# Praktische Mathematik - Symbolisches Rechnen

Vorlesung im
Sommersemester 2012
TU-Kaiserslautern

gehalten von
C. Fieker
Version vom 12. Juli 2012

# Inhaltsverzeichnis


Kapitel 0. Einführung 1

Kapitel 1. Arithmetik 3
1. Addition und Multiplikation von Zahlen und Polynomen 3
2. Multiplikation 5
3. Komplexität von Algorithmen 7
4. Modulare Darstellung 9
5. Fouriertransformation 11

Kapitel 2. Division und ggT 15
1. Division 15
2. Der Euklidische Algorithmus 17
3. ggT in $\mathbb{Q}[x]$, erster Versuch 19
4. Modulare Algorithmen 21

Kapitel 3. Gleichungssysteme 27
1. Moduln 27
2. Hermite Form 28
3. $p$-adische Approximation 31
4. Gitter 32
5. Der LLL Algorithmus 34
6. Gröbnerbasen 37
7. Freie $R$-Moduln 40
8. Buchbergers Algorithmus 42
9. Nicht lineare Gleichungen 45

Kapitel 4. Faktorisierung und Irreduzibilität 49
1. Quadratfreie Faktorisierung 49
2. Faktorisierung über $\mathbb{F}_q$ 51
3. Das Henselsche Lemma 56
4. van Hoeij: Faktorisierung mit LLL 59
5. Primzahltest 60
5.1. Pseudoprimzahlen 61
5.2. Miller-Rabin 63
5.3. Faktorisieren in $\mathbb{Z}$ 66
5.4. Quadratisches Sieb 69

Kapitel 5. Resultanten und Faktorisierung 73
1. Der Satz vom Primitiven Element 77

Anhang A. Übungszettel 79

KAPITEL 0

# Einführung

Praktische Mathematik - Symbolisches Rechnen, was ist das eigentlich? Eine Antwort wäre: Numerik in der reinen Mathematik - aber das ist falsch. Sehen wir uns mal an wo es herkommt: Um die Mathematik weiter zu entwickeln und neue Sätze zu finden und zu beweisen, braucht man Ideen - oft ist es viel schwieriger einen Satz zu forumlieren als ihn dan zu beweisen. Daher haben Mathematiker immer Beispiele gerechnet, traditionel von Hand, mit Papier und Bleistift, oder, seit es (preiswerte) Computer gibt, auch mit Hilfe von Computern. In der angewandten Mathematik wird dies nun seit vielen Jahren systematisch betrieben, sog. numerische Methoden werden benutzt um den Klimawandel zu modellieren, das Wetter vorherzusagen oder um Autos zu entwickeln. Zentral in der Numerik ist es, dass nichts exakt berechnet werden kann, hier geht es um *Näherungslösungen* (oft weil man beweisen kann, dass eine exakte Lösung nicht aufgeschrieben werden kann, aber auch oft, weil das Problem (Wetter?) schon in-exakt ist).

Auf der anderen Seite, in der *reinen* Mathematik, speziell in der Algebra, geht es um exakte Lösungen. Auch hier sind Computer im Einsatz seit es sie gibt. Ursprünglich waren es oft Programme die Lösungen in einer endlichen (grossen) Menge von Lösungen suchen (Gruppentheorie, 4-Farb-Satz) oder einfach nur viele kleine Probleme lösen um einen Überblick zu erhalten.

Diese Programme sind im Laufe der Zeit immer komplizierter geworden. Obwohl die Computer jedes Jahr schneller werden, gibt es Probleme die beliebig groß sind - und jeden Computer sprengen würden. Daher ist nun ein zentraler Bestandteil des Symbolischen Rechnens auch die Analyse der Programme: wie lange wird es dauern ... zu lösen. Wenn man mal an Verschlüsselung denkt (RSA) ist sofort klar, warum dies wichtig ist: Alle modernen Systeme können gebrochen werden, Teil der Spezifikation ist im Normalfall ein mathematisches Problem, dessen Lösung den Code bricht. In der Natur der Sache ist es, dass dieses Problem immer in endlicher Zeit lösbar ist - aber was heisst das? Lösbar in $> 10^6$ Jahren - dann ist der Code sicher. Lösbar in 10 min, dann vielleicht nicht.

Oft ist es auch so, dass Probleme einfache *algorithmische* Lösungen haben (es gibt ein Programm das Lösungen findet), aber es gibt keine *explizieten* Lösungsformeln.

Also: in der Vorlesung geht es darum für bestimme (einfache) mathematische Probleme Lösungen zu finden und diese dann zu analysieren und auszuprobieren. Dazu werden wir mit Algorithmen für ganze Zahlen und Polynome anfangen. Wir werden sehen, dass selbt das Multiplizieren von ganzen Zahlen oder Polynomen nicht völlig trivial ist - zumindest wenn die Zahlen gross genug sind. Zur Zeit wird mit Zahlen $> 10^{10}$ Stellen gerechnet - Zahlen die zu groß sind um sie im Hauptspeicher zu halten!

Nachdem wir mit Zahlen (und Polynomen) umgehen können, werden wir dies auf das Lösen von linaeren und algebraischen Gleichungssystemen anwenden - diese bilden die Grundlage für fast alle komplizierteren Algorithmen.

Zum Abschluss wollen wir uns dann noch mit Primzahltests und der Faktorisierung von Polynomen beschäftigen. Seltsamerweise wird sich herausstellen, dass es für Zahlen *viel einfacher* ist, zu entscheiden ob eine Primzahl vorliegt oder nicht, als eine Faktorisierung zu finden. Andererseits werden wir sehen, dass das Faktorisieren von Polynomen über $\mathbb{Q}$ viel einfacher ist als das von Zahlen.

KAPITEL 1

# Arithmetik

## 1. Addition und Multiplikation von Zahlen und Polynomen

Als erstes müssen wir uns überlegen wie Zahlen überhaupt auf dem Rechner dargesellt werden sollen. Auch dies ist nicht so einfach, verschiedene Probleme und Methoden bevorzugen verschiedene Darstellungen. Ein fundamentales Problem ist es offenbar, dass Rechner endlich sind, Zahlen jedoch nicht.

DEFINITION 1.1: Sei $B > 1$ eine feste ganze Zahl. Ferner seien $0 \leq a_i < B$ $(0 \leq i \leq n)$ beliebig. Dann heißt die Folge $(a_n, \ldots, a_1, a_0)$ die $B$-adische Entwicklung von $\sum_{i=0}^{n} a_i B^i =: (a_n, \ldots, a_0)_B$.

BEMERKUNG 1.2: Die Abbildung

$$\phi : \{0, \ldots, B-1\}^n \to \{0, \ldots, B^{n+1} - 1\} : (a_n, \ldots, a_0) \to \sum_{i=0}^{n} a_i B^i$$

ist bijektiv, daher ist die $B$-adische Entwicklung eindeutig - bis auf führende Nullen.

Für $B > 10$ werden auch Buchstaben als „Ziffern“ benutzt, z.B. für $B = 16$ benutzt man $\{0, \ldots, 9, A, \ldots F\}$.

BEISPIEL 1.3: $(1001)_2 = 9_{10}$ und $(1001)_3 = 28$.

$(D3)_{16} = (323)_8 = (11010011)_2 = 211$.

Aufgabe: Finde einen Algorithmus um von $B$-adischer Darstellung auf $C$-adisch umzurechnen - unter der Annahme, dass wir mit beliebig grossen Zahlen bereits rechnen können.

Auf Computern wird (heute) oft $B = 2^{32}$ oder $B = 2^{64}$ benutzt. Ebenfalls populär ist $B = 2^{23}$, aber auch $B = 10^4$ wurde ernsthaft eingesetzt. Welche Basis tatsächlich benutzt werden sollte, ist nicht so ganz offensichtlich, wir kommen dazu später nochmals zurück. Dies hängt ganz entscheidend von der Anwendung und der Harware ab!

BEMERKUNG 1.4: (1) Für $a \in \mathbb{Z}$ können wir eine eindeutige $B$-adische Darstellung erhalten: finde $n$ mit $B^{n-1} \leq |a| < B^n$. Dann gibt es eine $B$-adische Darstellung $|a| = (a_{n-1}, \ldots a_0)_B$ mit $a_{n-1} \neq 0$. Für $a$ verwenden wir dann $(\operatorname{sign} a, (a_{n-1}, \ldots, a_0))$ mit $\operatorname{sign} a := \begin{cases} 1 & a > 0, \\ 0 & a = 0, \\ -1 & a < 0. \end{cases}$.

(2) Für $b \in \mathbb{Q}$, $b = r/s$ mit $s > 0$ und $\operatorname{ggT}(r, s) = 1$ erhalten wir eine Darstellung via $(\operatorname{sign} r, r_B, s_B)$. Es ist oft sinnvoll eine ungekürzte Darstellung zu erlauben.

(3) Reelle Zahlen sind schwieriger da sie i.Allg. nicht exakt und vollständig dargestellt werden können. Auf dem Computer werden sie oft in der Form $(\operatorname{sign} x, m, e) = \operatorname{sign} x \cdot m \cdot B^e$ Dargestellt wobei noch zusätlich $m \in \mathbb{Z}$, $0 < m < X$ und $B \nmid m$ gefordert wird. Für $c$-doubles gilt etwa $X = 2^{56}$, $B = 2$ und $|e| < 1023$.
Im Gegensatz zu anderen Zahlen haben doubles damit eine feste Länge!

DEFINITION 1.5: Sei $x \in \mathbb{R}$. Dann ist $\lfloor x \rfloor := \max\{z \in \mathbb{Z} : z \leq x\}$ (Floor von $x$) und $\lceil x \rceil := \min\{z \in \mathbb{Z} : z \geq x\}$ (Ceiling von $x$).
Ferner sei $z \bmod B := z - \lfloor \frac{z}{B} \rfloor B$ „der Rest bei Division durch $B$“ (also ist der Rest hier *immer* positiv!)

BEMERKUNG 1.6: Oft wird auch $\lfloor x \rceil := \lfloor x + 1/2 \rfloor$ zum „Runden“ benutzt. Achtung: das Vorzeichen des Restes ist in vielen Programmiersprachen (c!) nicht definiert und daher Prozessorabhängig.

Damit können wir jetzt einen Algorithmus zur Addition von positiven ganzen Zahlen formulieren:

ALGORITHMUS 1.7: Vorzeichenfreihe Addition von Zahlen

Input: Zahlen $(a_n, \ldots, a_0)$ und $(b_n, \ldots, b_0)$ mit $0 \leq a_i, b_i < B$
Output: Zahlen $(c_m, \ldots, c_0)$ mit $(c_m, \ldots, c_0)_B = (a_n, \ldots, a_0)_B + (b_n, \ldots, b_0)_B$ und $m = n + 1$.
$r := 0$
for $i := 0$ to $n$ do
  $c_i := a_i + b_i + r \bmod B$
  $r := \lfloor (a_i + b_i + r)/B \rfloor$
$c_{n+1} := r$

BEMERKUNG 1.8: (1) So kann der Algorithmus natürlich nur schlecht implementiert werden. Damit der Algorithmus funktioniert müssen wir zwei Zahlen $a_i$ und $b_i$ addieren können. In der $c$-Sprache z.B. wären $a_i$ und $b_i$ `int` oder `unsigned int` Datentypen. In diesem Fall gibt es keine operation „+“, es gibt nur $a_i + b_i \bmod 2^?$. Also muss z.B. $B$ so gewählt werden, dass dieses „+“ exakt ausgeführt werden kann...
Alternativ gibt es oft eine CPU-Instruktion „Add-With-Carry“ die genau die beiden inneren Schritte für $B = 2^{64}$ implementiert.
$B = 2^{23}$ kommt von Versuchen `double` statt `int` zu benutzen.
(2) In jedem Schleifendurchlauf haben wir $r = 0, 1$, $0 \leq a_i + b_i + r < 2B - 1$ und $(a_i, \ldots, a_0)_B + (b_i, \ldots, b_0)_B = (r, c_i, \ldots, c_0)_B$
(3) Offensichtlich sollten wir $a_i + b_i + r$ nur einmal berechnen.
(4) Die (möglicherweise) teuren Divisionen in mod und $r$ können z.B. durch

$$s := a_i + b_i + r, \text{ if } s > B \text{ then } c_i := s - B, r := 1 \text{ else } c_i := s, r := 0$$

ersetzt werden. Aber je nach Wahl von $B$ und dem Prozessor, kann beides durch eine einzige Instruktion ersetzt werden.

Vieles, was man mit ganzen Zahlen machen kann, auch algorithmisch, kann auch mit Polynomen durchgeführt werden. Oft ist dies sogar einfacher da man keinen Übertrag braucht. Die Grundidee ist einfach, dass Polynome wie $x$-adische Zahlen aussehen.

Sei $R$ ein kommutativer Ring mit 1 (alle Ringe hier sind kommutativ mit 1, daher werden wir dies in Zukunft weglassen). Wir nehmen zusätzlich noch an, dass $R$ konstruktiv ist, d.h. alle Ringoperationen können algorithmisch realisiert werden. $R[x]$ ist nun der Ring der Polynome über $R$ mit der Unbestimmten $x$, also $f \in R[x]$ hat die Form

$$f = \sum_{i=0}^{n} f_i x^i$$

Der grösste Index $i$ mit $f_i \neq 0$ heisst der Grad von $f$, formal $\deg f$. Wir definieren $\deg 0 := -\infty$. Polynome werden (oft) als Koeffizientenvektor dargestellt

$$f = (f_n, \ldots, f_0) \in R^{n+1}.$$

Andere Darstellungen speichern Paare $(i, f_i)$ von Null verschiedenen Koeffizienten.

Offenbar gilt für $g = \sum_{i=0}^{n} g_i x^i$:

$$f + g = \sum_{i=0}^{n} (f_i + g_i) x^i$$

Daher ist der Algorithmus für Polynomaddition einfacher als für Zahlen, da wir keinen Übertrag brauchen.

ALGORITHMUS 1.9: Addition von Polynmen

Input: Polynome $(f_n, \ldots, f_0)$ und $(g_n, \ldots, g_0)$
Output: Polynom $(c_n, \ldots, c_0) = (a_n, \ldots, a_0) + (b_n, \ldots, b_0)$
for $i := 0$ to $n$ do
$c_i := f_i + g_i$

## 2. Multiplikation

Ziel: gegeben $a := (a_n, \ldots, a_0)_B$ und $b := (b_n, \ldots, b_0)_B$, gesucht $(c_m, \ldots, c_0)_B = ab$. Klar ist, mittels Kroncker Produkt von Polynomen:

$$(\sum_{i=0}^{n} a_i B^i)(\sum_{i=0}^{n} b_i B^i) = \sum_{l=0}^{2n} (\sum_{i+j=l} a_i b_j) B^l$$

Leider ist jedoch i. Allg. $\sum_{i+j=l} a_i b_j > B$. In der Schule wird dies etwas anders geschrieben und ist die Basis für die Multiplikation:

$$(\sum_{i=0}^{n} a_i B^i)(\sum_{i=0}^{n} b_i B^i) = \sum_{i=0}^{n} a_i (\sum_{j=0}^{n} b_j B^j) B^i$$

In dem normalen Diagramm wird in jeder Zeile $a_i(\sum_{j=0}^{n} b_j B^j)$ berechnet, Diagonal aufgeschrieben $(\cdot B^j)$ und dann mit Übertrag addiert:

| | | | | | |
|---|---|---|---|---|---|
| 1 | 6 | 3 | $\cdot$ | 6 | 7 |
| | 9 | 7 | 8 | | |
| | 1 | 1 | 4 | 1 | |
| 1 | 0 | 9 | 2 | 1 | |

Wenn wir dies mit dem Additionsalgorithms verbinden (in jeder Zeile werden die einzelnen Stellen in der richtigen Reihenfolge erzeugt!) erhalten wir:

ALGORITHMUS 1.10: Schul-Multiplikation von Zahlen

Input: Zahlen $a = (a_n, \ldots, a_0)_B$ und $b = (b_m, \ldots, b_0)_B$
Output: $c = (c_{n+m+1}, \ldots, c_0)_B = ab$
for $i := 0$ to $n + m + 1$ do $c_i := 0$
for $j := 0$ to $n$ do
  $r := 0$
  for $i := 0$ to $m$ do
    $s := a_i b_j + c_{i+j} + r$
    $c_{i+j} := s \bmod B$
    $r := \lfloor s/B \rfloor$
  $c_{m+i+1} := r \bmod B$

LEMMA 1.11: Der Algorithmus ist korrekt.

BEWEIS. In jedem Schleifendurchlauf gilt $s < B^2$ und $r < B$ denn am Anfang wird $r = 0$ gesetzt. Dann per Induktion: $s = a_i b_j + c_{i+j} + r \leq (B-1)^2 + (B-1) + (B-1) = B^2 - 1$. Daher ist das neue $r := \lfloor s/B \rfloor < B$. □

Wenn wir ungefähr mitzählen, so sehen wir, dass $(n+1)(m+1)$ viele Multiplikationen benötigt werden, um diese $s$ zu erhalten. Falls also $n = m$ gilt, so benötigen wir $(n+1)^2$ viele Elementarmultiplikationen.

BEMERKUNG 1.12: Um $s = ab + c + r$ zu berechnen brauchen wir bereits Operationen für Zahlen $> B$, wie wir gesehen haben bis $B^2 - 1$. Oft gibt es auch hier spezielle Prozessorinstruktionen um zwei Zahlen der Grösse $B = 2^{64}$ zu multiplizieren und das Ergebnis in zwei Variablen $l$ und $u$ abzuspeichern $ab = l + 2^{64}u$. Die Berechnung des neuen $r$ ist dann einfach. In GNU-c (gcc) kann dafür *long long* oder inline assembler benutzt werden. Alle (schnellen) Bibliotheken für Ganzzahl-Arithmetik nutzen diese Prozessorinstruktionen.

Auf einigen Prozessoren ist es *schneller* `double` Multiplikationen zu benutzen als `int`, daher wird oft $B = 2^{23}$ eingesetzt.

Es gibt jedoch substantiell schnellere Methoden ganze Zahlen zu multipizieren. Dies ist immer noch ein aktives Forschungsgebiet: erst in 2007 wurde der bisher schnellste Algorithmus veröffentlicht! Leider muss man in der Praxis jedoch alle Methoden implementieren da die Vorteile der prinzipiell schnelleren Methoden erst bei sehr grossen Zahlen greifen.

Ein einfaches Beispiel ist die Karatsuba-Multiplikation, ein typischer Divide-and-conquer (divide-et-impera, teile-und-hersche) Algorithmus. Seien $0 \leq a, b, c, d < B^n$ gegeben. Wir wollen $xy := (a + bB^n)(c + dB^n)$ berechnen (in der $B$-adischen Darstellung von $x$ enthält $a$ nur die letzten $n$ Stellen, $b$ die ersten $n$ Stellen). Dann gilt

$$xy = bdB^{2n} + (ad + bc)B^n + ac$$

Also kann $xy$ durch 4 kleinere Multiplikationen erhalten werden (plus Verschieben und Addieren). Karatsuba hatte nun die Idee es anders zu schreiben:

$$xy = bdB^{2n} + (ac + bd + (b - a)(c - d))B^n + ac$$

Da hier $ac$ und $bd$ doppelt verwendet werden kommen wir mit 3 kleineren Multiplikationen aus! Leider brauchen wir jetzt mehr Additionen, die aber im Vergleich „preiswert“ sind. Diese Idee führt zu dem nächsten Algorithmus:

ALGORITHMUS 1.13: Karatsuba-Multiplikation von Zahlen

Input: Zahlen $a = (\epsilon, (a_n, \dots, a_0)_B)$ und $b = (\sigma, (b_n, \dots, b_0)_B)$ mit Vorzeichen $\epsilon, \sigma \in \{\pm 1, 0\}$ mit $n + 1 = 2^k$.
Output: $c = (\tau, (c_{2n+1}, \dots, c_0)_B) = ab$
- if $n = 1$ then return $(\sigma\epsilon, (\lfloor a_0 b_0 / B \rfloor, a_0 b_0 \bmod B))$
- $x_1 := (a_n, \dots, a_{n+1/2})$, $x_2 := (a_{n-1/2}, \dots, a_0$
- $y_1 := (b_n, \dots, b_{n+1/2})$, $y_2 := (b_{n-1/2}, \dots, b_0$
- $m_1 := \texttt{Karatsuba}((1, x_1), (1, y_1))$
- $m_2 := \texttt{Karatsuba}(x_1 - x_2, y_1 - y_2)$
- $m_3 := \texttt{Karatsuba}((1, x_2), (1, y_2))$
- return $(\epsilon\sigma, m_1 B^n + (m_1 + m_2 + m_3) B^{n-1/2} + m_3)$

BEWEIS. Es ist einfach zu sehen, dass der Algorithmus rekursiv auf Zahlen der halben Länge angewendet wird, die inneren Aufrufe also korrekt sind. □

Klar ist, dass der Algorithmus sich sofort auch auf Polynome verallgemeinern lässt. Was nicht völlig klar ist, ist die Frage, ob er besser ist als die „Schul-Multiplikation“. Wir werden dies im weiteren Untersuchen, müssen jedoch erst Terminologie sammeln.

## 3. Komplexität von Algorithmen

Wir wollen Algorithmen am besten nach Laufzeit (und Speicherbedarf) vergleichen, müssen also die Laufzeit als „Funktion der Eingabedaten“ finden. Konkret werden wir die Laufzeit als Funktion der *Länge* der Eingabe bestimmen, wobei Länge sich am tatsächlichen Platzbedarf auf dem Rechner (oder dem Papier) orientiert. Noch konkreter: die Anzahl der $B$-adischen Stellen liefert diese Länge. Statt der Laufzeit (die stark vom Rechner abhängt) werden wir die Anzahl der elementaren Operationen benutzen, in der Hoffnung, dass alle elementaren Operationen etwa die selbe Zeit dauern. Leider hängt dies stark davon ab, was „elementar“ ist: bei den Polynomalgorithmen haben wir die Ringooperationen als elementar angesehen, während bei den ganzen Zahlen Operationen mit Zahlen $< B$ als elementar gelten. Dies liefert dann auch die Unterscheidung von

- Algebraischer Komplexität (alle Ringoperationen sind elementar)
- Binärer Komplexität (nur Bit-Operationen, $B = 2$ sind elementar)

Das Problem ist hier, dass z.B. für $R = \mathbb{Z}$ oder $R = \mathbb{Q}$ die Laufzeit einer einzelnen Ringoperation stark von der Größe der Zahlen abhängt. Oft ist daher das wesentliche Problem der binären Komplexität genau diese Größen abzuschätzen.

DEFINITION 1.14: Sei $f : \mathbb{N} \to \mathbb{R}_{>0}$. Dann definieren wir

$$O(f) := \{h : \mathbb{N} \to \mathbb{R}_{>0} : \exists C \in \mathbb{R}, N \in \mathbb{N} \text{ mit } \forall n > N : h(n) \leq C f(n)\}$$

Für $g \in O(f)$ wird auch oft $g = O(f)$ geschrieben.

(Im Prinzip $g = O(f)$ falls $g/f$ beschränkt ist, $g$ darf nicht schneller als $f$ wachsen). Diese Schreibweise wird auch auf Funktionen ausgedehnt, die nur „asymptotisch positiv“ sind, also $\exists N \forall n > N : f(n) > 0$ statt $f > 0$.

BEMERKUNG 1.15: (1) $f \in O(f)$

(2) $cf \in O(f)$ für alle $c > 0$.
(3) $O(f)O(g) = O(fg)$
(4) $gO(f) = O(fg)$

BEISPIEL 1.16: (1) $O(1)$ ist die Menge der beschränkten positiven Funktionen
(2) Sei $f \in \mathbb{R}[x]$ ein Polynom mit positivem Leitkoeffizienten. Dann gilt $f \in O(x^{\deg f})$i.

BEMERKUNG 1.17: (1) Die Laufzeit des Additionsalgorithmus für zwei (positive) Zahlen der $B$-adischen Länge $n$ ist $O(n)$
(2) Die „Schul-Multiplikation“ von zwei Zahlen der Längen $n$ und $m$ hat eine Laufzeit von $O(nm)$
(3) Der Karatsuba Algorithmus zur Multiplikation zweier Zahlen der Länge $n$ hat eine Laufzeit von $O(n^{\log_2(3)}) \sim O(n^{1.58})$

BEWEIS. Wir zählen Operationen:

(1) Die Schleife läuft über alle $n$ Stellen des Inputs. In jedem Durchlauf wird zunächst $s := a_i + b_i + r$ berechnet (2 $B$-Additionen), und dann $s \bmod B$ und $\lfloor s/B \rfloor$ was im wesentlichen 1 $B$-Division mit Rest ist. In der Summe ergibt dies 3 $B$-Operationen pro Durchlauf, also $3n$ insgesamt. Dann gibt es noch die Initialisierung $r = 0$ und die letzte Stelle $c_{m+1} = r$, also erhalten wir die Gesamtlaufzeit in $O(3m + 2) = O(m)$
(2) Wir haben $n+m+1$ in der Initialisierung $c_i = 0$, dann $nm$ $B$-Multiplikationen und $2nm$ Additionen $s := a_i b_j + c_{i+j} + r$ gefolgt von einer $B$-Division mit Rest. Also etwa $4mn$ $B$-Operationen insgesamt, daher $O(mn)$.
(3) Sei $T(n)$ die Zeit (Anzahl der $B$-Opertionen) für einen Aufruf von Karatsuba. Dann kann man sofort sehen, dass $T(n) = 3T(n/2)+cn$ für ein geeignetes $c > 0$ (drei rekursive Aufrufe plus Verwaltung und Addition) gilt. Damit:

$$\begin{aligned} T(2^m) &= 3T(2^{m-1}) + c2^m \\ &= 3(3T(2^{m-2}) + c2^{m-1}) + c2^m \\ &= 3^m T(1) + c2^m \sum_{i=0}^{m-1} (\frac{3}{2})^i \\ &= 3^m T(1) + c2^m \frac{(\frac{3}{2})^m - 1}{\frac{3}{2} - 1} \\ &\leq 3^m (T(1) + 2c) \\ &= 3^{\log_2 n} \tilde{c} = n^{\log_2 3} \tilde{c} \end{aligned}$$

□

Was man hier im Beweis sieht ist typisch: das genaue Zählen von Operationen ist sehr aufwendig und liefert sehr lange und komplizierte Formeln. Daher werden die Teilschritte oft nur abgeschätzt. Die Kunst liegt dann darin, gute Abschätzungen zu finden. Häufig wird der Algorithmus sogar so abgeändert, dass er sich besser analysieren lässt.

Was auch typisch ist, ist dass schnellere Methoden (Karatsuba) komplizierter als naive Algorithmen sind. Speziell sind sie oft langsamer für kleinen Input (z.B. ist

Karatsuba für $n = 2$ sicherlich nicht schnell) daher werden dann mehrere Algorithmen implementiert und dann automatisch benutzt - je nach Größe der Eingabe. Andererseits ist Karatsuba schon für Polynome vom Grad 2 mit großen Koeffizienten vom Vorteil. Im Prinzip kann der Algorithmus auch zur Multiplikation von komplexen Zahlen benutzt werden, hat jedoch schlechte numerische Eigenschaften.

Wann genau ein Algorithmus besser ist als ein anderer (eine Implementierung) hängt von vielen Dingen ab. In der Praxis wird dieser „Crossoverpoint“ immer experimentell bestimmt - und für jeden Rechner neu.

Es gibt noch schnellere Multiplikationsalgorithmen. Der bisher schnellste (leider noch nicht praktikabel) wurde erst 2007 gefunden.

In GMP (eine der schnellsten freien Bibliotheken für Ganz-zahl Arithmetik) wird Karatsuba auf Zahlen mit $> 32$ $B$-Stellen angewendet, wo $B = 2^{64}$ ist, also etwa ab 600 Dezimalstellen. Was auch auffällt ist, dass Karatsuba nur effizient ist, falls beide Zahlen eine ähnliche Grösse haben. Für gemischte Multiplikation sollte man anders vorgehen.

