwei Algorithmen aus der Dissertation von Gerold Jäger [Jäg01] zur Berechnung der Hermite-Normalform (Algorithmus 4.6) und Zeilen-Hermite-Normalform (Algorithmus 4.8) vorgestellt.

Grundsätzlich arbeiten beide Algorithmen identisch, wobei einmal unimodulare Spalten- und einmal unimodulare Zeilenoperationen ausgeführt werden. Die folgende Dokumentation konzentriert sich auf Algorithmus 4.6, der durch die Erzeugung einer unteren Dreiecksmatrix den in früheren Kapiteln vorgestellten Algorithmen ähnlicher ist. Die einzelnen Schritte sind in Tabelle 4.4 dargestellt.

Während bei den Algorithmen für Matrizen mit vollem Zeilenrang die Zeilenpositionen der Diagonalelemente bereits zu Beginn bekannt ist, ermitteln die Algorithmen von Gerold Jäger diese dynamisch. Aus diesem Grund werden Eliminations- und Normalisierungsschritte in Abhängigkeit vom Wert einer Variable r ausgeführt.

Schritt	Beschreibung
0.1	Initialisierung: $H := A$, $U := I_n$. Starte mit Spalte $t = 1$.
0.2	Setze $r := 0$, $s := 1$.
0.3	Ist mindestens einer der Einträge $h_{s,r+1}$ oder $h_{s,t}$ von Null verschieden, kann ein Eliminationsschritt ausgeführt werden. Setze $r := r + 1$.
1	Gilt $t = r$, so ist $h_{s,t}$ das Pseudodiagonalelement. Ist dieses negativ, so wird Spalte t mit (-1) multipliziert.
2	Ist $t \neq r$, so findet ein Eliminationsschritt statt. Dazu wird Algorithmus ROWGCD aufgerufen.
3	Ist ein Diagonalelement vorhanden, werden alle Einträge in Zeile s links vom Diagonalelement normalisiert. Ist nun $t = r$, wähle die nächste Spalte $t + 1$ und gehe zu 0.2. Ansonsten wähle die nächste Zeile $s + 1$ und gehe zu 0.3.

Tabelle 4.4: *Schritte des HNF-Algorithmus nach Jäger*

In r, wobei bei jedem Durchlauf einer Spalte t zunächst $r := 0$ gilt, wird der Spaltenindex des Diagonalelements der Zeile s gespeichert. Wenn mindestens einer der Einträge $h_{s,t}$ oder $h_{s,r+1}$ von Null verschieden ist, wird r um 1 hochgezählt und es werden Eliminations- und Normalisierungsschritte ausgeführt, denn diese Zeile besitzt dann ein Diagonalelement. Besteht eine Zeile z.B. nur aus Nullen, wird somit kein Diagonalelement gefunden.

Algorithmus 4.6 stellt den gesamten Algorithmus dar. Neben der Hermite-Normalform wird die Transformationsmatrix U berechnet.

Algorithmus 4.6: COLUMNHNF

Eingabe: $A_{m\times n}$
Ausgabe: $(H_{m\times n}, U_{n\times n})$, wobei $H = HNF(A)$ und $H = A \cdot U$.

```
1  //Initialisierung
2  H := A;
3  U := I_n;
4
5  //HNF-Berechnung
6  for t := 1 to n do
7     r := 0;
8     for s := 1 to m do
9        if (h_{s,r+1} ≠ 0) ∨ (h_{s,t} ≠ 0) then
10          r := r + 1;
```

```
11        if t = r then
12          //Schritt 1: Pseudodiagonalelemente müssen positiv sein
13          if h_{s,t} < 0 then
14            h_{*,t} := −h_{*,t};
15            u_{*,t} := −u_{*,t};
16          end_if
17        else
18          //Schritt 2: Elimination
19          (H, U) := rowGCD(H, U, s, r, t);
20        end_if
21        //Schritt 3: Normalisierung
22        for l := 1 to r − 1 do
23          f := ⌊h_{s,l}/h_{s,r}⌋;
24          h_{*,l} := h_{*,l} − f · h_{*,r};
25          u_{*,l} := u_{*,l} − f · u_{*,r};
26        end_for
27        if t = r then
28          break;
29        end_if
30      end_if
31    end_for
32  end_for
33
34  return (H, U);
```

Der eigentliche Eliminationsschritt in Algorithmus 4.6 erfolgt durch den Algorithmus rowGCD. Übergeben werden ihm neben den Matrizen H und U auch die Positionen des aktuellen Pseudodiagonalelements und zu annullierenden Elements. Durch die Konstruktion einer geeigneten Permutationsmatrix und Verwendung des extended-GCD wird dann ein Element annulliert, das Diagonalelement wird durch den größten gemeinsamen Teiler beider Einträge ersetzt.

Algorithmus 4.7: rowGCD

Eingabe: $(H_{m\times n}, U_{n\times n}, z, s_1, s_2)$ mit $1 \le z \le m$, $1 \le s_1 < s_2 \le n$
Ausgabe: (H, U) mit $h^{neu}_{z,s_1} = ggT(h^{alt}_{z,s_1}, h^{alt}_{z,s_2})$ und $h^{neu}_{z,s_2} = 0$

```
1  if (h_{z,s_1} ≠ 0) ∨ (h_{z,s_2} ≠ 0) then
2    //ggT−Berechnung
3    (d, u, v) := extendedGCD(h_{z,s_1}, h_{z,s_2});
4    //Konstruktion der Transformationsmatrix
5    T := I_n;
```

```
6     t_{s_1,s_1} := u;
7     t_{s_1,s_2} := -h_{z,s_2}/d;
8     t_{s_2,s_1} := v;
9     t_{s_2,s_2} := h_{z,s_1}/d;
10    //Annullierung des Matrixelements
11    H := H · T;
12    U := U · T;
13  end_if
14
15  return (H, U);
```

Jetzt stellt sich die Frage, wie die Zeilen ermittelt werden, in denen *keine* Diagonalelemente stehen. Laut Definition 3.7 ist die Hermite-Normalform so gestaltet, dass jede ihrer ersten $rg(A)$ Spalten ein Diagonalelement besitzt. Gegeben sei der Ausschnitt einer Matrix H, für die gerade die Schleifendurchläufe $t = 5$ (Spalte) und $s = 4$ (Zeile) abgearbeitet wurden:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 1 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 1 & 0 & 0 & \dots \\ \mathbf{-1} & \mathbf{-1} & \mathbf{-1} & \mathbf{-1} & \mathbf{0} & \mathbf{0} & \mathbf{\dots} \\ 0 & 0 & 0 & 0 & 0 & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Der nächste Durchlauf ist damit $t = 5$ und $s = 5$, der aktuelle Wert von r ist $r = 4$, denn bei den Durchläufen $s = 1$ bis $s = 4$ wurde r jeweils um 1 hochgezählt. In Zeile $s = 5$ wird nun überprüft, ob $h_{s,r+1} = h_{s,t} = h_{5,5}$ von Null verschieden ist. Da dies nicht der Fall ist, wird r nicht erhöht und die gesamte Zeile nicht weiter bearbeitet. Damit ist die erste Zeile gefunden, die kein Diagonalelement enthält.

Der Wert für r wird erst wieder hochgezählt, wenn in den folgenden Zeilendurchläufen $s > 5$ ein Element $h_{s,5} \neq 0$ ist. Dann werden Eliminationsschritte auf dieser Zeile s ausgeführt. Falls keine solche Zeile gefunden wird, geschieht es bei Spaltendurchläufen $t > 5$: Nach Durchlauf der ersten 4 Zeilen von H erhält man zum Beispiel im Schritt $t = 6$ in jedem Fall wieder den Wert $r = 4$. Zeile $s = 5$ wird wieder übersprungen. In Zeile $s = 6$ gilt dann $h_{s,t} = h_{6,6} = 1$, sodass r hochgezählt wird. Da $t \neq r$ gilt, $r = 5$ aber Diagonalelementsspalte sein muss, wird $h_{6,6}$ annulliert und ein Diagonalelement an Position $h_{6,5}$ erzeugt. Das Diagonalelement für Spalte 5 wurde also erst durch Bearbeitung von Spalte 6 gefunden.

Der folgende Algorithmus ROWHNF berechnet die Zeilen-Hermite-Normalform. Verwendet man ihn unter Eingabe der transponierten Matrix A^T einer gegebenen Matrix A, so ist die resultierende Hermite-Normalform $\text{ROWHNF}(A^T)$ eine obere Dreiecksmatrix, für die gilt:

$$(\text{ROWHNF}(A^T))^T = \text{COLUMNHNF}(A).$$

Eine Besonderheit der Zeilen-Hermite-Normalform ist ihre Linksäquivalenz zur Ausgangsmatrix (vgl. Definition 3.3), es werden also elementare Zeilenoperationen ausgeführt.

Algorithmus 4.8: ROWHNF

Eingabe: $A_{m\times n}$
Ausgabe: $(H_{m\times n}, U_{m\times m})$, wobei $H = LHNF(A)$ und $H = U \cdot A$.

```
1  //Initialisierung
2  H := A;
3  U := I_m;
4
5  //HNF-Berechnung
6  for t := 1 to m do
7    r := 0;
8    for s := 1 to n do
9      if (h_{r+1,s} ≠ 0) ∨ (h_{t,s} ≠ 0) then
10       r := r + 1;
11       if t = r then
12         //Schritt 1: Pseudodiagonalelemente müssen positiv sein
13         if h_{t,s} < 0 then
14           h_{t,*} := −h_{t,*};
15           u_{t,*} := −u_{t,*};
16         end_if
17       else
18         //Schritt 2: Elimination
19         (H, U) := COLUMNGCD(H, U, s, r, t);
20       end_if
21       //Schritt 3: Normalisierung
22       for l := 1 to r − 1 do
23         f := ⌊h_{l,s}/h_{r,s}⌋;
24         h_{l,*} := h_{l,*} − f · h_{r,*};
25         u_{l,*} := u_{l,*} − f · u_{r,*};
26       end_for
27       if t = r then
```

```
28            break;
29          end_if
30        end_if
31      end_if
32  end_for
33
34  return (H,U);
```

Auch bei dem zu Algorithmus 4.8 gehörenden Eliminationsschritt, dargestellt durch Algorithmus 4.9, ist die Ausgabematrix linksäquivalent zur Eingabematrix. Wieder werden also unimodulare Zeilenoperationen ausgeführt.

Algorithmus 4.9: COLUMNGCD

Eingabe: $(H_{m\times n}, U_{m\times m}, s, z_1, z_2)$ mit $1 \leq s \leq n$, $1 \leq z_1 < z_2 \leq m$
Ausgabe: (H,U) mit $h^{neu}_{z_1,s} = ggT(h^{alt}_{z_1,s}, h^{alt}_{z_2,s})$ und $h^{neu}_{z_2,s} = 0$

```
1  if (h_{z1,s} ≠ 0) ∨ (h_{z2,s} ≠ 0) then
2    //ggT−Berechnung
3    (d,u,v) := EXTENDEDGCD(h_{z1,s}, h_{z2,s});
4    //Konstruktion der Transformationsmatrix
5    T := I_m;
6    t_{z1,z1} := u;
7    t_{z2,z1} := −h_{z2,s}/d;
8    t_{z1,z2} := v;
9    t_{z2,z2} := h_{z1,s}/d;
10   //Annullierung des Matrixelements
11   H := T · H;
12   U := T · U;
13 end_if
14
15 return (H,U);
```

4.2.2.2 Anwendungsbeispiel

Mit Algorithmus 4.6 wird nun die Hermite-Normalform für die Matrix

$$A_{4\times 4} = \begin{pmatrix} 0 & 2 & 0 & -3 \\ 2 & 4 & -3 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

berechnet. Man erhält die zugehörige Zeilen-Hermite-Normalform, indem man Algorithmus 4.8 auf auf die transponierte Matrix A^T anwendet. Interessant ist nun die Frage, ob die nun folgenden Berechnungen das gleiche Ergebnis liefern wie Algorithmus 4.2, da A vollen Zeilenrang hat und die Definitionen der Hermite-Normalform von Schrijver und Jäger in diesem Fall identisch sind. Gestartet wird wieder mit der Initialisierung:

$$H := A,\ U := I_4.$$

Anschließend werden die Schleifen t und s durchlaufen:

$[t = 1]$:
$r := 0$ wird gesetzt.

$[t = 1, s = 1]$:
Es gilt $h_{s,r+1} = h_{1,1} = h_{s,t} = 0$, daher werden keine weiteren Schritte ausgeführt.

$[t = 1, s = 2]$:
Es ist $h_{s,r+1} = h_{2,1} = 2 \neq 0$. Man setze $r := r + 1 = 1$. Nun gilt $t = r$.

1. Pseudodiagonalelement auf Negativität testen
Es wird überprüft, ob das Pseudodiagonalelement $h_{s,t} = h_{2,1}$ negativ ist. Dies ist nicht der Fall, somit müssen keine weiteren Schritte ausgeführt werden.

3. Normalisierung
Es werden keine Normalisierungsschritte ausgeführt.

Da $t = r$ gilt, wird die Schleife s abgebrochen.

$[t = 2]$:
$r := 0$ wird gesetzt.

$[t = 2, s = 1]$:
Es gilt $h_{s,t} = h_{1,2} = 2 \neq 0$. Man setze $r := r + 1 = 1$. Nun ist $t \neq r$, d.h. es wird ein Eliminationsschritt ausgeführt.

2. Elimination
Der Algorithmus ROWGCD$(H, U, 1, 1, 2)$ wird aufgerufen. Innerhalb dieses Algorith-

mus wird nun der EXTENDEDGCD berechnet:

$$(d, u, v) = \text{EXTENDEDGCD}(h_{1,1}, h_{1,2}) = (2, 0, 1).$$

Die Permutationsmatrix T wird konstruiert, mit der man H und U multipliziert:

$$T = \begin{pmatrix} \mathbf{0} & \mathbf{-1} & 0 & 0 \\ \mathbf{1} & \mathbf{0} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, H := \begin{pmatrix} 2 & 0 & 0 & -3 \\ 4 & -2 & -3 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, U := \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

3. Normalisierung
Es finden keine Normalisierungsschritte statt.

$[t = 2, s = 2]$:
Es gilt $h_{s,r+1} = h_{2,2} = -2 \neq 0$. Man setze $r := r + 1 = 2$. Nun ist $t = r = 2$.