Aufgabe: Seien $a_i$ ($1 \leq i \leq n$, $n = 2^k$) ganze Zahlen. Ziel ist es $\prod a_i$ zu erhalten. Vergleichen sie die Komplexität von $\prod_{i=1}^{n} a_i$ mit der von $(\prod_{i=1}^{n/2} a_{2i})(\prod_{i=1}^{n/2} a_{2i-1})$, sowohl unter der Benutzung von Karatsuba, als auch der Schulmultiplikation. Entwickeln sie einen rekursiven Algorithmus um das Produkt zu berechnen. Kann dies noch verbessert werden?

Es gibt Verallgemeinerungen (Toom-Cook) die Karatsubas Idee auch für mehr als 2 Teile verallgemeinern.

## 4. Modulare Darstellung

Bisher haben wir Zahlen immer $B$-adisch dargestellt - aber es gibt natürlich auch andere Möglichkeiten. Eine solche ergibt sich z.B. aus dem Chinesischen Restsatz.

DEFINITION 1.18: Sei $m \in \mathbb{Z}$. Eine Teilmenge $M \subseteq \mathbb{Z}$ heißt vollständiges Restsystem modulo $m$, falls $\#M \cap (a + m\mathbb{Z}) = 1$ gilt für alle $a \in \mathbb{Z}$.

BEISPIEL 1.19:
(1) Sei $m = 0$, dann ist $\mathbb{Z}/m\mathbb{Z} \sim \mathbb{Z}$, also ist $\mathbb{Z}$ das einzige vollständige Restsystem
(2) Sei $m > 1$, dann ist $\{0, \ldots, n-1\}$ vollständig. Dieses wird auch mit mod$m$ erhalten.
(3) Sei $m > 1$ ungerade, dann ist $-(n-1)/2, \ldots, 0, \ldots (n-1)/2$ ebenfalls Vollständig. Dies ist das *symmetrische Restsystem*. (Analog kann es natürlich auch für $m$ gerade definiert werden)

SATZ 1.20 (Chinesischer Restsatz): Seien $m_i \in \mathbb{Z}$ paarweise koprim und $m := \prod_{i=1}^{n} m_i$. Ferner seien $M_i$ vollständige Restsysteme modulo $m_i$ und $M$ vollständig modulo $m$. Dann gilt:
(1) Die Abbildung
$$\phi : M \to \times M_i : x \mapsto (x \bmod m_i)_i$$
ist bijektiv
(2) Ferner gilt $\phi(a + b \bmod m) = (a + b \bmod m_i)_i$ falls $x \bmod m \in M$ gilt.

Dies ist nur eine leicht andere Fassung (mit Restsystemen) des Chinesischen Restsatzes aus der AGS Vorlesung.

Damit können wir eine *modulare* Darstellung definieren: Für $a \in M$ heißt $\phi(a) = (a \bmod m_i)_i$ die modulare Darstellung von $a$. Diese kann (und wird!) alternativ zur $B$-adischen Darstellung benutzt.

BEMERKUNG 1.21: (1) Wenn wir davon ausgehen, dass $\bmod m_i$, d.h. Operationen in $\mathbb{Z}/m_i\mathbb{Z}$ elementar sind, so kann $a + b$ in Zeit $O(n)$ berechnet werden (wie $B$-adisch).
Unter der selben Veraussetzung kann auch $ab$ in Zeit $O(n)$ berechnet werden - was viel schneller ist als selbst Karatsuba!
(2) Problematisch sind Überläufe: was passiert wenn $a+b > m$ ist? $B$-adisch fügen wir einfach eine Stelle an, aber hier?
Allgemeiner: wie testet man $a + b \in M$?
(3) Wie kann, effizient, zwischen den verschiedenen Darstellungen gewechselt werden?
(4) Wie kann $a > b$ entschieden werden?

Obwohl es Probleme gibt wird die modulare Darstellung oft eingesetzt. Ein wesentlicher Vorteil ist in einem von den Problemen enthalten: die Zahlen die verwarbeitet werden müssen sind in der größe beschränkt. Speziell wird dies daher eingesetzt wenn *kleine* Zahlen berechnet werden sollen, aber Zwischenergebnisse sehr Groß sein können.

SATZ 1.22 (Chinesischer Restsatz für Polynome): Sei $K$ ein Körper und $m_i \in K$ $(1 \le i \le n)$ paarweise verschieden. Dann gilt
(1) Die Abbildung
$$\phi : \{f \in K[x] : \deg f < n\} \to K^n : f \mapsto (f(m_i))_i$$
ist eine Bijektion.
(2) Für $f, g \in K[x]$ mit $\deg f, g < n$ gilt $\phi(f+g) = \phi(f)+\phi(g)$. Falls $\deg fg < n$, so gilt auch $\phi(fg) = \phi(f)\phi(g)$.

BEWEIS. Dies ist eine Variante des Chinesischen Restsatzes für Polynome via Evaluation/Interpolation. Die Injektivität der Abbildung folgt aus der Tatsache, dass Polynome von Grad $\le n$ höchstes $n$ Nullstellen haben können. Die Surjektivität ist konstruktiv (Übung) und wird durch (Lagrange) Interpolation gelöst. □

Um eine modulare Darstellung von einem Polynom $f$ zu erhalten, müssen wir $f$ an verschiedenen Stellen auswerten. Naiv benötigt jede Auswertung etwa $O(n^2)$ Multiplikationen in $K$, aber es geht besser:

ALGORITHMUS 1.23 (Horner Schema): Polynomevaluation mittels Horner Schema

Input: Polynom $f = \sum_{i=0}^{n} f_i x^i$ und $m \in K$.
Output: $f(m)$
- $s := f_n$
- for $i := n - 1$ to $0$ do $s := sm + f_i$
- return $s$

Der Algorithmus benötigt offenbar $O(n)$ Multiplikationen und Additionen. Es ist nicht offensichtlich, aber auch das kann man verbessern - zumindest wenn man viele Auswertungen haben will.

Eine weitere Operation die oft benötigt wird ist das Potenzieren:

ALGORITHMUS 1.24: Schnelles Potenzieren

Input: $x$, $n \in \mathbb{N}$
Output: $x^n$
- if $n = 0$ then return 1
- $y := x^{\lfloor n/2 \rfloor}$
- if IsOdd($n$) then return $xy$ else return $y$

Dieser Algorithmus braucht offenbar nur $O(\log n)$ Multiplikationen um $x^n$ zu berechnen - aber Achtung: für viele $x$ wird $x^n$ sehr sehr groß, sodass die Laufzeit von $x^n$ NICHT $O(\log n)$ sein wird. Der Algorithmus kann auch iterativ (nicht-rekursiv) implementiert werden. Es gibt auch Varianten, die Potenzprodukte $\prod x_i^{m_i}$ berechnen.

Unter Einsatz des Hornerschemas kann daher $f(m_i)$ für $n$ verschiedene $m_i$ in Zeit $O(n^2)$ berechnet werden. Wir werden jetzt sehen, dass zumindest für spezielle $m_i$ dies in Zeit $O(n \log n)$ berechnet werden kann, mittels schneller Fourier Transformationen (FFT: fast Fourier transform, DFT: discrete Fourier transform, FFTW: fastest Fourier transform in the west)

## 5. Fouriertransformation

Fouriertransformationen kommen eigentlich aus der Analysis (oder Physik oder Elektrotechnik). Sie werden eingesetzt um bestimmte, komplizierte Integrale durch einfache Produkte zu ersetzen. Man kann eine allgemeine Theorie von Fouriertransformation und (Haar)-Maßen aufbauen die all diese Fälle enthält. Hier jedoch werden wir es einfach „zu Fuß“ tun. Hier sind Fouriertransformationen einfach geschickte, rekursive Auswertungen von Funktionen an speziellen Stützstellen.

Sei $R$ kommutativ mit 1, $f \in R[x]$, $f = \sum_{i=0}^{n-1} f_i x^i$ mit $n$ gerade. Dann gilt

$$f(x) = g(x^2) + xh(x^2)$$

mit geeigneten Polynomen $g$ und $h$: $g = \sum_{i=0}^{n/2-1} f_{2i} x^i$ und $h = \sum_{i=0}^{n/2-1} f_{2i+1} x^i$. Dies, zusammen mit geeigneten Stützstellen ist der Schlüssel zu schneller Polynommultiplikation:

LEMMA 1.25: Sei $n$, $n/2$ gerade und $x_0, \ldots, x_n \in R$ mit
(1) $x_i \neq x_j$ für $i \neq j$
(2) $x_{n/2+i} = -x_i$ für $0 \leq i \leq n/2 - 1$.
Dann gilt für $y_i := x_i^2 = (x_{n/2+i})^2$ $(0 \leq i \leq n/2 - 1)$ und die $y_i$ erfüllen ebenfalls (1) und (2). Um nun ein Polynom $f \in R[x]$ vom Grad $\deg f = n - 1$ an den $x_i$ Auszuwerten benötigen wir nur $T(n) = 2T(n/2) + cn/2$ elementare Operationen in $R$.

BEWEIS. Wir benutzen die Zerlegung $f = g(x^2)+xh(x^2)$ und werten (rekursiv) $g(y_i)$ und $h(y_i)$ aus. Wegen (2) benötigen wir $n/2$ Operationen um $y_i$ zu berechnen und $2T(n/2)+cn/2$ um $g(y_i)$ und $h(y_i)$ zu bestimmen. Also noch $n/2$ Multiplikationen für $y_ih(y_i)$, $n/2$ Additionen für $g(y_i) + x_ih(y_i)$ und schliesslich noch $n/2$ Subtraktionen für $f(x_{n/2+i}) = f(-x_i) = g(y_i) - y_ih(y_i)$. □

Um das Lemma anwenden zu können, brauchen wir geeignete Evaluierungspunkte. Da wir zusätzlich auch noch $f(x_i)$ klein haben wollen, sollten die $x_i$ auch nicht zu groß sein.

DEFINITION 1.26: Seien $R$ ein Ring und $n > 0$. Ein $\zeta \in R$ heißt primitive $n$-te Einheitswurzel, falls $\zeta^n = 1$ und $\zeta^j \neq 1$ für $0 < j < n$.

BEISPIEL 1.27:
(1) $\zeta := \exp(2\pi i/n) \in \mathbb{C}$ ist eine (komplexe) primitive $n$-te Einheitswurzel
(2) $2 \in \mathbb{Z}/9\mathbb{Z}$ ist primitive 6-te Einheitswurzel: $2^2 = 4$, $2^3 = 8 = -1$, $2^4 = 16 = 7$, $2^5 = 2 \cdot 7 = 14 = 5$ und $2^6 = 1$.
(3) $4 \in \mathbb{Z}/41\mathbb{Z}$ ist primitive 8-te Einheitswurzel
(4) 2 ist primitive $2n$-te Einheitswurzel in $\mathbb{Z}/(1 + 2^n\mathbb{Z})$

LEMMA 1.28: Sei $R$ ein Ring, $n > 0$, $\zeta \in R$ primitive $n$-te Einheitswurzel und $x_i := \zeta^i$ für $0 \leq i < n$. Dann gilt:
(1) $\{x_0, \ldots, x_{n-1}\}$ ist eine (multiplikative) Gruppe.
(2) Für $n$ gerade ist $\zeta^2$ eine primitive $n/2$-te Einheitswurzel.
(3) Sei $n$ gerade, $\zeta^{n/2} = -1$ und $\operatorname{char} R \neq 2$. Dann gilt $x_{n/2+i} = -x_i$.
(4) Seien $n$, $n/2$ gerade und $\zeta^{n/2} = -1$. Dann erfüllen die $x_i$ die Bedingungen von Lemma 1.25

BEWEIS. Setze $\phi : (\mathbb{Z}/m\mathbb{Z}, +) \to (R^*, \cdot) : i \mapsto \zeta^i$. Dies ist ein Gruppenhomomorphismus, also ist $\operatorname{Im}\phi = \{x_i : 0 \leq i < n\}$ eine Gruppe. Der Rest ist klar. □

BEMERKUNG 1.29: In Charakteristik 2 muss man mit Quadrieren und Quadratwurzeln aufpassen: $R = \mathbb{F}_2[x]/(x^4 + 1)$, $\zeta := x + (x^4 + 1)\mathbb{F}_2[x]$. Dann ist $\zeta$ primitive 4-te Einheitswurzel, aber $\zeta^2 \neq \pm 1 = \zeta^0$

DEFINITION 1.30: Sei $R$ Ring, $n > 0$ und $\zeta$ eine primitive $n$-te Einheitswurzel. Die Abbildung

$$\phi : \{f \in R[x] : \deg f < n\} \to R^n : f \mapsto (f(\zeta^i))_{i=0}^{n-1}$$

heißt diskrete Fourier Transformation (DFT).

BEMERKUNG 1.31: Die DFT ist ein Spezialfall der modularen Darstellung von Polynomen.

LEMMA 1.32: Seien $R$, $n$ und $\zeta$ wie in Definition 1.30. Dann gilt:
(1) Bzgl. der Standardbasis $0, x, \ldots, x^{n-1}$ von $\{f \in R[x] : \deg f < n\}$ und der Standardbasis von $R^n$ hat $\phi$ die Matrix. $(\zeta^{ij})_{0 \leq i,j < n}$
(2) Gilt $\sum_{i=0}^{n-1} \zeta^{ij} = 0$ für alle $1 < j < n$ und ist $n$ invertierbar in $R$, so ist $(\zeta^{ij})_{i,j}$ invertierbar und damit $\phi$ ein Isomorphismus. Es gilt $(\zeta^{ij})_{i,j}^{-1} = \frac{1}{n}(\zeta^{-ij})_{i,j}$

BEWEIS. (1) Klar $\phi(x^i) = (\zeta^{ij})_j$ nach Definition.

(2) Setze $z := \sum_{j=0}^{n-1} \zeta^{ij}\zeta^{-jk} = \sum_{j=0}^{n-1} \zeta^{j(i-k)}$. (Eintrag von $(\zeta^{ij})_{j,k}(\zeta^{jk})_{j,k}^{-1}$). Gilt $i = k$ so ist $z = n$. Für $i > k$ gilt $z = 0$ nach Voraussetzung. Sei nun $i < k$. Dann gilt $z\zeta^{(n-1)(k-i)} = \sum_{j=0}^{n-1} \zeta^{j(i-k)+(n-1)(k-i)} = \sum_{j=0}^{n-1} \zeta^{(n-1-j)(k-i)} = 0$ (die selbe Summe wie vorher, andere Reihenfolge). Da $\zeta^{(n-1)(k-i)}$ eine Einheit ist, folgt $z = 0$. Also ist $\frac{1}{n}(\zeta^{-ij})_{i,j}$ die inverse Matrix wie behauptet.

□

Die Bedingung $\sum_{i=0}^{n-1} \zeta^{ij} = 0$ ist z.B. erfüllt, falls $n$ invertierbar und $\zeta^k - 1$ kein Nullteiler für $1 \leq k < n$ ist, denn $(\sum_{j=0}^{n-1} \zeta^{kj})(\zeta^k - 1) = \zeta^{kn} - 1 = 0$ (Teleskopsumme).

Dies klappt z.B. in $\mathbb{C}$ für $\zeta := \exp(2\pi i/n)$, ist jedoch falsch in $\mathbb{Z}/9\mathbb{Z}$ und $\zeta = 4$, da $1 + 4 + 4^2 = 21 = 3 \neq 0$ in $\mathbb{Z}/9\mathbb{Z}$ gilt.

Damit können wir einen (schnellen) Algorithmus für die modulare Darstellung erhalten:

ALGORITHMUS 1.33: Berechnung der FFT über einem geeigneten Ring

Input: $n = 2^k$, $\zeta$ geeignet und $R[x] \ni f = \sum_{i=0}^{n-1} f_i x^i$,
Output: $(f(\zeta^i))_{i=0}^{n-1}$
- if $n = 1$ then return $f_0$.
- Schreibe $f = g(x^2) + xh(x)$
- $b := \texttt{FFT}(n/2, g, \zeta^2)$
- $c := \texttt{FFT}(n/2, h, \zeta^2)$
- for $i := 0$ to $n/2 - 1$ do
- $\quad a_i := b_i + \zeta^i c_i$
- $\quad a_{n/2+i} := b_i - \zeta^i c_i$
- return $a \in R^n$

BEMERKUNG 1.34: Sei $R$ ein Ring mit 1, $R$ enthalte eine $n = 2^k$-te primitive Einheitswurzel für alle $k$. Dann benötigt der Algorithmus $O(n \log n)$ elementare Operationen in $R$.

BEWEIS. Nach Lemma 1.28 erfüllt $x_i := \zeta^i$ die Bedingungen in Lemma 1.25. Damit gilt dann $T(n) = 2T(n/2) + cn/2$. Per Induktion folgt damit $T(n) = 2^k T(1) + 2^{k-1} c \log_2 n = O(n \log n)$. □

Damit können wir in geeigneten Ringen/Körpern Polynome schnell multiplizieren:

ALGORITHMUS 1.35: Polynonmultiplikation per FFT

Input: Polynome $f$, $g$ mit $\deg f, \deg g$
Output: $fg$
- Finde minimales $k$ mit $\deg f + \deg g < 2^k$ und berechne eine primitive $2^k$-te Einheitswurzel
- $a := \texttt{FFT}(2^k, \zeta, f)$
- $b := \texttt{FFT}(2^k, \zeta, g)$
- for $i = 0$ to $2^k - 1$ do $c_i := a_i b_i$
- return $\frac{1}{n}\texttt{FFT}(2^k, \zeta^{-1}, c)$

(Hier wird $\{f \in R[x] | \deg f < n\}$ mit $R^n$ identifiziert.)

BEMERKUNG 1.36: (1) Prinzipiell können so auch ganze Zahlen multipliziert werden. Man denkt sich $\sum_{i=0}^{l} b_i B^i$ als Polynom (mit $B$ beeignet), benutzt dann FFT um die Polynome zu multiplizieren und sortiert dann die Überträge (im Produkt werden die Koeffizienten zu groß sein).

(2) Um ganze Zahlen zu multiplizieren kann z.B. entweder $\zeta = \exp(2\pi i/n) \in \mathbb{C}$ benutzt werden - oder $\zeta = 2 \in \mathbb{Z}/(2^{2^n}+1)\mathbb{Z}$. Was genau benutzt werden sollte, hängt wieder vom Prozessor ab.

(3) Jede Multiplikationsfunktion für Zahlen kann auch sofort benutzt werden um Polynome über $\mathbb{Z}$ zu multiplizieren. Aus $f = \sum_{i=0}^{n} f_i x^i$ wird dann $z := \sum_{i=0}^{n} f_i B^i$ mit $B$ so groß, dass alle Koeffizienten des Produktes kleiner als $2B$ sind.

(4) Wenn mehrere Multiplikationen durchgeführt werden sollen, kann man ggf. die FFTs nur einmal am Anfang durchführen.

KAPITEL 2

# Division und ggT

## 1. Division

**Definition 2.1:** Ein Ring $R$ heißt euklidisch, wenn es eine Abbildung $\nu : R \to \mathbb{N}_0$ gibt, so dass es für alle $a, b \in R$, $b \neq 0$ ein $q$ und $r \in R$ gibt mit

$$a = qb + r \quad \text{und} \quad \nu(r) < \nu(b)$$

**Beispiel 2.2:** (1) $R = \mathbb{Z}$ mit $\nu(x) := |x|$ ist euklidisch.
(2) $K$ ein Körper, $R = K[x]$ mit $\nu(x) := a^{\deg x}$ ist euklidisch für $a > 1$ (wegen $\deg 0 = -\infty$ ist dann $\nu(0) := 0$).
(3) Jeder Körper $K$ ist euklidisch mit $\nu(0) = 0$ und $\nu(x) = 1$

Jetzt wollen wir Algorithmen ifür $\mathbb{Z}$ untersuchen die die Division mit Rest die in der Definition von euklidischen Ringen gefordert ist, realisieren. Zunächst nehmen wir an, dass $a, b > 0$ gilt. Also $a = \sum_{i=0}^{n} a_i B^i$, $b = \sum_{i=0}^{m} b_i B^i$ mit $a_i, b_i \in \{0, \ldots, B-1\}$ gesucht ist $c = \sum_{i=0}^{k} c_i B^i$ mit $0 \leq c_i < B$ und $c = \lfloor a/b \rfloor$.

Die Idee ist es, die normale Schuldivision umzusetzen:

$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 1 & 1 & 9 & \\ \hline & & 4 & 4 \\ & & 3 & 4 \\ \hline & & 1 & 0 \end{array} \quad = 17 \cdot 72 + 10$$

Wie auch bei der Division von Hand ist das Problem zu „raten“, wie oft der Divisor abgezogen werden kann. Wir werden versuchen, dies an den ersten zwei Stellen abzulesen.

**Lemma 2.3:** (1) Für $a \in \mathbb{Z}$ und $x \in \mathbb{R}$ gilt $\lfloor a + x \rfloor = a + \lfloor x \rfloor$
(2) Für $a \in \mathbb{Z}$ und $x \in \mathbb{R}$ gilt $\lfloor ax \rfloor \geq a \lfloor x \rfloor$ Für $x \notin \mathbb{Z}$ ist die Ungleichung strikt.
(3) $B - \lfloor B/(x+1) \rfloor \geq x \lfloor B/(x+1) \rfloor \geq B/2$ für $1 \leq x \leq B-1$ und $x, B \in \mathbb{Z}$, $B > 1$.

**Beweis.** (1) klar
(2) $\lfloor x \rfloor = x - \epsilon$ für $0 < \epsilon < 1$. Damit folgt dann $a\lfloor x \rfloor = ax - a\epsilon$ und $\lfloor ax \rfloor = a\lfloor x \rfloor + a\epsilon$.
(3) $B - \lfloor B/(x+1) \rfloor = \lfloor B - B/(x+1) \rfloor = \lfloor xB/(x+1) \rfloor > x\lfloor B/(x+1) \rfloor$. Schreiben wir $B = q(x+1) + r$ mit $0 \leq r \leq x$, so gilt $\lfloor B/(x+1) \rfloor = q = (B-r)/(x+1) \geq (B-x)/(x+1)$. Die Funktion $x \to x(B-x)/(x+1)$ hat ein Maximum bei $-1/2 + \sqrt{1/4 + B} \approx \sqrt{B} > B/2$. Damit ist das Minimum auf $[1, B/2]$ bei $x = 1$, also $(B-1)/2$. Damit folgt dann $x\lfloor B/(x+1) \rfloor \geq (B-1)/2$. Da die linke Seite ganz ist, gilt dann weiter $x\lfloor B/(x+1) \rfloor = \lceil x\lfloor B/(x+1) \rfloor \rceil \geq \lceil (B-1)/2 \rceil = \lfloor B/2 \rfloor$. (Nach Knuth)

□

LEMMA 2.4: Seien $n \in \mathbb{N}$, $n > 0$, $a = (a_n, \dots, a_0)_B$, $b = (b_{n-1}, \dots, b_0)_B$, $b_{n-1} \neq 0$ und $a/b < B$. Setze nun $q := \lfloor a/b \rfloor$ und $\tilde{q} := \min(\lfloor (a_n B + a_{n-1})/b_{n-1} \rfloor, B-1)$. Dann gilt:

(1) $\tilde{q} \geq q$
(2) Ist $b_{n-1} \geq \lfloor B/2 \rfloor$, so gilt $\tilde{q} \leq q + 2$
(3) Setze $d := \lfloor B/(b_{n-1}+1) \rfloor$, $a' := ad$, $b' := bd$, so gilt für die $B$-adische Entwicklung $a' = (a'_{n+1}, \dots, a'_0)$ und $b' = (b'_{n-1}, \dots, b'_0)$ mit $b'_{n+1} \geq \lfloor B/2 \rfloor$

BEWEIS. (1) Aus $a/b < B$ folgt sofort $q = \lfloor a/b \rfloor < B$, also gilt die Aussage für $\tilde{q} = B-1$. Sei nun $\tilde{q} < B-1$, also $\tilde{q} = \lfloor (a_n B + a_{n-1})/b_{n-1} \rfloor$. Damit haben wir auch $a_n B + a_{n-1} = b_{n-1}\tilde{q} + r$ mit $0 \leq r < b_{n-1}$.

$$\begin{aligned} a - \tilde{q}b &\leq a - \tilde{q} b_{n-1} B^{n-1} \\ &= \sum_{i=0}^{n} a_i B^i - B^{n-1}(a_n B + a_{n-1} - r) \\ &= \sum_{i=0}^{n-2} a_i B^i + r B^{n-1} \\ &\leq (B^{n-1} - 1) + (b_{n-1} - 1) B^{n-1} = b_{n-1} B^{n-1} - 1 < b \end{aligned}$$

Damit sehen wir $a/b - \tilde{q} < 1$ und damit $q = \lfloor a/b \rfloor \leq a/b < \tilde{q} + 1$ und damit die Behauptung.

(2) Angenommen $\tilde{q} \geq q + 3$. Wegen $b_{n-1} \geq \lfloor B/2 \rfloor$ folgt sofort $b \neq B^{n-1}$. Damit können wir dann

$$\tilde{q} \leq \frac{a_n B + a_{n-1}}{b_{n-1}} = \frac{a_n B^n + a_{n-1} B^{n-1}}{b_{n-1} B^{n-1}} \leq \frac{a}{b_{n-1} B^{n-1}} < \frac{a}{b - B^{n-1}}$$

Damit folgt dann auch (wegen $q = \lfloor a/b \rfloor \geq a/b - 1$)

$$3 \leq \tilde{q} - q < \frac{a}{b - B^{n-1}} - \frac{a}{b} + 1 = \frac{a}{b} \frac{B^{n-1}}{b - B^{n-1}} + 1$$

Also $\frac{a}{b} > 2\frac{b - B^{n-1}}{B^{n-1}} \geq 2(b_{n-1} - 1)$ und $q = \lfloor a/b \rfloor \geq 2(b_{n-1} - 1)$ da dies ebenfalls ganz ist. Zum Schluss folgt dann

$$B - 4 \geq \tilde{q} - 3 \geq q \geq 2(b_{n-1} - 1)$$

und $B/2 - 1 \geq b_{n-1}$ und damit ein Widerspruch!

(3) Die Aussage über die Länge von $a'$ ist klar. Da $b_{n-1} \neq 0$ folgt $B/2 \geq \lfloor B/2 \rfloor \geq d$. Es gilt $a'_{n+1} = \lfloor ad/B^{n+1} \rfloor \leq \lfloor (B^{n+1} - 1)d/B^{n+1} \rfloor \leq \lfloor d \rfloor \leq B - 2$
Gilt $b_{n-1} \geq B/2$ so folgt $d = 1$ und der Rest ist klar. Sei nun $b_{n-1} < B/2$ (und $d > 1$). Aus Lemma 2.3 folgt $B - d > d b_{n-1} \geq B/2$. Wegen $d(b - b_{n-1}B^{n-1}) \leq d(B^{n-1} - 1)$ folgt dann $b'_{n-1} = d b_{n-1} + \lfloor d/B^{n-1}(b - b_{n-1}B^{n-1}) \rfloor \leq B - 1$.

□

LEMMA 2.5: Seien $a = (a_n, \dots, a_0)_B$, $b = (b_{n-1}, \dots, b_0)_B$ $B$-adische Zahlen. Dann gilt $a/b < B$ genau dann, wenn $(a_n, \dots, a_1)_B < b$ gilt.

BEWEIS. Wir haben $a/b < B$ falls $a/B < b$ gilt. Aber $a/B = (a_n \dots, a_1)_B + a_0/B = \lfloor a/B \rfloor + a_0/B$, also $a/B < b$ genau dann, falls $\lfloor a/B \rfloor < b$ gilt (da $b$ ganz ist!) □

Damit können wir jetzt einen ersten Divisionsalgorithmus bauen:

ALGORITHMUS 2.6: Input: $a = (a_m, \dots a_0)_B$, $b = (b_n, \dots, b_0)_B$ mit $b_n \neq 0$
Output: $q$ und $r$ mit $a = qb + r$ und $0 \leq r < b$ in $B$-adischer Darstellung.
1: $d := \lfloor B/(b_n + 1) \rfloor$ und $b := bd$, $a := ad$
2: for $j := 0$ to $m - n - 1$ do
3: $\quad \tilde{q} := \min(\lfloor \frac{a_{m-j}B + a_{m-j-1}}{b_n} \rfloor, B - 1)$
4: $\quad$ while $(a_{m-j}, \dots, a_{m-n-j-1})_B - \tilde{q}b < 0$ do $\tilde{q} := \tilde{q} - 1$
5: $\quad q_{m-n-j} := \tilde{q}$
6: $\quad a := a - \tilde{q}bB^{m-n-j-1}$
7: $r := (a_m, \dots, a_0)_B/d$

BEMERKUNG 2.7: In dem Algorithmus gilt:

(1) Die „while“ Schleife (Schritt 4) um $\tilde{q}$ zu bestimmen wird maximal 2x durchlaufen. Ferner haben wir immer $(a_{m-j}, \dots, a_{m-n-j})_B/b < B$.
(2) Der Algorithmus benötig $O(n(m-n))$ $B$-Operationen

BEWEIS. (1) Durch Schritt 1 wird erreicht, dass das Lemma anwendbar ist: Ab Schritt 2 gilt $b_n > B/2$ und $\tilde{q} \leq q + 2$. Damit ist auch sofort klar, dass die Scheife in Schritt 4 nur 2x ausgeführt werden kann. Da sich $b$ nicht ändert, bleibt $b_n > B/2$ bis zum Ende korrekt.
Wir wollen Lemma Lemma 2.4 auf $(a_{m-j}, \dots, a_{m-n-j})_B$ und $b$ anwenden. Per Induktion gilt $j = 0$ :

□

## 2. Der Euklidische Algorithmus

Hier sei $R$ immer ein euklidischer Ring. Ausgangspunkt ist der klassische euklidische Algorithmus:

ALGORITHMUS 2.8: Einfacher ggT

Input: $a$, $b \in R$.
Output: $\mathrm{ggT}(a, b)$
- $r_0 := a$, $r_1 := b$, $i := 1$
- while $r_i \neq 0$ do
- $\quad r_{i-1} =: q_i r_i + r_{i+1}$, $i := i + 1$
- return $r_{i-1}$

LEMMA 2.9: Der Algorithmus terminiert und ist korrekt.

BEWEIS. Die Terminierung folgt daraus, dass in jedem Schritt entweder $r_i = 0$ gilt, oder $\nu(r_{i+1}) < \nu(r_i)$. Für die Korrektheit bemerken wir $\mathrm{ggT}(a, b) = \mathrm{ggT}(a, b + qa)$ für $q \in \mathbb{Z}$, also gilt immer $\mathrm{ggT}(a, b) = \mathrm{ggT}(r_i, r_{i+1})$. □

BEMERKUNG 2.10: Der Algorihmus wird natürlich anders implementiert. Die $r_i$ sind jedoch nützlich für den Beweis. Hier ist eine vereinfachte, typischere Variante:

Input: $a$, $b \in R$.
Output: $\mathrm{ggT}(a, b)$
while $b \neq 0$ do
$\quad a =: qb + r$

$a := b$, $b := r$
return $a$

BEMERKUNG 2.11: (1) Durch unsere Normierung sind sowohl $q$ und $r$ als auch der ggT selber eindeutig bestimmt - zumindest, falls $R = \mathbb{Z}$ oder $R = K[t]$ gilt.
(2) Da $ab = \mathrm{ggT}(a,b)\,\mathrm{kgV}\,(a,b)$ gilt können wir auch kgV s berechnen.
(3) Wegen $\mathrm{ggT}(a,b,c) = \mathrm{ggT}(a,\mathrm{ggT}(b,c))$ können wir auch ggTs von mehreren Elementen berechnen.

Der nächste Schritt ist der erweiterte ggT: es gibt $s$, $t$ mit $\mathrm{ggT}(a,b) = sa + tb$:

ALGORITHMUS 2.12: Erweiterter ggT

Input: $a, b \in R$
Output: $s, t \in R$ mit $\mathrm{ggT}(a,b) = sa + tb$.
$r_0 := a$, $r_1 := b$, $i := 1$
$s_0 := 1$, $s_1 := 0$
$t_0 := 0$, $t_1 := 1$
while $r_i \neq 0$ do
$r_{i-1} =: q_i r_i + r_{i+1}$
$s_{i+1} := s_{i-1} - q_i s_i$
$t_{i+1} := t_{i-1} - q_i t_i$
return $s_i$, $t_i$

Es ist einfach zu sehen, dass immer $r_i = as_i + bt_i$ gilt. Der Rest folgt dann aus dem Beweis des ggT Algorithmus.

BEISPIEL 2.13: (1) $R = \mathbb{Z}$, $\mathrm{ggT}(2238, 168) = 6 = 2238 \cdot (-3) + 168 \cdot 40$
(2) $R = \mathbb{Q}[t]$, $\mathrm{ggT}(t^8 - 1, t^5 - 2t^4 + t - 2) = 15(t^4 + 1) = (t^8 - 1) \cdot 1 - (t^5 - 2t^4 + t - 2)(t^3 + 2t^2 + 4t + 8)$

BEMERKUNG 2.14: Seien $a = (a_n, \ldots, a_0)_B$ und $b = (b_m, \ldots, b_0)_B$, dann kann der $\mathrm{ggT}(a,b)$ in $O(mn)$ $B$-Operationen berechnet werden. Die selbe Aussage gilt auch für den erweiterten Algorithmus.

Der ggT von $f, g \in K[t]$ kann in $O(\deg f \deg g)$ $K$-Operationen berechnet werden.

BEWEIS. Jeder Durchlauf der Schleife führt eine Division durch:

$$r_{i-1} =: q_i r_i + r_{i+1}.$$

Wie wir gesehen haben, benötigt dies $O((\log_B r_i)(\log_B q_i))$ $B$-Operationen, also insgesamt

$$\sum \log_B r_i \log_B q_i \leq n \sum \log_B q_i = n \log_B(\prod q_i) \leq n \log_B a = nm$$

Das selbe Argument klappt für $K[t]$. Für die erweiterten Algorithmen werden zusätzlich zu der Division noch 2 Multiplikationen durchgeführt. Der Aufwand dort ist ähnlich zu dem der Division. □

## 3. ggT in $\mathbb{Q}[x]$, erster Versuch

Wir wollen jetzt mehr Ringe zulassen, bisher haben wir ja nur ggT in euklidischen Ringen, also speziell in $K[t]$ berechnet. In der Praxis will man aber auch in $\mathbb{Z}[t]$ arbeiten oder sogar in $\mathbb{Q}(x)[t]$. Wenn wir den euklidischen Algorithmus über $\mathbb{Q}[x]$ oder sogar über $\mathbb{Q}(x)[y]$ durchführen, passiert etwas hässliches: die Koeffizienten werden sehr, sehr groß: $f := x^8 + 8x^7 + 7x^6 + 6x^5 + 5x^4 + 4x^3 + 3x^2 + 2x + 1$ und $g := f'$, die Ableitung. Dann gilt

$$\begin{aligned} r_2 &= -21/4x^6 - 3x^5 - 5/4x^4 + 3/4x^2 + x + 3/4 \\ r_3 &= 1574/147x^5 + 870/49x^4 + 148/7x^3 + 3068/147x^2 + 830/49x + 458/49 \\ r_4 &= -213248/619369x^4 - 639744/619369x^3 - 1279488/619369x^2 \\ &\quad -2132480/619369x - 2620275/619369 \\ r_5 &= 3096845/106624x + 619369/3332 \\ r_6 &= -144653722563/387105625 \end{aligned}$$

Und damit ist der ggT = 1 - aber wir müssen mit sehr großen Zahlen rechnen.

Ziel ist es einen Algorithmus zu erhalten der diese sog. Koeffizientenexplosion vermeidet. Ferner wäre es auch schön auf die rationale Arithmetik verzichten zu können.

**Definition 2.15:** Sei $f = \sum f_i x^i \in \mathbb{Z}[x]$. Dann heißt $c(f) := \mathrm{ggT}(f_i : 0 \le i \le \deg(f))$ der Inhalt (content) von $f$. Ein Polynom $f$ mit $c(f) = 1$ heißt primitiv, $\tilde{f} := f/c(f)$ heißt der primitive Teil von $f$.

Über allgemeineren Ringen wird der Inhalt ein Ideal werden: $c(f) := \langle f_i : 0 \le i \le \deg f \rangle$, aber über $\mathbb{Z}$ (oder $\mathbb{Q}$) brauchen wir dies nicht.

**Bemerkung 2.16:** (1) Es gilt das Lemma von Gauss: Seien $f$ und $g$ primitiv, so ist auch $fg$ primitiv.
(2) $\mathrm{ggT}(f, g) = \mathrm{ggT}(c(f), c(g))\,\mathrm{ggT}(\tilde{f}, \tilde{g})$
(3) Division mit Rest über Ringen: $f$, $g \in R[t]$, $g \neq 0$, $\deg f =: m \ge \deg g =: n$, $b := l(g)$ ($l(g)$ ist der Leitkoeffizient). Angenommen $b^{m-n+1} | c(f)$, dann exisitieren $q$, $r \in R[t]$ mit $f = qg + r$ mit $r = 0$ oder $\deg r < \deg g$.

**Beweis.** (1) Standard Ergebnis, zB. Lorenz, Algebra I, siehe nächstes Lemma.
(2) Folgt aus (1)
(3) Per Induktion über $m$: Die Polynonmdivision von $f = l(f)x^m + f'$ durch $g = l(g)x^n + g'$ berechnet $\mu = l(f)/l(g)$ und setzt $f_1 := f - l(f)/l(g)x^{m-n}g$. Dann gilt $\deg f_1 < m$ und $b^{m-n} | c(f_1)$

□

**Lemma 2.17 (Gauß):** Seien $f$, $g \in \mathbb{Z}[t]$ Polynome, dann gilt $c(fg) = c(f)c(g)$.

**Beweis.** Wir zeigen dies in mehreren Schritten:

Ein Ideal $p \subset R$ heißt Primideal falls $R/p$ ein Integritätsring ist. Für $p = (p) \subset \mathbb{Z}$ und $p \in \mathbb{P}$ gilt $\mathbb{Z}/p\mathbb{Z} = \mathbb{F}_p$ ist ein Körper, also ein Integriätsring. Für $p$ Primideal ist dann $p[t]$ ein Primideal in $\mathbb{Z}[t]$, da $\mathbb{Z}[t]/p[t] \simeq (\mathbb{Z}/p)[t]$ gilt. Für $R$ Integritätsring ist $R[t]$ ebenfalls Nullteilerfrei.

Sei nun $p \in \mathbb{P}$ eine Primzahl. Wir setzen $w_p(f) := \min\{v_p(f_i) | f = \sum f_i t^i\}$ und zeigen $w_p(fg) = w_p(f) + w_p(g)$ woraus dann die Aussage folgt. ObdA gehen wir davon aus, dass $w_p(f) = w_p(g) = 0$ gilt (sonst kann durch $p$ geteilt werden!). Angenommen $w_p(fg) > 0$. Dann betrachten wir $fg \in (\mathbb{Z}/p\mathbb{Z})[t] =: S$. In dem Integritätsring $S$ gilt nun $fg = 0$, also $f = 0$ oder $g = 0$. Aber $f = 0$ in $S$ impliziert $v_p(f_i) > 0 \ \forall i$ also $w_p(f) > 0$. Analog für $g$ und damit haben wir einen Widerspruch. □

DEFINITION 2.18: Seien $f, g \in R[x]$, $\deg f =: m \geq \deg g =: n$, $b = l(g)$. Schreibe $b^{m-n+1} f = qg + r$ mit $\deg r < n$ oder $r = 0$. Dann heißt $q$ der Pseudoquotient und $r$ der Pseudorest von $f$ und $g$

Damit kann nun der euklidische Algorithmus mit Pseudoquotienten aufgezogen werden. Es ist klar, das dies einen ganzzahligen Algortihmus liefert. Wenn wir damit das Bsp vom Anfang nochmals durchrechnen bekommen wir:

$$\begin{aligned}
r_2 &= -336x^6 - 192x^5 - 80x^4 + 48x^2 + 64x + 48 \\
r_3 &= 1208832x^5 + 2004480x^4 + 2386944x^3 + 2356224x^2 + \ldots \\
r_4 &= -321993698181 12x^4 - 96598109454336x^3 - \ldots \\
r_5 &= 339967809482754034804056781750 2720x + \ldots \\
r_6 &\approx -46683476497246545242983637168 18 \cdot 10^{120}
\end{aligned}$$

Offensichtlich ist dies nicht so gut - die Zahlen sind viel zu gross.

Verbesserung: nur primitive Teile benutzen.

LEMMA 2.19: Seien $f$, $g$, $q$ und $r$ wie in Definition 2.18, so gilt

$$\mathrm{ggT}(f, g) = \mathrm{ggT}(g, \tilde{r})$$

BEWEIS. $\mathrm{ggT}(g, r) = \mathrm{ggT}(g, b^{m-n+1} f - qg) = \mathrm{ggT}(g, b^{m-n+1} f) = \mathrm{ggT}(c(g), b^{m-n+1} c(f)) \, \mathrm{ggT}(\tilde{g}, \tilde{f})$ Wenn $g$ primtiv war, so gilt $\mathrm{ggT}(c(g), b^{m-n+1} c(f)) = 1$. Ferner gilt $\mathrm{ggT}(g, r) = \mathrm{ggT}(c(g), c(r)) \, \mathrm{ggT}(\tilde{g}, \tilde{r})$, also für $f$, $g$ primitiv sofort $\mathrm{ggT}(g, \tilde{r}) = \mathrm{ggT}(f, g)$. □

Damit ändert sich unser Beispiel zu:

$$\begin{aligned}
\tilde{r}_2 &= -21x^6 - 12x^5 - 5x^4 + 3x^2 + 4x + 3 \\
\tilde{r}_3 &= 787x^5 + 1305x^4 + 1554x^3 + 1534x^2 + 1245x + 687 \\
\tilde{r}_4 &= -4352x^4 - 13056x^3 - 26112x^2 - 43520x - 53475 \\
\tilde{r}_5 &= 5x + 32 \\
\tilde{r}_6 &= 1
\end{aligned}$$

Was natürlich viel besser ist. Andererseits haben wir jetzt viele ggT Berechnungen und können nur in euklidischen Ringen arbeiten. Es gibt auch „Kompromisse“, also Algorithmen die in beliebigen Ringen arbeiten und „zwischen“ Pseudoquotienten und primitiven Resten arbeiten. Dort werden in jedem Schritt Divisionen durchgeführt die aufgehen müssen. Die Zahlen sind nicht ganz so klein wie bei dem primitive Algorihmus, aber viel kleiner als die Pseudoquotienten-Methode. Diese heißen z.B. Subresultanten Algorithmen.

## 4. Modulare Algorithmen

Seien $R$, $S$ Ringe (wir werden sie später einschränken), $\phi : R \to S$ ein Ringhomomorphismus. Dieser induziert einen Homomorphismus von Polynomen:

$$\phi : R[t] \to S[t] : f = \sum f_i t^i \mapsto \sum \phi(f_i) t^i.$$

Offensichtlich gilt $\deg \phi(f) \leq \deg f$ wobei $<$ genau dann gilt, falls $l(f) \in \ker \phi$ gilt.

Wir wollen diesen Homomorphismus nutzen um die ggT Berechnung in $R[t]$, wo sie schwierig ist, nach $S[t]$ zurückführen, wo es hoffentlich besser geht.

LEMMA 2.20: Seien $0 \neq f, g \in R[t]$ und $h := \mathrm{ggT}(f, g)$. Dann gilt

(1) Ist $\phi(f)$ irreduzibel und $\deg(f) = \deg(\phi(f))$, so ist $f$ irreduzibel
(2) Gilt $\deg \phi(f) = \deg f$ so folgt $\deg \phi(h) = \deg h$
(3) Gilt $\deg \phi(h) = \deg h$ so folgt $\deg \mathrm{ggT}(\phi(f), \phi(g)) \geq \deg h$
(4) Sei $f$ primitiv und $\deg \phi(h) = \deg h$. Ist $k \in R[t]$ mit $k|f$, $k|g$ und $\phi(k) = \mathrm{ggT}(\phi(f), \phi(g))$, so gilt $k = \mathrm{ggT}(f, g)$

BEWEIS. (1) Ang. $f = ab$ mit $\deg a, \deg b > 0$, so folgt $\phi(f) = \phi(a)\phi(b)$. Wegen $\deg f = \deg \phi(f)$ und $\deg a \geq \deg \phi(a)$ folgt sofort auch $\deg \phi(a) = \deg a > 0$ und, analog, $\deg \phi(b) = \deg b > 0$. Also $\phi(f)$ nicht irreduzibel.
(2) Sei $f = qh$ und $\deg f = \deg \phi(f)$, so folgt wie in (1) sofort $\deg \phi(h) = \deg h$.
(3) Wegen $h|f, h|g$ folgt sofort $\phi(h)|\phi(f)$ und $\phi(h)|\phi(g)$. Also auch $\phi(h)|\,\mathrm{ggT}(\phi(f), \phi(g))$ und damit $\deg h = \deg \phi(h) \leq \deg \mathrm{ggT}(\phi(f), \phi(g))$.
(4) Wie oben folgt sofort $\deg k \leq \deg \mathrm{ggT}(f, g)$ und $\deg \phi(k) = \deg k$. Damit folgt die Aussage über den ggT aus der Primitivität ($f$ primitiv $\Rightarrow k$ primitiv und ggT primitiv)

□

Für $R = \mathbb{Z}$ wollen wir $S = \mathbb{Z}/p\mathbb{Z} = \mathbb{F}_p$ benutzen. Hier schreiben wir $\phi_p : \mathbb{Z} \to \mathbb{F}_p$ für die kanonische Abbildung. Nehmen wir wieder $f = \sum_{i=1}^{8} i t^{i-1} + t^8$ und $f'$ wie oben. Dann gilt (via Computer):

$$\begin{aligned} \mathrm{ggT}(\phi_2(f), \phi_2(f')) &= t^8 + t^6 + t^4 + t^2 + 1 \\ \mathrm{ggT}(\phi_3(f), \phi_3(f')) &= t + 1 \\ \mathrm{ggT}(\phi_5(f), \phi_5(f')) &= 1 \end{aligned}$$

Also wissen wir sofort (von $\phi_5$), dass $f$ und $f'$ Teilerfremd sind - und $f$ quadratfrei. Ebenso sehen wir, dass $\phi_3$, $\phi_2$ nicht klappen.

DEFINITION 2.21: Seien $f, g \in R[t]$, $g \neq 0$, $\deg f \geq \deg g$. Eine Polynomrestfolge (polynomial remainder sequence) ist eine Folge $r_0 = f$, $r_1 = g$, $r_i \in R[t]$ so, dass es $\alpha_i$, $\beta_i \in R$ und $q_i \in R[t]$ gibt mit $\alpha_i r_{i-1} = q_i r_i + \beta_i r_{i+1}$ $1 \leq i \leq l - 1$. Ferner muss der Pseudorest von $r_{l-1}$ und $r_l$ gleich 0 sein.

Beispiele hierfür haben wir bereits gesehen: die normale euklidische Division in $\mathbb{Q}[t]$ liefert eine Restfolge, die Pseudodivision eine andere.

LEMMA 2.22: Sei $(r_i)_{0 \leq i \leq l}$ eine Polynomrestfolge für $f$ und $g$. Dann gilt $\tilde{r}_l = \mathrm{ggT}(\tilde{f}, \tilde{g})$

BEWEIS. Nach Definition der Polynomrestfolge gilt $\alpha_i r_{i-1} = q_i r_i + \beta_i r_{i+1}$ und daher $\mathrm{ggT}(\tilde{r}_i, \tilde{r}_i - 1) = \mathrm{ggT}(\tilde{r}_i, \tilde{r}_{i+1})$. □

SATZ 2.23: Seien $f$, $g \in \mathbb{Z}[t]$, $h := \mathrm{ggT}(f, g)$. Dann ist $\phi_p(h) = \mathrm{ggT}(\phi_p(f), \phi_p(g))$ für fast alle (also bis auf endlich viele Ausnahmen) $p \in \mathbb{P}$.

BEWEIS. Ist $f = 0$ oder $g = 0$ so ist die Aussage trivial. Sei nun $f$ primitiv, $f$, $g \neq 0$ und $(r_i)_i$ eine Polynomrestfolge für $f$ und $g$. Ist nun $p \in \mathbb{P}$ mit $p \not| l(r_i)$ für $0 \leq i < l$, so ist $\phi_p(r_i)$ eine Polynomrestfolge für $\phi_p(f)$, $\phi_p(g)$ da die Grade gleich sind. Also ist $\tilde{\phi}_p(r_l) = \mathrm{ggT}(\phi_p(f), \phi_p(g))$ nach Lemma 2.20.

Wenn nun $f$ oder $g$ nicht primitiv sind, so wende den Satz zunächst auf $\tilde{f}$ und $\tilde{g}$ an und benutze $\mathrm{ggT}(f, g) = \mathrm{ggT}(c(f), c(g))\,\mathrm{ggT}(\tilde{f}, \tilde{g})$. Für $p \in \mathbb{P}$ mit $p \not| l(f)l(g)$ und $p \not| l(r_i)$ folgt wieder $\phi_p(\tilde{h}) = \mathrm{ggT}(\phi_p(\tilde{f}), \phi_p(\tilde{g}))$. Aus $\phi_p(h) = \phi_p(c(h))\phi_p(\tilde{h})$ folgt dann die Behauptung. □

Um dies jetzt schön anwenden zu können, müssen wir noch Aussagen über die Größe der Koeffizienten machen.

DEFINITION 2.24: Sei $f := \sum f_i t^i \in \mathbb{C}[t]$ beliebig. Dann definieren wir $\|f\|_2 := (\sum |f_i^2|)^{1/2}$ und $\|f\| := \|f\|_\infty = \max\{|f_i|\}$ als die Norm von $f$.

SATZ 2.25: Seien $h|f \in \mathbb{C}[t]$ dann gilt

$$\|h\| \leq 2^{\deg h} |\frac{l(h)}{l(f)}| \|f\|_2$$

BEWEIS. Übung! □

KOROLLAR 2.26: Für $h|f \in \mathbb{Z}[t]$ gilt $\|h\| \leq 2^{\deg h} \|f\|_2$

BEWEIS. Aus $h|f$ folgt sofort $l(h)|l(f)$. Daher ist der Quotient $\leq 1$ und kann weggelassen werden. □

Damit gilt auch $\|\mathrm{ggT}(f, g)\| \leq \min(2^{\deg f}\|f\|_2, 2^{\deg g}\|g\|_2)$.

SATZ 2.27: Seien $f$, $g \in \mathbb{Z}[t]$ primitiv, $h := \mathrm{ggT}(f, g)$ sowie $a = l(f)$, $b = l(g)$ $c = \mathrm{ggT}(a, b)$. Ferner sei $M \subset \mathbb{P}$ endlich mit

(1) $\phi_p(h) = \mathrm{ggT}(\phi_p(f), \phi_p(g))$ für $p \in M$
(2) $\phi_p(c) \neq 0$ für $p \in M$
(3) $q := \prod_{p \in M} p > 2|c|\|h\|$

Ist nun $k \in \mathbb{Z}[t]$ mit

(4) $\phi_p(k) = \mathrm{ggT}(\phi_p(f), \phi_p(g))$
(5) $\phi_p(l(k)) = \phi_p(c)$
(6) $\|k\| < q/2$

So gilt $\tilde{k} = \pm h$

BEWEIS. Sei $d = l(h)$, dann gilt $d|c$. Setze $c' := c/d$, $h' := c'h$, so folgt $l(h') = c$. Modulo Multiplikation mit Elementen in $\mathbb{F}_p^*$ gilt nun für $p \in M$: $\phi_p(k) = \operatorname{ggT}(\phi_p(f), \phi_p(g)) = \phi_p(h')$, also exisitiert ein $s_p \in \mathbb{F}_p^*$ mit $\phi_p(k) = s_p\phi_p(h')$. Wegen $0 \neq \phi_p(c) = \phi_p(l(k))$ gilt nun $l(\phi_p(k)) = \phi_p(c) = l(\phi_p(h'))$ und sogar $\phi_p(k) = \phi_p(h')$. Wegen $\|h'\| = |c'|\|h\| \leq |c|\|h\| < q/2$ und $\|k\| < q/2$ folgt dann $\|h' - k\| \leq \|k\| + \|h'\| < q$ also $k = h'$. (Nach dem Chinesischen Restsatz folgt für die Koeffizienten von $k$ und $h'$: $k_i - h'_i \equiv 0 \bmod q$, also sind alle nicht-Null Koeffizienten von $k - h'$ mindestens $q$) Somit $\tilde{k} = \pm h$ da $h$ primitiv ist. □

Damit können wir jetzt einen modularen ggT Algorithmus angeben. Da dieser etwas komplizierter ist, machen wir es informal, angefangen mit einem Beispiel: Sei $f := x^7+12x^6+12x^5+123x^4+56x^3+36x^2+360x+60$ und $g := x^6+12x^5+2x^4+3x^2+36x+6$.

Also gilt $\|\operatorname{ggT}(f,g)\| \leq \min(2^7 \cdot 392, 2^6 \cdot 39) = 2496$. Nun berechnen wir $\operatorname{ggT}(\phi_p(f), \phi_p(g))$ für verschiedene $p$ (und kombinieren Ergebnisse mit dem Chinesischen Restsatz)

$$\begin{aligned}
p = 2 &\qquad x^3 + x^2 \\
p = 3 &\qquad x^5 + 2x^3 \\
p = 5 &\qquad x^2 + 2x + 2 \\
p = 7 &\qquad x^2 + 5x + 2 \\
5 \cdot 7 = 35 &\qquad x^2 + 12x + 2 \\
p = 11 &\qquad x^2 + x + 2 \\
5 \cdot 7 \cdot 11 = 385 &\qquad x^2 + 12x + 2 \\
p = 13 &\qquad x^2 + 12x + 2 \\
5 \cdot 7 \cdot 11 \cdot 13 = 5005 &\qquad x^2 + 12x + 2
\end{aligned}$$

Wie wir sehen, brauchen wir eigentlich weder $p = 11$ noch $p = 13$, da wir modulo $5 \cdot 7$ bereits das korrekte Ergebnis hatten. Nun versuchen wir einen Algorithmus zu formulieren:

ALGORITHMUS 2.28: Modularer ggT von normierten Polynomen in $\mathbb{Z}[t]$

Input: $f$, $g \in \mathbb{Z}[t]$ normiert.
Output: $h = \operatorname{ggT}(f,g)$
- $p := 1$, $m := 1$
- $B := \min(2^{\deg f}\|f\|_2, 2^{\deg g}\|g\|_2)$
- while true do
- $p :=$NextPrime$(p)$
- $h := \operatorname{ggT}(\phi_p(f), \phi_p(g))$
- Falls $m = 1$ oder $\deg h < \deg k$ dann $k := h$, $m := p$
- Falls $\deg h > \deg k$ dann nächste Primzahl.
- Mit dem Chinesischen Restsatz kombiniere $h \bmod m$ und $k \bmod p$ um $h \bmod pm$ zu erhalten. Setze $m := mp$.
- Falls $m > 2B$ gilt und $h|f$, $h|g$ dann gib $h$ zurück.

Im Prinzip brauchen wir die Schranke gar nicht: Im letzten Schritt testen wir die Teilbarkeit. Wenn sie erfüllt ist sind wir fertig, sonst nicht.

Falls $f$, $g$ nicht normiert sind, so gibt es mehrere Möglichkeiten:

(1) Wir können, wie in dem Satz, die Leitkoeffizienten einbauen
(2) Wir können die Polynome normieren und die Transformation dann für den ggT wieder rückgängig machen.

BEMERKUNG 2.29: Sei $f = \sum f_i t^i$ primitiv vom Grad $n := \deg f$. Dann ist $f(t/(cf_n))f_n^{n-1}c^n$ normiert. (Die Division die hier gefordert ist ist exakt und geht auf)

Weiters Problem: Die Teilbarkeit zu testen ist nicht preiswert - speziell wenn $h$ noch falsch ist. In der Praxis optimiert man:

(1) Teste die Teilbarkeit nur wenn $h$ sich bei $2 \dots 5$ Primzahlen nach dem Chinesischen Restsatz nicht mehr ändert
(2) Teste ob die konstanten Koeffizienten sich teilen
(3) Teste ob die Norm $\|h\|$ nicht zu Groß ist
(4) Teste erst wenn $m > C$ gilt wobei $C$ kleiner als $B$ gewählt wird (wie wir im Beispiel gesehen haben ist $B$ viel zu Groß!)
(5) Statt $p = 2$ nimmt man $p > 2^{20}$ oder noch größer. $p$ wird als größte Primzahl gewählt mit der noch schnell (single precision) gerechnet werden kann. Statt NextPrime benutzt man PreviousPrime.