1. Pseudodiagonalelement auf Negativität testen
$h_{s,t} = h_{2,2} = -2 < 0$, d.h. in H und U wird jeweils Spalte $t = 2$ mit (-1) multipliziert:

$$H := \begin{pmatrix} 2 & 0 & 0 & -3 \\ 4 & 2 & -3 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, U := \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

3. Normalisierung
Normalisiert wird nun für den Fall $l = 1 = r - 1$. $f := \lfloor h_{2,1}/h_{2,2} \rfloor = 2$ wird berechnet und folglich 2-mal Spalte 2 von Spalte 1 subtrahiert. Da $t = r$ gilt, wird danach die Schleife s abgebrochen.

$$H := \begin{pmatrix} 2 & 0 & 0 & -3 \\ 0 & 2 & -3 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, U := \begin{pmatrix} -2 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

$[t = 3]$:
$r := 0$ wird gesetzt.

$[t = 3, s = 1]$:
Es gilt $h_{s,r+1} = h_{1,1} = 2 \neq 0$. Man setze $r := r + 1 = 1$. Nun ist $t \neq r$, d.h. es wird ein

Eliminationsschritt ausgeführt.

2. Elimination
Der Algorithmus ROWGCD$(H, U, 1, 1, 3)$ wird aufgerufen. Innerhalb dieses Algorithmus wird nun der EXTENDEDGCD berechnet:

$$(d, u, v) = \text{EXTENDEDGCD}(h_{1,1}, h_{1,3}) = (2, 1, 0).$$

Anschließend wird die Permutationsmatrix T konstruiert, die in diesem Fall der Einheitsmatrix I_4 entspricht. H und U werden dadurch nicht verändert, denn es gilt ohnehin schon $h_{1,3} = 0$.

3. Normalisierung
Es finden keine Normalisierungsschritte statt.

$[t = 3, s = 2]$:
Es gilt $h_{s,r+1} = h_{2,2} = 2 \neq 0$. Man setze $r := r + 1 = 2$. Es gilt $t \neq r$.

2. Elimination
ROWGCD$(H, U, 2, 2, 3)$ wird aufgerufen. Es wird der EXTENDEDGCD berechnet:

$$(d, u, v) = \text{EXTENDEDGCD}(h_{2,2}, h_{2,3}) = (1, -1, -1).$$

Anschließend wird die Permutationsmatrix T konstruiert, mit der H und U multipliziert werden:

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \mathbf{-1} & \mathbf{3} & 0 \\ 0 & \mathbf{-1} & \mathbf{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \; H := \begin{pmatrix} 2 & 0 & 0 & -3 \\ 0 & 1 & 0 & -1 \\ 0 & -1 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \; U := \begin{pmatrix} -2 & -1 & 3 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Mit Hilfe von Diagonalelement $h_{2,2}$ dieser Zeile wurde also Eintrag $h_{2,3}$ annulliert.

3. Normalisierung
Normalisiert wird für $l = 1$. Da jedoch $f := \lfloor h_{2,1}/h_{2,2} \rfloor = 0$ ist, wird an H und U nichts verändert.

$[t = 3, s = 3]$:
Es gilt $h_{s,r+1} = h_{3,3} = 2 \neq 0$. Man setze $r := r + 1 = 3$. Es gilt $t = r$.

1. Pseudodiagonalelement auf Negativität testen
$h_{3,3} = 2 > 0$, hier wird nichts verändert.

3. Normalisierung
Normalisiert wird von $l = 1$ bis $l = r - 1 = 2$. Für den Fall $l = 1$ erhält man wieder $f = 0$, H und U werden also nicht verändert. Im Fall $l = 2$ ist $f := \lfloor h_{3,2}/h_{3,3} \rfloor = -1$, damit wird in H und U jeweils (-1)-mal die 3. von der 2. Spalte subtrahiert:

$$H := \begin{pmatrix} 2 & 0 & 0 & -3 \\ 0 & 1 & 0 & -1 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \; U := \begin{pmatrix} -2 & 2 & 3 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Da $t = r$ gilt, wird die Schleife s abgebrochen.

$[t = 4]$:
$r := 0$ wird gesetzt.

$[t = 4, s = 1]$:
Es gilt $h_{s,r+1} = h_{1,1} = 2 \neq 0$. Man setze $r := r + 1 = 1$. Es gilt $t \neq r$.

2. Elimination
ROWGCD$(H, U, 1, 1, 4)$ wird aufgerufen. Es wird nun der EXTENDEDGCD berechnet:

$$(d, u, v) = \text{EXTENDEDGCD}(h_{1,1}, h_{1,4}) = (1, -1, -1).$$

Anschließend wird die Permutationsmatrix T konstruiert, mit der H und U multipliziert werden:

$$T = \begin{pmatrix} \mathbf{-1} & 0 & 0 & \mathbf{3} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \mathbf{-1} & 0 & 0 & \mathbf{2} \end{pmatrix}, \; H := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & -2 \\ 0 & 1 & 2 & 0 \\ -1 & 0 & 0 & 2 \end{pmatrix}, \; U := \begin{pmatrix} 2 & 2 & 3 & -6 \\ -1 & 0 & 0 & 3 \\ 0 & 1 & 2 & 0 \\ -1 & 0 & 0 & 2 \end{pmatrix}.$$

3. Normalisierung
Es werden keine Normalisierungsschritte ausgeführt.

$[t = 4, s = 2]$:
Es gilt $h_{s,r+1} = h_{2,2} = 1 \neq 0$. Man setze $r := r + 1 = 2$. Es gilt $t \neq r$.

2. Elimination
ROWGCD$(H, U, 2, 2, 4)$ wird aufgerufen. Innerhalb dieses Algorithmus wird nun der EXTENDEDGCD berechnet:

$$(d, u, v) = \text{EXTENDEDGCD}(h_{2,2}, h_{2,4}) = (1, 1, 0).$$

Anschließend wird die Permutationsmatrix T konstruiert, mit der H und U multipliziert werden:

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & \mathbf{2} \\ 0 & 0 & 1 & 0 \\ 0 & \mathbf{0} & 0 & \mathbf{1} \end{pmatrix}, H := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 2 & 2 \\ -1 & 0 & 0 & 2 \end{pmatrix}, U := \begin{pmatrix} 2 & 2 & 3 & -2 \\ -1 & 0 & 0 & 3 \\ 0 & 1 & 2 & 2 \\ -1 & 0 & 0 & 2 \end{pmatrix}.$$

3. Normalisierung
Normalisiert wird nun für den Fall $l = 1 = r - 1$. Man erhält $f := \lfloor h_{2,1}/h_{2,2} \rfloor = 1$ und Spalte 2 wird 1-mal von Spalte 1 subtrahiert:

$$H := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 1 & 2 & 2 \\ -1 & 0 & 0 & 2 \end{pmatrix}, U := \begin{pmatrix} 0 & 2 & 3 & -2 \\ -1 & 0 & 0 & 3 \\ -1 & 1 & 2 & 2 \\ -1 & 0 & 0 & 2 \end{pmatrix}.$$

$[t = 4, s = 3]$:
Es gilt $h_{s,r+1} = h_{3,3} = 2 \neq 0$. Man setze $r := r + 1 = 3$. Es gilt $t \neq r$.

2. Elimination
ROWGCD$(H, U, 3, 3, 4)$ wird aufgerufen. Innerhalb dieses Algorithmus wird nun der EXTENDEDGCD berechnet:

$$(d, u, v) = \text{EXTENDEDGCD}(h_{3,3}, h_{3,4}) = (2, 0, 1).$$

Anschließend wird T konstruiert, mit der H und U multipliziert werden:

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \mathbf{0} & \mathbf{-1} \\ 0 & 0 & \mathbf{1} & \mathbf{1} \end{pmatrix}, H := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 1 & 2 & 0 \\ -1 & 0 & 2 & 2 \end{pmatrix}, U := \begin{pmatrix} 0 & 2 & -2 & -5 \\ -1 & 0 & 3 & 3 \\ -1 & 1 & 2 & 0 \\ -1 & 0 & 2 & 2 \end{pmatrix}.$$

3. Normalisierung
Normalisiert wird von $l = 1$ bis $l = r - 1 = 2$. Im Fall $l = 1$ ist $f := \lfloor h_{3,1}/h_{3,3} \rfloor = -1$, damit wird in H und U jeweils (-1)-mal die 3. von der 1. Spalte subtrahiert:

$$H := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 2 & 0 \\ 1 & 0 & 2 & 2 \end{pmatrix}, \; U := \begin{pmatrix} -2 & 2 & -2 & -5 \\ 2 & 0 & 3 & 3 \\ 1 & 1 & 2 & 0 \\ 1 & 0 & 2 & 2 \end{pmatrix}.$$

Für den Fall $l = 2$ ist $f = 0$, H und U werden also nicht verändert.

$[t = 4, s = 4]$:
Es gilt $h_{s,r+1} = h_{4,4} = 2 \neq 0$. Man setze $r := r + 1 = 4$. Es gilt $t = r$.

1. Pseudodiagonalelement auf Negativität testen
$h_{4,4} = 2 > 0$, hier wird nichts verändert.

3. Normalisierung
Normalisiert wird von $l = 1$ bis $l = r - 1 = 3$. In den Fällen $l = 1$ und $l = 2$ ist $f = 0$, H und U werden nicht verändert. Im Fall $l = 3$ ist $f := \lfloor h_{4,3}/h_{4,4} \rfloor = 1$, damit wird in H und U jeweils 1-mal die 4. von der 3. Spalte subtrahiert:

$$H := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix}, \; U := \begin{pmatrix} -2 & 2 & 3 & -5 \\ 2 & 0 & 0 & 3 \\ 1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix}.$$

Nun ist Schleife t an ihrem Ende angelangt und der Algorithmus stoppt. Die Hermite-Normalform H und Transformationsmatrix U sind damit die in Abbildung 4.5 dargestellten Matrizen. Die Ergebnisse sind identisch mit denen des Schrijver-Algorithmus.

$$H := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix}, \; U := \begin{pmatrix} -2 & 2 & 3 & -5 \\ 2 & 0 & 0 & 3 \\ 1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix}$$

Abbildung 4.5: *Ergebnismatrizen nach Ablauf von Algorithmus 4.6*

4.3 Zusammenfassung

In den letzten beiden Abschnitten wurde die Vorgehensweise vier verschiedener HNF-Algorithmen anhand eines ausführlichen Beispiels erläutert. Alle Berechnungsverfahren erhielten die gleiche Eingabematrix A und lieferten als Rückgabewert die in Abbildung 4.6 dargestellten Hermite-Normalformen.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & -1 & 2 & 0 \\ -1 & 0 & 0 & 2 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix} \quad \begin{pmatrix} 2 & 0 & 0 & 1 \\ 0 & 2 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix}$$

(a) HNFNW (b) HNFSCHRIJVER (c) HNFTHEO (d) COLUMNHNF

Abbildung 4.6: *Resultate der verschiedenen HNF-Algorithmen im Vergleich*

In allen Fällen sind die Matrizen korrekt diagonalisiert und liefern den zugrunde liegenden Definitionen entsprechende Ergebnisse. Die Ausgaben der Algorithmen HNFSCHRIJVER und COLUMNHNF sind sogar identisch, da Algorithmus COLUMNHNF bei Eingabe einer Matrix mit vollem Zeilenrang eine Hermite-Normalform erzeugen muss, die identisch mit der von HNFSCHRIJVER berechneten ist.

Die Ergebnismatrix von HNFTHEO ist eine horizontal und vertikal gespiegelte Version der Matrizen 4.6(b) und 4.6(d). Dies ist nicht allgemein der Fall. Matrix 4.6(a) hat die gleiche Form wie diese beiden Matrizen, nur dass die normalisierten Einträge negativ sind. Auch diese Gleichheit der Absolutwerte ist allgemein nicht der Fall. Was zeichnet nun jeden Algorithmus aus?

Algorithmus HNFNW
Der Algorithmus ist sehr leicht zu verstehen. Die Zeilen der Matrix werden nacheinander bearbeitet und darauf erst Eliminations-, dann Normalisierungsschritte ausgeführt. Die Position der Diagonalelemente ist von Anfang an bekannt, da nur Matrizen mit vollem Zeilenrang akzeptiert werden. In [Nem88] wird ein Verfahren beschrieben, wie - ähnlich wie bei Schrijver - die Matrixeinträge gering gehalten werden können. Soll der Algorithmus implementiert werden, ist dies eine sinnvolle Ergänzung.

Algorithmus HNFSCHRIJVER
Dieser Algorithmus unterscheidet sich von allen anderen, da er bei jedem Iterationsschritt k die Hermite-Normalform nicht zeilen- oder spaltenweise aufbaut, sondern immer den $k+1$-ten Minor in eine untere Dreiecksmatrix transformiert. Dies wird da-

durch ermöglicht, dass nur Matrizen mit vollem Zeilenrang bearbeitet werden und die Position der Diagonalelemente damit bekannt ist. Der Normalisierungsschritt erfolgt erst nach Ablauf der Diagonalisierung, allerdings werden alle Einträge durch Normalisierung mit einer vorher berechneten Unterdeterminante gering gehalten.

Algorithmus HNFschrijverComplete
Dieser Algorithmus erweitert den Algorithmus HNFschrijver, um zusätzlich die Transformationsmatrix U zu berechnen. Die Eingabematrix A wird zunächst zu einer regulären Matrix A_{neu} erweitert und anschließend mit Algorithmus 4.2 deren Hermite-Normalform berechnet. Danach wird aus der inversen Matrix von A_{neu} und der HNF die Transformationsmatrix berechnet.

Algorithmus HNFtheo
Dies ist der einzige Algorithmus dieser Arbeit, der eine Hermite-Normalform in oberer Dreiecksmatrixform erzeugt und dabei seine Berechnung in der letzten Zeile und Spalte beginnt. Die Spaltenindizes aller Diagonalelemente sind von Anfang an bekannt, die Zeilenindizes werden dynamisch ermittelt. Dazu wird in jeder Zeile durch Eliminationsschritte ein von Null verschiedenes Element so lange nach rechts verschoben, bis es an der korrekten Spaltenposition steht. Die anderen Einträge links von diesem Element werden jeweils annulliert. Nachdem der Diagonaleintrag erzeugt wurde, erfolgt der Normalisierungsschritt für die gleiche Zeile. Zeilen, die nur Nullen bis zum Spaltenindex des letzten berechneten Diagonalelements enthalten, werden ignoriert.

Algorithmus columnHNF
Dieser und sein verwandter Algorithmus rowHNF erzeugen eine untere bzw. obere Dreiecksmatrix, die bei Eingabe einer Matrix und deren transponierter Form in jeweils einen der Algorithmen in Beziehung zueinander stehen: Die resultierenden Hermite-Normalformen liegen auch in normaler und transponierter Form vor. Übergibt man an rowHNF eine beliebige Matrix $A_{m\times n}$, so hat die zugehörige HNF nicht $n - rg(A)$, sondern $m - rg(A)$ Nullzeilen, da unimodulare Zeilenoperationen ausgeführt werden. Nähere Informationen sind in Abschnitt 5 zu finden.

Auch innerhalb dieser Algorithmen wird der Zeilenindex (bzw. Spaltenindex bei rowHNF) dynamisch ermittelt und ist an Eliminationsschritte gekoppelt: In columnHNF (und in rowHNF analog, nur alle Operationen transponiert) werden während der Bearbeitung einer Spalte t nacheinander alle Zeilen s durchlaufen. Die Spaltenposition des Diagonalelements der Zeile s wird dabei in der Variablen r gespeichert. In jeder Zeile s wird mit Hilfe des Diagonalelements $h_{s,r}$ dann das Element $h_{s,t}$ annulliert.

Hat eine Zeile kein Diagonalelement, so wird in den nächsten Zeilen fortgefahren und dabei danach gesucht. Wird dort auch keins gefunden, so erfolgt die Bearbeitung der nächsten Spalte. Durch die Variable r bleibt gewährleistet, dass der Spaltenindex des nächsten zu findenden Diagonalelements gespeichert bleibt. Die Normalisierungsschritte erfolgen für jede Zeile, nachdem ihr Diagonalelement festgelegt wurde.

5 Lösen linearer Gleichungssysteme mittels der Hermite-Normalform

Bereits in Kapitel 2.3 auf Seite 11 wurden lineare Gleichungssysteme vorgestellt und unter Verwendung des Gaußschen Eliminationsverfahrens Kriterien aufgelistet, mit deren Hilfe Aussagen über die Existenz und Gestalt von Lösungen getroffen werden können. Die existierenden Lösungen sind dabei im Allgemeinen rational.

Die Herausforderung ist es nun, ganzzahlige Lösungen für ein lineares inhomogenes Gleichungssystem

$$A \cdot \vec{x} = \vec{b}$$

zu finden, genauer gesagt eine *ganzzahlige partielle Lösung* für das System $A \cdot \vec{x} = \vec{b}$ und – falls vorhanden – eine ganzzahlige *Basis* des Systems $A \cdot \vec{x} = \vec{0}$. Da die Forderung der Ganzzahligkeit auch für Linearkombinationen der Basisvektoren gilt, spricht man von einer $\mathbb{Z}$-*Basis*.

In den nächsten Abschnitten werden zwei Verfahren vorgestellt, bei denen aus der Hermite-Normalform und zugehöriger Transformationsmatrix wiederum Kriterien abgeleitet werden können, die über die Existenz von Lösungen entscheiden. Anschließend wird jeweils ein Berechnungsverfahren für die Lösungen vorgestellt und an geeigneten Beispielen erläutert.

5.1 Berechnung aller Lösungen für ganzzahlige Matrizen mit vollem Zeilenrang

5.1.1 Vorgehensweise

Dieser Abschnitt beschreibt ein Verfahren, wie ganzzahlige Lösungen linearer inhomogener Gleichungssysteme für Matrizen mit vollem Zeilenrang gewonnen werden können. Kriterium für die Existenz einer solchen Lösung ist wie bekannt

$$rg(A) = rg([A|\vec{b}]).$$

Da nur Matrizen $A \in \mathbf{Mat}_{m\times n}(\mathbb{Z})$ mit vollem Zeilenrang betrachtet werden, gilt in jedem Fall $rg(A) = m \leq n$. Im Folgenden wird davon ausgegangen, dass die zugehörige Hermite-Normalform die Gestalt $(H, 0)$ hat, d.h. es sind genau die ersten m Spalten in Normalform, die verbleibenden $n - m$ Spalten sind Nullspalten. Es gilt $H = A \cdot U$ und man unterscheidet zwei Fälle:

Fall 1: $rg(A) = n$
In diesem Fall ist A quadratisch und invertierbar und das Gleichungssystem hat genau eine eindeutige Lösung. H und U sind damit ebenfalls quadratisch und nichtsingulär, da sie durch unimodulare Operationen aus A bzw. einer Einheitsmatrix entstanden sind. H besitzt somit keine Nullspalten. Die eindeutige Lösung $\vec{x}$ kann durch einfache algebraische Umformungen hergeleitet werden:

$$\begin{aligned} & H &=& A \cdot U \\ \Rightarrow\quad & H \cdot U^{-1} &=& A \cdot U \cdot U^{-1} \\ \Rightarrow\quad & H \cdot U^{-1} &=& A. \end{aligned}$$

Nun wird A in der Ausgangsgleichung ersetzt und man erhält

$$\begin{aligned} & A \cdot \vec{x} &=& \vec{b} \\ \Rightarrow\quad & (H \cdot U^{-1}) \cdot \vec{x} &=& \vec{b} \\ \Rightarrow\quad & (H^{-1} \cdot H) \cdot U^{-1} \cdot \vec{x} &=& H^{-1} \cdot \vec{b} \\ \Rightarrow\quad & U \cdot U^{-1} \cdot \vec{x} &=& U \cdot H^{-1} \cdot \vec{b} \\ \Rightarrow\quad & \vec{x} &=& U \cdot H^{-1} \cdot \vec{b}. \end{aligned}$$

Die eindeutige Lösung $\vec{x}$ für ein lineares inhomogenes Gleichungssystem $A \cdot \vec{x} = \vec{b}$ mit $rg(A) = m = n$ ist damit

$$\vec{x} = U \cdot H^{-1} \cdot \vec{b}.$$

Ist $\vec{x}$ nicht ganzzahlig, so besitzt das System keine ganzzahlige eindeutige Lösung.

Fall 2: $rg(A) \leq n$
In diesem Fall hat H möglicherweise das Format $(H, 0)$, d.h. die letzten $n-m$ Spalten sind Nullspalten. Man erhält also eine Gleichung der Form

$$\left(\begin{array}{ccc|ccc} h_{1,1} & \dots & h_{1,m} & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ h_{m,1} & \dots & h_{m,m} & 0 & \dots & 0 \end{array}\right) = A \cdot \left(\begin{array}{ccc|ccc} u_{1,1} & \dots & u_{1,m} & u_{1,m+1} & \dots & u_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ u_{n,1} & \dots & u_{n,m} & u_{n,m+1} & \dots & u_{n,n} \end{array}\right).$$

Folglich gilt auch

$$\begin{pmatrix} h_{1,1} & \dots & h_{1,m} \\ \vdots & \ddots & \vdots \\ h_{m,1} & \dots & h_{m,m} \end{pmatrix} = A \cdot \begin{pmatrix} u_{1,1} & \dots & u_{1,m} \\ \vdots & \ddots & \vdots \\ u_{n,1} & \dots & u_{n,m} \end{pmatrix}$$

sowie

$$\begin{pmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{pmatrix} = A \cdot \begin{pmatrix} u_{1,m+1} & \dots & u_{1,n} \\ \vdots & \ddots & \vdots \\ u_{n,m+1} & \dots & u_{n,n} \end{pmatrix}.$$

Die $n - m$ Spaltenvektoren

$$\begin{pmatrix} u_{1,m+1} \\ \vdots \\ u_{n,m+1} \end{pmatrix}, \dots, \begin{pmatrix} u_{1,n} \\ \vdots \\ u_{n,n} \end{pmatrix}$$

bilden eine $\mathbb{Z}$-*Basis* des homogenen Gleichungssystems $A \cdot \vec{x} = \vec{0}$, denn sie sind linear unabhängig (da U vollen Spaltenrang hat) und ganzzahlig. Nun muss nur noch die partielle Lösung $\vec{x}_0$ des inhomogenen Gleichungssystems ermittelt werden. Dies erfolgt wie in Fall 1:

$$\vec{x}_0 = \begin{pmatrix} u_{1,1} & \dots & u_{1,m} \\ \vdots & \ddots & \vdots \\ u_{n,1} & \dots & u_{n,m} \end{pmatrix} \cdot \begin{pmatrix} h_{1,1} & \dots & h_{1,m} \\ \vdots & \ddots & \vdots \\ h_{m,1} & \dots & h_{m,m} \end{pmatrix}^{-1} \cdot \vec{b},$$

also

$$\vec{x}_0 = U_{1..n,1..m} \cdot (H_{1..m,1..m})^{-1} \cdot \vec{b}.$$

Ist $\vec{x}_0$ nicht ganzzahlig, so hat das inhomogene Gleichungssystem keine ganzzahlige Lösung.

In [Sch99] ist ein Verfahren angegeben, mit dem für Matrizen mit vollem Zeilenrang in einem Schritt mittels der Hermite-Normalform und Transformationsmatrix alle ganzzahligen Lösungen eines inhomogenen linearen Gleichungssystems

$$A \cdot \vec{x} = \vec{b}$$

berechnet werden können:

$$[\vec{x}_0, \vec{x}_1, ..., \vec{x}_{n-m}] = U \cdot \begin{bmatrix} (H_{1..m,1..m})^{-1} \cdot \vec{b} & \mathbf{0}_{m \times (n-m)} \\ \mathbf{0}_{(n-m) \times 1} & I_{n-m} \end{bmatrix},$$

wobei im Fall $(H_{1..m,1..m})^{-1} \cdot \vec{b} \in \mathbb{Z}^n$ gilt:

$$\vec{x} = \vec{x}_0 + \sum_{i=1}^{n-m} \lambda_i \cdot \vec{x}_i, \quad \lambda_i \in \mathbb{Z}.$$

5.1.2 Anwendungsbeispiele

In diesem Kapitel wird das Verfahren von Schrijver (vgl. [Sch99, S. 59]) anhand von Beispielen veranschaulicht. Gegeben sei das inhomogene LGS $A_{m\times n} \cdot \vec{x} = \vec{b}$ mit $rg(A) = m$. Durch den vollen Zeilenrang besitzt das System mindestens eine Lösung, weil keine zwei Gleichungen zu einem Widerspruch führen können. Die vier nun zu untersuchenden Fälle sind:

1. Das System hat genau eine eindeutige ganzzahlige Lösung.
Gegeben sei das System

$$\begin{pmatrix} 0 & 2 & 0 & -3 \\ 2 & 4 & -3 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \vec{x} = \begin{pmatrix} -16 \\ -6 \\ 6 \\ 8 \end{pmatrix}.$$

Mit Algorithmus 4.4 erhält man die Hermite-Normalform H und Transformationsmatrix U:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix}, \; U = \begin{pmatrix} -2 & 2 & 3 & -5 \\ 2 & 0 & 0 & 3 \\ 1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix}.$$

H hat keine Nullspalten, folglich gilt $rg(A) = n = m = 4$. Die Werte werden eingesetzt:

$$[\vec{x}_0, \vec{x}_1, ..., \vec{x}_{n-m}] = U \cdot \begin{bmatrix} (H_{1..m,1..m})^{-1} \cdot \vec{b} & \mathbf{0}_{m\times(n-m)} \\ \mathbf{0}_{(n-m)\times 1} & I_{n-m} \end{bmatrix},$$

also

$$\vec{x}_0 = U \cdot \begin{bmatrix} (H_{1..4,1..4})^{-1} \cdot \vec{b} & \mathbf{0}_{4\times 0} \\ \mathbf{0}_{0\times 1} & I_0 \end{bmatrix}$$

und somit

$$\vec{x}_0 = U \cdot (H_{1..4,1..4})^{-1} \cdot \vec{b}.$$

Im nächsten Schritt muss die inverse Matrix zu $H_{1..4,1..4}$ ermittelt werden. Die Berechnung wird hier nicht weiter ausgeführt, da die Vorgehensweise in Abschnitt 4.1.2.3 im

Detail beschrieben ist[1]. Man erhält

$$(H_{1..4,1..4})^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & 0 & \frac{1}{2} \end{pmatrix}$$

und damit als *eindeutige ganzzahlige* Lösung des Gleichungssystems den Vektor

$$\vec{x}_0 = \begin{pmatrix} -2 & 2 & 3 & -5 \\ 2 & 0 & 0 & 3 \\ 1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & 0 & \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} -16 \\ -6 \\ 6 \\ 8 \end{pmatrix} = \begin{pmatrix} \mathbf{2} \\ \mathbf{4} \\ \mathbf{6} \\ \mathbf{8} \end{pmatrix}.$$

2. Das System hat genau eine eindeutige nicht ganzzahlige Lösung.
Gegeben sei das System

$$\begin{pmatrix} 0 & 2 & 0 & -3 \\ 2 & 4 & -3 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \vec{x} = \begin{pmatrix} -16 \\ -6 \\ 7 \\ 8 \end{pmatrix}.$$

H, U und somit auch $(H_{1..4,1..4})^{-1}$ sind identisch zum vorherigen Fall und man erhält

$$\vec{x}_0 = \begin{pmatrix} -2 & 2 & 3 & -5 \\ 2 & 0 & 0 & 3 \\ 1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & 0 & \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} -16 \\ -6 \\ 7 \\ 8 \end{pmatrix} = \begin{pmatrix} \mathbf{\frac{7}{2}} \\ \mathbf{4} \\ \mathbf{7} \\ \mathbf{8} \end{pmatrix}$$

als *eindeutige nicht ganzzahlige* Lösung des Gleichungssystems.

Für die folgenden zwei Fälle wird zur Berechnung der Algorithmus von Nemhauser/Wolsey (Algorithmus 4.1) verwendet.

3. Das System hat unendlich viele Lösungen. Die partielle Lösung $\vec{x}_0$ ist ganzzahlig.
Gegeben sei das System

$$\begin{pmatrix} 2 & 6 & 1 \\ 4 & 7 & 7 \end{pmatrix} \cdot \vec{x} = \begin{pmatrix} 7 \\ 4 \end{pmatrix}.$$

[1] Ein hilfreiches Java-Applet, das die inverse Matrix einer (4×4)-Matrix berechnet, findet man z.B. unter `http://www.math.ubc.ca/~israel/applet/mcalc/matcalc.html` [Abruf: 10.12.2007].