Achtung: Das Ganze funktioniert natürlich nur wenn wir das symmetrische Restsystem für $\mathbb{Z}/m\mathbb{Z}$ benutzen - sonst können wir nie negative Koeffizienten finden.

Für Polynome über $\mathbb{Q}$ gibt es mehrere Möglichkeiten:

(1) Wir können sie einfach mit dem Hauptnenner skalieren und dann den $\mathbb{Z}[x]$ Algorithmus benutzen.
(2) Wir können auch direkt mit $\mathbb{Q}$ und $\phi_p$ arbeiten. Hier dürfen die Primzahlen keine Nenner der Koeffizienten teilen. Zusätzlich brauchen wir noch Techniken um von $x \in \mathbb{Z}/m\mathbb{Z}$ wieder nach $x \in \mathbb{Q}$ zurück zu finden (die gibt es)

Andere Ringe: Um diesen Algorithmus (diese Idee) zu benutzen brauchen wir Ringe $S$ die wir statt $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ benutzen können. Z.B. geht das direkt auch in $K[x](t)$ also für Polynome wo die Koeffizienten bereits Polynome sind. Falls $K = \mathbb{Q}$ gilt so gibt es natürlich auch wieder die Koeffizientenexplosion, aber für $K = \mathbb{F}_q[x]$ klappt dies sehr gut.

Varianten werden auch für Zahlkörper benutzt, hier gilt $R = (\mathbb{Z}[x]/f)[t]$. Die Ringe $\mathbb{Z}/p\mathbb{Z}$ sind etwas komplizierter, aber auch hier können wir $\operatorname{ggT}(f, g) \bmod p$ berechnen und den Algorithmus wie oben verwenden.

Zum Abschluss wollen wir uns noch die Komplexität ansehen, zumindest so ungefähr:

SATZ 2.30: Der Algorithmus 2.28 für $\deg f \geq \deg g$ hat eine Laufzeit von

$$O(\deg^4 f \max^3(\|f\|, \|g\|))$$

BEWEIS. ObdA: $\deg f \geq \deg g$. Wir haben $\log B = O(\deg f \log \max(\|f\|, \|g\|))$. Die äußere Schleife wird maximal $\log_2 B$ mal durchlaufen (da alle Primzahlen $\geq 2$ sind). In jedem Durchlauf haben wir einen ggT in $\mathbb{F}_p$, also $O(\deg f \deg g)$ viele $\mathbb{F}_p$ Operationen. Da hier $p$ durch eine Konstante beschränkt ist, nehmen wir diese Operationen als konstant an. Zusätzlich haben wir noch den Chinesischen Restsatz mit 2 Moduln. Dieser benötigt 2 modulare Invertierungen, 5 Produkte und 1 Summe, also ist der Aufwand hier $O(\log^2 m) = O(\log^2 B)$ (klassische Schulmultiplikation). Die Probedivisionen brauchen $O((\deg f - \deg h) \deg h)$ viele $\mathbb{Z}$-Operationen mit Zahlen der Größe $B$, also $O(\deg f^2 \log^2 B)$. Wenn wir jetzt alles zusammensetzen, erhalten wir:

$$O(\deg f \max(\|f\|, \|g\|)(\deg^2 f + \deg f(\deg^2 f \max(\|f\|, \|g\|))^2)) = O(\deg^4 f \max^3(\|f\|, \|g\|))$$

□

(Was wir im Beweis auch sehen ist das die Probedivisionen sehr teuer sind. Ohne diese erhalten wir $O(\deg^3 f \max^3(\|f\|, \|g\|))$). Mit schneller Multiplikation werden die Exponenten sofort kleiner.

KAPITEL 3

# Gleichungssysteme

Ein Grundproblem der Algebra ist es Gleichungssysteme zu lösen - ein großer Teil der modernen Algebra ist entstanden, um Polynomgleichungen zu verstehen. Daher ist es nicht überraschend, dass wir uns in der praktischen Mathematik ebenfalls damit auseinandersetzen müssen.

In der linearen Algebra wurden lineare Gleichungssystem über Körpern behandelt. Ein zentrales Ergebnis ist der Gauß Algorithmus und das Verständnis der Vektorraumstruktur. Obwohl dies klassische Ergebnisse sind, so ist selbst das Lösen von linearen Gleichungen über $\mathbb{Q}$ immer noch aktuelles Forschungsgebiet.

Ähnlich wie bei den Polynomen kommt es auch hier zu einer Koeffizientenexplosion die wir zunächst durch den Übergang zu $\mathbb{Z}$ vermeiden wollen. Hier haben wir dann wieder das Problem, dass es keine $\mathbb{Z}$-Vektorräume gibt - ähnlich wie das iProblem, dass $\mathbb{Z}[t]$ nicht euklidisch ist.

## 1. Moduln

Im weiteren sei $R$ wieder kommutativ mit 1.

DEFINITION 3.1: Eine abelsche Gruppe $M$ heißt $R$-Modul, falls es eine bilineare Abbildung von $R \times M \to M : (r, m) \mapsto rm$ gibt mit $(rs)m = r(sm)$ und $1m = m$ für $r$, $s \in R$ und $m \in M$.
Eine Teilmenge $B \subset M$ heißt frei (linear unabhängig), falls aus $\sum r_i b_i = 0$ stets $r_i = 0$ folgt für jede endliche Summe.
Eine freie Teilmenge $B$ heißt Basis von $M$ falls jedes $m \in M$ als endliche Linearkombination dargestellt werden kann: $m = \sum r_i b_i$.
Ein Modul mit Basis heißt frei.

BEISPIEL 3.2:
(1) Jede abelsche Gruppe ist ein $\mathbb{Z}$-Modul.
(2) $\mathbb{Z}/N\mathbb{Z}$ ist nicht frei für $N > 1$ da es keine freie Teilmenge geben kann: $Nm = 0$ für jedes $m$.
(3) $R^n$ ist frei mit Basis $\{e_i\}$.
(4) Ist $M$ frei mit endlicher Basis $B$, $n := |B|$, so gilt $M \sim R^n$. $n$ heißt auch der Rang $\mathrm{rk}(M)$ von $M$.
(5) Sei $K$ ein Körper, dann ist jeder $K$-Vektorraum ein freier $K$-Modul.

SATZ 3.3: Sei $R$ ein Hauptidealring, $M$ ein freier Modul mit endlicher Basis und $V < M$ ein Teilmodul. Dann ist $V$ ebenfalls frei vom Rang $\leq n$.

BEWEIS. Per Induktion über $n$ =Rang$(M)$. $n = 1$: Dann gilt $B = \{b_1\}$ und jedes Element von $M$ (und $V$) ist ein Vielfaches von $b_1$. Betrachte

$$A := \{r \in R | rb_1 \in V\}$$

Da $V$ ein $R$-Modul ist, so ist $A < R$ ein Ideal. Da $R$ Hauptidealring ist, so gibt es ein $a \in R$ mit $A = (a)$. Damit folgt dann $V = ab_1R$.

Sei nun $n > 1$ und $B = \{b_1, \ldots, b_n\}$ eine Basis von $M$. Dann definieren wir die Abbildung

$$\phi : M \to R : \sum r_i b_i \to r_n$$

Diese Abbildung ist $R$-linear und $\phi(V)$ wieder, wie oben, ein Ideal in $R$, also gilt $\phi(V) = (a_n)$. Wenn wir jetzt

$$\psi : M \to M : m \mapsto m - \phi(m)b_n$$

betrachten, so ist $\psi(M)$ offenbar frei vom Rang $n-1$ mit Basis $\{b_1, \ldots, b_{n-1}\}$. Nach Induktionsvoraussetzung ist nun $\tilde{V} := V \cap \psi(M)$ frei mit Basis $\{v_1, \ldots, v_l\}$ $l < n$. Offenbar gilt $v = \psi(v) + \phi(v)b_n$ also ist $\{v_1, \ldots, v_l, a_n b_n\}$ eine Basis von $V$. □

BEISPIEL 3.4: (1) $M = \mathbb{Z}^3$, $U = \langle (1,1,0), (1,0,1), (0,1,1) \rangle$. Dann ist $U$ frei vom Rang 3 aber $U \neq \mathbb{Z}^3$ was bei Vektorräumen nicht passieren kann! ($U$ hat nur Elemente bei denen die Summe der Koorpdinaten gerade ist)

(2) $M = \mathbb{Z}$, $U = \langle 2, 3 \rangle$ ist frei ($U = M$) aber keine freie Teilmenge des Erzeugendensystems ist eine Basis.

DEFINITION 3.5: Sei $S \in R^{m \times n}$, dann heißt $\ker A := \{x \in R^n | Ax = 0\}$ der Nullraum von $A$. Der Spaltenraum ist definiert als $\text{bild } A := \{Ax | x \in \mathbb{R}^n\} \subseteq R^m$.

Klar ist, dass bild $A$ und $\ker A$ Untermoduln (Teilmoduln) sind. Damit ist auch klar, dass sie frei sind - falls $R$ ein Hauptidealring ist. Im Prinzip ist damit die Lösbarkeit von linearen Gleichungssystemen klar: $\ker A$ beschreibt das homogene System, $Ax = b$ ist lösbar, genau dann wenn $b \in \text{bild } A$ gilt. Ist $c$ eine Lösung, so sind alle anderen Lösungen von der Form $c + k$ mit $k \in \ker A$ - wie bei Vektorräumen. Was noch fehlt ist der Gauß-Algorithmus um eine Basis für $\ker A$ und bild $A$ zu finden.

## 2. Hermite Form

DEFINITION 3.6: Sei $R$ euklidisch und $v : R \to \mathbb{N}$ die zugehörige euklidische Funktion. $A = (a_{i,j})_{i,j} \in R^{m \times n}$ ist in Hermite Normalform (HNF) (oder auch Hermite Spaltenreduziert) wenn gilt:

(1) Es exisitiert $0 \leq r \leq n$ mit $a_{i,j} = 0$ für $j > r$ und $0 < i \leq m$
(2) Für $1 \leq j \leq r$ sei $f(j) := \min\{i : a_{i,j} \neq 0\}$. Dann ist $f$ als Funktion $f : \{1, \ldots, r\} \to \{1, \ldots, m\}$ streng monoton wachsend.
(3) Für $1 \leq j \leq r$ gilt $v(a_{f(j),k}) < v(a_{f(j),j})$ für $k < j$.

Also ist $A$ in Zeilen-Stufen-Form, eine untere Dreiecksmatrix.
Für $R = \mathbb{Z}$ verlangt man noch $a_{f(j),j} > 0$. Im Allgemeinen kann die „Diagonale“ modulo Einheiten geändert werden.

Es gibt Varianten der HNF wo die Matrix in eine obere Dreiecksmatrix transformiert wird oder die Nullspalten links statt rechts stehen. Natürlich gibt es auch Zeilen HNFs.

Beispiel 3.7:

$$\begin{pmatrix} 5 & 0 & 0 & 0 \\ -4 & 9 & 0 & 0 \\ 1 & 2 & 0 & 0 \end{pmatrix}$$

Wie wir sehen, hängt die HNF auch von der euklidischen Division ab.

Nun brauchen wir einen Algorithmus um $A$ in HNF zu überführen. Aber studieren wir zunächst einmal die Eigenschaften:

Bemerkung 3.8: Sei $R$ euklidisch und $A = (a_{i,j}) \in R^{m\times n}$ in HNF, wobei die Spalten $1, \ldots, r$ nicht identisch Null sind und die Spalten $r+1, \ldots, n$ identisch Null sind. Dann gilt:

(1) Die nicht-Null Spalten von $A$ bilden eine Basis für $\operatorname{bild} A$
(2) Die Standard Basis Vektoren $e_{r+1}, \ldots, e_n$ bilden eine Basis für $\ker A$.

Beides ist eine direkte Folgerung aus der Zeilen-Stufen-Form von $A$.

Bemerkung 3.9: Sei $R$ euklidisch und $a, b \in R$. Dann gibt es $s, t \in R$ mit $g := \operatorname{ggT}(a, b) = sa + tb$. Damit folgt dann

$$\det \begin{pmatrix} s & -b/g \\ t & a/g \end{pmatrix} = 1$$

und

$$(a, b) \begin{pmatrix} s & -b/g \\ t & a/g \end{pmatrix} = (g, 0)$$

Satz 3.10: Sei $R$ euklidisch, $A \in R^{m\times n}$. Dann gilt

(1) Es gibt ein $U \in \operatorname{Gl}(n, R)$ (also $\det U \in R^*$), so dass $B := AU$ in HNF ist.
(2) Es gibt ein $U$ wie in (1), welches ein Produkt von Elementarmatrizen der Form

$E_{i,j}(a) := I_n + \delta_{i,j}a, \quad P_{i,j} := I_n - \delta_{i,i} - \delta_{j,j} + \delta_{i,j} + \delta_{j,i}$ und $S_i(u) := I_n + (u - \delta_{i,i})$

für $a \in R$ und $u \in R^*$ ist. Also kann $A$ durch elementare Transformationen in HNF überführt werden.

Für $R = \mathbb{Z}$ oder $R = K[t]$, $K$ ein Körper, ist die HNF eindeutig (modulo euklidischer Division): Ist $B = AU$ eine HNF von $A$ und $C = AV$ eine weitere HNF, so folgt $B = C$.

Beweis. Wir zeigen (2) per Induktion über $n$: $n = 1$, $A = (a)$ ist in HNF (für (3) muss noch mit Einheiten skaliert werden)

Sei nun $n > 1$. ObdA ist die 1. Zeile von $A$ nicht identisch Null, sonst kann sie (im Prinzip) weggelassen werden. Durch geeignete Spaltenvertauschungen erreichen wir $a_{1,1} \neq 0$.

Für $j := 2$ bis $n$ tun wir nun das folgende: Falls $a_{1,1} | a_{1,j}$ so multiplizieren $A$ mit $E_{1,j}(-a_{1,j}/a_{1,1})$ (ziehe das $a_{1,j}/a_{1,1}$-fache der 1ten Spalte von der $j$ten ab) um $a_{1,j} = 0$ zu erreichen.

Sonst multipliziere mit einer Variante von Bemerkung 3.9 um $a_{1,1} = \operatorname{ggT}(a_{1,1}, a_{1,j})$ und $a_{1,j} = 0$ zu erreichen.

Zum Schluss gibt es dann in der 1. Zeile nur einen von 0 verschiedenen Eintrag.

Per Induktion kann nun die Teilmatrix, die durch das Weglassen der 1. Zeile und 1. Spalte entsteht, in HNF überführt werden.

Schließlich muss noch Definition 3.6.(2) erreicht werden. Aber dies folgt mit einer euklidischen Division um $a_{f(j),1}$ modulo $a_{f(j),j}$ zu reduzieren.

Um (2) zu zeigen muss noch die Matrix aus Bemerkung 3.9 in Elementarmatrizen zerlegt werden. Aber dies ist eine direkte Folge aus dem erweiterten ggT Algorithmus. (Übung)

Die Eindeutigkeit für $\mathbb{Z}$ und $K[t]$ folgt aus unserer Normierung des ggT und einem fixierten Divisionsalgorithmus (fixes Restsystem). □

Beispiel 3.11:

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix} \to \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 \\ 1 & 8 & 0 & 0 \end{pmatrix}$$

Bemerkung 3.12: Sei $R$ euklidisch, $A \in R^{m\times n}$ und $U \in \mathrm{Gl}(n, R)$ mit $B := AU$ ist in HNF, die Spalten $1, \ldots, r$ von $B$ seien $\neq 0$, die lezten seien 0. Dann gilt:

(1) Die Spalten $1, \ldots, r$ von $B$ bilden eine Basis für $\operatorname{bild} A$
(2) Die Spalten $r+1, \ldots, n$ von $U$ bilden eine Basis von $\ker A$
(3) Gilt $n = m$, so folgt $\det A = u \prod b_{i,i}$ für $u \in R^*$

Beweis. Direkte Folge von $U \in \mathrm{Gl}(n, R)$ □

Beispiel 3.13: Oben gilt $\operatorname{bild} A = \langle (1, 1, 1)^t, (0, 4, 8)^t \rangle$

Mit der HNF können dann lineare Gleichungssysteme über $\mathbb{Z}$ gelöst werden. Das Problem hier ist das unser Algorithmus eine starke Koeffizientenexplosion zeigt. Ohne es Formal zu beweisen, können wir trozdem sehen warum: Wir fangen an mit einer Matrix $A$, bei der alle Einträge die Größe $M$ haben (oder kleiner sind). Dann berechnen wir den ggT mit Darstellung von zwei Einträgen. Nach dem Übungsblatt gilt $\mathrm{ggT}(a, b) = sa + tb$ mit $s = O(b)$, $t = O(a)$, also haben $s$, $t$ die Größe $M$. Dann multiplizieren wir mit der ggT-Matrix. Die Elemente der 2. Spalte haben nun die Größe $2M$ (bis auf den 1. Eintrag der ist 0). Die 1. Spalte ist jetzt auch von der Größe $2M$ (bis auf den 1. Eintrag der ist (wahrscheinlich) 1).

Jetzt geht es zur 3. Spalte, hier wird warscheinlich kein ggT mehr berechnet, also wird nur ein $M$-Faches der 1. Spalte abgezogen. Die 3. Spalte hat daher eine Größe von $3M$. Analog haben alle weiteren ebenfalls die Größe $3M$.

Jetzt geht es in die 2. Zeile, hier fangen wir bereits mit eine Matrix an wo alle Einträge die Größe $2M$ bzw. $3M$ haben. Die analoge Prozedur ändert die 2. und 3. Spalte erst auf $(2 + 3)M = 5M$ und dann alle Weiteren auf $(5 + 3)M = 8M$. Die 3. Zeile erhält dann $(5 + 8)M$ und $(5 + 8 + 8)M$. Induktiv sehen wir das die Zahlen in jedem Schritt mehr als doppelt so groß werden, also exponentiell wachsen!

Die ist natürlich kein Beweis, erklärt jedoch die Beobachtung das der Algorithmus unter extremer Koeffizientenexplosion leidet.

Hier gibt es nun zwei Lösungen die eingesetzt werden. Die offensichtliche Idee modulo $p$ zu rechnen und den Chinesischen Restsatz zu benutzen geht leider nicht. Aber es gibt andere Ideen.

BEMERKUNG 3.14: Sei $A \in \mathbb{Z}^{n\times n}$ mit $d := \det A \neq 0$ und $d|D$ beliebig. Dann gilt $(\mathrm{HNF}(A)|0) = \mathrm{HNF}(A|DI_n)$. Insbesondere können in dem Algorithmus im Beweis von Satz 3.10 alle Zahlen modulo $D$ reduziert werden.

BEWEIS. Sei $B := \mathrm{HNF}(A)$ mit $B = (b_{i,j})_{i,j}$ dann gilt $d = \prod b_{i,i}$. Wir wollen zeigen, dass $De_i \in \mathrm{bild}\, A$ gilt. Wegen $b_{1,1}|D$ ist $c_1 := D/b_{1,1} \in \mathbb{Z}$, dann gilt $De_1 - c_1(b_{i,1})_i = -(0, c_1b_{2,1}, \ldots)^t$. Da $b_{2,2}|c_1$ gilt, so folgt mit $c_2 := c_1b_{2,1}/b_{2,2}$ dann $De1 - c_1(b_{i,1})_i - c_2(b_{i,2})_i = -(0, 0, c_1b_{3,1} + c_2b_{3,2}, \ldots)^t$ Induktiv folgt damit $De_1 \in \mathrm{bild}\, A$. Analog ebenso für $De_i$. Da die HNF eine eindeutige Modulbasis berechnet, folgt so die Behauptung. □

Achtung: es gibt zwei verschiedene Varianten modularer HNFs:

(1) Die in Bemerkung 3.14 beschrieben wird
(2) Eine direkte Berechnung über $\mathbb{Z}/D\mathbb{Z}$ (wenn wir akzeptieren, dass $\mathbb{Z}/D\mathbb{Z}$ im Prinzip euklidisch ist (Übung)).

Obwohl diese in einem engen Zusammenhang stehen, sind sie doch verschieden. Z.B.: für $A = (D)$ folgt $\mathrm{HNF}(D|D) = (D|0)$ aber, in $\mathbb{Z}/D\mathbb{Z}$ gilt $\mathrm{HNF}(D) = (0)$. Dies ist wichtig weil der Algorithmus implizit in Bemerkung 3.14 in $\mathbb{Z}$ arbeitet und daher, je nach Implementierung, sehr wohl Zahlen $> D$ benutzt! In einem typischen HNF Algorithmus wird die HNF von $(A|DI_n)$ dadurch ausgerechnet, dass zuerst $\mathrm{HNF}(A)$ berechnet wird, und dann $DI_n$ modulo der HNF reduziert wird. Andererseits, in $\mathbb{Z}/D\mathbb{Z}$ sind alle Zahlen offensichtlich beschränkt. Was hier noch gezeigt werden muss ist:

BEMERKUNG 3.15: Sei $A \in \mathbb{Z}^{n\times n}$ mit $d := \det A \neq 0$ und $d|D$ beliebig. Ferner sei $\tilde{B} \in \mathbb{Z}^{n\times n}$ mit $\tilde{B} \bmod D\mathbb{Z} = \mathrm{HNF}(A \bmod D\mathbb{Z})$. Dann gilt $\mathrm{HNF}(A) = \mathrm{HNF}(\tilde{B}|DI_n)$ und die letzte, unmodulare HNF kann einfach berechnet werden.

## 3. $p$-adische Approximation

Sei $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^m$. Wir suchen $x \in \mathbb{Z}^n$ mit $Ax = b$. Wir können dies natürlich mit der HNF lösen - haben jedoch das Problem, dass Koeffizientenexplosion auftreten wird. Die modulare HNF können wir aus zwei Gründen nicht benutzen:

(1) Wir kennen die Determinante nicht.
(2) Wir brauchen die Transformationsmatrix.

Die Determinante könnten wir sogar noch berechnen, aber dann? Anderer Ansatz, motiviert von der Numerik.

Sei $p \in \mathbb{P}$ eine Primzahl und $\bar{\cdot} : \mathbb{Z} \to \mathbb{Z}/p\mathbb{Z} = \mathbb{F}_p$ die Restklassenabbildung. Dann schreiben wir $\bar{A} = (\bar{a}_{i,j})$ und $\bar{b} = (\bar{b}_i)$ für die Matrizen modulo $p$. Da $\bar{A}$ und $\bar{b}$ über $\mathbb{F}_p$, einem Körper, gegeben sind ist es einfach $\bar{x}$ zu finden - oder, falls $\bar{x}$ nicht exisitiert, zu entscheiden, dass es keine Lösung in $\mathbb{Z}$ geben kann.

Sei nun $\bar{x}_1$ mit $\bar{A}\bar{x}_1 = \bar{b}$ gegeben. Also existieren $x_1 \in \mathbb{Z}^n$, $c_1 \in \mathbb{Z}^m$ mit $Ax_1 = b + pc_1$ via $c_1 := (Ax_1 - b)/p$. Wenn wir nun $\bar{x}_2$ finden können mit $\bar{A}\bar{x}_2 = -\bar{c}_1$, so folgt

$A(x_1 + px_2) = b + pc_1 + p(-c_1 + pc_2) = b + p^2c_2$ für $c_2 := (Ax_2 - c_1)/p$. Durch iterieren können wir so $x_i$, $c_i$ finden mit

$$A\sum p^{i-1}x_i = b + p^ic_i$$

Falls nun $Ax = b$ genau eine Lösung in $\mathbb{Q}^n$ hat und diese in $\mathbb{Z}^n$ lebt, so können wir sie mit dieser Methode finden. Für $p^i > 2\|x\|_\infty$ muss offensichtlich $x_i = x$ gelten - wenn wir $x_i$ im symmetrischen Restsystem modulo $p^i$ darstellen. Speziell klappt dies immer wenn $A \in \mathrm{Gl}(n, \mathbb{Z})$ ist und hat eine Chance wenn $A \in \mathrm{Gl}(n, \mathbb{Q}) \cap \mathbb{Z}^{n\times n}$ gilt. Das Problem hier kommt von Gleichungen die in $\mathbb{Q}$ (eindeutig) lösbar sind, aber nicht in $\mathbb{Z}$:

$$2x = 1$$

hat für $p = 3$ immer Lösungen: $2 \cdot 2 \equiv 1 \bmod 3$, $2 \cdot 5 \equiv 1 \bmod 9$, $2 \cdot 14 \equiv 1 \bmod 27$, ..., $2 \cdot (3^i + 1)/2 \equiv 1 \bmod 3^i$ aber **nie** $2x = 1$. Damit brauchen wir noch weitere Hilfsmittel:

(1) um zu entscheiden ob eine Lösung existiert brauchen wir die Größe der potentiellen Lösung
(2) i. Allg. müssen wir entweder den Nenner von $x$ finden oder irgendwie wieder nach $\mathbb{Q}$ kommen
(3) was passiert wenn $Ax = b$ eindeutig lösbar sein soll, $\bar{A}\bar{x} = \bar{b}$ jedoch viele Lösungen hat?

Das letzte ist einfach: in diesem Fall nehmen wir eine andere Primzahl. Die beiden anderen Punkte müssen wir vertagen: die Nenner und die Größe der Lösung werden aus der Cramerschen Regel und der Hadarmad Ungleichung bzw. dem Chinesischen Restsatz folgen, das „zurück nach $\mathbb{Q}$“ benötigt die Gitter.

## 4. Gitter

Sehen wir uns mal einen kleinen Modul an, gegeben als Spaltenmodul einer Matrix:

$$\begin{pmatrix} 1 & 0 \\ 83846179880 & 100000000003 \end{pmatrix}$$

Dann ist die Matrix bereits in HNF, aber, mit ein bischen ausprobieren sehen wir

$$\begin{pmatrix} 1 & 0 \\ 83846179880 & 100000000003 \end{pmatrix}\begin{pmatrix} -29541 & 413704 \\ 24769 & -346875 \end{pmatrix} = \begin{pmatrix} -29541 & 413704 \\ 239227 & 34895 \end{pmatrix}$$

was auch nich schön ist - aber wenigstens sind jetzt alle Zahlen nur halb so lang wie oben! In gewissem Sinne, ist die neue Matrix „kleiner“ als die Alte. Dies soll nun systematisch untersucht werden, wobei „klein“ im Sinne von euklischer Norm gelesen wird.

Definition 3.16: Ein $\mathbb{Z}$-Modul $L \subseteq R^n$ heißt Gitter, wenn er frei von Rang $\operatorname{rk} L =: m \leq n$ ist, also $L = \sum_{i=1}^m \mathbb{Z}x_i$ gilt und die $x_i$ zusätzlich noch $\mathbb{R}$-linear unabhängig sind. Die Menge $\{x_1, \ldots, x_m\}$ heißt dann Gitterbasis von $L$.

Beispiel 3.17: (1) $L = \mathbb{Z}^n$
(2) $L = \langle (1, 0)^t, (-1/2, \sqrt{3}/2)^t\rangle$
(3) $\langle e, \pi\rangle \subseteq \mathbb{R}^1$ ist kein Gitter

Bemerkung 3.18: Sei $L$ ein Gitter und $\{x_1, \ldots, x_m\}$ und $\{y_1, \ldots, y_m\}$ Gitterbasen. Dann gibt es $U \in \mathrm{Gl}(m, \mathbb{Z})$ mit $(x_1, \ldots, x_m) = (y_1, \ldots, y_m)U$.

BEWEIS. Da $(y_i)$ Basis, gibt es $u_{i,j}$ mit $x_i = \sum u_{i,j} y_j$. Damit gilt $(x_1, \ldots, x_m) = (y_1, \ldots, y_m)U$. Da wir analog ein $V$ finden können mit $(y_1, \ldots, y_m) = (x_1, \ldots, x_m)V$ sind $U$ und $V$ über $\mathbb{Z}$ invertierbar, also in $\mathrm{Gl}(m, \mathbb{Z})$. □

DEFINITION 3.19: Sei $L \subseteq \mathbb{R}^m$ ein Gitter vom Rang $n$ mit Basis $\{x_1, \ldots, x_n\}$. Sei $X := (x_1, \ldots, x_n) \in \mathbb{R}^{m \times n}$ die Basismatrix. Dann heißt

$$d(L) := \sqrt{\det X^t X} = \sqrt{(x_i^t x_j)_{i,j}}$$

die Diskriminante von $L$.