Algorithmus 4.1 liefert folgende Matrizen H und U:

$$H = \begin{pmatrix} 1 & 0 & 0 \\ -3 & 5 & 0 \end{pmatrix},\ U = \begin{pmatrix} -6 & 3 & 7 \\ 2 & -1 & -2 \\ 1 & 0 & -2 \end{pmatrix}.$$

Die Hermite-Normalform hat hier die Gestalt $(H, 0)$, denn die letzte Spalte ist eine Nullspalte. Es existiert eine $(n-m)$-elementige, also einelementige $\mathbb{Z}$-Basis. Nun wird die inverse Matrix zu $H_{1..2,1..2}$ ermittelt:

$$(H_{1..2,1..2})^{-1} = \begin{pmatrix} 1 & 0 \\ \frac{3}{5} & \frac{1}{5} \end{pmatrix}.$$

Anschließend wird wieder eingesetzt:

$$[\vec{x}_0, \vec{x}_1] = U \cdot \begin{bmatrix} (H_{1..2,1..2})^{-1} \cdot \vec{b} & \mathbf{0}_{2\times 1} \\ \mathbf{0}_{1\times 1} & I_1 \end{bmatrix},$$

also

$$[\vec{x}_0, \vec{x}_1] = \begin{pmatrix} -6 & 3 & 7 \\ 2 & -1 & -2 \\ 1 & 0 & -2 \end{pmatrix} \cdot \left[\begin{array}{cc|c} \multicolumn{2}{c|}{\begin{pmatrix} 1 & 0 \\ \frac{3}{5} & \frac{1}{5} \end{pmatrix} \cdot \begin{pmatrix} 7 \\ 4 \end{pmatrix}} & \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ \hline \multicolumn{2}{c|}{0} & 1 \end{array}\right]$$

und somit

$$[\vec{x}_0, \vec{x}_1] = \left[\begin{pmatrix} -27 \\ 9 \\ 7 \end{pmatrix}, \begin{pmatrix} 7 \\ -2 \\ -2 \end{pmatrix}\right].$$

Alle *ganzzahligen* Lösungen des Gleichungssystems sind damit gegeben als

$$\vec{x} = \begin{pmatrix} -27 \\ 9 \\ 7 \end{pmatrix} + \lambda \cdot \begin{pmatrix} 7 \\ -2 \\ -2 \end{pmatrix}, \quad \lambda \in \mathbb{Z}.$$

4. Das System hat unendlich viele Lösungen. Die partielle Lösung $\vec{x}_0$ ist nicht ganzzahlig.
Gegeben sei das System

$$\begin{pmatrix} 2 & 6 & 1 \\ 4 & 7 & 7 \end{pmatrix} \cdot \vec{x} = \begin{pmatrix} 7 \\ 5 \end{pmatrix}.$$

Die Matrizen H, U und $(H_{1..2,1..2})^{-1}$ sind identisch mit denen aus Fall 3. Damit kann direkt eingesetzt werden:

$$[\vec{x}_0, \vec{x}_1] = \begin{pmatrix} -6 & 3 & 7 \\ 2 & -1 & -2 \\ 1 & 0 & -2 \end{pmatrix} \cdot \left[\begin{array}{c|c} \begin{pmatrix} 1 & 0 \\ \frac{3}{5} & \frac{1}{5} \end{pmatrix} \cdot \begin{pmatrix} 7 \\ 5 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ \hline 0 & 1 \end{array} \right]$$

und somit

$$[\vec{x}_0, \vec{x}_1] = \left[\begin{pmatrix} \frac{-132}{5} \\ \frac{44}{5} \\ 7 \end{pmatrix}, \begin{pmatrix} 7 \\ -2 \\ -2 \end{pmatrix} \right].$$

Da die partielle Lösung nicht ganzzahlig ist, hat das Gleichungssystem keine ganzzahligen Lösungen.

5.2 Berechnung aller Lösungen für beliebige ganzzahlige Matrizen

Bereits in Kapitel 3.3.2 wurde darauf hingewiesen, dass die Hermite-Normalform für Matrizen ohne vollen Zeilenrang nicht zwangsläufig den von Hermite genannten Kriterien wie z.B. der Nichtnegativität aller Einträge entspricht: Für eine $(m \times n)$-Matrix A mit $rg(A) < m$ besitzen genau $rg(A)$ Zeilen ein Diagonalelement. Auf den verbleibenden $m - rg(A)$ Zeilen erfolgen keine Eliminations- oder Normalisierungsschritte.

Im vorherigen Abschnitt wurde beschrieben, wie die partielle Lösung eines inhomogenen Gleichungssystems mittels der inversen Hermite-Normalform berechnet werden kann. Dies funktioniert nur für quadratische Matrizen mit vollem Zeilen- und Spaltenrang. Aus diesem Grund wird nun ein anderes Verfahren beschrieben, das auf beliebige ganzzahlige Matrizen anwendbar ist.

Dieses Verfahren zum Ablesen der Lösungen ist konkret vom verwendeten Algorithmus abhängig und muss somit auf diesen zugeschnitten werden. Es basiert auf der Existenz und Analyse von $n - rg(A)$ Nullspalten in der Hermite-Normalform und den entsprechenden Spalten der zugehörigen Transformationsmatrix U. Insbesondere werden spezielle Matrixeinträge in U betrachtet, die nur bestimmte Werte annehmen können und damit entscheidende Kriterien für die Lösungsgewinnung sind.

Die Vorgehensweise wird im nächsten Abschnitt anhand von Algorithmus 4.6 erläutert. Eine Beschreibung der Verfahren für die restlichen Algorithmen aus dieser Arbeit erfolgt in Kapitel 5.2.2.

5.2.1 Vorgehensweise am Beispiel von Algorithmus 4.6

Die Grundlage für das nun präsentierte Verfahren bildet das inhomogene Gleichungssystem

$$A \cdot \vec{x} = \vec{b}.$$

Es hat genau dann mindestens eine Lösung, wenn

$$rg(A) = rg([A|\vec{b}]) \leq n$$

gilt. Berechnet man die Hermite-Normalform von A mit Algorithmus 4.6, so besitzen die ersten $rg(A)$ Spalten ein Diagonalelement, alle nachfolgenden Spalten sind Nullspalten. Für die HNF der Matrix $[A|\vec{b}]$ muss dies ebenso gelten, sonst hat das Gleichungssystem keine Lösung. Zur Berechnung aller Lösungen wird das System umgeformt zu

$$A' \cdot \vec{y} = \vec{0}$$

mit

$$A' = (A| - \vec{b}).$$

Das modifizierte Gleichungssystem hat damit die Form

$$\left(\begin{array}{ccc|c} a_{1,1} & \dots & a_{1,n} & -b_1 \\ \vdots & \ddots & \vdots & \vdots \\ a_{m,1} & \dots & a_{m,n} & -b_m \end{array}\right) \cdot \begin{pmatrix} y_1 \\ \vdots \\ y_{n+1} \end{pmatrix} = \vec{0}.$$

Anschließend werden die Hermite-Normalform H' und zugehörige Transformationsmatrix U' für die Matrix A' berechnet. Zum Ablesen der Lösungen werden mehrere Fälle unterschieden. Unterscheidungskriterium ist neben der Anzahl der Nullspalten von H' (gemeint sind hier die letzten $n + 1 - rg(A')$ Spalten) der Wert bestimmter Matrixeinträge in U'. Dazu muss die Arbeitsweise des Algorithmus genauer betrachtet werden.

columnHNF arbeitet sich nacheinander von Spalte 1 bis Spalte $n + 1$ durch die Matrix und erzeugt in den ersten $rg(A')$ Spalten ein Diagonalelement, dessen Zeilenposition dynamisch festgelegt wird:

$$H' = \left(\begin{array}{ccc|ccc} h'_{1,1} & \dots & h'_{1,rg(A')} & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \underbrace{h'_{m,1} \quad \dots \quad h'_{m,rg(A')}}_{\text{mit Diagonalelement}} & & & \underbrace{0 \quad \dots \quad 0}_{\text{Nullspalten}} & & \end{array}\right).$$

Interessant sind nun die möglichen Nullspalten von H' und entsprechende Spalten in U'. Da U' als Einheitsmatrix I_{n+1} initalisiert wird, enthält Zeile $u'_{n+1,*}$ zunächst nur einen von Null verschiedenen Wert an Position $u'_{n+1,n+1}$. Dadurch wird diese Zeile bei der Bearbeitung der ersten n Spalten nicht verändert, da alle Berechnungen unimodulare *Spalten*operationen sind: Es werden entweder

- Spalten mit -1 multipliziert (Nullen bleiben erhalten) oder
- Vielfache von Spalten addiert, sodass in der letzten Zeile von U' nur Nullen aufsummiert werden.

Erst wenn der Algorithmus Spalte $n+1$ bearbeitet, können Einträge verändert werden. In jeder durchlaufenen Zeile s, die ein Diagonalelement $h'_{s,r}$ besitzt, wird unter deren Verwendung der Eintrag $h'_{s,n+1}$ annulliert. Es wird demnach auf den Spalten r und $n+1$ gearbeitet und die Einträge $u'_{n+1,r+1}, ..., u'_{n+1,n}$ bleiben unverändert:

$$\left(\begin{array}{ccc|c|ccc|c} u'_{1,1} & \dots & u'_{1,r-1} & u'_{1,r} & u'_{1,r+1} & \dots & u'_{1,n} & u'_{1,n+1} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \underbrace{u'_{n+1,1}}_{} & \dots & \underbrace{u'_{n+1,r-1}}_{} & \underbrace{u'_{n+1,r}}_{} & \underbrace{0}_{} & \dots & \underbrace{0}_{} & \underbrace{u'_{n+1,n+1}}_{} \\ \multicolumn{3}{c|}{\text{erste } r{-}1 \text{ Spalten}} & \text{Spalte } r \text{ mit } r{\leq}rg(A') & \multicolumn{3}{c|}{\text{letzte Zeile konstant } =0} & \text{Spalte } n{+}1 \end{array}\right).$$

Der Spaltenindex r kann maximal den Wert $rg(A')$ erhalten. Im Fall $r = n+1$ (Schritt 1) wird $u'_{n+1,n+1}$ möglicherweise mit -1 multipliziert, bleibt aber von Null verschieden. Bei einem Eliminationsschritt, der den Algorithmus ROWGCD aufruft (Schritt 2), also $r < n+1$ ist, gilt es zwei Fälle zu unterscheiden. Einer der beiden Parameter $h'_{s,r}$ oder $h'_{s,n+1}$ ist zwingend von Null verschieden, sonst würde keine Elimination stattfinden. Der EXTENDEDGCD berechnet Werte (d, u, v).

1. Ist $h'_{s,r} = 0$, so muss an diese Position ein Diagonalelement gesetzt werden. Die erzeugte Permutationsmatrix T ersetzt $u'_{*,n+1}$ durch folgende Linearkombination der Spalten r und $n+1$:

$$-\frac{h'_{s,n+1}}{d} \cdot u'_{*,r} + \underbrace{\frac{h'_{s,r}}{d}}_{=0} \cdot u'_{*,n+1},$$

   also

$$-\frac{h'_{s,n+1}}{d} \cdot u'_{*,r}.$$

   Da $u'_{n+1,r} = 0$ ist, wird $u'_{n+1,n+1} = 0$ gesetzt.

2. Ist $h'_{s,r} \neq 0$, so stellt dieser Eintrag das Diagonalelement dar und Eintrag $h'_{s,n+1}$ wird annulliert. Der Eintrag $u'_{n+1,n+1}$ bleibt durch die Konstruktion der Permutationsmatrix in jedem Fall von Null verschieden, da Spalte $n+1$ durch folgende Linearkombination ersetzt wird:

$$-\frac{h'_{s,n+1}}{d} \cdot u'_{*,r} + \underbrace{\frac{h'_{s,r}}{d}}_{\neq 0} \cdot u'_{*,n+1}.$$

Im gesamten Ablauf des Algorithmus erhält somit maximal Spalte $rg(A')$ ein Diagonalelement und der Eintrag $u'_{n+1,n+1}$ erhält dabei möglicherweise eine neuen Wert. Die Elemente

$$u'_{n+1,rg(A')+1}, ..., u'_{n+1,n}$$

bleiben immer unverändert.

Insgesamt gilt also bei einer Matrix H' mit $n+1-rg(A')$ Nullspalten:

$$u'_{n+1,rg(A')+1} = ... = u'_{n+1,n} = 0$$

sowie

$$u'_{n+1,n+1} \in \mathbb{Z}.$$

Die Position und Eigenschaften dieser Werte sind ausreichend zur Konstruktion eines Verfahrens, das das direkte Ablesen ganzzahliger Lösungen für ein lineares inhomogenes Gleichungssystem $A \cdot \vec{x} = \vec{b}$ erlaubt. Zur besseren Verständlichkeit werden mehrere Fälle unterschieden, basierend auf der HNF H' und Transformationsmatrix U' bei Eingabe von Matrix $A' = [A| - \vec{b}]$.

Fall 1: H' hat genau eine Nullspalte.
Es gilt

$$H'_{m\times(n+1)} = \left(\begin{array}{ccc|c} h'_{1,1} & \dots & h'_{1,n} & 0 \\ \vdots & \ddots & \vdots & \vdots \\ h'_{m,1} & \dots & h'_{m,n} & 0 \end{array}\right),$$

damit auch $rg(A') = n$ und das System hat entweder *keine* ($rg(A) < n$) oder *genau eine* ($rg(A) = n$) Lösung. Nun muss die letzte Spalte der Transformationsmatrix U' betrachtet werden. Es gilt in jedem Fall

$$H' = A' \cdot U'.$$

U' hat die Form

$$U' = \left(\begin{array}{ccc|c} u'_{1,1} & \dots & u'_{1,n} & u'_{1,n+1} \\ \vdots & \ddots & \vdots & \vdots \\ u'_{n+1,1} & \dots & u'_{n+1,n} & u'_{n+1,n+1} \end{array}\right).$$

Da die letzte Spalte von H' eine Nullspalte ist, kann die Spalte $u'_{*,n+1}$ für $\vec{y}$ in die Gleichung $A' \cdot \vec{y} = \vec{0}$ eingesetzt werden und man erhält

$$\left(\begin{array}{ccc|c} a_{1,1} & \dots & a_{1,n} & -b_1 \\ \vdots & \ddots & \vdots & \vdots \\ a_{m-1,1} & \dots & a_{m-1,n} & -b_{m-1} \\ a_{m,1} & \dots & a_{m,n} & -b_m \end{array}\right) \cdot \begin{pmatrix} u'_{1,n+1} \\ \vdots \\ u'_{n,n+1} \\ \mathbf{u'_{n+1,n+1}} \end{pmatrix} = \vec{0}$$

und damit

$$A \cdot \begin{pmatrix} u'_{1,n+1} \\ \vdots \\ u'_{n,n+1} \end{pmatrix} - \mathbf{u'_{n+1,n+1}} \cdot \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix} = \vec{0},$$

also

$$A \cdot \begin{pmatrix} u'_{1,n+1} \\ \vdots \\ u'_{n,n+1} \end{pmatrix} = \mathbf{u'_{n+1,n+1}} \cdot \vec{b}.$$

Nun gilt es wieder drei Fälle zu unterscheiden:

- **Fall 1.1:** $u'_{n+1,n+1} = 0$
  Man erhält
  $$\mathbf{u'_{n+1,n+1}} \cdot \vec{b} = \vec{0},$$
  damit ist der Spaltenvektor $u'_{1..n,n+1}$ eine Lösung des homogenen Gleichungssystems $A \cdot \vec{x} = \vec{0}$ und kann nicht gleichzeitig Lösung des inhomogenen Systems sein. $A \cdot \vec{x} = \vec{b}$ hat damit *keine* Lösung ($rg(A) < n$).