BEMERKUNG 3.20: Die Diskriminante $d(L)$ von $L$ ist unabhängig von der Wahl der Gitterbasis.

BEWEIS. Seien $\{x_1, \ldots, x_n\}$ und $\{y_1, \ldots, y_n\}$ Gitterbasen. Nach Bemerkung 3.18 existiert dann eine unimodulare Matrix $U \in \mathrm{Gl}(n, \mathbb{Z})$ mit $X = YU$. Damit gilt dann

$$d(L)^2 = \det X^t X = \det(YU)^t (YU) = \det U^t \det Y^t Y \det U = \det Y^t Y$$

□

BEMERKUNG 3.21: Sei $L \subseteq \mathbb{R}^n$ Gitter vom Rank $n$ mit Basis $\{x_1, \ldots, x_n\}$. Eine (messbare) Menge $F \subseteq \mathbb{R}^n$ heißt Fundamentalbereich von $L$ falls gilt

(1) $\mathbb{R}^n = \bigcup_{\lambda \in L} \lambda + F$
(2) Für $\lambda \neq \mu \in L$ gilt $\lambda + F \cap \mu + F = \emptyset$

$\Gamma := \{\sum r_i x_i : 0 \leq r_i < 1\}$ ist ein Fundamentalbereich oder Grundmasche von $L$. Ferner haben wir für jeden Fundamentalbereich $G$, dass $\mathrm{Vol}\, G = d(L)$ gilt. $G$ ist ein Vertretersystem für $\mathbb{R}^n / L$ als $\mathbb{Z}$-Moduln.

SATZ 3.22 (Gram-Schmidt): Seien $x_1, \ldots, x_n \in \mathbb{R}^m$ $\mathbb{R}$-linear unabhängig. Definiere $x_1^*, \ldots, x_n^* \in \mathbb{R}^m$ mittels:

(1) $x_1^* := x_1$
(2) Für $i > j$ sei $\mu_{i,j} := \frac{x_i^t x_j}{B_j}$, $B_j := (x_j^*)^t x_j^*$ und $x_i^* := x_i - \sum_{j=1}^{i-1} \mu_{i,j} x_j^*$

Ferner setze $M = (M_{i,j}) \in \mathbb{R}^{n \times n}$ mit $M_{i,j} := \begin{cases} -\mu_{i,j} & i > j \\ 1 & i = j \\ 0 & \text{sonst} \end{cases}$. Dann gilt:

(1) $x_i^* \neq 0$, also $B_i \neq 0$
(2) $x_i^*$ ist die Projektion von $x_i$ auf $\langle x_1, \ldots, x_{i-1} \rangle^\perp$
(3) $(x_1^*, \ldots, x_n^*) = (x_1, \ldots, x_n)M$
(4) $(x_i^*)^t x_j^* = 0$ für $i \neq j$, d.h. die $x_i^*$ sind ein Orthogonalsystem

BEWEIS. Direktes Nachrechnen bzw. Nachschlagen in GdM. □

KOROLLAR 3.23 (Hadarmad Ungleichung): Seien $x_i \in \mathbb{R}^n$ $(1 \leq i \leq n)$ Vektoren und $X := (x_1, \ldots, x_n)$. Dann gilt

$$|\det X| \leq \prod_{i=1}^{n} \|x_i\|_2$$

BEWEIS. Seien $M$ und $x_i^*$ wie in Satz 3.22 und $X^* = (x_1^*, \dots, x_n^*)$. Dann gilt $X^* = XM$ und $\det M = 1$ da $M$ Dreiecksmatrix mit 1sen auf der Diagonale ist. Also auch

$$\det X^2 = \det X^t X = \det(X^*)^t X^*$$

Da die $x_i^*$ orthogonal sind, folgt $(X^*)^t X^* = \operatorname{diag}(B_1, \dots, B_n)$ mit $B_i = (x_i^*)^t x_i^*$. Aus Satz 3.22.(2) folgt $B_i \le x_i^t x_i$ (Vektoren werden kürzer unter einer Projektion) und so die Behauptung. □

BEMERKUNG 3.24: Die Hadarmad Ungleichung ist Ausgangspunkt für die Berechung von Determinanten über $\mathbb{Z}$ wie wir in der Übung sehen werden.

BEMERKUNG 3.25: Die Determinanten können benutzt werden um Lineare Gleichungssyteme zu lösen. Sei $A = (a_1, \dots, a_n) \in \mathbb{Z}^{n\times n}$ und $b \in \mathbb{Z}^n$. Für $x \in \mathbb{Z}^n$ mit $Ax = b$ gilt $x_i = \det(a_1, \dots, a_{i-1}, b, a_{i+1}, \dots, a_n)/\det A$. Damit haben wir jetzt zwei weitere Methoden $x$ zu finden:

(1) wir können alle Determinanten mit der letzten Bemerkung berechnen
(2) wir berechnen nur $\det A$ und benutzen dann die $p$-adische Methode um $x_i$ zu berechnen. Nach der Cramerschen Regel ist dann $x_i \det A \in \mathbb{Z}$ und $x_i$ kann daher gefunden werden, falls $p^i > 2|x_i \det A|$ gilt.

Beide Methoden habe gegenüber der HNF den Vorteil, dass sie die Koeffizientenexplosion vollständig vermeiden. Wir werden später noch eine dritte Methode kennenlernen.

## 5. Der LLL Algorithmus

DEFINITION 3.26: Sei $L \subseteq \mathbb{R}^n$ ein Gitter vom Rang $n$. Eine Basis $\{x_1, \dots, x_n\}$ von $L$ heißt LLL reduziert, falls gilt

(1) $|\mu_{i,j}| \le 1/2$ für $1 \le j < i \le n$
(2) $\|x_i^* + \mu_{i,i-1} x_{i-1}^*\|^2 \ge 3/4 \|x_{i-1}^*\|^2$ für $2 \le i \le n$

(LLL steht für HW Lenstra, AK Lenstra und L Lovász die in 1982 LLL erfunden haben um damit Polynome zu faktorisieren).

SATZ 3.27: Sei $L \subseteq \mathbb{R}^n$ ein Gitter vom Rang $n$ und $\{x_1, \dots, x_n\}$ eine LLL-reduzierte Basis von $L$. Setze $A := (x_1, \dots, x_n)$. Dann gilt

(1) $|\det A| \le \prod \|x_i\| \le 2^{n(n-1)/4} |\det A|$
(2) $\|x_j\| \le 2^{(i-1)/2} \|x_i^*\|$ für $1 \le j < i \le n$
(3) $\|x_1\| \le 2^{(n-1)/4} \sqrt[n]{|\det A|}$
(4) für jedes $0 \ne x \in L$ gilt $\|x_1\| \le 2^{(n-1)/2} \|x\|$
(5) für alle $y_1, \dots, y_l$ linear unabhängig gilt $\|x_j\| \le 2^{(n-1)/2} \max(\|y_1\|, \dots, \|y_l\|)$ für $1 \le j \le l$

BEWEIS. (1) Nach Hadarmad gilt $|\det A| \le \prod \|x_i\|$ und $|\det A| = \prod \|x_i^*\|$. Aus der LLL Bedingung folgt $3/4 \|x_{i-1}^*\|^2 \le \|x_i^* + \mu_{i,i-1} x_{i-1}^*\|^2 = \|x_i\|^2 + |\mu_{i,i-1}|^2 \|x_{i-1}^*\|^2 \le \|x_i^*\| 2 + 1/4 \|x_{i-1}^*\|^2$.
Also $\|x_i^*\|^2 \ge 1/2 \|x_{i-1}^*\|^2$, und per Induktion $\|x_i^*\|^2 \ge 2^{j-i} \|x_j^*\|^2$ für $j < i$. Damit folgt dann $\|x_i\|^2 = \|x_i^*\|^2 + \sum \mu_{i,j}^2 \|x_j^*\|^2 \le \|x_i^*\|^2 (1 + 2^i/4 \sum 2^{-j}) = 1/2(2^{i-1} + 1) \|x_i^*\|^2$.

Für die Determinante folgt dann:

$$|\det A\| \leq \prod \|x_i\|^2 \leq \prod 1/2(2^{i-1}+1)\|x_i^*\|^2 \leq \prod 2^{i-1}\|x_i^*\| = 2^{n(n-1)/2}|\det A\|$$

(2) Für $1 \leq j < i \leq n$ gilt

$$\|x_j\|^2 \leq 1/2(2^{j-1}+1)\|x_j^*\|^2 \leq 1/2(2^{j-1}+1)2^{i-j}\|x_i^*\|^2 = 2^{i-1}(1/2+2^{-j})\|x_i^*\|^2 \leq 2^{i-1}\|x_i^*\|^2$$

(3) $\|x_1\|^2 \leq 2^{i-1}\|x_i^*\|^2$, damit $\|x_1\|^{2n} \leq \prod 2^{i-1}\|x_i^*\|^2 = 2^{n(n-1)/2}\det^2 A$

(4) Sei $i$ minimal mit $x \in \langle x_1, \ldots, x_i\rangle = \langle x_1^*, \ldots, x_i^*\rangle$. Dann exisitieren $z_j \in \mathbb{Z}$ und $r_j \in \mathbb{R}$ mit $x = \sum_{j=1}^{i} z_j x_j = \sum r_j x_j^*$. Aus dem Gram-Schmidt Verfahren kommt $r_i = z_i$ und damit

$$\|x\|^2 = \sum r_j^2\|x_j^*\|^2 \geq r_i^2\|x_i^*\|^2 = z_i^2\|x_i^*\|^2 \geq \|x_i^*\|^2 \geq 2^{i-1}\|x_1\|^2 \geq 2^{1-n}\|x_1\|^2$$

(5) Übung, via Induktion und (4).

□

OK, eine LLL-reduzierte Basis hat sehr schöene Eigenschaften, aber was noch fehlt ist der LLL-Algorithmus der, ausgehend von einer beliebigen Basis eine LLL-reduzierte Basis berechnet. Der Algorithmus ist sehr einfach - zumindest auf den 1. Blick:

ALGORITHMUS 3.28: Input: Ein Gitter, gegeben mittels einer Basis $x_1, \ldots, x_n$.
Output: Eine LLL-reduzierte Basis

1: $k = 2$
2: Berechne $\mu_{k,k-1}$
3: Falls $|\mu_{k,k-1}| > 1/2$, so setze $x_k := x_k - \lfloor \mu_{k,k-1} \rceil x_{k-1}$ und ändere die $\mu_{k,k-1}$ entsprechend.
4: Falls $\|x_k^* + \mu_{k,k-1}x_{k-1}^*\|^2 < 3/4\|x_{k-1}^*\|^2$ gilt, so vertausche $x_k, x_{k-1}$. Falls $k > 2$, so setze $k := k-1$. Gehe wieder zu (2).
5: Reduziere $\mu_{k,j}$ für $j < k-1$ wie in (3), setze $k := k+1$.
6: Falls $k > n$ return $x_1, \ldots, x_n$, sonst gehe nach (2).

So wie hier beschrieben, ist es natürlich schlecht zu implementieren, wir müssten mit reellen Zahlen arbeiten und das können wir nicht (gut). Hier gibt es jetzt im wesentlichen zwei Methoden. Die erste ist anzunehmen, dass $L \subseteq \mathbb{Q}^n$ gilt, dann sind alle Operationen in $\mathbb{Q}$ und daher exakt. Die 2. Möglichkeit ist es mit Näherungen zu arbeiten und die Rundungsfehler zu kontrollieren. Beide Methoden wurden eingesetzt, heutige, moderene Algorithmen (Stehle, $L^2$) sind jedoch deutlich komplizierter (und viel schneller) und favorisieren die 2. Variante.

Es gibt keinen (publizierten) Algorithmus der tatsächlich ein nicht rationales Gitter reduzieren kann.

SATZ 3.29: Der Algorithmus terminiert (und liefert eine LLL-reduzierte Basis).

BEWEIS. OK, Beweisidee. Setze $d_i := \prod_{j=1}^{k} \|x_j^*\|^2$ und $D := \prod_{i=1}^{n-1} d_i$. Dann gilt $d_n = \det^2 A$. Der Schritt (3) (die Reduktion $|\mu| < 1/2$) ändert die orthogonale Basis und damit $d_i$ und $D$ nicht. Falls im nächsten Schritt $x_k$ und $x_{k-1}$ vertauscht werden, so wird um mindestens einen Faktor 3/4 reduziert und damit $D$ um 3/4 verkleinert (alle anderen $d_i$ bleiben gleich). Andererseits gelten für die Teilgitter

$$L_i := \sum_{j=1}^{i} \mathbb{Z}x_i$$

$d(L_i) = d_i$. Man kann zeigen, dass $d(L_i) \geq c > 0$ gilt (z.B. für $L_i \subseteq \mathbb{Z}^n$ gilt $d(L_i) \geq 1$), damit gibt es eine untere Schranke für $D$ und die Vertauchung kann nur endlich oft durchgeführt werden. □

(Wir haben fast alles was wir bräuchten um den Beweis zu vervollständigen. Im wesentlichen fehlt nur der Formalismus um mit Gittern von Rang $< n$ vernünftig rechnen zu können und ein formaler Beweis der Diskretheit).

BEISPIEL 3.30: Algebraische Relationen. Der LLL Algorithmus wird für viele Dinge eingesetzt. Als ein (typisches) Beispiel sehen wir uns mal die folgende Frage an: ist $\xi := 1.64575131106459059050161575364\ldots$ (wahrscheinlich) algebraisch? Also: gibt es ein Polynom $f \in \mathbb{Z}[t]$ mit $f(\xi) = 0$?

Hierzu bauen wir eine Matrix

$$\begin{pmatrix} 1 & & \\ 0 & 1 & \\ 0 & 0 & 1 \\ \lambda & \lambda\xi & \lambda\xi^2 \end{pmatrix}$$

für ein geeignetes (großes) $\lambda > 0$. Dann rechnen wir LLL-reduzierte Basen aus und sehen uns die 1. Basisvektoren an:

| $\lambda$ | $x_1$ |
|---|---|
| 10 | $(-1, -1, 1, 0.627)$ |
| 100 | $(-6, 2, 1, 0)$ |
| 1000 | $(-6, 2, 1, 0)$ |

Also können wir vermuten, dass $\xi^2 + 2\xi - 6 = 0$ gilt, also $\xi = -1 \pm \sqrt{7}$. Natürlich ist die kein Beweis - und kann auch nie einen Beweis liefern. Aber wenn wir wissen *welche* algebraische Zahl (wahrscheinlich) herauskommt, so können wir dies u.U. beweisen.

BEISPIEL 3.31: Sei $x = a/b \in \mathbb{Q}$ und $M \in \mathbb{Z}$, $\mathrm{ggT}(M, b) = 1$. Dann gibt es $d \in \mathbb{Z}$ mit $db \equiv 1 \bmod M$, also können wir sagen: $x = a/b \equiv ad =: r \bmod M$. Nehmen wir jetzt an, wir haben $0 \leq r < M$ gegeben und wollen $a$, $b$ finden mit $a/b \equiv r \bmod M$ oder $a \equiv br \bmod M$. Dann können wir dies über Gitter erledigen!

Sei $L := \langle (M, 0)^t, (r, 1)^t \rangle$ und sei $b_1 = (a, b)$ der 1. Basisvektor einer LLL-Basis von $L$ (Wir brauchen einen kurzen Vektor).

Es gilt der Satz: Sei $M > 2B$. Dann gibt es ein $x \in \mathbb{Q}$, so dass aus $(a, b)^t \in L$ mit $a^2 + b^2 < B$ sofort $a/b = x$ folgt.

Damit gilt dann $a/b = x$ falls $M > 2\sqrt{2}(a^2 + b^2)$ gilt.

Dies liefert nun die 3. Möglichkeit lineare Gleichungssysteme über $\mathbb{Q}$ zu lösen: mit der Cramerschen Regel und der Hadarmad Ungleichung werden Schranken für die Zähler und Nenner der Lösung berechnet. Dann wird die Lösung z.B. mit Hilfe der $p$-adischen Methode modulo $M = p^k$ berechnet. Falls $k$ groß genug ist, kann dann mit dem Gitter die Lösung gefunden werden:

$2x = 1$ und $p = 3$. Dann gilt $2 \cdot 2 \equiv 1 \bmod 3$, $2 \cdot 5 \bmod 9$ und $2 \cdot 14 \bmod 27$. Nun $L := \langle (27, 0)^t, (14, 1)^t \rangle$ und der erste LLL-Basisvektor ist $b_1 = (1, 2)^t$. In der Tat, $x = 1/2$ ist eine Lösung.

## 6. Gröbnerbasen

Nachdem wir jetzt lineare Gleichungssysteme umfassend (aber natürlich nicht abschliessend) behandelt haben, wenden wir uns jetzt nicht linearen Systemen zu.

Ausgangspunkt sind hier ein Körper $K$ und $R := K[X_1, \dots, X_n]$ ein Polynomring in $n$ Variabeln über $K$. Speziell können wir nicht lineare, algebraische Systeme oft mit Idealen in $R$ in Verbindung setzen. Da $R$ für $n > 1$ kein Hauptidealring ist, ist die Idealstruktur und alles was damit zusammen hängt kompliziert. Gröbnerbasen sind entwickelt worden um diese Ideale besser untersuchen und darstellen zu können. Daher sind sie fundamental für sehr viele Probleme in der algebraischen Geometrie.

Ein erstes Problem bei Polynomen in mehreren Veränderlichen ist es eine *eindeutige* Darstellung zu erhalten. Für $n = 1$ haben wir z.B. die Polynome oft als Vektoren interpretiert. Dies war einfach weil die Koeffizienten von den $x^i$ eine natürliche Reihenfolge haben. Hier ist dies anders ...

(Wir brauchen dies auch um die wichtige Division mit Rest zu übertragen. $R$ ist nicht mehr euklidisch, aber wir wollen so viel wie möglich retten)

DEFINITION 3.32: Ein Polynom $X_1^{m_1} \dots X_n^{m_n} =: \underline{X}^{\underline{m}}$ mit $\underline{m} \in \mathbb{N}_0^n$ heißt Monom, $|\underline{m}| := \sum m_i$ heißt der Totalgrad des Monoms. Sei nun $f = \sum a_{\underline{m}} \underline{X}^{\underline{m}} \in R$ ein beliebiges Polynom. Ist $a_{\underline{m}} \neq 0$ so heißt $a_{\underline{m}} \underline{X}^{\underline{m}}$ ein Term von $f$, $a_{\underline{m}}$ der Koeffizient von $\underline{X}^{\underline{m}}$. Der Totalgrad von $f$ ist das Maximum über die Totalgrade der Monome in der Termen von $f$.

Es gibt eine Bijektion von {Monome} zu $\mathbb{N}_0^n$ mittels $\underline{X}^{\underline{m}} \mapsto \underline{m}$.

DEFINITION 3.33 (Monomordnung): Eine Monomordnung auf $R$ ist eine Relation $>$ auf $\mathbb{N}_0^n$ (oder den Monomen) mit

(1) $>$ ist eine Totalordnung (also wir haben immer $\underline{a} < \underline{b}$ oder $\underline{b} < \underline{a}$ für Monome $\underline{a}$ und $\underline{b} \in \mathbb{N}_0^n$)
(2) für $\underline{a}$, $\underline{b}$, $\underline{c} \in \mathbb{N}_0^n$ mit $\underline{a} < \underline{b}$ gilt immer $\underline{a} + \underline{c} < \underline{b} + \underline{c}$
(3) $(\mathbb{N}_0^n, <)$ ist wohlgeordnet (also besitzt jede nicht leere Teilmenge ein kleinstes Element)

BEISPIEL 3.34:
(1) $>$ auf $\mathbb{N}_0$ mit $1 > 2 > 3 \dots$ ist keine Monomordnung da $\mathbb{N}_0$ offenbar nicht beschränkt ist und daher kein „kleinstes“ Element enthält.
(2) $\underline{a} >_{\text{lex}} \underline{b} :\Leftrightarrow$ der 1. Nicht-Null-Eintrag von $\underline{a} - \underline{b}$ (von links) ist positiv, die sog. lexikographische Ordnung.
(3) $\underline{a} >_{\text{grlex}} \underline{b} :\Leftrightarrow |\underline{a}| > \underline{b}|$ oder $|\underline{a}| = |\underline{b}|$ und $\underline{a} <_{\text{lex}} \underline{b}$ (graduiert lexikographisch)
(4) $\underline{a} >_{\text{grevlex}} \underline{b} :\Leftrightarrow |\underline{a}| > \underline{b}|$ oder $|\underline{a}| = |\underline{b}|$ und $\underline{b} <_{\text{lex}} \underline{a}$ (graduiert revers lexikographisch)

Wenn eine Termordnung fixiert ist, kann ebenfalls eine eindeutige Darstellung der Polynome erreicht werden: als Summe von Termen die absteigend geordnet sind.

Zum Beispiel: in $R = \mathbb{Q}[x, y, z]$ mit $x > y > z$ und $f = 4xy^2z + 4z^2 - 5x^3 + 7x^2z^2$.

| Ordnung | Polynom |
|---|---|
| lex | $-5x^3 + 7x^2z^2 + 4xy^2z + 4z^2$ |
| grlex | $7x^2z^2 - 4xy^2z - 5x^3 + 4z^2$ |
| grevlex | $-4xy^2z + 7x^2z^2 - 5x^3 + 4z^2$ |

Prinzipiell gibt es zwei Darstellungen für multivariate Polynome auf dem Rechner: dichte (rekursive) und dünn besetzte (distributive) Darstellung. In der ersten Form wird $R = K[X_1, \ldots, X_n] = K[X_1][X_2] \ldots [X_n]$ benutzt, d.h. $f \in R$ ist ein Polynom mit Koeffizienten in $K[X_1, \ldots, X_{n-1}]$. Für dieses univariate Polynom wird einfach ein Vektor mit Koeffizienten $f_i \in K[X_1, \ldots, X_{n-1}]$ gespeichert, die Koeffizienten werden dann rekursiv ebenfalls als univariate Polynome implementiert. Auf der anderen Seite werden multivariate Polynome auch als Liste von Termen gespeichert, wobei nur Terme mit von Null verschiedenen Koeffizienten tatsächlich gespeichert werden. Oft wird diese Liste dann nach der Monomordnung sortiert. Andere Möglichkeiten ergeben sich aus Hash-Funktionen oder Suchbäumen. Da in vielen Anwendungen die meisten Exponentenvektoren klein sind werden diese oft komprimiert (4 Bits je Exponent, 8 Bits, ...).

Wenn wir uns z.B. die Multiplikation von Polynomen ansehen, so folgt, dass die Multiplikation von zwei Polynomen vom Totalgrad $m$ etwa $O(n^{2m})$ Operationen in $K$ benötigt (um die $n^m$ Koeffizienten zu multiplizieren), während in der distributiven Darstellung nur die Anzahl der nicht-Null Koeffizienten entscheidend ist. Es gibt auch Mischformen, welche in der Regel eingestetzt werden, wenn ein Teil der Variablen als Parameter verstanden werden soll, also z.B. Polynome über $\mathbb{Q}(x)$ betrachtet werden.

DEFINITION 3.35: Sei $>$ eine Monomordnung, $f = \sum a_{\underline{m}} \underline{X^m}$ ein Polynom. Der Multigrad von $f$ ist $\operatorname{mdeg}(f) := \max_>\{\underline{m} | a_{\underline{m}} \neq 0\}$. Der Leitkoeffizient von $f$ ist $\operatorname{lc}(f) := a_{\operatorname{mdeg} f}$, das Leitmonom ist $\operatorname{lm}(f) = \underline{X}^{\operatorname{mdeg} f}$ und der Leitterm $\operatorname{lt}(f) = \operatorname{lc}(f) \operatorname{lm}(f)$.

Es gilt $\operatorname{mdeg}(fg) = \operatorname{mdeg}(f) + \operatorname{mdeg}(g)$ und $\operatorname{mdeg}(f+g) \leq \max_>(\operatorname{mdeg} f, \operatorname{mdeg} g)$ wobei Gleichheit gilt, falls $\operatorname{mdeg} f \neq \operatorname{mdeg} g$. Zur Erinnerung: $\langle F \rangle$ ist das von $F \subseteq R$ erzeugte Ideal.

DEFINITION 3.36: Sei $0 \neq I \leq R$ ein Ideal. Dann definieren wir $\operatorname{lt}(I) := \{\operatorname{lt}(f) : f \in I\}$

Wir fangen mit der einfachen Feststellung an:

LEMMA 3.37: Sei $O \neq I = \langle f_1, \ldots, f_m \rangle$. Dann gilt $\langle \operatorname{lt}(f_1), \ldots, \operatorname{lt}(f_n) \rangle \subseteq \operatorname{lt}(I)$.

BEWEIS. Trivial wegen $A \subseteq B \Rightarrow \langle A \rangle \subseteq \langle B \rangle$ □

Im Allgemeinen gilt hier jedoch keine Gleichheit: $R = \mathbb{Q}[x, y]$, $x > y$ und grlex. $f_1 := x^3 - 2xy$, $f_2 := x^2y - 2y^2 + x$. Dann ist $xf_2 - yf_1 = x^2$ aber $x^2 \notin \langle x^3, x^2y \rangle$. Dies wollen wir im weiteren genauer untersuchen.

DEFINITION 3.38: Sei $>$ eine Monomordnung, $I \leq R$ ein Ideal. $G = \{g_1, \ldots, g_i\} \subseteq I$ heißt Gröbnerbasis von $I$, falls gilt

$$\operatorname{lt}(I) = \langle \operatorname{lt}(g_1), \ldots, \operatorname{lt}(g_i) \rangle$$

(Dies hängt von der gewählten Monomordnung ab!)

BEMERKUNG 3.39: Unter den Voraussetzungen wie in der Definition gilt: $G$ ist Gröbnerbasis von $I$ genau dann, wenn für alle $0 \neq f \in I$ ein $g \in G$ exisitiert mit $\operatorname{lt}(g) | \operatorname{lt}(f)$.

BEWEIS. Angenommen, für alle $0 \neq f \in I$ ex. ein $g \in G$ mit $\operatorname{lt}(g) | \operatorname{lt}(f)$. Nach Lemma 3.37 gilt $\langle \operatorname{lt}(G) \rangle \subseteq \operatorname{lt}(I)$. Sei nun $0 \neq f \in I$ beliebig. Dann gilt $\operatorname{lt}(g) | \operatorname{lt}(f)$ für $g \in G$ geeignet, und damit $\operatorname{lt}(f) \in \langle \operatorname{lt}(G) \rangle$, also $\operatorname{lt}(I) \subseteq \langle \operatorname{lt}(G) \rangle$.

Sei nun $G$ eine Gröbnerbasis und $f \in I$ beliebig. Dann gilt $\operatorname{lt}(f) \in \langle \operatorname{lt}(G) \rangle$ und somit $\operatorname{lt}(f) = \sum h_i \operatorname{lt}(g_i)$ für $h_i \in R$. Also gibt es $g_i$i, so dass $\operatorname{lm}(f)$ ein Monom von $h_i \operatorname{lt}(g_i)$ ist (es kann mehrere geben). Offensichtlich ist $h_i \operatorname{lt}(g_i)$ durch $\operatorname{lt}(g_i)$ teilbar, also auch $\operatorname{lt}(f)$. □

Damit können wir jetzt zumindest schon einmal die Division mit Rest verallgemeinern:

SATZ 3.40: Sei $>$ Monomordnung, $F = (f_1, \ldots, f_s) \in (R \setminus \{0\})^s$ und $f \in R$. Dann gibt es $q_i \in R$ und $r \in R$ mit $f = \sum q_i f_i + r$, $\operatorname{lm}(f) \geq \operatorname{lm}(q_i f_i)$ und $r = 0$ oder kein Term von $r$ ist durch einen der Terme $\operatorname{lt}(f_i)$ teilbar.