- **Fall 1.2:** $u'_{n+1,n+1} = \pm 1$
  Man erhält
  $$A \cdot \begin{pmatrix} u'_{1,n+1} \\ \vdots \\ u'_{n,n+1} \end{pmatrix} = \vec{b} \quad \text{oder} \quad A \cdot \begin{pmatrix} u'_{1,n+1} \\ \vdots \\ u'_{n,n+1} \end{pmatrix} = -\vec{b},$$

damit stellt der Spaltenvektor

$$u'_{n+1,n+1} \cdot \begin{pmatrix} u'_{1,n+1} \\ \vdots \\ u'_{n,n+1} \end{pmatrix}$$

die *eindeutige ganzzahlige* Lösung des gesamten Gleichungssystems dar.

- **Fall 1.3:** $u'_{n+1,n+1} = z \in \mathbb{Z}\backslash\{0, 1, -1\}$
  In diesem Fall gilt
  $$A \cdot \begin{pmatrix} u'_{1,n+1} \\ \vdots \\ u'_{n,n+1} \end{pmatrix} = z \cdot \vec{b},$$
  das System besitzt somit zwar eine eindeutige Lösung, diese ist jedoch *nicht* ganzzahlig.

Fall 2: H' hat mehrere Nullspalten.
In diesem Fall hat H' die Form

$$H'_{m\times(n+1)} = \left(\begin{array}{ccc|ccc} h'_{1,1} & \dots & h'_{1,r} & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ h'_{m,1} & \dots & h'_{m,r} & 0 & \dots & 0 \end{array}\right)$$

und es gilt

$$rg(A) = r < n.$$

Damit gibt es in jedem Fall $n-r$ Lösungen des homogenen Gleichungssystems $A\cdot\vec{x} = \vec{0}$, die eine $\mathbb{Z}$-Basis bilden. Es werden nun die gleichen Kriterien diskutiert wie in Fall 1.

- **Fall 2.1:** $u'_{n+1,n+1} = 0$
  Auch hier erhält man wieder
  $$u'_{n+1,n+1} \cdot \vec{b} = \vec{0},$$
  Es gilt wieder $rg(A) < rg([A|\vec{b}])$ und das System $A \cdot \vec{x} = \vec{b}$ hat damit *keine* Lösung, denn die letzte Spalte $u'_{1..n,n+1}$ kann wieder *keine* Lösung des inhomogenen Systems sein.

  Die letzten $n - r$ Spalten von U' (ohne die Elemente der letzten Zeile) bilden jedoch eine $\mathbb{Z}$-Basis des homogenen Gleichungssystems $A \cdot \vec{x} = \vec{0}$.

- **Fall 2.2:** $u'_{n+1,n+1} = \pm 1$
  Man erhält
  $$A \cdot \begin{pmatrix} u'_{1,n+1} \\ \vdots \\ u'_{n,n+1} \end{pmatrix} = \pm\vec{b},$$
  damit stellt der Spaltenvektor
  $$\vec{x}_0 = u'_{n+1,n+1} \cdot \begin{pmatrix} u'_{1,n+1} \\ \vdots \\ u'_{n,n+1} \end{pmatrix}$$
  eine *ganzzahlige* Lösung des Gleichungssystems $A \cdot \vec{x} = \vec{b}$ dar. Links von Spalte $h'_{*,n+1}$ gibt es jedoch noch $n - r =: l$ weitere Nullspalten. Die Spaltenvektoren
  $$\vec{x}_1 = \begin{pmatrix} u'_{1,r+1} \\ \vdots \\ u'_{n,r+1} \end{pmatrix}, ..., \vec{x}_l = \begin{pmatrix} u'_{1,n} \\ \vdots \\ u'_{n,n} \end{pmatrix}$$
  bilden somit eine $\mathbb{Z}$-Basis des zugehörigen homogenen Gleichungssystems $A \cdot \vec{x} = \vec{0}$. Die allgemeine Lösung ist damit
  $$\vec{x} = \vec{x}_0 + \sum_{i=1}^{l} \lambda_i \vec{x}_i, \quad \lambda_i \in \mathbb{Z}.$$

- **Fall 2.3:** $u'_{n+1,n+1} = z \in \mathbb{Z}\backslash\{0, 1, -1\}$
  In diesem Fall gilt
  $$A \cdot \begin{pmatrix} u'_{1,n+1} \\ \vdots \\ u'_{n,n+1} \end{pmatrix} = z \cdot \vec{b},$$
  das System besitzt somit zwar eine partielle Lösung, diese ist jedoch *nicht* ganzzahlig. Wie in Fall 2.2 bilden die Spaltenvektoren $\vec{x}_1, ..., \vec{x}_l$ eine $\mathbb{Z}$-Basis des homogenen Gleichungssystems $A \cdot \vec{x} = \vec{0}$.

Fall 3: H' hat keine Nullspalte.
In diesem Fall ist $rg(A') = n + 1 > rg(A)$, somit existiert *keine Lösung.*

In Abbildung 5.1 ist das Verfahren für Algorithmus 4.6 (COLUMNHNF) noch einmal graphisch dargestellt.

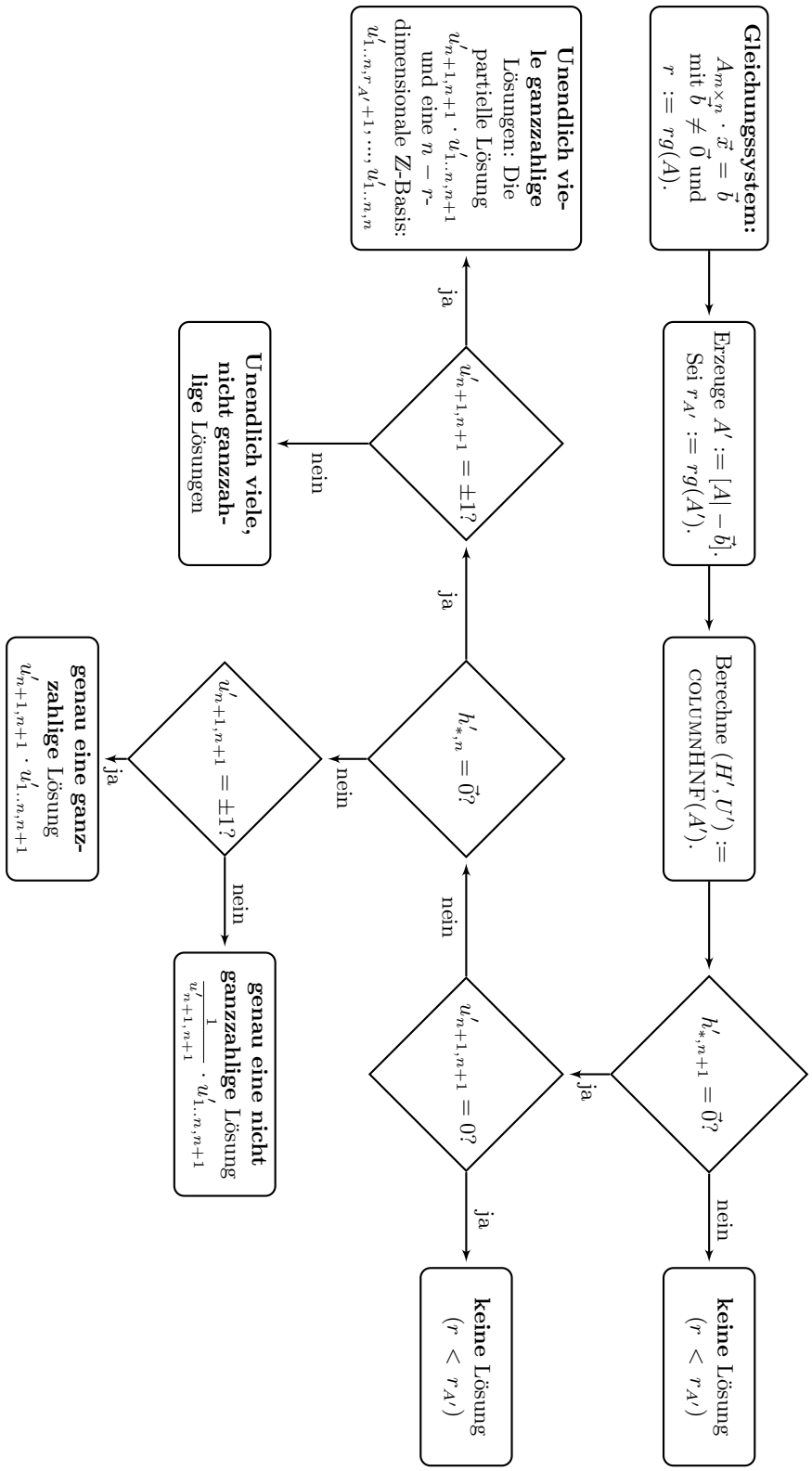

Abbildung 5.1: *Ermittlung aller Lösungen eines linearen inhomogenen Gleichungssystems mittels Algorithmus 4.6 (*columnHNF*).*

5.2.2 Vorgehensweise für die restlichen Algorithmen

Auch für zwei weitere Algorithmen dieser Arbeit können Verfahren zum Ablesen der Lösungen konstruiert werden. Dieser Abschnitt fasst die einzelnen Schritte für jeden Algorithmus zusammen. Nachfolgend wird davon ausgegangen, dass den Algorithmen ein Gleichungssystem der Form

$$A_{m\times n} \cdot \vec{x} = \vec{b}$$

mit $\vec{b} \neq \vec{0}$, $r := rg(A)$ und $A' := [A| - \vec{b}]$ zugrunde liegt.

5.2.2.1 Verfahren für den Algorithmus von Nemhauser/Wolsey

Zur Erinnerung: Bei diesem Algorithmus werden ausgehend vom Diagonalelement $h'_{i,i}$ einer Zeile i erst die Elemente in den Spalten $j > i$ eliminiert, anschließend die Elemente in den Spalten $k < i$ normalisiert.

Nach Initialisierung von U' ist in dessen $(n+1)$-ter Zeile zunächst nur das letzte Element $u'_{n+1,n+1} = 1$. Bei Eliminationsvorgängen auf den ersten n Spalten wird dieser Eintrag nicht verändert. Ebenso bleiben die Einträge $u'_{n+1,1}, ..., u'_{n+1,n}$ vor der ersten Abarbeitung von Spalte $n+1$ unverändert, da bei Eliminationsvorgängen nur Vielfache von Spalten zu anderen Spalten und damit in der letzten Zeile nur Vielfache von Nullen addiert werden.

Der Eintrag $u'_{n+1,n+1}$ bleibt von Null verschieden, solange in der $(n+1)$-ten Spalte von H' nicht das Diagonalelement für eine Zeile i gefunden wird – was bedeutet, dass alle anderen Elemente dieser Zeile Null sein müssen und zwangsweise auf ein Element der letzten Spalte zurückgegriffen wird. Tritt dieser Fall ein, gilt also $h'_{i,i} = 0$. An diese Position muss aber ein Diagonalelement gesetzt werden[2]. Der

$$\text{EXTENDEDGCD}(h'_{i,i}, h'_{i,n+1})$$

wird berechnet und liefert einen Tripel

$$(r, p, q) = (h'_{i,n+1}, 0, 1) \quad \text{oder} \quad (r, p, q) = (-h'_{i,n+1}, 0, -1).$$

Die anschließend konstruierte Permutationsmatrix erzeugt eine Null an Stelle $u'_{n+1,n+1}$, da ein Vielfaches der i-ten Spalte zu 0-mal der $(n+1)$-ten Spalte addiert wird. Gleichzeitig wird an Position $u'_{n+1,i}$ ein von Null verschiedener Eintrag 1 oder -1 erzeugt.

Zeile i ist nun fertig bearbeitet. Der Eintrag $u'_{n+1,i}$ wirkt sich nicht auf weitere

[2] Es gilt dann in jedem Fall $h'_{i,n+1} \neq 0$ durch die Definition, dass bei vollem Zeilenrang jede Zeile ein Diagonalelement besitzt.

Eliminationsschritte aus, da diese nun für die Zeile $i+1$ erst ab Spalte $i+2$ durchgeführt werden. Die Einträge $u'_{n+1,i+1}, ..., u'_{n+1,n}$ sind damit noch Null.

Insgesamt bedeutet dieser Schritt, dass die letzte Spalte von H', die ja die rechte Seite des Gleichungssystems war, zur Erzeugung eines Diagonalelements gebraucht wird und somit der Rang von H' größer ist als der von A, da die Einträge aus A nicht zur Diagonalisierung ausreichten. Das Gleichungssystem hat damit keine Lösung; nur durch die Erweiterung zu A' wurde der volle Zeilenrang hergestellt und der Algorithmus akzeptierte die Eingabematrix.

Weiterhin besteht durch die Arbeitsweise des Algorithmus noch die Möglichkeit, dass in der Spalte i des aktuellen Diagonalelements der Eintrag $u'_{n+1,i}$ einen von Null verschiedenen Wert enthält und Eintrag $u'_{n+1,n+1} \neq 0$ bleibt. Dies kann nur passieren, wenn ein Eliminationsschritt auf Spalte $n+1$ erfolgt und $h'_{i,i} \neq 0$ ist. Weitere Berechnungen erfolgen aber wieder ab Zeile $i+1$ und Spalte $i+2$, sodass der Eintrag $u'_{n+1,i}$ danach keine weiteren Eliminationsschritte beeinflussen kann.

Schlussendlich können also in U' nur die ersten $rg(A') = m$ Spalten und Spalte $n+1$ in ihren $(n+1)$-ten Zeilen von Null verschiedene Einträge besitzen. Hat H' mehrere Nullspalten, kann in Zeile $n+1$ von U' in den Spalten $rg(A')+1, .., n+1$ also nur der Eintrag $u'_{n+1,n+1} \neq 0$ sein.

Man erkennt schnell, dass dieses Verfahren dem in Abbildung 5.1 beschriebenen Verfahren entspricht, obwohl in dieser Variante der Hermite-Normalform negative Einträge zugelassen sind. Gemäß Definition 3.4 hat die Hermite-Normalform entweder das Format

- H' ($rg(A') = n+1$, d.h. H' hat keine Nullspalten),
- $[H'|\vec{0}]$ ($rg(A') = n$, d.h. H' hat genau eine Nullspalte) oder
- $(H', 0)$ ($rg(A') = r < n$, d.h. H' hat mehrere Nullspalten).