BEWEIS. Wir zeigen dies konstruktiv, per Induktion nach $\operatorname{mdeg} f$. Da wir eine Wohlordnung haben, geht dies. Angenommen, $\operatorname{lt}(f_i) \not| \operatorname{lt}(f)$ für alle $i$. Betrachte nun $f' := f - \operatorname{lt}(f)$. Da $\operatorname{mdeg} f' < \operatorname{mdeg} f$ so folgt per Induktion, $f' = \sum q_i f_i + r'$, also $f = \sum q_i f_i + (r + \operatorname{lt}(f))$ wie behauptet.

Also können wir annehmen, dass es ein minimales $i$ gibt mit $\operatorname{lt}(f_i) | \operatorname{lt}(f)$. Setze nun $f' := f - \operatorname{lt}(f)/\operatorname{lt}(f_i) f_i$, dann folgt, wie oben, $\operatorname{mdeg} f' < \operatorname{mdeg} f$, also $f' = \sum q'_j f_j + r$ und damit $f = \sum q'_j f_j + r + \operatorname{lt}(f)/\operatorname{lt}(f_i) f_i$ wie behauptet. □

In diesem Fall schreiben wir $\bar{f}^F = r$ falls $f = \sum q_i f_i + r$ gilt. Achtung: der Rest ist nicht eindeutig. In der Prozedur hängt er von der Reihenfolge der Operation und damit von der Reihenfolge der $f_i$ ab.

BEISPIEL 3.41: Sei $R := \mathbb{Q}[x, y]$, $x > y$, $>_{\text{lex}}$ und $f = x^2 y + x y^2 + y^2$.

(1) $f_1 = y^2 - 1$, $f_2 = xy - 1$. Dann ist $f' = f - x f_2 = xy^2 + x + y^2$, $f'' = f' - y f_1 = 2x + y^2$, $f''' = f'' - f_1 = 2x + 1$
(2) $f_1 = xy - 1$, $f_2 = y^2 - 1$. Dann ist $f' = f - x f_1 = xy^2 + x + y^2$, $f'' = f' - y f_1 = x + y^2 + y$, $f''' = f'' - f_2 = x + y + 1$

KOROLLAR 3.42: Sei $>$ eine Monomordnung, $I \leq R$ ein Ideal und $G$ eine Gröbnerbasis. Dann gilt $\langle G \rangle = I$.

BEWEIS. Division mit Rest: sei $f \in I$ beliebig, dann gilt $f = \sum h_i g_i + r$ mit $h_i \in R$ und $g_i \in G$ und $r = 0$ oder $\operatorname{lt}(g_i) \not| \operatorname{lt}(r)$. Aber $r \in I$, also muss es (für $r \neq 0$) ein $g_i \in G$ geben mit $\operatorname{lt}(g_i) | \operatorname{lt}(r)$, daher folgt $r = 0$ und $f \in \langle G \rangle$. □

BEISPIEL 3.43: $R = \mathbb{R}[X, Y, Z]$, $X > Y > Z$, $>_{\text{lex}}$, dann ist $G := \{g_1, g_2\}$ mit $g_1 = X + Z$, $g_2 = Y - Z$ Gröbnerbasis von $I = \langle X + Z, Y - Z \rangle$. Wir haben $\operatorname{lt}(g_1) = X$, $\operatorname{lt}(g_2) = Y$ und müssen zeigen: $\forall 0 \neq f \in I$ gilt $X | \operatorname{lt}(f)$ oder $Y | \operatorname{lt}(f)$. Sei nun $0 \neq f = h(X + Z) + k(Y - Z) \in I$ mit $h, k \in R$ beliebig. Falls $X \not| \operatorname{lt}(f)$ und $Y \not| \operatorname{lt}(f)$ so folgt $f \in \mathbb{R}[Z]$, $f$ ist ein Polynom in $Z$. Aber $0 = h(-t, t, t)(-t + t) + k(-t, t, t)(t - t) = f(-t, t, t)$ für jedes $t \in \mathbb{R}$ im Widerspruch zu $0 \neq f \in \mathbb{R}[Z]$.

BEMERKUNG 3.44: Sei $I < R$ Ideal, $G = \{g_1, \ldots, g_t\}$ Gröbnerbasis von $I$ und $f \in R$ beliebig. Dann ex. ein eindeutig bestimmtes $r \in R$ mit

(1) kein Term von $r$ ist durch ein $\mathrm{lt}(g_i)$ teilbar
(2) $f - r \in I$

BEWEIS. Division mit Rest: $f = \sum f_i g_i + r$, dann sind (1) und (2) erfüllt. Sei nun $\tilde{r}$ mit $f = \sum \tilde{f}_i g_i + \tilde{r}$ geben mit (1) und (2), dann folgt sofort $r - \tilde{r} \in I$. Angenommen $r \neq \tilde{r}$, so muss es ein $g_i$ geben mit $\mathrm{lt}(g_i) | \mathrm{lt}(r - \tilde{r})$, was nicht sein kann (Widerspruch zu (1)). □

KOROLLAR 3.45: Sei $I < R$ Ideal mit Gröbnerbasis $G = \{g_1, \ldots, g_t\}$
(1) Sei $\sigma \in S_t$ und $h_i := g_{\sigma i}$, so gilt für jedes $f \in R$: $\bar{f}^{\underline{h}} = \bar{f}^{\underline{g}}$, die Reduktion ist von der Reihenfolge unabhängig.
(2) $f \in I$ genau dann, wenn $\bar{f}^{\underline{g}} = 0$.

BEWEIS. Beides folgt aus der letzten Bemerkung. □

## 7. Freie $R$-Moduln

Bevor wir uns der Frage widmen ob und wie man Gröbnerbasen berechnen kann, wollen wir unsere Terminologie noch etwas verallgemeinern. Zur Illustration seien $f_1, \ldots, f_r$ Polynome in $R$. Dann betrachten wir die Abbildung

$$\phi : R^r \to R : (h_i)_i \to \sum h_i f_i$$

von dem freien $R$-Modul $R^r$ nach $R$. Die Gröbnerbasis von $\langle \underline{f} \rangle$ wird uns zeigen was das Bild der Abbildung ist, aber natürlich wollen wir auch den Kern berechnen. Um dies tun zu können müssen wir freie $R$-Moduln und „Termordnungen“ studieren. Da die Ordnung von der Basis abhängen wird, so werden wir auch immer eine Basis fixieren.

DEFINITION 3.46: Sei $M = \langle f_1, \ldots, f_r \rangle$ ein $R$-Modul. Eine Relation $\sum r_i f_i = 0$ mit $r_i \in R$ heißt *Syzygie*

Da offensichtlich die Menge der Syzygien einen $R$-Modul bilden, so können wir die Frage nach dem Kern auch so formulieren: wir wollen den Syzygien Modul berechnen.

Am einfachsten ist dies Falls der Modul $M$ von Monomen erzeugt wird. Wir vereinbaren für Monome $m = \underline{X}^{\underline{\alpha}}$ und $n = \underline{X}^{\underline{\beta}}$:

$$\mathrm{ggT}(m, n) := \underline{X}^{(\min \alpha_i, \beta_i)_i} \quad \text{und} \quad \mathrm{kgV}\,(m, n) := \underline{X}^{(\max(\alpha_i, \beta_i)_i)}$$

.

Für Elemente in $R^r$ mit Basis $e_i$ nennen wir einen Ausdruck in der Form $e_i \underline{X}^{\underline{\alpha}}$ ebenfalls ein Monom. Analog ist ein Term von der Form $s e_i \underline{X}^{\underline{\alpha}}$ mit $s \in K$, und für Terme mit dem selben Basiselement $e_i$ erweitern wir die Definition von ggT und kgV entsprechend.

Für 2 Monome $M_1 = m_1 e_i$ und $M_2 = m_2 e_i$ mit dem selben Basiselement $e_i$ definieren wir $M_1 / M_2 := m_1 / m_2$. Analog definieren wir auch den ggT und kgV von Monomen mit dem selben Basiselement.

Zunächst halten wir fest das von Monomen erzeugten Moduln „einfach“ sind:

BEMERKUNG 3.47: Sei $F$ ein freier $R$-Modul mit fixierter Basis, ferner sei $M := \langle m_1, \ldots, m_r \rangle \subseteq F$ ein $R$-Modul erzeugt von Monomen. Für $f \in F$ gilt nun $f \in M$ genau dann, wenn jeder Term $t$ von $f$ bereits in $M$ liegt. Dies ist der Fall, wenn es ein $m_i$ gibt mit $m_i | t$.

Denn: $f = \sum h_i m_i$ mit $h_i \in R$. Alo muss jeder Term von $f$ irgendwo in $h_i m_i$ auftreten (und somit auch durch $m_i$ teilbar sein).

Wir sehen auch, dass falls $m_i | m_j$ gilt (für $i \neq j$), so können wir $m_j$ weglassen ohne den Modul zu ändern. Somit können wir ein reduziertes Erzeugendensystem erhalten.

LEMMA 3.48: Sei $F$ ein freier $R$-Modul mit fixierter Basis, ferner sei $M := \langle m_1, \ldots, m_r \rangle \subseteq F$ ein $R$-Modul erzeugt von Monomen $m_i \in F$. Wie oben:

$$\phi : R^r \to F : \sum r_i e_i \to \sum r_i m_i$$

. Definiere für jedes Paar $i \neq j$ wo $m_i$ und $m_j$ das selbe Basiselement ethalten $m_{i,j} := m_i / \operatorname{ggT}(m_i, m_j)$ und $\sigma_{i,j} := m_{i,j} e_j - m_{j,i} e_i$. Dann ist $\ker \phi$ erzeugt von den $m_{i,j}$.

BEWEIS. Als Vektorraum über $K$ zerfällt $\ker \phi$ in die direkte Summe der $(\ker \phi)_n$ für Monome $n \in F$ mit

$$(\ker \phi)_n := \{ \sum a_v (n/m_v) e_v : m_v | n, a_v \in K \}$$

Denn: sei $\sigma = \sum p_i e_i \in R^r$ eine Syzygie (also $\sigma \in \ker \phi$). Damit gilt dann $\sum p_i m_i = 0$. Für jedes Monom $n \in F$ das in der Darstellung von $p_i m_i$ irgendwo vorkommt sei nun $p_{i,n}$ der Term von $p_i$, so dass $p_{i,n} m_i = kn$ für $k \in K$ (damit gilt für den Term $p_{i,n} = kn/m_i$). Dann gilt offenbar $\sum p_{i,n} e_i = 0$ und $\sum p_{i,n} e_i \in (\ker \phi)_n$. Offensichtlich ist dies Eindeutig (und die Summe direkt).

Sei nun $\sigma = \sum a_v n_v e_v \in (\ker \phi)_n$. Wir zeigen per Induktion über die Anzahl der nicht-Null-Terme von $\sigma$ das dies in dem von $\sigma_{i,j}$ erzeugen Modul liegt. Falls $\sigma \neq 0$ ist, angenommen, in $\sigma$ gibt es Terme mit verschiedenen Basiselementen von $F$, so können wir $\sigma$ sofort aufspalten. Also können wir oBdA annehmen, dass alle Terme in $\sigma$ nur ein Basiselement enthalten. Da $\sigma$ eine Syzygie ist, so muss es min. 2 nicht-Null-Terme $a_1 n_1$ und $a_2 n_2$ geben (oBdA 1 und 2). Zunächst nehmen wir mal an, wir haben genau 2 nicht-Null-Terme, also $\sigma = a_1 n_1 e_1 + a_2 n_2 e_2 = a_1 n / m_1 e_1 + a_2 n / m_2 e_2$. Dann gilt $\phi(\sigma) = a_1 n + a_2 n = 0$, also $a_1 = -a_2$. Wegen $m_i | n$ folgt auch $\operatorname{kgV}(m_1, m_2) | n$. Mit $m_1 m_2 = \operatorname{ggT}(m_1, m_2) \operatorname{kgV}(m_1, m_2)$ erhalten wir dann, dass $n \operatorname{ggT}(m_1, m_2)/(m_1 m_2)$ ein Monom ist. Nun folgt $\sigma = a_1 n \operatorname{ggT}(m_1, m_2)/(m_1 m_2) \sigma_{1,2}$ durch Einsetzen: $n \operatorname{ggT}(m_1, m_2)/(m_1 m_2) \sigma_{1,2} = n \operatorname{ggT}(m_1, m_2)/(m_1 m_2)(m_2 / \operatorname{ggT}(m_1, m_2) e_1 - m_1 / \operatorname{ggT}(m_1, m_2) e_2 = n/m_1 e_1 - n/m_2 e_2$.

Falls es nun mehr als 2 nicht-Null-Terme gibt, so können wir mit der selben Prozedur $\sigma$ nach und nach abändern: Sei $\sigma = \sum a_i n / m_i e_i$ mit $a_1, a_2 \neq 0$. Dann können wir $\sigma_{1,2}$ benutzen um $a_1$ zu eleminieren und erhalten so ein neues $\sigma$ mit weniger Termen. □

BEISPIEL 3.49: $R = \mathbb{Q}[x, y, z]$ und $m_1 := x^2 y$, $m_2 = xy^2$ und $m_3 = yz^2$. Dann ist $\sigma_{x,y} = y e_x - x e_y$, $\sigma_{y,z} = z e_y - y e_z$ und $\sigma_{x,z} = z e_x - x e_z$. Das Lemma besagt nun, dass jede Kombination $\sum p_i e_i$ mit $p_i \in R$ und $\sum p_i m_i = 0$ aus diesen drei $\sigma$ aufgebaut wird.

Um dieses Beispiel nun weiterzuführen, können wir fragen ob es $q_i \in R$ gibt mit $q_{x,y}\sigma_{x,y} + q_{x,z}\sigma_{x,z} + q_{y,z}\sigma_{y,z} = 0$. Offenbar gilt $z\sigma_{x,y} - y\sigma_{x,z} + x\sigma_{y,z} = 0$

Nun können wir den Begriff der Gröbnerbasis verallgemeinern:

DEFINITION 3.50: Sei $F$ ein freier $R$-Modul mit fixierter Basis und Monomordnung $>$ und $M \subseteq F$ ein $R$-Modul. Eine Teilmenge $G = \{g_1, \ldots, g_t\} \subseteq F$ heißt Gröbnerbasis von $M$ falls $\langle \mathrm{lt}(G)\rangle = \mathrm{lt}(M)$ gilt.

BEISPIEL 3.51: Sei $R = K$ und $F$ ein $K$-Vektorraum. Die Monome sind dann einfach die fest gewählten Basisvektoren. Als Ordnung können wir die Basisvektoren irgendwie ordnen, z¿B. $e_1 > e_2 > \ldots > e_n$. Damit können wir Elemente von $F$ einfach als Spaltenvektoren schreiben und (endliche) Teilmengen mit Matrizen identifizieren. Eine Grobnerbasis von $M$ ist nun jede endliche Teilmenge die ein maximalunabhängiges System von Vektoren enthält.

Eine *reduzierte* Gröbnerbasis währe eine Vektorraum Basis.

BEISPIEL 3.52: $R = K[x]$ und $F = R$, hier nehmen wir den Grad als (einzige) Ordnung. Ein $R$-Modul ist dann einfach ein Ideal und $G$ ist eine Gröbnerbasis genau dann, wenn $G$ ein Polynom minimalen Grades (also ein Erzeuger für das Hauptideal $M$) enthält.

Genauso wie im Falle von Polynomen können wir auch hier zeigen:

LEMMA 3.53: $G \subseteq M$ ist eine Gröbnerbasis für $M$, dann gilt $M = \langle G\rangle$.

BEMERKUNG 3.54: Mit dem Hilbertschen Basissatz ($R$-ist Noethersch) folgt die Existenz von Gröbnerbasen: Sei $M = \langle A\rangle$ beliebig mit $A \subseteq F$ endlich. Dann ist $\mathrm{lt}(M)$ endlich erzeugt und wir müssen $A$ nur so erweitern, dass diese endlichen Erzeuger da sind.

Der Divisionsalgorithmus klappt auch hier:

LEMMA 3.55: Sei $F$ ein freier $R$-Modul mit fester Basis und Termordnung $>$. Für $f$, $g_1$, …, $g_s \in F$ gibt es $q_i$, $r \in F$ mit

$$f = \sum q_i f_i + r$$

und kein Term von $r$ ist durch einen Leitterm von $f_i$ Teilbar. Falls die $g_i$ eine Gröbnerbasis bilden, so ist $r$ eindeutig.

## 8. Buchbergers Algorithmus

Sei für den Rest des Kapitels $F$ ein freier $R$-Modul mit Basis $b_1$, …, $b_r$ mit Termordnung $>$ fixiert.

Für eine endliche Menge $g_1$, …, $g_t \in F$ sei

$$\phi : R^t \to F : (r_i) = \sum r_i e_i \to \sum r_i g_i.$$

Für beliebige $f$, $g \in F$ mit $\mathrm{lt}(f) = a\tilde{f}b_i$ und $\mathrm{lt}(g) = c\tilde{g}b_i$ (also die Leitmonome involvieren das selbe Basiselement $b_i$) definieren wir

$$m(f,g) := \mathrm{lt}(f)/\,\mathrm{ggT}(\mathrm{lm}(f), \mathrm{lm}(g)) = a\tilde{f}/\,\mathrm{ggT}(\tilde{f}, \tilde{g}) \in R$$

Speziell setzen wir $m_{i,j} := m(g_i, g_j)$ wo dies Sinn macht und

$$\sigma_{i,j} = m_{j,i}e_i = m_{i,j}e_j$$

so, dass $\phi(\sigma) = 0$ gilt. Wir definieren das $S$-Polynom

$$S(f, g) := m(g, f)f - m(f, g)g$$

wo dies Sinn macht. Dann haben wir

SATZ 3.56 (Buchbergers Kriterium): $G = \{g_1, \ldots, g_n\}$ ist eine Gröbnerbasis genau dann, wenn

$$(\bar{S}(g_i, g_j))^G = 0$$

gilt.

BEWEIS. Die Definition der Division und der Gröbnerbasis zeigen sofort, dass für Gröbnerbases $G$ der „Rest“ immer 0 sein muss.

Sei nun $G$ mit dieser Eigenschaft. Angenommen, $G$ ist keine Gröbnerbasis, dann gibt es ein

$$f = \sum f_i e_i$$

mit $\phi(f) = \sum f_i g_i$ und $\mathrm{lt}(\phi(f)) \notin \langle \mathrm{lt}(G) \rangle$. Sei nun $m$ das größte Monom das in $\mathrm{lt}(f_i g_i)$ auftaucht. Aufgrund der Wohlordnung können wir annehmen, dass $f$ so gewählt ist, dass $m$ minimal ist und das $m$ in so wenigen $f_i g_i$ wir möglich auftaucht. Da $\mathrm{lt}\,\phi(f) \notin \langle \mathrm{lt}(G) \rangle$ gilt, und $m$ durch $\mathrm{lt}(g_i)$ teilbar ist, also muss es in min. zwei verschiedenen $\mathrm{lt}(f_i g_i)$ auftauchen (damit es sich Wegkürzen kann), oBdA, in $\mathrm{lt}(b_1 g_1) = n_1 m_1$ und $\mathrm{lt}(b_2 g_2) = n_2 m_2$ wobei $n_i$ ein Term von $f_i$ ist und $m_i = \mathrm{lt}(g_i)$ gilt. Da sich $n_1 m_1$ und $n_2 m_2$ nur um ein Skalar unterscheiden (wegen des Wegkürzens) gilt $m_2 / \operatorname{ggT}(m_1, m_2) | n_1$. Nun gilt $m_{2,1} = \mathrm{lm}(g_2) / \operatorname{ggT}(\mathrm{lm}(g_1), \mathrm{lm}(g_2)) = m_2 / \operatorname{ggT}(m_1, m_2)$ also $n m_{2,1} = n_1$. Nun schreiben wir $m_{2,1} g_1 - m_{1,2} g_2 = \sum t_i g_i + r$ (Division mit Rest). Nach Voraussetzung $r = (\overline{m_{2,1} g_1 - m_{1,2} g_2})^G = 0$. Betrachte nun ein neues $f$:

$$f' := f - n(\sigma_{2,1} - \sum t_i e_i) = \sum f'_i e_i.$$

Nach Annahme/ Konstruktion gilt $\phi(f') = \phi(f)$. In der In dieser Darstellung gibt es nun weniger $m$, also war die Minimalität verletzt. (Nach Konstruktion kürzt sich $n_i e_1$ in $f_1 e_1$ mit $n m_{2,1} e_1$ in $n\sigma_{1,2}$. Der andere Term in $\sigma_{1,2}$ ändert nur den Koeffizienten in $n_2 e_2$, alle anderen Terme sind kleiner). □

Damit erhalten wir dann sofort Buchbergers Algorithmus um Gröbnerbasen zu berechnen:

ALGORITHMUS 3.57: Input: $g_1, \ldots, g_t \in F$.
Output: Gröbnerbasis
1: Berechne $S := \{(\overline{S(g_i, g_j)})^G | i, j\}$ für alle zulässigen Paare $i \neq j$.
2: Falls $Si \setminus \{0\} = \{s_1, \ldots, s_r\} \neq \{\}$ dann $g_{t+i-1} := s_i$ $(1 \leq i \leq r)$, $t := t + r$ und gehe nach (1).

Da das $\langle M \rangle$ in jedem Schritt echt grösser wird, muss der Algorithmus terminieren (Noethersch!). Die Korrektheit folgt dann aus Buchbergers Kriterium.

So ist der Algorithmus offensichtlich nicht so optimal - wir würden viele Paare mehrfach testen. Ausserdem gibt es noch viele gute Ideen um gewisse Paare ausschliessen zu können (wenn man weiss das die Reduktion 0 liefern wird). Andererseits ist dies immer noch die Grundstruktur des Algorithmus.

BEISPIEL 3.58: Sei $f := x^2y^2 - x^2y + x$ und $g := x^2y + y^2$. Dann gilt $S(f,g) = -x^2y + x - y^3$. Division mit Rest liefert eine Rest von $S(f,g) + g = x - y^3 + y^2$, also ist $G = \{f, g\}$ keine Gröbnerbasis.

Nächster Versuch: $G = \{f, g, h := x - y^3 + y^2\}$. $S(f,g) \to 0$, $S(f,h) = -x^2y + xy^5 - xy^4 + x \to_g xy^5 - xy^4 + x + y^2 \to_h -xy^4 + x + y^8 - y^7 + y^2 \to_h x + y^8 - 2y^7 + y^6 + y^2 \to_h y^8 - 2y^7 + y^6 + y^3$ $S(g,h) = xy^4 - xy^3 + y^2 \to_h -xy^3 + y^7 - y^6 + y^2 \to_h y^7 - 2y^6 + y^5 + y^2$. Also brauchen wir noch zwei weitere Polynome $y^8 - 2y^7 + y^6 + y^3$ und $y^7 - 2y^6 + y^5 + y^2$, diese haben einen ggT von $y^7 - 2y^6 + y^5 + y^2$.

Daher haben wir eine Gröbnerbasis von

$$f, g, h, y^8 - 2y^7 + y^6 + y^3, y^7 - 2y^6 + y^5 + y^2, \dots$$

Wenn wir uns die Leitterme ansehen, so folgt

$$h = x - y^3 + y^2, y^7 - 2y^6 + y^5 + y^2$$

ist eine reduzierte Basis.

Wie schon angekündigt, wollen wir jetzt noch die Relationen (Syzygien) bestimmen: Zusätzlich zu der Notation für Satz 3.56 sei für $i < j$ wo $\operatorname{lt} g_i$ und $\operatorname{lt} g_j$ das selbe Basiselement involvieren noch

$$\tau_{i,j} := m_{j,i}e_i - m_{i,j}e_j - \sum f_u^{(i,j)} e_u$$

definiert mit $m_{j,i}g_i - m_{i,j}g_j = \sum f_u^{(i,j)} g_u + h_{i,j}$ für eine beliebige (aber fest gewählte) Darstellung aus der Division mit Rest. Dann gilt:

SATZ 3.59 (Schreyer): Angenommen, $G = \{g_1, \dots, g_t\}$ ist eien Gröbnerbasis. Sei $>$ die Termordnung auf $R^t = \oplus Re_i$ mit

$$me_i > ne_j :\iff \operatorname{lt}(mg_i) > \operatorname{lt}(ng_j) \text{ or } \operatorname{lt}(mg_i) = \operatorname{lt}(ng_j) \text{ und} i < j.$$

Dann erzeugen die $\tau_{i,j}$ alle Relationen, sie bilden sogar eine Gröbnerbasis für den Syzygien-Modul für die angegebene Termordnung. Ferner gilt $\operatorname{lt}(\tau_{i,j}) = m_{i,j}e_i$

BEWEIS. Zuerst zeigen wir $\operatorname{lt}(\tau_{i,j}) = m_{i,j}e_i$. Es gilt

$$m_{j,i} \operatorname{lt} g_i = m_{i,j} \operatorname{lt} g_j$$

und die Terme sind grösser als alle Terme die in $f_u^{(i,j)} g_u$ auftauchen (da dies der „Rest“ bei der Division durch die Gröbnerbasis war). Daher muss $\operatorname{lt} \tau_{i,j}$ entweder $m_{j,i}e_i$ oder $-m_{i,j}e_j$ sein. Nach Def. gilt $i < j$ und daher $m_{j,i}e_i > m_{i,j}e_j$ und die Aussage ist gezeigt.

Als nächstes wollen wir zeigen, dass die $\tau_{i,j}$ eine Gröbnerbasis bilden. Sei dazu $\tau = \sum f_u e_u$ eine bliebige Syzygie (Relation), wir müssen nun zeigen, dass $\operatorname{lt} \tau$ durch ein $\operatorname{lt} \tau_{i,j}$ teilbar ist. Setze $n_v e_v := \operatorname{lt}(f_v e_v)$. Da diese Terme verschiedene Basiselemente involvieren können sie sich nicht wegkürzen, also gilt $\operatorname{lt} \sum f_v e_v = n_i e_i$ für ein $i$. Sein nun $\sigma := \sum' n_v e_v$ wobei die Summe über alle $v$ geht mit $n_v \operatorname{lm} g_v = n_i \operatorname{lm} g_i$. Für diese $v$ muss $v \geq i$ gelten, da $n_i e_i$ der Leitterm von $\tau$ ist. Daher muss nun $\sigma$ eine Relation für die Monome $\operatorname{lm} g_v$ $v \geq i$ sein. Nach Lemma 3.48 ist $\sigma$ eine Linearkombination von $\sigma_{u,v}$ mit $u, v \geq i$. Da speziell $e_i$ gebraucht wird, so reichen sogar die $\sigma_{i,j}$ mit $j > i$. Also ist $n_i \in \langle m_{j,i} | j > i \rangle$ und wir sind fertig. □

BEISPIEL 3.60: Sei $R = \mathbb{Q}[x, y]$, $x > y$ in $>_{\text{lex}}$ und $g_1 := x^2$, $g_2 := xy + y^2$. Wir wollen eine Gröbnerbasis und den Syzygien-Modul bestimmen.

$$S(g_1, g_2) = (\text{lt}(g_2)/x)g_1 - (\text{lt}(g_1)/x)g_2 = -xy^2$$

(wegen $x = \text{ggT}(\text{lt}(g_1), \text{lt}(g_2)) = \text{ggT}(x^2, xy)$). Division mit Rest liefert $g_3 := y^3$ und die Syzygie

$$\tau_{1,2} := ye_1 - xe_2 + ye_2 - e_3$$

Da $g_1$ und $g_3$ Monome sind, so erhalten wir eine Syzygie aus Lemma 3.48 als

$$\tau_{1,3} := y^3e_1 - x^2e_3$$

Da wir (für Satz 3.56) das Paar $(g_1, g_2)$ bereits getestet haben, (und $(g_1, g_3)$ ebenfalls) brauchen wir noch $(g_2, g_3)$:

$$S(g_2, g_3) = (\text{lt}(g_3)/y)g_2 - (\text{lt}(g_2)/y)g_3 = y^4$$

Die Division liefert dann einen Rest von 0 und die Syzygie

$$\tau_{2,3} := y^2e_2 - xe_3 - ye_3$$

Satz 3.56 zeigt nun, dass $\{g_1, g_2, g_3\}$ eine Gröbnerbasis bilden und das $\{\tau_{1,2}, \tau_{1,3}, \tau_{2,3}\}$ eine Gröbnerbasis für den Syzygien-Modul sind. Wenn wir nun lieber Relationen für $g_1$ und $g_2$ wollen, so müssen wir $e_3$ noch eleminieren. Wegen $e_3 = ye_1 - xe_2 + ye_2 - \tau_{1,2}$ ist dies einfach.