Nun existieren drei zu unterscheidende Fälle:

1. ($u'_{n+1,n+1} = 0$ und $h'_{*,n+1} = \vec{0}$) oder $h'_{*,n+1} \neq \vec{0}$:
   Das Gleichungssystem hat *keine* Lösung.
2. $u'_{n+1,n+1} = \pm 1$ und $h'_{*,n+1} = \vec{0}$:
   Das Gleichungssystem hat mindestens eine ganzzahlige partielle Lösung

$$u'_{n+1,n+1} \cdot \begin{pmatrix} u'_{1,n+1} \\ \vdots \\ u'_{n,n+1} \end{pmatrix}.$$

Hat H' keine weiteren Nullspalten links neben der letzten Spalte, so hat das System nur genau diese Lösung. Wenn $r < n$ gilt, gibt es weitere $n-r$ Nullspalten. Die entsprechenden Spalten

$$\begin{pmatrix} u'_{1,rg(A')+1} \\ \vdots \\ u'_{n,rg(A')+1} \end{pmatrix}, ..., \begin{pmatrix} u'_{1,n} \\ \vdots \\ u'_{n,n} \end{pmatrix}$$

bilden eine $\mathbb{Z}$-Basis des zugehörigen homogenen Gleichungssystems und das gesamte System hat damit *unendlich viele ganzzahlige* Lösungen.

3. $u'_{n+1,n+1} \in \mathbb{Z}\backslash\{0, 1, -1\}$ und $h'_{*,n+1} = \vec{0}$:
   Das Gleichungssystem hat mindestens eine partielle, nicht ganzzahlige Lösung

$$\begin{pmatrix} \frac{u'_{1,n+1}}{u'_{n+1,n+1}} \\ \vdots \\ \frac{u'_{n,n+1}}{u'_{n+1,n+1}} \end{pmatrix}.$$

   Falls in H' weitere Nullspalten links von der $(n+1)$-ten Spalte existieren, so kann die $\mathbb{Z}$-Basis wie in Punkt 2 ermittelt werden und das gesamte Gleichungssystem hat damit unendlich viele, allerdings *nicht ganzzahlige* Lösungen.

5.2.2.2 Verfahren für den Algorithmus von Schrijver

Bei HNFSchrijverComplete wird die Transformationsmatrix U' erst nachträglich berechnet. A' wurde möglicherweise zu einer invertierbaren Matrix A_{neu} ergänzt (vgl. Abschnitt 4.1.2.3), damit U' erzeugt werden kann. Das System

$$\begin{pmatrix} 0 & 0 & 2 & -3 \\ 2 & -3 & 4 & -1 \end{pmatrix} \cdot \vec{x} = \begin{pmatrix} -6 \\ 4 \end{pmatrix}$$

liefert beispielsweise

$$H' = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}, \; U' = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ -3 & -11 & -10 & -2 & -16 \\ -1 & -6 & -6 & 0 & -9 \\ 1 & 0 & 0 & 2 & 0 \\ 1 & 2 & 2 & 1 & 3 \end{pmatrix}.$$

Die Bildung von A_{neu} führt zunächst zu einer Hermite-Normalform H_{neu} mit $n-m$ zusätzlichen Zeilen. Mit der inversen Matrix A_{neu}^{-1} und H_{neu} wird anschließend U' berechnet. Es ist jedoch nicht gewährleistet, dass bei dieser Matrixmultiplikation die Einträge $u'_{n+1,rg(A')+1}, ..., u'_{n+1,n+1}$ aussagekräftige Werte annehmen, wie es für die vorherigen Algorithmen gezeigt wurde. Insbesondere gilt allgemein nicht

$$u'_{n+1,rg(A')+1} = ... = u'_{n+1,n} = 0.$$

Auch wenn die letzten $n+1-rg(A')$ Spalten von U' eine $\mathbb{Z}$-Basis für das System $A' \cdot \vec{y} = \vec{0}$ bilden, kann für diesen Algorithmus kein Verfahren nach der in Abschnitt 5.2.1 vorgestellten Methode für das System $A \cdot \vec{x} = \vec{b}$ konstruiert werden. Für diesen Algorithmus ist demnach das in Abschnitt 5.1 vorgestellte Verfahren anzuwenden.

5.2.2.3 Verfahren für den Algorithmus von Patrick Theobald

Dieser Algorithmus erzeugt eine obere Dreiecksmatrix, bei der die ersten

$$l := n + 1 - rg(A')$$

Spalten Nullspalten sind. Folglich müssen wie bei den anderen Algorithmen entsprechende Spalten von U', insbesondere wiederum die $(n+1)$-te Zeile, untersucht werden.

Auch bei diesem Verfahren ist zu Beginn in der $(n+1)$-ten Zeile von U' nur der Eintrag $u'_{n+1,n+1}$ von Null verschieden. Der Algorithmus beginnt in der *letzten* Zeile und Spalte. Diagonalelemente werden in den letzten $rg(A')$ Spalten erzeugt und deren Zeilenindex dynamisch ermittelt. Alle Einträge links vom Diagonalelement werden annulliert.

Wie bei den vorherigen Algorithmen stellt sich nun die Frage, ob die Elemente $u'_{n+1,1}, ..., u'_{n+1,l}$ als Kriterium zum Ablesen der Lösungen akzeptiert werden können. Die Arbeitsweise des Algorithmus, insbesondere die Auswirkung der Eliminations- und Normalisierungsschritte auf die gerade erwähnten Elemente, muss dazu genauer betrachtet werden.

HNFTHEO beginnt in der letzten Zeile i und letzten Spalte $n+1$. O.B.d.A. sei i keine Nullzeile, es werden also Eliminationsschritte ausgeführt. Die Zeile $u'_{n+1,*}$ wird so lange nicht verändert, bis die Bearbeitung der Spalten n und $n+1$ erfolgt, da in den anderen Fällen nur Vielfache der ersten n Spalten und damit in Zeile i nur Nullen addiert werden. $u'_{n+1,n}$ wird nur verändert, wenn $h'_{i,n} \neq 0$ ist und es gibt zwei Fälle:

1. Ist $h'_{i,n+1} = 0$, werden die Spalten n und $n+1$ getauscht und damit $u'_{n+1,n} \neq 0$ und $u'_{n+1,n+1} = 0$ gesetzt.

2. Ist $h'_{i,n+1} \neq 0$, wird der EXTENDEDGCD (g, a, b) zweier Elemente berechnet und eine Permutationsmatrix konstruiert, die einen von Null verschiedenen Wert an Position $u'_{n+1,n}$ erzeugt:

$$-\frac{h'_{i,n+1}}{g} \cdot u'_{n+1,n} + \underbrace{\frac{h'_{i,n}}{g}}_{\neq 0} \cdot u'_{n+1,n+1}.$$

Der Algorithmus arbeitet anschließend auf Zeile $i-1$ und Spalte n weiter. Falls diese Zeile Nullen in den ersten n Spalten hat, wird sie übersprungen und mit Zeile $i-2$ fortgefahren. Die Arbeitsspalte n bleibt erhalten. Damit sei die nächste nun zu bearbeitende Zeile eine Zeile $j < i$. Auch in diesem Fall ist erst die Bearbeitung der Spalten $n-1$ und n interessant, da sonst in der letzten Zeile von U' nur Nullen addiert werden. Der Eintrag $u'_{n+1,n}$ hatte nach Bearbeitung von Zeile i einen von Null verschiedenen Wert erhalten. Bei einem nächsten Eliminationsschritt ($h'_{j,n-1} \neq 0$), der die Spalten n und $n-1$ umfasst, sind die gleichen Fälle wie oben zu betrachten: Ist $h'_{j,n} = 0$, so werden die Spalten n und $n-1$ getauscht und der Eintrag $u'_{n+1,n-1}$ erhält damit den Wert von $u'_{n+1,n}$. Im Fall $h'_{j,n} \neq 0$ erhält $u'_{n+1,n-1}$ ebenfalls einen von Null verschiedenen Wert.

Nun ist noch ein letzter Fall zu betrachten: Angenommen, die aktuelle Zeile ist p und die zu bearbeitenden Spalten sind k ($k > 1$, sonst muss nicht eliminiert werden) und $k-1$. Tritt hierbei der Fall $h'_{p,k-1} = 0$ ein, so findet kein Eliminationsschritt statt und der Eintrag $u'_{n+1,k-1}$ bleibt unverändert ($= 0$). Dadurch kann dann im weiteren Verlauf kein Eintrag $u'_{n+1,z}$, $1 \leq z \leq k-1$, einen von Null verschiedenen Wert erhalten.

Insgesamt wird *maximal* $rg(A')$-mal mit Hilfe eines Eliminationsschritts ein Diagonalelement in H' erzeugt, zuletzt also *maximal* in der Spalte $l+1$, sodass *maximal* der Eintrag $u'_{n+1,l}$ einen von Null verschiedenen Wert erhält[3]: Die Einträge der letzten Zeile von U' mit Spaltenindex $< l$ werden dabei nicht verändert. Für die Einträge $u'_{n+1,1}, ..., u'_{n+1,l-1}$ gilt also

$$u'_{n+1,1} = ... = u'_{n+1,l-1} = 0$$

und es können gleiche Kriterien wie bei den anderen Algorithmen angewandt werden:

Gemäß Definition 3.6 hat die Hermite-Normalform im Fall $rg(A') = n$ das Format $[\vec{0}|H']$, im Fall $rg(A') < n$ die Form $(0, H')$, d.h. die ersten $n+1-rg(A')$ Spalten sind

[3] Zur Erinnerung: $l = n + 1 - rg(A')$.

Nullspalten, und im Fall $rg(A') = n + 1$ das Format H'. Wieder existieren mehrere Fälle:

1. $u'_{n+1,1} = \pm 1$ und $h'_{*,1} = \vec{0}$:
   Das Gleichungssystem hat genau eine ganzzahlige Lösung, nämlich den Vektor

$$u'_{n+1,1} \cdot \begin{pmatrix} u'_{1,1} \\ \vdots \\ u'_{n,1} \end{pmatrix}.$$

2. $u'_{n+1,1} \in \mathbb{Z}\backslash\{0, 1, -1\}$ und $h'_{*,1} = \vec{0}$:
   Das Gleichungssystem hat genau eine nicht ganzzahlige Lösung

$$\begin{pmatrix} \frac{u'_{1,1}}{u'_{n+1,1}} \\ \vdots \\ \frac{u'_{n,1}}{u'_{n+1,1}} \end{pmatrix}.$$

3. $(u'_{n+1,1} = u'_{n+1,n+1-rg(A')} = 0$ und $h'_{*,1} = ... = h'_{*,n+1-rg(A')} = \vec{0})$ oder $h'_{*,1} \neq \vec{0}$:
   Das Gleichungssystem hat keine Lösung.

4. $u'_{n+1,1} = 0$ und $h'_{*,1} = ... = h'_{*,n+1-rg(A')} = \vec{0}$ und $u'_{n+1,n+1-rg(A')} = \pm 1$:
   Das System hat unendlich viele ganzzahlige Lösungen. Die allgemeine Lösung ist:

$$\vec{x} = u'_{n+1,n+1-rg(A')} \cdot \begin{pmatrix} u'_{1,n+1-rg(A')} \\ \vdots \\ u'_{n,n+1-rg(A')} \end{pmatrix} + \sum_{i=1}^{n-rg(A')} \lambda_i \begin{pmatrix} u'_{1,i} \\ \vdots \\ u'_{n,i} \end{pmatrix}, \quad \lambda_i \in \mathbb{Z}.$$

5. $u'_{n+1,1} = 0$ und $h'_{*,1} = ... = h'_{*,n+1-rg(A')} = \vec{0}$ und $u'_{n+1,n+1-rg(A')} \in \mathbb{Z}\backslash\{0, 1, -1\}$:
   Das System hat unendlich viele, nicht ganzzahlige Lösungen.

5.2.2.4 Verfahren für den Algorithmus von Gerold Jäger

Bei ROWHNF wird zunächst A' erzeugt und dem Algorithmus dann $(A')^T$ übergeben. Die Ergebnisse liegen in transponierter Form zu COLUMNHNF vor, d.h. bei der Überprüfung der Fälle müssen Zeilen- und Spaltenindizes vertauscht werden.

Übergibt man Algorithmus 4.8 stattdessen die Matrix A', so wird nicht das System $A' \cdot \vec{y} = \vec{0}$ gelöst, sondern das System $(A')^T \cdot \vec{y}' = \vec{0}$, also

$$[A| - \vec{b}]^T \cdot \vec{y}' = \vec{0},$$

damit

$$\left[\begin{array}{c|c} (A_{1..m-1,*})^T & (A_{m,*})^T \\ \hline -b_1, ..., -b_{m-1} & -b_m \end{array}\right] \cdot \vec{y}' = \vec{0},$$

und somit das System

$$\left(\begin{array}{ccc} a_{1,1} & \dots & a_{m-1,1} \\ \vdots & \ddots & \vdots \\ a_{1,n} & \dots & a_{m-1,n} \\ \hline -b_1 & \dots & -b_{m-1} \end{array}\right) \cdot \vec{x}' = \left(\begin{array}{c} -a_{m,1} \\ \vdots \\ -a_{m,n} \\ \hline b_m \end{array}\right).$$

Im Fall $\vec{b} = \vec{0}$ kann die letzte Zeile des Systems gestrichen werden; ROWHNF berechnet folglich eine $\mathbb{Z}$-Basis des $(n \times m)$-Systems $A^T \cdot \vec{x} = \vec{0}$.

5.2.3 Anwendungsbeispiele

Die Verfahren aus den beiden letzten Abschnitten werden nun anhand von Beispielen veranschaulicht. Für Fall 1 (genau eine Nullspalte) wird bei allen Berechnungen der Hermite-Normalform und der zugehörigen Transformationsmatrix Algorithmus 4.6 (COLUMNHNF) verwendet, für Fall 2 (mehrere Nullspalten) Algorithmus 4.5 (HNFTHEO) und für Fall 3 (keine Nullspalten) Algorithmus 4.1 (HNFNW).