## 9. Nicht lineare Gleichungen

DEFINITION 3.61: Gegeben sind $f_1, f_s \in R = K[X_1, \ldots, X_n]$. Dann ist die Lösungsmenge

$$v(f_1, \ldots, f_s) := \{x \in K^n : \forall i : f_i(x) = 0\}.$$

Für ein Ideal $I$ sei analog $v(I) := \{x \in K^n : \forall f \in I : f(x) = 0\}$.

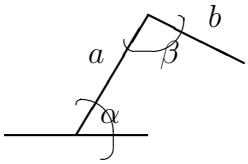
Falls $I = \langle f_1, \ldots, f_s \rangle$ so gilt $v(I) = v(f_1, \ldots, f_s)$. Also reicht es $v(G)$ zu berechnen falls $G$ ein beliebiges (endliches) Erzeugendensystem ist, z.b. eine Gröbnerbasis.

BEISPIEL 3.62: $K = \mathbb{C}$ und $f_1 := x^2 + y^2 + z^1 - 1$, $f_2 = x^2 + y^2 - z$, $f_3 := x - y \in \mathbb{C}[x, y, z]$. Als erstes berechnen wir eine lex-Gröbnerbasis

$$g_1 = f_3 = x - y, g_2 = 2y^2 - z = f_2 - (x + y)f_3, g_3 = z^2 + z - 1 = f_1 - f_2$$

Dann bestimmen wir alle Lösungen für $z$: $z = 1/2(-1 \pm \sqrt{5})$ setzen dies in $g_2$ ein um $y$ zu bestimmen und dann $x$.

BEISPIEL 3.63 (Roboter): Gröbnerbasen werden z.B. eingesetzt um Robotarme zu positionieren - zumindest wurden sie in der Forschung dafür eingesetzt.




Wir nehmen an, dass der Robotarm bei $(0, 0)$ anfängt. Gesucht sind die Winkel $\alpha$ und $\beta$ in Abhängigkeit der Handposition (Ende des Arms) $(x, y)$. Dann gilt $u =$

$a\cos\alpha, v = a\sin\alpha$ und $x - u = b\sin\gamma$, $v - y = b\cos\gamma$, $\gamma + (\pi/2 - \alpha) = \beta$. Hier betrachten wir zunächst $s_\alpha := \sin\alpha$, $c_\alpha := \cos\alpha$, $s_\gamma := \sin\gamma$ und $c_\gamma := \cos\gamma$ als Unbekannte und $x$, $y$, $a$, $b$ als Parameter. Damit können wir die möglichen Winkel/ Positinen für jedes $(x, y)$ als Ideal beschreiben. $I \subseteq \mathbb{R}(a, b)$ beschreiben, mit $I := \langle x - ac_\alpha - bs_\gamma, as_\alpha - y - bc_\gamma, s_\alpha^2 + c_\alpha^2 - 1, s_\gamma^2 + c_\gamma^2 - 1\rangle$ Punkte in der Varietät $\{\underline{a} := (a_\alpha, c_\alpha, a_\gamma, b_\gamma) \in \mathbb{R}^4 | f(a) = 0 \forall f \in I\}$ entsprechen dann den Lösungen. Wenn wir wollen, so können wir auch noch $(x, y)$ als Parameter einführen: Für gegebenes $(x, y)$ und $(a, b)$ kann nun „einfach“ eine Gröbnerbasis berechnet werden um „einfache“ Gleichungen (univariat) für $\cos\gamma = c_\gamma$ zu erhalten. Damit können dann die anderen Variblen berechnet werden. Wir können dies mal für $a = 3\sqrt{2}$, $b = \sqrt{5}$ und $x = 4$, $y = 1$ durchrechnen. Ein Gröbnerbasis für $I$ (lex, $s_\alpha > c_\alpha > s_\gamma > c_\gamma$) ist dann

$$s_\alpha - 1/6\sqrt{2}\sqrt{5}c_\gamma - 1/6\sqrt{2}, c_\alpha + 1/24\sqrt{2}\sqrt{5}c_\gamma - 7/12\sqrt{2}, s_\gamma - 1/4c_\gamma - 1/10\sqrt{5}, c_\gamma^2 + 4/85\sqrt{5}c_\gamma - 76/85$$

Dies gibt sofort zwei Lösungen für $c_\gamma$, alle anderen Variabeln hängen davon dann linear ab. Es gibt damit zwei Lösungen: $s_\alpha = 1/\sqrt{2}$ und damit $\alpha = \pi/4 = 45°$, $s_\gamma = 2/\sqrt{5}$ und $\gamma = \pi/6 = 30°$. Die Andere ist $s_\alpha = 23/34\sqrt{2}$, $s_\gamma = -38/85\sqrt{5}$.

In „realen“ Anwendungen haben wir oft mehr Gelenke (also mehr Variablen) und arbeiten in 3 Dimensionen. Damit sind die resultierenden Gleichungen dann komplizierter und es ist ein Problem die univariate Gleichung vom hohen Grad explizit und schnell zu lösen.

Um jetzt noch Gleichungssyteme etwas abstrakter behandeln zu können brauchen wir noch ein paar Ergebnisse:

LEMMA 3.64: Sei $F \geq K$ ein Erweiterungskörper, $f_1, \ldots, f_s \in K[\underline{X}]$, $I := \langle f_1, \ldots, f_s\rangle_K < K[\underline{X}]$ und $G$ eine Gröbnerbasis von $I$. Dann ist $I$ auch eine Gröbnerbasis von $\langle f_1, \ldots, f_s\rangle_F \leq F[\underline{X}]$

BEWEIS. Folgt direkt aus dem Buchbergerkriterium da die Division nicht von dem Körper abhängt: es werden nur Operationen mit den Koeffizienten durchgeführt. □

SATZ 3.65 (Hilberts Nullstellensatz): Sei $K$ algebraisch abgeschlossen, $I \leq R$ ein Ideal. Dann gilt $v(I) = \emptyset$ genau dann, falls $I = R$ ist.

BEWEIS. Kommutative Algebra. □

KOROLLAR 3.66: Sei $F \geq K$ eine Körpererweiterung wobei $F$ algebraisch Abgeschlossen ist und $G$ eine Gröbnerbasis von $I = \langle f_1, \ldots, f_s\rangle$ mit $\mathrm{lc}(f) = 1$ für jedes $f \in G$. Dann gilt

$$v_F(G) = \emptyset \iff 1 \in I$$

wobei $v_F(G) := v(\langle f_1, \ldots, f_s\rangle_F)$, die Nullstellenmenge über $F$ ist.

BEWEIS. Sei $v_F(G) = \emptyset$ dann gilt $1 \in \langle G\rangle_F$. $G$ ist Gröbnerbasis, also muss es $g \in G$ geben mit $\mathrm{lt}(g) | \mathrm{lt}(1) = 1$. Wegen $\mathrm{lc}(g) = 1$ folgt daher $g = 1$ und $1 \in G \subseteq I$.

Andererseits, sei $1 \in I$. Dann ist natürlich sofort $1 \in \langle I\rangle_F$ und damit $v_F(G) = \emptyset$. □

BEISPIEL 3.67: Sei $K = \mathbb{R}[x]$ und $G = \{x^2 + 1\}$. Dann ist $G$ eine Gröbnerbasis, $\langle G\rangle \neq R$ aber $v(G) = \emptyset$. Die algebraische Abgeschlossenheit ist also wirklich notwendig.

BEMERKUNG 3.68: Sei $I := \langle f_1, \ldots, f_s \rangle \neq \{0\}$ und $G$ eine Gröbnerbasis von $I$. Dann ist $R/I$ ein $K$-Vektorraum mit Basis

$$\{f + I : f \in R \text{ ist Monom und } \forall g \in G : \mathrm{lt}(g) \not| f\}$$

BEWEIS. Die einzige Aussage die zu zeigen ist ist die Basis.

Sei $h \in R$ beliebig. Dann ist $h \equiv \bar{h}^G \bmod I$. Nach Bemerkung 3.44 ist dann $\bar{h}^G \in [B]_K$ eine $K$-linear Kombination. (Bemerkung 3.44 charkterisiert den Rest als „durch keinen Leitterm von $G$ Teilbar. Alle Monome mit dieser Eigenschaft sind sind $B$.) Also ist $B$ ein Erzeugendensystem für $R/I$.

Angenommen, wir haben $h_i + I \in B$, $a_i \in K$ mit $\sum a_i h_i + I = 0$. Das impliziert dann $\sum a_i h_i \in I$ also $\overline{\sum a_i h_i}^G = 0$ und schliesslich $a_i = 0$ mit Bemerkung 3.44. Also sind die $h_i$ unabhängig. □

SATZ 3.69: Sei $K$ algebraisch abgeschlossen, $I \leq R$. Dann ist $R/I$ endlich dimensional als $K$-Vektorraum genau dann, falls $v(I)$ endlich ist.

BEWEIS. Kommutative Algebra oder Übung? □

KOROLLAR 3.70: Sei $F \geq K$ und $F$ algebraisch abgeschlossen. Ferner sei $I := \langle f_1, \ldots, f_s \rangle$, $J := IF[\underline{X}]$ und $G$ eine Gröbnerbasis für $I$ mit $\mathrm{lt}(g) = 1$ für jedes $g \in G$. Dann ist $v_F(I)$ endlich genau dann, falls es für jedes $X_i$ ein $g \in G$ und $w \in \mathbb{N}$ gibt mit $\mathrm{lt}(g) = X_i^w$.

BEWEIS. □

KAPITEL 4

# Faktorisierung und Irreduzibilität

Wir wollen für unsere „normalen“ Ringe $\mathbb{Z}$, $\mathbb{F}_q[x]$ und $\mathbb{Q}[x]$ die zerlegung in Primelemente oder Irreduzible untersuchen. Es wird sich herausstellen, dass es

- einfach ist (wahrscheinlich) festzustellen ob eine Zahl Prim ist oder nicht
- schwierig ist eine Zahl zu faktorisieren
- einfach ist ein Polynom über $\mathbb{F}_q$ zu faktorisieren
- irreduzibilität in $\mathbb{F}_q[x]$ ist wie faktorisieren
- $\mathbb{Q}[x]$ folgt aus $\mathbb{F}_q[x]$
- $\mathbb{Z}[x]$ ist so kompliziert wie beide zusammen...

## 1. Quadratfreie Faktorisierung

Sei $R$ ZPE-Ring. Für den Rest dieses Abschnittes werden wir immer $R[x]$ betrachten.

DEFINITION 4.1: Die Abbildung

$$()' : R[x] \to R[x] : f = \sum_{i=0}^{n} f_i x^i \mapsto \sum_{i=1}^{n} i f_i x^{i-1}$$

heißt *formale Ableitung*

Wie in der Analysis/ GdM kann hier, rein algebraisch, gezeigt werden, dass die „normalen“ Regeln gelten:

- Ableiten ist $R$-linear: $(af + bg)' = af' + bg'$
- Produktregel: $(fg)' = f'g + g'f$

DEFINITION 4.2: Sei $0 \neq f \in R[x]$. $f$ heißt qudratfrei, falls aus $g \in R[x]$ mit $g^2 | f$ immer $g \in R$ folgt. Eine Darstellung $f = \prod f_i^i$ für $f_i \in R[x]$ heißt quadratfreie Faktorisierung falls

- Die $f_i$ sind paarweise Teilerfremd: $\mathrm{ggT}(f_i, f_j) \in R$
- Die $f_i$ sind quadratfrei.

Da $R$ nur ein Ring ist, so ist der ggT über Pseudodivision zu berechnen.

BEMERKUNG 4.3: Sei $R$ Integritätsring mit $\operatorname{char} R = p > 0$. Dann gelten:

(1) Die Abbildung $:F : R \to R : x \mapsto x^p$ ist ein Ringhomomorphismus
(2) Ist $|R| < \infty$, also $R$ ein endlicher Ring, so ist $F$ bijektiv.

BEWEIS. (1) $F(xy) = (xy)^p = x^p y^p = F(x)F(y)$ da $R$ kommutativer Ring. Für die Summe benutzen wir die Binomische Formel:

$$F(x+y) = (x+y)^p = \sum_{i=0}^{p} \binom{p}{i} x^i y^{p-i}$$

Wegen $p|\binom{p}{i}$ für $1 \leq i < p$ folgt dann die Behauptung.

(2) Da $F$ offensichtlich injektiv ist, so folgt die Behauptung aus der Tatsache, dass $R$ endlich ist.
Wenn wir benutzten, dass ein endlicher Integritätsring immer ein Körper ist, so können wir das sogar konstruktiv zeigen: Nun gilt $|R| = q = p^r$ und damit für $x \neq 0$ immer $x^{q-1} = 1$, also $x^q = x$. Damit folgt $(x^{p^{r-1}})^r = x^q = x$ und wir haben das Urbild sogar gefunden.

□

BEMERKUNG 4.4: Sei $f = \prod f_i^i \in R[x]$ mit quadratfreier Faktorisierung. Dann gilt:

(1) Für $\operatorname{char} R = 0$ gilt $\operatorname{ggT}(f, f') = r \prod f_i^{i-1}$ für $r \in R$ also ist speziell $f$ quadratfrei wenn $\operatorname{ggT}(f, f') \in R$ gilt.
(2) Für $\operatorname{char} R = p > 0$ und $R^p = R$ gilt $\operatorname{ggT}(f, f') = r \prod_{p \nmid i} f_i^{i-1} \prod_{i|p} f_i^i$ für $r \in R$

BEWEIS. Zunächst gilt $(g^i h)' = i g^{i-1} g' h + g^i h$ damit folgt dann sofort $f_i^{i-1} | \operatorname{ggT}(f, f')$ und, da die $f_i$ paarweise teilerfremd sind, auch $\prod f_i^{i-1} | \operatorname{ggT}(f, f')$. Angenommen, $f = g^i h$ mit $g$ irreduzibel, $\operatorname{ggT}(g, h) \in R$ und $g^i | f'$. Wegen $f' = i g^{i-1} g' h + g^i h'$ folgt dann sofort $g | i g' h$ was aber nicht sein kann - außer $p|i$ oder $g' = 0$ und $\operatorname{char} R = p$: $\operatorname{ggT}(g, h) \in R$ und $\deg g' < \deg g$. Falls $\operatorname{char} R = p$ und $g' = 0$, so ist $g = \sum g_i x^{p^i}$. Wegen $R^p = R$ ist aber dann $g = h^p$, also ist $g$ nicht irreduzibel. In $\operatorname{char} R = 0$ gilt $g' = 0$ genau dann, falls $g \in R$ ist. Also folgt (1).

Für (2) beachten wir, dass $(f_p^p)' = p f_p^{p-1} f' = 0$ gilt und daher $f_p^p$ den ggT teilt. □

BEISPIEL 4.5: Sei $R := \mathbb{F}_p[y]$ und $f := x^p - y \in R[x]$. Dann ist $f$ irreduzibel (nach Eisenstein mit $y$), also ist die quadratfreie Faktorisierung $f = f_1$, aber $f' = 0$ und damit $\operatorname{ggT}(f, f') = f$. Hier gilt nicht $R^p = R$ da $y$ keine $p$-te Potenz ist.

Einen Algorithmus um die quadratfreie Faktorisierung zu erhalten haben wir bereits bei den Übungsaufgaben zum ggT gesehen. Für positive Charakteristik muss der Algorithmus noch abgeändert werden um ggf. $p$-te Wurzeln aus den Polynomen zu ziehen.

BEISPIEL 4.6: $R = \mathbb{Z}$, $f := x^8 - 2x^6 + 2x^2 - 1$. Dann gilt $f_3 := \operatorname{ggT}(f, f') = x^4 - 2x^2 + 1 = (x^2 - 1)^2$ und $f/f_3 = x^4 - 1 = (x^2 - 1)(x^2 + 1)$. Damit erhalten wir $f = (x^2 - 1)^3 (x^2 + 1)$.

BEISPIEL 4.7: $R = \mathbb{F}_3$, $f = x^{11} + 2x^9 + 2x^8 + x^6 + x^5 + 2x^3 + 2x^2 + 1$.

Dann ist $h := \operatorname{ggT}(f, f') = x^9 + 2x^6 + x^3 + 2$ und $f/h = x^2 + 2$, $\operatorname{ggT}(x^2 + 2, h) = x + 2$, damit gilt $f_1 = x + 1$. Jetzt machen wir mit $f \leftarrow h$ weiter: $f' = 0$, und $f = (x^3 + 2x^2 + x + 2)^3$. Also nun $f \leftarrow x^3 + 2x^2 + x + 2$.

$\operatorname{ggT}(f, f') = 1$ also ist das neue $f$ Quadratfrei - jedoch nicht koprim zu bekannten Faktoren. Es gilt $x^3 + 2x^2 + x + 2 = (x+2)(x^2+1)$. Insgesamt folgt $f = (x+1)(x^2+1)^3(x+2)^4$.

ALGORITHMUS 4.8: Input: $f \in R[x]$
Output: qudratfreie Faktorisierung $f_i$
1 Falls $f' = 0$ gilt, so schreibe $f = g^p$ und rufe den Algorithmus für $g$ auf: $g = \prod g_i^i$, also $f = g^p = \prod g_i^{pi}$.
2 $c := \mathrm{ggT}(f, f')$ (also $c = \prod f_i^{i-1}$)
3 $w := f/c$ (also $w = \prod f_i$)
4 $i := 1$
5 while $\deg w \neq 0$ do
6 $y := \mathrm{ggT}(w, c)$
7 $z := w/y$
8 $f_i := z$
9 $i := i + 1$
10 if $c' = 0$ then break
11 $c := c/y$
12 $w := y$
13 Rufe den Algorithmus für $\sqrt[p]{c}$ auf, also $c = \prod c_i^{pi}$ und schreibe $f$ entsprechend.

## 2. Faktorisierung über $\mathbb{F}_q$

In diesem Abschnitt sei $R = \mathbb{F}_q$ mit $q = p^r$, also ein endlicher Körper der Charakteristik $p > 0$. Wir wollen Polynome faktorisieren, wie wir gesehen haben, reicht es hier quadratfreie Polynome zu betrachten.

LEMMA 4.9: Sei $K \geq \mathbb{F}_q$ eine Körpererweiterung. Dann ist $F_q : K \to K : x \mapsto x^q$ ein Körperhomomorphismus mit Fixkörper $\{x \in K : F_q(x) = x\} = \mathbb{F}_q$.

BEWEIS. Es gilt $F_q = F^r$ für $F$ wir in Bemerkung 4.3 und $q = p^r$, also ein Homomorphismus. Das Polynom $t^q - t$ hat höchstens $q$ Nullstellen in $K$, aber für $x \in \mathbb{F}_q$ gilt $x^q = x$, damit ist die Aussage dann klar. □

LEMMA 4.10: Die Abbildung $F_q : \mathbb{F}_q[t] \to \mathbb{F}_q[t] : f \mapsto f^q$ ist ein Ringhomomorphismus. Für jedes Ideal $I \leq \mathbb{F}_q[t]$ gilt $F_q(I) \subseteq I$, also induziert $F_q$ einen Ringhomomorphismus

$$\mathbb{F}_q[t]/I \to \mathbb{F}_q[t]/I : x + I \mapsto F_q(x) + I$$

BEWEIS. Klar, da $F_q(f) = f^q$, also $f \in I$ impliziert $f^q \in I$. Damit ist die Abbildung wohldefiniert, der rest ist dann Klar. □

SATZ 4.11: Sei $f \in \mathbb{F}_q[t]$ mit $\deg f \geq 1$, $f = \prod_{i=1}^{s} f_i$ mit $f_i$ paarweise koprim und irreduzibel. Seien $V := \mathbb{F}_q[t]/f$, $K_i := \mathbb{F}_q[t]/f_i$ und

$$\phi : V \to \times K_i : g + \langle f \rangle \mapsto (g + \langle f_i \rangle)_i$$

Dann gilt:

(1) $\phi$ ist ein Ringisomorphismus
(2) $K_i$ ist ein Erweiterungskörper von $K$ vom Grad $K_i : K = \deg f_i$
(3) $\phi$ kommutiert mit $F_q$, also $\phi(F_q(g) + (f)) = (F_q(g) + (f_i))_i$

$$\begin{array}{ccc} V & \xrightarrow{\phi} & \times K_i \\ {\scriptstyle F_q}\downarrow & & \downarrow{\scriptstyle F_q \times \ldots \times F_q} \\ V & \xrightarrow{\phi} & \times K_i \end{array}$$

Kommutiert.
(4) $W := \{v \in V : F_q(v) = v\} \simeq \{(x_i)_i : F_q(x_i) = x_i\}$ ist ein $s$-dimensionaler $\mathbb{F}_q$-Vektorraum.

BEWEIS. (1) Dies ist der Chinesische Restsatz für Polynome.
(2) Da $f_i$ irreduzibel ist, so ist $K_i = \mathbb{F}_q[t]/f_i$ ein Körper vom Grad $\deg f_i$ (AGS)
(3) In Lemma 4.10 haben wir gezeigt, dass $F_q$ ein Homomorphismus auf den Restklassenringen induziert. Der Rest folgt dann aus der Definition.
(4) Lemma 4.9 zeigt $\{x \in K_i : x^q = x\} = \mathbb{F}_q$, damit folgt dann $W = \mathbb{F}_q^s$.

□

KOROLLAR 4.12: Mit den Bezeichnungen von Satz 4.11 gilt:

(1) $f$ irreduzibel $\iff \dim_{\mathbb{F}_q} W = 1$
(2) Sei $\dim_{\mathbb{F}_q} W = s > 1$ und für $1 \leq i \neq j \leq s$ seien $a_i \neq a_j \in \mathbb{F}_q$, sowie $g \in \mathbb{F}_q[t]$ mit $\deg g < \deg f$ und $g + (f_i) = a_i + (f_i)$, $g + (f_j) = a_j + (f_j)$. Dann gelten:
   (a) $f_i | \operatorname{ggT}(g - a_i, f)$, $f_j \nmid \operatorname{ggT}(g - a_i, f)$
   (b) $1 \leq \deg \operatorname{ggT}(g - a_i, f) < \deg f$

BEWEIS. (1) Satz 4.11.(4)
(2) Nach Vorraussetzung: $f_i | g - a_i$ und $g - a_i \equiv a_j - a_i \bmod f_j$ aber $0 \neq a_i - a_j \in \mathbb{F}_q$, also $f_j \nmid g - a_i$. Der Rest folgt dann aus (1).

□

Damit erhalten wir den Berlekamp Algorithmus zur Faktorisierung von Polynomen über $\mathbb{F}_q$: Berechne eine Basis $g_i \in \mathbb{F}_q[t]$ von $W$ mit $1 \leq \deg g_i < \deg f$ $(2 \leq i \leq s)$ und $g_1 := 1$. Falls $s = 1$, so ist $f$ irreduzibel. Sonst: berechne $\operatorname{ggT}(g_i - a, f)$ für alle $a \in \mathbb{F}_q$. Dies muss alle Faktoren liefern.

Also Frage: wie finden wir die $g_i$, die Basis von $W$? Mit linearer Algebra!

LEMMA 4.13: Sei $f \in \mathbb{F}_q[t]$ mit $\deg f = m \geq 1$ gegeben./ Sei $T := t + (f) \in \mathbb{F}_q[t]/f$. Dann bildet $1, T, \ldots, T^{m-1}$ eine $\mathbb{F}_q$-Basis für $F_q[t]/f$. (AGS). Setze $Q = (q_{i,j})_{i,j} \in \mathbb{F}_q^{m\times m}$ mit

$$T^{iq} = \sum q_{i,j} T^j$$

Dann gilt für $0 \neq g = \sum_{i=0}^{m-1} b_i t^i \in \mathbb{F}_q[t]$:

$$g^q = g \bmod f \iff (b_0, \ldots, b_{m-1})(Q - I_m) = 0$$

Also finden wir $g_i$ als Basis des Kerns von $Q - I_m$.

BEWEIS. Im wesentlichen direkte Folge davon, dass $F_q$ ein Homomorphismus ist: $g^q = (\sum b_i t^i)^q = \sum b_i^q t^{iq} = \sum b_i t^{iq} = \sum_i b_i \sum_j q_{i,j} t^j = \sum_j (\sum_i b_i q_{i,j}) t^j \bmod f$. Also $g^q = g \bmod f$ genau, dann, falls $\sum_i b_i q_{i,j} = b_j$ gilt für alle $j$. □

Damit können wir jetzt den Algorithmus komplettieren:

ALGORITHMUS 4.14: Input: $f \in \mathbb{F}_q[t]$, normiert, quadratfrei, $\deg f = m$
Output: $f = \prod f_i$ mit $f_i$ irreduzibel
1 Berechne $Q \in \mathbb{F}_q^{m\times m}$ wie oben
2 Berechne $\ker Q - I_m$ und damit eine Basis $g_i$ $(1 \leq i \leq s)$ von $W$.
3 Falls $s = 1$ so ist $f$ irreduzibel.
4 $F := \{f\}$
5 Für $i = 2$ bis $s$ tue
6 Für $a \in \mathbb{F}_q$ tue
7 $u_h := \mathrm{ggT}(g_i - a, h)$ für $h \in F$.
8 Falls $u_h$ nicht trivial, so setze

$$F := (F \cup \{u_h, h/u_h\}) \setminus \{h\}$$

9 Falls $|F| = s$ Faktoren gefunden sind, so sind wir fertig.

BEISPIEL 4.15: Sei $K = \mathbb{F}_2$ und $f := t^3 + 1$. Dann berechnen wir $Q$: $t^0 \bmod f = t^0$, $t^2 \bmod f = t^2$ und $t^4 \bmod f = t$. Damit ist

$$Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad Q - I_3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

Damit erhalten wir $g_1 = 1$ und $g_2 = t + 1$ und $\mathrm{ggT}(g2, f) = t + 1$, $\mathrm{ggT}(g2 + 1, f) = t^2 + t + 1$ und damit die Faktorisierung.

BEMERKUNG 4.16: Der Algorithmus terminiert und liefert die korrekte Faktorisierung.

BEWEIS. Die Terminierung ist klar da alle Schleifen endlich sind. Angenommen einer der Faktoren $h$ die gefunden worden ist nicht irreduzibel. Dann gibt es $i \neq j$ mit $f_i | h$ und $f_j | h$. Da die $g_i$ eine Basis für $W$ bilden ex. ein $l$ mit $g_l + (f_i) = a_i + (f_i)$ und $g_l + (f_j) = a_j + (f_j)$ und $a_i \neq a_j$, also $f_i \neq f_j \bmod g$. Nach Korollar 4.12 folgt $f_i | \mathrm{ggT}(g_l - a_i, h)$ und $f_j \nmid \mathrm{ggT}(g_l - a_i, h)$ also wird $h$ für $r = l$ zerlegt.

In jedem Schritt haben wir $\prod_{g\in F} = f$, also sind die Faktoren immer paarweise koprim (da $f$ quadratfrei ist). □

BEMERKUNG 4.17: Der Algorithmus benötigt $O(qm^4)$ Operationen in $\mathbb{F}_q$ wo $m = \deg f$ gilt.

BEWEIS. Wir brauchen $m$ Potenzierungen mit $q$ in $\mathbb{F}_q[t]$ und $m$ Divisionen um $Q$ zu berechnen, also $O(m^3 \log q)$ Operationen in $\mathbb{F}_q$. Der Kern wird mit Gauß berechnet, was $O(m^3)$ Operationen benötigt. Schliesslich brauchen wir noch $s^2 q$ viele ggT-s, also $O(m^4 q)$ Operationen. □

Wenn wir klassische Multiplikationen in $\mathbb{F}_q$ benutzen so erhalten wir $O(m^4 q r^2 \log^2 p)$ Bit-Operationen ($q = p^r$). Was wir auch sehen, ist, dass es für $q$ groß $q > 10^{100}$ nicht praktikabel ist.

Hier gibt es nun Varianten...