Fall 1.1:
Für Fall 1.1 sei das Gleichungssystem $A \cdot \vec{x} = \vec{b}$ als

$$\begin{pmatrix} -2 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{pmatrix} \cdot \vec{x} = \begin{pmatrix} 3 \\ -1 \\ 2 \end{pmatrix}$$

gegeben. Die Matrix $A' := [A| - \vec{b}]$ wird erzeugt. Die resultierenden Matrizen H' und U' sind

$$H' = \begin{pmatrix} 1 & 0 & 0 & \mathbf{0} \\ 0 & 1 & 0 & \mathbf{0} \\ 3 & 3 & 4 & \mathbf{0} \end{pmatrix}, \; U' = \begin{pmatrix} -1 & -2 & -1 & \mathbf{1} \\ -2 & -4 & -2 & \mathbf{1} \\ -2 & -3 & -3 & \mathbf{1} \\ -1 & -1 & -1 & \mathbf{0} \end{pmatrix}.$$

Aus H' ist ablesbar, dass $rg(A') = n = 3$ gilt. Da aber $u'_{n+1,n+1} = u'_{4,4} = 0$ gilt, ist $rg(A) = r < n$. Das Gleichungssystem $A \cdot \vec{x} = \vec{b}$ hat damit *keine* Lösung. Die letzte Spalte von U' ohne das Element der letzten Zeile, also der Vektor $(\; 1 \quad 1 \quad 1 \;)^T$, bildet eine Lösung des homogenen Gleichungssystems $A \cdot \vec{x} = \vec{0}$.

Fälle 1.2/1.3:
Nun werden die zwei Fälle betrachtet, in denen eine partielle Lösung existiert. Es gilt noch zu unterscheiden, ob sie ganzzahlig ist oder nicht. Das zugehörige homogene LGS $A \cdot \vec{x} = \vec{0}$ besitzt nur die triviale Lösung $\vec{0}$. In den Fällen 1.2 und 1.3 sei

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

die rechten Seiten $\vec{b}$ seien

$$\vec{b}_{1.2} = \begin{pmatrix} -16 \\ -6 \\ 6 \\ 8 \end{pmatrix}, \ \vec{b}_{1.3} = \begin{pmatrix} -16 \\ -6 \\ 7 \\ 8 \end{pmatrix}.$$

Wieder wird zunächst $A' := [A|-\vec{b}]$ erzeugt. In Fall 1.2 sind die resultierenden Matrizen

$$H'_{1.2} = \begin{pmatrix} 1 & 0 & 0 & 0 & \mathbf{0} \\ 0 & 1 & 0 & 0 & \mathbf{0} \\ 1 & 1 & 2 & 0 & \mathbf{0} \\ 1 & 0 & 0 & 2 & \mathbf{0} \end{pmatrix}, \ U'_{1.2} = \begin{pmatrix} -2 & 2 & 3 & -5 & \mathbf{2} \\ 2 & 0 & 0 & 3 & \mathbf{4} \\ 1 & 1 & 2 & 0 & \mathbf{6} \\ 1 & 0 & 0 & 2 & \mathbf{8} \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{pmatrix}.$$

Hier ist $u'_{n+1,n+1} = u'_{5,5} = 1$, d.h. es gibt *genau eine ganzzahlige* Lösung $\vec{x}$:

$$\vec{x} = \begin{pmatrix} 2 \\ 4 \\ 6 \\ 8 \end{pmatrix}.$$

In Fall 1.3 haben die Matrizen die Gestalt

$$H'_{1.3} = \begin{pmatrix} 1 & 0 & 0 & 0 & \mathbf{0} \\ 0 & 1 & 0 & 0 & \mathbf{0} \\ 0 & 0 & 1 & 0 & \mathbf{0} \\ 1 & 0 & 0 & 2 & \mathbf{0} \end{pmatrix}, \ U'_{1.3} = \begin{pmatrix} 0 & 4 & -2 & -5 & \mathbf{7} \\ 6 & 4 & -4 & 3 & \mathbf{8} \\ 7 & 7 & -6 & 0 & \mathbf{14} \\ 9 & 8 & -8 & 2 & \mathbf{16} \\ 1 & 1 & -1 & 0 & \mathbf{2} \end{pmatrix}.$$

Es gilt $u'_{n+1,n+1} = u'_{5,5} = 2$. Das LGS hat *keine ganzzahlige* Lösung, denn

$$A \cdot \vec{x} = \mathbf{2} \cdot \vec{b}_{1.3},$$

also

$$A \cdot \mathbf{\frac{1}{2}} \cdot \begin{pmatrix} 7 \\ 8 \\ 14 \\ 16 \end{pmatrix} = A \cdot \begin{pmatrix} \frac{7}{2} \\ \mathbf{4} \\ \mathbf{7} \\ \mathbf{8} \end{pmatrix} = \begin{pmatrix} -16 \\ -6 \\ 7 \\ 8 \end{pmatrix}.$$

Nun wird Fall 2 mit Hilfe von Algorithmus 4.5 veranschaulicht. Das Verfahren ist hier ähnlich, wobei dieser Algorithmus eine obere Dreiecksmatrix erzeugt und die Nullspalten links in der Matrix stehen. Die Kriterien für die Existenz einer Lösung basieren dadurch auf anderen Matrixeinträgen von U'. Die Ausgangsmatrix A sei für die Fälle 2.1 und 2.2

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

Es gilt $r := rg(A) = 2$, damit hat ein zugehöriges homogenes Gleichungssystem eine Lösung mit $n - r = 2$ Elementen in der $\mathbb{Z}$-Basis.

Fall 2.1:
Mittels der Hermite-Normalform werden nun die Lösungen für das Gleichungssystem $A \cdot \vec{x} = \vec{b}$ bei gegebener rechter Seite

$$\vec{b}_{2.1} = \begin{pmatrix} -6 \\ 4 \\ 11 \\ -8 \end{pmatrix}$$

ermittelt. Zuerst wird A' gebildet. Algorithmus 4.5 liefert folgende Hermite-Normalform H' und Transformationsmatrix U':

$$H'_{2.1} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & 1 & 0 & 0 \\ \mathbf{0} & \mathbf{0} & 0 & 0 & -1 \\ \mathbf{0} & \mathbf{0} & 0 & 1 & 0 \\ \mathbf{0} & \mathbf{0} & 0 & 0 & 2 \end{pmatrix}, \; U'_{2.1} = \begin{pmatrix} \mathbf{-3} & \mathbf{-10} & 105 & 54 & -9 \\ \mathbf{-2} & \mathbf{-10} & 105 & 54 & -9 \\ \mathbf{0} & \mathbf{-3} & 35 & 18 & -3 \\ \mathbf{0} & \mathbf{-2} & 27 & 14 & -2 \\ \mathbf{0} & \mathbf{0} & 2 & 1 & 0 \end{pmatrix}.$$

Gemäß Definition 3.6 sind die ersten $n+1-rg(A')$ Spalten in der Hermite-Normalform Nullspalten, in diesem Fall $5 - rg(A') = 2$ Spalten von H'. Hier sieht man sofort, dass

$$r = rg(A) = 2 < rg(A') = 3$$

gilt und das inhomogene Gleichungssystem damit *keine* Lösung hat.

Die ersten $n - r$ Spalten von U' bilden (ohne die letzte Zeile) eine $\mathbb{Z}$-Basis des zugehörigen homogenen Gleichungssystems. Man erkennt diese Spalten daran, dass für folgende Einträge der letzten Zeile von U' gilt:

$$u'_{n+1,1} = u'_{n+1,2} = ... = u'_{n+1,n-r} = 0.$$

Fall 2.2:
Besitzt $A \cdot \vec{x} = \vec{b}$ mindestens eine Lösung, so hat die zugehörige Hermite-Normalform H' der erweiterten Matrix A' genau $n - r + 1$ Nullspalten, denn genau dann gilt $rg(A) = rg([A|\vec{b}])$. Man betrachte hierzu das Gleichungssystem mit rechter Seite

$$\vec{b}_{2.2} = \begin{pmatrix} -6 \\ 4 \\ 12 \\ -8 \end{pmatrix}.$$

Die ermittelten Ergebnismatrizen sind

$$H'_{2.2} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & -1 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & -1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 2 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 2 \end{pmatrix}, \; U'_{2.2} = \begin{pmatrix} \mathbf{-3} & \mathbf{-10} & \mathbf{54} & 3 & -9 \\ \mathbf{-2} & \mathbf{-10} & \mathbf{54} & 3 & -9 \\ \mathbf{0} & \mathbf{-3} & \mathbf{18} & 1 & -3 \\ \mathbf{0} & \mathbf{-2} & \mathbf{14} & 1 & -2 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 & 0 \end{pmatrix}.$$

A hat den Rang 2, damit bilden die ersten $n-r = 2$ Nullspalten von U' (wieder ohne die letzte Zeile) eine $\mathbb{Z}$-Basis des homogenen Gleichungssystems. Die 3. Nullspalte liefert die Lösung des inhomogenen Gleichungssystems. Da $u'_{5,3} = 1$ gilt, bildet der Vektor

$$\begin{pmatrix} 54 \\ 54 \\ 18 \\ 14 \end{pmatrix}$$

die *ganzzahlige partielle* Lösung. Alle ganzzahligen Lösungen für das System $A \cdot \vec{x} = \vec{b}_{2.2}$ sind damit gegeben als

$$\vec{x} = \begin{pmatrix} 54 \\ 54 \\ 18 \\ 14 \end{pmatrix} + \lambda_1 \cdot \begin{pmatrix} -3 \\ -2 \\ 0 \\ 0 \end{pmatrix} + \lambda_2 \cdot \begin{pmatrix} -10 \\ -10 \\ -3 \\ -2 \end{pmatrix}, \quad \lambda_1, \lambda_2 \in \mathbb{Z}.$$

Fall 2.3:
Die Lösungen für das Gleichungssystem

$$\begin{pmatrix} -2 & 1 & 1 \\ 3 & -3 & 0 \\ 3 & 0 & -3 \end{pmatrix} \cdot \vec{x} = \begin{pmatrix} 1 \\ -1 \\ -2 \end{pmatrix}$$

werden ermittelt. Man erweitert A zu A',

$$A' = \begin{pmatrix} -2 & 1 & 1 & -1 \\ 3 & -3 & 0 & 1 \\ 3 & 0 & -3 & 2 \end{pmatrix},$$

und berechnet die HNF und Transformationsmatrix mittels Algorithmus 4.5:

$$H'_{2.3} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & -1 & -1 \\ \mathbf{0} & \mathbf{0} & 3 & 2 \\ \mathbf{0} & \mathbf{0} & 0 & 1 \end{pmatrix},\ U'_{2.3} = \begin{pmatrix} \mathbf{-1} & \mathbf{-1} & 0 & 0 \\ \mathbf{-1} & \mathbf{0} & 0 & 0 \\ \mathbf{-1} & \mathbf{1} & 2 & 1 \\ \mathbf{0} & \mathbf{3} & 3 & 2 \end{pmatrix}.$$

H' hat zwei Nullspalten, somit müssen die ersten zwei Spalten von U' als Lösung in Betracht gezogen werden. Die erste Spalte stellt die $\mathbb{Z}$-Basis des homogenen Gleichungssystems dar. Anschließend wird Spalte 2 untersucht. Es gilt

$$u'_{n+1,2} = u'_{4,2} = \mathbf{3},$$

damit stellt diese Spalte eine Lösung des inhomogenen Gleichungssystems dar, allerdings *keine ganzzahlige*. Setzt man sie für $\vec{y}$ in das System $A' \cdot \vec{y} = \vec{0}$ ein, so erhält man

$$A \cdot \vec{x} = \mathbf{3} \cdot \vec{b},$$

also

$$A \cdot \frac{\mathbf{1}}{\mathbf{3}} \cdot \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} = A \cdot \begin{pmatrix} -\frac{1}{3} \\ 0 \\ \frac{1}{3} \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ -2 \end{pmatrix}.$$

Fall 3:
Das Gleichungssystem

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} \cdot \vec{x} = \begin{pmatrix} 5 \\ 5 \end{pmatrix}$$

wird mittels Algorithmus 4.1 auf Lösungen untersucht. A wird zu A' erweitert, an-

schließend H' und U' bestimmt:

$$H'_3 = \begin{pmatrix} 1 & 0 \\ -3 & 5 \end{pmatrix},\ U'_3 = \begin{pmatrix} -4 & 5 \\ -1 & 1 \end{pmatrix}.$$

Die letzte Spalte von H' ist keine Nullspalte, damit ist $rg(A') = n + 1$ und das Gleichungssystem besitzt *keine* Lösung.

6 Bereitstellung der Algorithmen in Mathematica

Mathematica[1] ist eines der leistungsfähigsten weltweit verfügbaren Computeralgebrasysteme und umfasst umfangreiche Bibliotheken für eine Vielzahl mathematischer Teilgebiete. Das System wird nicht nur von Mathematikern genutzt, sondern ist beispielsweise auch in der Sozialwissenschaft, Physik oder Biologie ein hilfreiches Werkzeug.

Im Bereich der linearen Algebra hat Mathematica 4.2, das für diese Arbeit verwendet wurde, jedoch noch einige Lücken. Es gibt zum Beispiel keine Funktion, die direkt den Rang einer Matrix berechnet. Für diese Zwecke müssen Umwege über andere Funktionen genommen werden, nicht zuletzt durch die Berechnung der Hermite-Normalform.

Die in Kapitel 4 diskutierten Algorithmen wurden für diese Arbeit in Mathematica-Code implementiert und können als Paket in das Computeralgebrasystem geladen werden. Abschnitt 6.1 beschreibt den Aufbau des Pakets und geht auf Besonderheiten ein. Abschnitt 6.2 enthält Informationen zur Nutzung des Pakets in der Mathematica-Umgebung.

6.1 Das Paket „HNFAlgorithms“

Der Vorteil von Mathematica gegenüber einer normalen Programmiersprache ist das Vorhandensein geeigneter Datenstrukturen (z.B. für Matrizen) und bereits integrierter Funktionen wie die Berechnung der inversen Matrix. Aus diesem Grund wurde Mathematica für diese Arbeit als Plattform für die Algorithmen ausgewählt. Das Paket `HNFAlgorithms` enthält die in Kapitel 4 vorgestellten Algorithmen in Form von Modulen. Wird das Paket in Mathematica geladen, so kann über so genannte *Notebooks* ein Modul mit geeigneten Parametern aufgerufen und die jeweilige Berechnung der Hermite-Normalform gestartet werden.

Die Programmcodes der einzelnen HNF-Algorithmen entsprechen den in dieser Arbeit abgebildeten Pseudocodes. Lediglich einige Anpassungen an die von Mathematica geforderte Syntax mussten vorgenommen werden. Der Algorithmus EXTENDEDGCD

[1] Informationen: http://www.wolfram.com/products/mathematica/history.html, Abruf: 19.12.2007.

wird von Mathematica als Funktion bereitgestellt und musste nicht implementiert werden. Tabelle 6.1 enthält die Zuordnung der Algorithmen zu den Mathematica-Modulen:

Algorithmus	**Modulname**
HNFNW (Algorithmus 4.1)	`HNFNW`
HNFSCHRIJVER (Algorithmus 4.2)	`HNFSchrijver`
HNFSCHRIJVERCOMPLETE (Algorithmus 4.4)	`HNFSchrijverComplete`
HNFTHEO (Algorithmus 4.5)	`HNFTHEO`
COLUMNHNF (Algorithmus 4.6)	`ColumnHNF`
ROWGCD (Algorithmus 4.7)	`rowGCD`
ROWHNF (Algorithmus 4.8)	`RowHNF`
COLUMNGCD (Algorithmus 4.9)	`columnGCD`

Tabelle 6.1: *Zuordnung der Algorithmen zu den Mathematica-Modulen*

Der einzige erwähnenswerte Unterschied liegt bei den Algorithmen von Schrijver: Der Hilfsalgorithmus ELIMINATE ist kein eigenständiges Modul, sondern in den Code von `HNFSchrijver` integriert. Lediglich zur besseren Verständlichkeit wurde ELIMINATE in dieser Arbeit als eigenständiger Algorithmus abgebildet.

Wird das Modul `HNFSchrijverComplete` aufgerufen und die inverse Matrix $(A^{neu})^{-1}$ berechnet, so wird in Mathematica dazu die bereits integrierte Funktion `Inverse[]` verwendet[2].

6.2 Benutzung der HNF-Algorithmen

Der einfachste Weg, das Package `HNFAlgorithms` zu verwenden, ist innerhalb von *Notebooks*. Durch Eingabe von

```
<< "<Laufwerk>:\<Pfad zum Package>\HNFAlgorithms.m",
```

also zum Beispiel

```
<< "E:\Mathematica\HNFAlgorithms.m"
```

in ein Notebook wird ein Paket geladen und die darin enthaltenen Module stehen von nun an zur Verfügung. Informationen über ein Modul können durch Eingabe von

```
<ModulName>::usage
```

[2]Informationen zu den von Mathematica bereitgestellten Funktionen liefert dessen Dokumentation ([Wol03]).

gewonnen werden. Die HNF-Module im Paket `HNFAlgorithms` erwarten als Eingabe genau zwei Parameter und werden wie folgt aufgerufen:

```
<ModulName>[<Parameter1>,<Parameter2>]
```

`<Parameter1>` muss eine Matrix sein. Ist diese nicht ganzzahlig oder bei den Modulen `HNFNW` und `HNFSchrijver` ohne vollen Zeilenrang, so melden die Algorithmen einen Eingabefehler vom Typ `HNFInputError`. In Mathematica werden Matrizen durch gschachtelte geschweifte Klammern definiert, die Matrix

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

also in der Form

```
{{1,2},{3,4}}.
```

`<Parameter2>` bekommt entweder den Wert `True` oder `False`. Wird `False` übergeben, so berechnet das Modul die Hermite-Normalform H und zugehörige Transformationsmatrix U und gibt die Ergebnisse zurück. Hat der 2. Parameter den Wert `True`, so wird neben den Matrizen H und U jeder wichtige Berechnungsschritt des Algorithmus in das Notebook ausgegeben. Insbesondere werden bei jedem Schleifendurchlauf oder Bearbeitungsschritt dem Benutzer die aktuellen Werte von H und U mitgeteilt. Auf diese Weise kann das Konstruktionsverfahren der Hermite-Normalform bequem nachverfolgt werden. Bei den Modulen `ColumnHNF` und `RowHNF` wird in beiden Fällen noch eine Liste der Zeilen bzw. Spalten, in denen die Pseudodiagonalelemente stehen, ausgegeben.

Eine Beispieleingabe für obige Matrix mit der Ausgabe detaillierter Schritte durch Modul `HNFTHEO` ist zum Beispiel

```
HNFTHEO[{{1,2},{3,4}},True]
```

und liefert die in Abbildung 6.1 abgebildete Ausgabe.

Die Beispiele aus den Kapiteln 4 und 5 wurden bereits in zwei Notebooks integriert, die sich auf der beiliegenden CD befinden. Das Notebook `AlgorithmTest.nb` enthält die Berechnungen der Hermite-Normalformen und Transformationsmatrizen mit jedem der HNF-Algorithmen in zwei Versionen: Einmal ohne Detailansicht der Berechnungsschritte (Parameter 2 hat den Wert `False`) und einmal in der ausführlichen Version (`True`). Die Beispiele aus Kapitel 5 befinden sich im Notebook `Solutions.nb`. Hier

```
In[2]:= HNFTHEO[{{1, 2}, {3, 4}}, True]

        Input Matrix: (1 2)
                      (3 4)

        --- Initialisation: ----------------------------------

            H = (1 2), U = (1 0), i = 2, j = 2
                (3 4)      (0 1)

        --- STEP 1: Working on row i = 2 and column j = 2. ---

            STEP 3: Elimination -------------------------------

                Working on elements v(1) = 3 and v(2) = 4.

            --- STEP 3.3 --

                Calculating ExtendedGCD for v(1) and v(2).

                --> (g,a,b) = (1,-1,1)

                Permutation Matrix T = (-4 -1)
                                       (3  1)

                Matrices after applying ExtendedGCD: H = (2 1), U = (-4 -1)
                                                         (0 1)      (3  1)

            STEP 4: Normalisation. ----------------------------

        --- STEP 5: Continue with i-1 and j-1 -----------------

        --- STEP 1: Working on row i = 1 and column j = 1. ---

            STEP 3: Elimination -------------------------------

            STEP 4: Normalisation. ----------------------------

            -- Working on column 1 = 2 ---

                Calculated q = 0

                Normalisation result: H = (2 1), U = (-4 -1)
                                          (0 1)      (3  1)

        --- STEP 5: Continue with i-1 and j-1 -----------------


        -------------------------------------------------------

        Result: HNF = (2 1), Transformation Matrix = (-4 -1)
                      (0 1)                          (3  1)

Out[2]= {{{2, 1}, {0, 1}}, {{-4, -1}, {3, 1}}}
```

Abbildung 6.1: *Ausgabe für das Modul HNFTHEO mit detaillierten Schritten.*

wurden ebenfalls für jeden der sieben Fälle die entsprechenden Eingaben konstruiert.

Anmerkung:
Es ist zu erwarten, dass Mathematica bei Eingabe von sehr großen Matrizen den Anforderungen nicht mehr gewachsen ist und während der Berechnung abstürzt. Sollen Matrizen mit mehreren Tausend Zeilen und Spalten in Hermite-Normalform transformiert werden, ist zum einen die Wahl eines effizienteren Algorithmus und eine Implementation in einer geeigneten Programmiersprache sinnvoll. Für den Zweck dieser Arbeit, nämlich den Vergleich der Arbeitsweise der Algorithmen und deren Verfahren zur Berechnung ganzzahliger Lösungen von Gleichungssystemen, reicht das Computeralgebrasystem jedoch aus.

7 Zusammenfassung und Ausblick

In dieser Arbeit wurden insgesamt fünf verwandte, aber variante Definitionen der Hermite-Normalform vorgestellt, angefangen bei der ursprünglichen Version von Charles Hermite. Anschließend wurden vier Basisalgorithmen zur Berechnung der Hermite-Normalform ganzzahliger Matrizen im Detail und anhand von vollständigen Beispielen präsentiert. Hierbei wurde vor allem Wert darauf gelegt, linear-algebraische Grundlagen wie beispielsweise die Verwendung von Permutationsmatrizen für unimodulare Spaltenoperationen verständlich zu vermitteln.

Die Bereitstellung der Algorithmen im Computeralgebrasystem Mathematica ermöglicht die automatisierte Veranschaulichung aller in dieser Arbeit diskutierten Beispiele und bietet ausreichend Gelegenheit für weitere Tests.

Zuletzt wurden Verfahren beschrieben, die die Hermite-Normalform und deren zugehörige Transformationsmatrix zur Lösung linearer inhomogener Gleichungssysteme heranziehen. Die Verfahren sind abhängig von der jeweils zugrunde liegenden Definition der HNF und den verwendeten Algorithmen, erzeugen aber bei Eingabe einer Matrix ohne vollen Spaltenrang jeweils eine korrekte $\mathbb{Z}$-Basis. Da diese nicht eindeutig ist, können die Algorithmen durchaus verschiedene Basen berechnen.

Insbesondere ist diese Gewinnung einer $\mathbb{Z}$-Basis direkt aus der Transformationsmatrix die Grundlage für weitere Verfahren, die sich mit dem Lösen linearer Gleichungssysteme beschäftigen: Aus einer ganzzahligen Basis eines linearen homogenen Gleichungssystems kann durch einige weitere Schritte ein nichtnegatives, ganzzahliges und eindeutiges Erzeugendensystem desselben Systems gewonnen werden (vgl. [Pas86, Han97, Krü87]). Diese weiterführenden Verfahren sind auf verschiedenste Problemstellungen anwendbar.

Diese Arbeit insbesondere entstand aus der Motivation heraus, alternative Techniken statt der in Fachkreisen altbekannten Verfahren (vgl. [Mar82, Col91]) zur Berechnung von Stellen- und Transitionsinvarianten in Petri-Netzen[1] zu untersuchen. Die Anwendbarkeit dieser und weiterer Verfahren in der Petri-Netz-Theorie bieten ausreichend Potenzial für weitere Studien im Bereich der Berechnung ganzzahliger, nichtnegativer Lösungen linearer (in-)homogener Gleichungssysteme.

[1] Informationen zu Petri-Netzen findet man z.B. unter [Bau96, Sta90].

Anhang: Inhalt der beiliegenden CD

Auf der beiliegenden CD befinden sich neben dieser Arbeit als PDF-Dokument im Ordner `Mathematica` folgende Komponenten:

- `HNFAlgorithms.m`: Mathematica-Package, das sämtliche in dieser Arbeit aufgeführten HNF-Algorithmen enthält.
- `AlgorithmTest.nb`: Mathematica-Notebook, das die in Kapitel 4 vorgerechneten Beispiele beinhaltet.
- `Solutions.nb`: Mathematica-Notebook, das die in Kapitel 5 vorgerechneten Beispiele enthält.

Literaturverzeichnis

[Bau96] Baumgarten, Bernd: *Petri-Netze. Grundlagen und Anwendungen.* Spektrum Akademischer Verlag, Heidelberg, Berlin, Oxford, 1996.

[Col91] Colom, J.M.; Silva, M.: *Convex Geometry and Semiflows in P/T Nets. A Comparative Study of Algorithms for Computation of Minimal P-Semiflows.* Lecture Notes in Computer Science; Advances in Petri Nets 1990, 483:79–112, 1991.

[Han97] Hanzálek, Zdeněk: *Parallel Algorithms for Distributed Control - A Petri Net Based Approach.* Dissertation, CTU Prague and UPS Toulouse, 1997.

[Her51] Hermite, Charles: *Sur l'introduction des variables continues dans la théorie des nombres.* J. Reine Angew. Math., 41:191–216, 1851.

[Jäg01] Jäger, Gerold: *Algorithmen zur Berechnung der Smith-Normalform und deren Implementation auf Parallelrechnern.* Vorlesungen aus dem Fachbereich Mathematik der Universität Essen, Heft 29, Essen: Universität Essen, 2001.

[Kan79] Kannan, R.; Bachem, A.: *Polynomial Algorithms for Computing the Smith and Hermite Normal Forms of an Integer Matrix.* SIAM J. Comput., 8(4):499–507, 1979.

[Kna01] Knauer, Ulrich: *Diskrete Strukturen - kurz gefasst.* Spektrum Akademischer Verlag, Heidelberg, Berlin, 2001.

[Krü87] Krückeberg, Fritz; Jaxy, Michael: *Mathematical methods for calculating invariants in Petri nets.* Lecture Notes in Computer Science; Advances in Petri Nets 1987, 266:104–131, 1987.

[Mar82] Martínez, J.; Silva, M.: *A simple and fast algorithm to obtain all invariants of a generalized Petri net.* In: Girault, C.; Reisig, W. (Herausgeber): *Application and theory of Petri Nets*, Seiten 301–310. Springer, Berlin, 1982.

[Mic01] Micciancio, D.; Warinschi, B.: *A linear space algorithm for computing the hermite normal form.* In: *ISSAC '01: Proceedings of the 2001 international symposium on Symbolic and algebraic computation*, Seiten 231–236, New York, NY, USA, 2001. ACM.

[Mnu07] Mnuk, Michal: *Lineare Gleichungssysteme.* `http://www14.in.tum.de/~mnuk/teaching/MAT1/Skriptum/K2node16.html`, Abruf: 08.09.2007.

[Nem88] Nemhauser, George L.; Wolsey, Laurence A.: *Integer and combinatorial optimization.* Wiley-Interscience, New York, NY, USA, 1988.

[Pas86] Pascoletti, K.H.: *Diophantische Systeme und Lösungsmethoden zur Bestimmung aller Invarianten in Petri-Netzen.* Berichte der GMD Nr. 160, Bonn, 1986.

[Sch99] Schrijver, Alexander: *Theory of linear and integer programming.* Wiley-Interscience, Chichester [u.a.], 1999.

[Sta90] Starke, Peter H.: *Analyse von Petri-Netz-Modellen.* Teubner, Stuttgart, 1990.

[Sto96] Storjohann, A.; Labahn, G.: *Asymptotically fast computation of Hermite normal forms of integer matrices.* In: *ISSAC '96: Proceedings of the 1996 international symposium on Symbolic and algebraic computation*, Seiten 259–266, New York, NY, USA, 1996. ACM.

[The00] Theobald, Patrick: *Ein Framework zur Berechnung der Hermite-Normalform von großen, dünnbesetzten, ganzzahligen Matrizen.* Dissertation, Technische Universität Darmstadt, Fachbereich Informatik, 2000.

[UT07] Universität Tübingen, Institut für Informatik: *Erweiterter Euklidischer Algorithmus für ganze Zahlen.* Materialien zu Kryptologie und Datensicherheit, Wintersemester 2006/2007. `http://www-dm.informatik.uni-tuebingen.de/lehre/kryptoVL/ws0607/EGGT.pdf`, Abruf: 26.11.2007.

[VZG99] Von Zur Gathen, J.; Gerhard, J.: *Modern Computer Algebra.* Cambridge University Press, Cambridge [u.a.], 1999.

[Wei07a] Weisstein, Eric W.: *Gauss-Jordan Elimination.* From MathWorld – A Wolfram Web Resource. `http://mathworld.wolfram.com/Gauss-JordanElimination.html`, Abruf: 12.12.2007.

[Wei07b] Weisstein, Eric W.: *Gaussian Elimination*. From MathWorld – A Wolfram Web Resource. `http://mathworld.wolfram.com/GaussianElimination.html`, Abruf: 20.11.2007.

[Wol03] Wolfram, Stephen: *The Mathematica Book*. Wolfram Media, Inc., 2003.