BEMERKUNG 4.18: Sei $f \in \mathbb{F}_q[t]$ normiert, quadratfrei und $g \in \mathbb{F}_q[t]$ mit $0 < \deg g < \deg f$, $g^q \equiv q \bmod f$ und $C := \{c \in \mathbb{F}_q : \operatorname{ggT}(g - c, f) \neq 1\}$. Dann ist $h(t) := \prod_{c \in C}(t - c)$ das normierte Polynom kleinsten Grades mit $f | h \circ g$

BEWEIS. $(h \circ g)(t) = h(g(t)) = \prod(g(t) - c)$. Sei $f_1$ ein irreduzibler Faktor von $f$, dann gilt $f_1 | g^q - g = \prod_{s \in \mathbb{F}_q}(g - s)$. Also ex. $s \in \mathbb{F}_q$ mit $f_1 | \operatorname{ggT}(g - s, f)$. Nach Konstruktion $s \in C$, also $f_1 | h \circ g$, also $f | h \circ g$.

Setze $H := \{\tilde{h} \in \mathbb{F}_q[t] r \mid f | \tilde{h} \circ g\}$. Dies ist ein Ideal in $\mathbb{F}_q[t]$ wie wir leicht nachrechnen. Da $\mathbb{F}_q[t]$ ein Hauptidealring ist, so folgt gilt $H = \langle h_0 \rangle$ für ein geeignetes $h_0$. Aber $h \in H$, also $h_0 | h$, also $h_0 = \prod_{c \in \tilde{C}}(t - c)$ für $\tilde{C} \subseteq C$. Also müssen wir noch $C = \tilde{C}$ zeigen. Sei nun $c \in C$, damit $\operatorname{ggT}(g - c, f) \neq 1$, also gibt es einen irreduziblem Teiler $f_1 | f$ mit $f_1 | g - c$ und $f_1 \not| g - \tilde{c}$ für $c \neq \tilde{c}$ (wegen $g^q = g$ folgt dies mit Satz 4.11). Wegen $h_0 \in H$ gilt $f | h_0 \circ g$ also $f_1 | \prod_{c \in \tilde{C}}(g - c)$ also $c \in \tilde{C}$ und wir sind fertig. □

Damit können wir das folgende probieren. Sei $g$ eines der Basiselemente von $W$.

(1) bereche $g^i \bmod f$ für $i \geq 0$ um die kleinste nicht-triviale Linearkombination mit $\sum a_i g^i = 0 \bmod f$ zu finden ($a_i \in \mathbb{F}_q$). Im Prinzip ist dies das Minimalpolynom von $g \in W$.
(2) Setze $h := \sum a_i t^i$, dann gilt $f | h \circ g$ und nach der Bemerkung $C = \{c \in \mathbb{F}_q : h(c) = 0\}$

Damit können wir die Suche über alle $q$ auf die Nullstellensuche für $h$ zurückführen. Also werden wir uns als nächstes mit den Nullstellen beschäftigen.

Nach Cantor-Zassenhaus:

Input: Sei $f \in \mathbb{F}_q[t]$ quadratfrei, normiert und ein Produkt von Linearfaktoren.
Output: Nullstellen von $f$.
1 Falls $\deg(f) = 1$, so gib $-f(0)$ aus.
2 Für $q$ ungerade, setze $r := t^{(q-1)/2} - 1$ und für $q = 2^e$, $r := \sum_{i=0}^{e-1} t^{2^i}$
3 Wähle zufällig $s \in \mathbb{F}_q$ und berechne $\operatorname{ggT}(f, r(t - s))$ bzw $\operatorname{ggT}(f, r(ts))$
4 Ist $h \neq f$ so berechne die Nullstellen von $f/h$ und $h$ mit dem selben Algorithmus.

Falls der Algorthmus terminiert, so findet er alle Nullstellen, also was fehlt ist die Terminierung.

BEMERKUNG 4.19: Für ungerades $q$ gilt für die Erfolgswahrscheinlichkeit in Schritt (3):

$$P(\mathrm{ggT}(f, r(t-s)) \neq 1, f) = 1 - (\frac{q-1}{2q})^n - (\frac{q+1}{2q})^n \geq 1/12,$$

für gerades $q$ sogar $P(\mathrm{ggT}(f, r(ts)) \neq 1, f) = 1 - 2(\frac{1}{2})^n > 1/2$.

BEWEIS. Sei $q$ ungerade. Wir werden später zeigen, dass $|\{x \in \mathbb{F}_q : x^{(q-1)/2} = -1\}| = (q-1)/2 = |\{x \in \mathbb{F}_q : x^{(q-1)/2} = 1\}|$ gilt. Beachte, dass $\mathrm{ggT}(f, r(t-a))(t) = \mathrm{ggT}(f(t+a), r)(t-a)$ gilt. Damit folgt $\mathrm{ggT}(f(t-s), r) = f$ genau dann, wenn $(\alpha - s)^{(q-1)/2} = 1$ gilt für alle Nullstellen $\alpha$ von $f$. Fixieren wir ein $\alpha$, so gibt es $(q-1)/2$ mögliche $s$, also können wir eins mit einer Wahrhscheinlichkeit von $(q-1)/(2q)$ zufällig wählen. Wir haben $n$ Nullstellen, also erhalten wir $((q-1)/(2q))^n$ für die Wahrscheinlichkeit alle zu vermeiden.

Analog: $\mathrm{ggT}(f(t-s), r) = 1$ falls $(\alpha - s)^{(q-1)/2} \neq 1$ gilt für alle Nullstellen $\alpha$ von $f$. Hier gibt es $(q-1)/2$ Möglichkeiten mit $\alpha^{(q-1)/2} = -1$ und zusätzlich 0, also erhalten wir eine Wahrscheinlichkeit von $((q+1)/2q)^n$. Damit ist die Aussage gezeigt.

Für gerades $q$ betrachten wir die Übung. □

BEISPIEL 4.20: Sei $K := \mathbb{F}_{3^{10}}$ und $f := t^4+t^2+2$. Dann ist $f$ quadratfrei $\mathrm{ggT}(f, f') = 1$. Für Berlekamp brauchen wir die Matrix $Q$ um $W$ zu berechnen: $1$, $t^{3^{10}} \equiv 2x$, $t^{2\cdot 3^{10}} \equiv t^2$, $t^{3\cdot 3^{10}} \equiv 2x^3 \bmod f$, also

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}, \quad Q - I_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Damit ist $g_1 = 1$, $g_2 = t^2$ eine Basis für $W$ und wir suchen 2 Faktoren.

Setzen wir $g = g_2$ so brauchen wir nun $a_i \in K$ mit $\sum a_i g^i = 0$. Wir berechnen $g_2^2 = 2x^2 + 1$ und sehen $g^2 + g + 2 = 0$. Nun brauchen wir die Nullstellen von $h := t^2 + t + 2$. $q$ ist ungerade, also ist $r := t^{(q-1)/2} - 1$.

Für $s = 1$ erhalten wir einen ggT von 1. Wählen wir nun zufällige $s \in \mathbb{F}_q$, so erhalten wir sehr häufig einen nicht trivialen ggT und die beiden Nullstellen.

Damit können wir dann wieder in der Berlekamp zurückgehen und erhalten die Faktorisierung von $f$ in 2 quadratische Faktoren.

Wir bemerken noch, dass wir natürlich nicht $r = t^{(q-1)/2} - 1$ tatsächlich berechnen - wir berechnen $(t-s)^n \bmod h$.

## 3. Das Henselsche Lemma

Wir wollen nun langsam versuchen Polynome $f$ über $\mathbb{Q}[x]$ zu faktorisieren. Wir können (und werden) annehmen, dass $f$ qudratfrei ist. Durch skalieren mit $a/b \in \mathbb{Q}$ können wir weiter annehmen, dass $f \in \mathbb{Z}[x]$ primitiv ist.

Die „Idee“, nach Zassenhaus, ist es $f$ über $\mathbb{Z}/p^k\mathbb{Z}$ zu faktorisiern, in der Hoffnung, dass wir, falls $k$ groß genug ist, damit die Faktorisierung in $\mathbb{Z}[x]$ erhalten können. Es gint nur noch ein paar Probleme:

(1) Wir können $k = 1$, also $\mathbb{F}_p$, aber nicht $k > 1$.
(2) Wir wissen nicht, wie groß $k$ sein sollte.
(3) Der Zusammenhang zwischen $\mathbb{Z}[x]$ und $(\mathbb{Z}/p^k\mathbb{Z})[x]$ ist nicht so einfach.

Also fangen wir mit (1) an. Als Bemerkung: um die modularen ggT zu berechnen haben wir Größenaussagen für Teiler von $f$ erhalten. Die werden wir im weiteren benutzen.

LEMMA 4.21: Sei $f \in \mathbb{Z}[x] \setminus \mathbb{Z}$ qudratfrei. Dann ist auch $f + p\mathbb{Z}[x] \in \mathbb{F}_p[x]$ quadratfrei für fast alle $p \in \mathbb{P}$.

BEWEIS. Falls $f$ quadratfrei ist, so gilt $\mathrm{ggT}(f, f') = 1$. Nach unseren Untersuchungen über modulare ggT Berechnungen, gilt daher $\mathrm{ggT}(f + p\mathbb{Z}[x], f' + p\mathbb{Z}[x]) = 1 \in \mathbb{F}_p[x]$ für fast alle (bis auf endlich viele Ausnahmen). Also ist $f + p\mathbb{Z}[x]$ dann quadratfrei für fast alle $p$. □

Um die Grundidee zu motivieren, gehen wir „verkehrtherum“ vor: Angenommen, $f = \prod f_i \in \mathbb{Z}[x]$ und $\phi_k : \mathbb{Z}[x] \to (\mathbb{Z}/p^k\mathbb{Z})[x]$ sei die Projektion für ein (geeignetes) fixiertes $p$. Dann gilt für $p^k > 2\|f_i\|_\infty$ und minimale Restsysteme

$$\phi_k(f_i) = f_i,$$

so dass wir hoffen können, dass die Faktorisierung modulo $p^k$ mit der in $\mathbb{Z}$ übereinstimmt - falls $k$ groß genug ist. Leider ist es ganz so einfach nicht, z.B. $f = x^4 + 1$ ist irreduzibel in $\mathbb{Z}[x]$, aber für alle $p$ und jedes $k$ hat $\phi_k(f)$ mindestens 2 Faktoren. Wir werden sehen, was dies bedeutet.

Frage: wie finden wir die Faktorisierung modulo $p^k$? Schranken habe wir ja früher schon einmal gesehen. Die Idee ist es mit einer Faktorisierung modulo $p$ anzufangen und diese dann „zu liften“. Hier wählen wie eine Primzahl so, dass $f$ modulo $p$ quadrat-frei ist. Sei nun $f = f_1 f_2 \bmod p$. Wir versuchen $g_1$, $g_2 \in \mathbb{Z}[x]$ zu finden, mit $(f_1 + pg_1)(f_2 + pg_2) = f \bmod p^2$. Ausmultiplizieren liefert $f_1 f_2 + p(g_1 f_2 + g_2 f_1) + p^2 g_1 g_2$. Also muss gelten $p^2 | f - f_1 f_2 - p(g_1 f_2 + g_2 f_1)$ und damit $p | 1/p(f - f_1 f_2) - g_1 f_2 + g_2 f_1$. Falls $f_1$ und $f_2$ teilerfremd sind, so können wir $g_i$ mit dem euklidischen Algorithmus finden. Das kann dann iteriert werden.

BEISPIEL 4.22: $f = x^3 + 10x^2 - 432x + 5040 \in \mathbb{Z}[x]$, $p = 5$. Dann gilt $f \equiv x(x^2 + 3) \bmod 5$ und $f - f_1 f_2 = 10x^2 - 435 * x + 5040$ und teilen durch $p = 5$ liefert $2x^2 - 87 * x + 1008$ und nun modulo 5: $2x^2 + 3x + 3$. Aus $1 = \mathrm{ggT}(x, x^2 + 3) = 3x \cdot x + 2 \cdot (x^2 + 3) \bmod 5$ folgt dann $2x^2 + 3x + 3 = (x + 3)x + 1(x^2 + 3)$. $g_1 := 4x + x$, $g_2 = (4x^2 + 1)$ und wir versuchen $(f_1 + 5)(f_2 + 5 \cdot (x + 3))$. Nun erhalten wir $f - (f_1 + 5)(f_2 + 5(x + 3)) = -475x + 4950 \equiv 0 \bmod 25$.

Wenn wir dies iterieren, so erhalten wir $(x + 30)(x^2 + 605x + 168) \equiv f \bmod 5^4$. Im symmtrischen Restsystem folgt dann sogar $(x + 30)(x^2 - 20x + 168) = f$.

Um dies vollständig nutzen zu können, brauchen wir eine genauere Kontrolle über den ggT, bzw. die Kofaktoren. Sonst wird es passieren, dass die Polynome die wir konstruieren wollen einen zu großen Grad haben. Schon in diesem Beispiel habe ich gemogelt: der direkte Ansatz mit aus dem euklidischen Algorithmus hätte eine Darstellung für $1/5(f - f_1 f_2)$ gegeben die Polynome von zu hohem Grad beinhaltet. Für $\mathbb{Z}$ haben wir dies schon in einer Übung gesehen, hier nur für Polynome:

LEMMA 4.23: Seien $K$ ein Körper, $g$, $h \in K[x] \setminus \{0\}$ mit $\operatorname{ggT}(g,h) = 1$. Dann ex. für alle $d \in K[x]$ eindeutige $r$, $s \in K[x]$ mit $rg + sh = d$ und $r = 0$ oder $\deg r < \deg h$. Für $\deg d < \deg g + \deg h$, so gilt $s = 0$ oder $\deg s < \deg g$.

BEWEIS. Der erweiterte euklidische Algorithmus liefert $a, b \in K[x]$ mit $ag+bhh = 1$, also $adg+bdh = d$. Division mit Rest liefert $ad = qh+r$ und $r = 0$ oder $\deg r < \deg h$. Damit folgt dann auch

$$d = adg + bdh = (qh + r)g + bdh = rg + (bd + qg)h.$$

Seien $r$, $s$ und $r'$, $s'$ wie im Lemma gegeben, also $d = rg + sh = r'g + s'h$ und somit $g(r - r') = h(s' - s)$ Wegen $\deg r < \deg h$ und $\operatorname{ggT}(g,h) = 1$ folgt dann $r = r'$ und somit $s = s'$.

Ist nun $\deg d < \deg g + \deg h$ und $\deg s \geq \deg g$ so folgt $\deg rg = \deg r + \deg g < \deg h + \deg g \leq \deg h + \deg s = \deg sh$. Damit folgt dann $\deg d = \deg(rg + sh) = \deg sh \geq \deg g + \deg h$ was nicht sein kann. □

BEMERKUNG 4.24 (Lineares Hensel Lifting): Seien $p \in \mathbb{P}$, $k > 0$ sowie $f$, $g$, $h \in \mathbb{Z}[x] \setminus \mathbb{Z}$ gegeben mit $\operatorname{ggT}(\phi_1(g), \phi_1(h)) = 1$, $\phi_k(f) = \phi_k(g)\phi_k(h)$. Dann ex. $g_1$, $h_1 \in \mathbb{Z}[x]$ mit $\phi_k(g_1) = \phi_k(g)$, $\phi_k(h_1) = \phi_k(h)$ und $\phi_{k+1}(f) = \phi_{k+1}(g_1)\phi_{k+1}(h_1)$. Sind $f$, $g$ und $h$ normiert, so können $g_1$, $h_1$ ebenfalls normiert gewählt werden.

BEWEIS. Setze $d := \phi_1(1/p^k(\phi_k(f - gh))) \in \mathbb{F}_p[x]$. Mit Lemma 4.23 finden wir nun $r$, $s \in \mathbb{Z}[x]$ mit $\deg \phi_1(r) = \deg r$, $\deg \phi_1(s) = \deg s$, $\phi_1(rg + sh) = d$ und $\deg \phi_1(r) < \deg \phi_1(h)$. Setze $g_1 := g + p^k s$ und $h_1 := h + p^k r$, damit ist die 1. Bedingung bereits erfüllt.

Weiter $f = g_1 h_1 = f - gh - p^k(rg + sh) - p^{2k}rs = p^k(1/p^k(f - gh) - (rg + sh)) - p^{2k}$. Wegen $1/p^k(f - gh) \equiv d \bmod p$ gilt $1/p^k(f - gh) - (rg + sh)) \equiv 0 \bmod p$ und damit $f \equiv g_1 h_1 \bmod p^{k+1}$ wie behauptet. Sind $f$, $g$ und $h$ normiert, so auch $\phi_k()$ für alle $k$. Ferner gilt $\deg f = \deg g + \deg h$ und damit $\deg d \leq \deg f - gh) < \deg f = \deg g + \deg h = \deg \phi_1(g) + \deg \phi_1(h)$. Mit Lemma 4.23 folgt dann $\deg r = \deg \phi_1(r) < \deg \phi_1(h) = \deg h$ und $\deg s < \deg g$, damit sind dann $g_1$ und $h_1$ wieder normiert. □

BEMERKUNG 4.25: Die Folgen die in Bemerkung 4.24 berechnet werden sind im wesentlichen Eindeutig: Seien $f$, $g$, $h$, $\tilde{g}$, $\tilde{h} \in \mathbb{Z}[x] \setminus \mathbb{Z}$ normiert mit $\operatorname{ggT}(\phi_1(g), \phi_1(h)) = 1$, $\phi_1(g) = \phi_1(\tilde{g})$, $\phi_1(h) = \phi_1(\tilde{h})$. Gibt es $k > 0$ mit $\phi_k(f) = \phi_k(g)\phi_k(h) = \phi_k(\tilde{g})\phi_k(\tilde{h})$ so folgt auch $\phi_k(g) = \phi_k(\tilde{g})$ und $\phi_k(h) = \phi_k(\tilde{h})$.

BEWEIS. Per Induktion. $k = 1$ klar: Faktorisierung modulo $p$ ist eindeutig. Sei nun $k > 1$. Wir haben $\phi_k(gh) = \phi_k(\tilde{g}\tilde{h})$, also auch $\phi_{k-1}(gh) = \phi_{k-1}(\tilde{g}\tilde{h})$ und damit per Induktion, $\phi_{k-1}(g) = \phi_{k-1}(\tilde{g})$, $\phi_{k-1}(h) = \phi_{k-1}(\tilde{h})$. Also folgt ebenfalls $\tilde{g} = g + p^{k-1}s$ und $\tilde{h} = h + p^{k-1}r$ mit $r$, $s \in \mathbb{Z}[x]$. Da $g$, $\tilde{g}$, $h$, $\tilde{h}$ normiert sind gilt $\deg g = \deg \tilde{g}$ und damit $\deg s < \deg g$ und analog $\deg r < \deg h$. Ferner $\deg \phi_1(r) < \deg \phi_1(g)$, $\deg \phi_1(s) < \deg \phi_1(h)$. Da $\phi_k(gh) = \phi_k(\tilde{g}\tilde{h})$ folgt, wie oben, $rg + sh \equiv 0 \bmod p^{k-1}$ also speziell $\phi_1(rg + sh) = 0$. Mit Lemma 4.23 folgt dann $\phi_1(r) = 0 = \phi_1(s)$, also $\phi_k(g) = \phi_k(\tilde{g})$ und $\phi_k(h) = \phi_k(\tilde{h})$ wie Behauptet. □

Mit Induktion können wir die beiden Bemerkungen natürlich auf mehr als 2 Faktoren verallgemeinern.

SATZ 4.26: Sei $f \in \mathbb{Z}[x] \setminus \mathbb{Z}$ quadratfrei, normiert, $f = gh$ mit $g$, $h \in \mathbb{Z}[x] \setminus \mathbb{Z}$. Sei $p \in \mathbb{P}$ so, dass $\phi_1(f)$ quadratfrei ist. Ferner sei $B$ mit $\|u\| \leq B$ für alle $u|f$, $u \in \mathbb{Z}[x]$ und $k$ mit $p^k > 2B$ gegeben.
Seinen nun $u_i \in \mathbb{Z}[x]$ normiert mit $\phi_k(f) = \prod \phi_k(u_i)$ und $\phi_1(u_i)$ irreduzibel. Dann exisitiert $S \subseteq \{1, \ldots, l\}$ mit: Ist $\bar{g} \in \mathbb{Z}[x]$ mit $\phi_k(g) = \phi_k(\bar{g}) = \prod_{s\in S} \phi_k(u_s)$, $\bar{g} = \sum g_i x^i$, $g_i \in [-p^k/2, p^k/2]$ so gilt $\bar{g} = g$

BEWEIS. Nach Vorraussetzung gilt $\phi_1(f) = \prod \phi_1(u_i)$ und die $\phi_1(u_i)$ sind irreduzibel. Wegen $g|f$ folgt $\phi_1(g)|\phi_1(f)$ also gibt es $S := \{i : \phi_1(u_i)|\phi_1(g)\}$. Mit Lemma 4.23, Bemerkung 4.24 können wir diese Faktoren dann liften und erhalten ein $\bar{g}$ als Produkt der gelifteten Faktoren. Wegen $\phi_k(\bar{g}) = \phi_k(g)$ folgt dann $g = \bar{g}$ aus der Eindeutigkeit und der Schranke $B$. □

Damit können wir jetzt einen Algorithmus zur Faktorisierung von Polynomen in $\mathbb{Z}[x]$ erhalten:

ALGORITHMUS 4.27: Input: $f \in \mathbb{Z}[x]$ normiert.
Output: $f_i \in \mathbb{Z}[x]$ irreduzibel mit $f = \prod f_i$
1 Berechne eine quadratfreie Faktorisierung $f = \prod g_i^i$
2 Für jedes $g_i$ tue
3 Finde eine Primzahl $p$ so, dass $g_i$ modulo $p$ quadratfrei ist.
4 Faktorisieren $\phi_1(g_i)$ mit Berlekamp, $\phi_1(g_i) = \prod_{i=1}^{l} \phi_1(f_{i,j})$
5 Lifte die Faktorisierung modulo einer geeigneten Schranke
6 Für alle $S \subseteq \{1, \ldots, l\}$ teste ob $\bar{h} := \prod_{s\in S} f_{i,j}$ ein Teiler von $f$ in $\mathbb{Z}[x]$ ist.

Das der Algorithmus korrekte Ergebnisse liefert ist nach Konstruktion klar. Ebenso ist die Terminierung offensichtlich, das einzige Problem ist die Laufzeit: es gibt irreduzible Polynome $f$ vom Grad $\deg f = n$ mit der Eigenschaft, dass für jede Primzahl $p$, $\phi_1(f)$ mindestens $n/2$ Faktoren hat. Damit können wir Polynome bauen wo $2^(n/2)$ Möglichkeiten getestet werden müssen.

Andererseits: die „meisten“ „normalen“ Polynome haben diese Eigenschaft nicht. Es kann sogar oft passieren, dass durch die Kombination mehrerer Primzahlen bereits bewiesen werden kann, dass $f$ irreduzibel ist.

Dieses Problem mit den $2^(n/2)$ Tests war die Motivation den LLL Algorithmus zu entwickeln. Damit werden wir uns als nächstes beschäftigen.

## 4. van Hoeij: Faktorisierung mit LLL

Obwohl der LLL-Algorithmus erfeunden wurde um Polynome zu faktorisieren, so war dieser Ansatz nie effektiv, in der Praxis ist er viel langsamer als die (potentiell) schlechteren Methoden. Im Jahre 2002 hat Mark van Hoeij eine andere Idee gehabt den LLL Algorithmus zu benutzen. Hier werden wir kurz skizzieren wie.

LEMMA 4.28: Die Abbildung $\phi : (R[x] \setminus \{0\}, \cdot) \to (R(x), +) : g \mapsto fg'/g$ bildet Produkte auf Summen ab: $\phi(gh) = \phi(g) + \phi(h)$.
Für $g_i | f$ gilt $\phi(g_i) \in R[x]$. Wenn wir Polynome mit den Koeffizientenvektoren identifizieren, so erhalten wir eine Abbildung von $(\langle g_i | i \rangle, \cdot) \to (R^n, +)$ für $n := \deg f$.

BEWEIS. Mit der Produktregel folgt $\phi(gh) = f(gh)'/gh = f(g'h+h'g)/gh = fg'/g + fh'/h$. Für $g|f$ gilt offensichtlich $\phi(g) \in R[x]$ und $\deg(fg'/g) = \deg f + \deg g' - \deg g \leq \deg f - 1$, damit gibt es im Bild maximal $\deg f - 1 + 1 = \deg f$ viele Koeffizienten. □

Die Idee, wir werden dies im Detail nicht beweisen, ist einfach: Wir wissen, dass Faktoren $g|f$ mit $g \in Z[x]$ „klein“ sind - speziell gilt $\|g\| < p^k$ für geeignetes $k$. Andererseits sind Polynome modulo $p^k$ i. Allg. *nicht* klein, wir erwarten immer $\|g\| \approx p^k$. Zusammen mit dem Lemma heißt dies, wir wollen das folgende Problem lösen:

Für $g_i$ mit $\phi_1(g_i)$ irreduzibel, $f = \prod g_i \bmod p^k$ finde $S \subseteq \{1, \ldots, l\}$ mit $\prod_{s \in S} g_i \in \mathbb{Z}[x]$. Mit dem Lemma heißt dies: Gegeben $G_i \in \mathbb{Z}^n$ mit $G_i = \phi(fg_i'/g_i)$ finde $e_i \in \{0, 1\}$ und $r \in p^k\mathbb{Z}^n$ mit $\| \sum e_i G_i + r \|$ klein. Dies können wir mit dem LLL Algorithmus effektiv lösen!

Bevor wir dies im Beispiel demonstrieren, noch ein paar Anmerkungen/ Beobachtungen: unter der Abbildung $\phi$ ist $\phi(g_i)$ eine Basis für den Raum der Faktoren. In diesem Raum gibt es einen Teilraum der Faktoren in $\mathbb{Z}[x]$. Mit dem oben Beschriebenen hat dieser eine Basis die aus $0, 1$-linearkonbinationen der $\phi(g_i)$ besteht. Betrachten wir jetzt eine Blockmatrix $B := \begin{pmatrix} G & p^k I_n \\ I_l & 0 \end{pmatrix}$ wo $G = (\phi(g_1), \ldots, \phi(g_l))$ ist, $n := \deg f$ gilt und die Faktoren modulo $p^k$ bekannt sind.

Auf diese Blockmatrix wenden wir den LLL Algorithmus an. Wenn alles gut klappt, so findet der Algorithmus ein $s$-Dimensionales Teilgitter mit sehr kleinen Einträgen wo $s$ die Anzahl der Faktoren in $\mathbb{Z}[x]$ ist. Wenn wir nun auf den unteren Teil der Basis dieses Teilraums den Gauß Algorithmus anwenden um eine Zeilen-Stufen-Form zu finden, so wird der untere Teil in eine Matrix mit nur $0, 1$ Einträgen transformiert. Diese geben dann die Teiler in $\mathbb{Z}[x]$ an.

BEISPIEL 4.29: Wir wollen $f = x^5 + 6x^4 - 16x^3 - 17x^2 + 133x - 143$ faktorisieren. Für $p = 11$ erhalten wir $f \equiv x(x+6)(x+9)(x^2+2x+10) \bmod 11$. Hensel-Lifting up to $11^{20}$ liefert $f \equiv (x - 24971481062072669171 3)(x + 249714810620726691719)(x - 131587204312918514646)(x^2 + 131587204312918514646x + 73716845854396012388)$.

Unter der Abbildung $\phi$: $G_1 = \phi(g_1) = (-117457436593353053658, 96925936761486559820, -5, 249714$
$G_2 = \phi(g_2) = (117457436593353053736, -96925936761486559824, -5, -249714810620726691713, 1)$

$G_3 = \phi(g_3) = (-138135309465796127067, -332408177383167577577, 190490076799347091052, 13158$
$G_4 = \phi(g_4) = (138135309465796127122, 332408177383167577547, -190490076799347091090, -13158$

Wir bauen jetzt eine Blockmatrix: $G := (g_1^t, g_2^t, g_3^t, g_4^t, 11^{20}e_1, 11^{20}e_2, 11^{20}e_3, 11^{20}e_4, 11^{20}e_5)$ $H := (I_4|0)$ und $L := \binom{G}{H}$. Wenn wir nun auf diese Matrix den LLL-Algorithmus anwenden, so finden wir 2 kurze Vektoren $G_1+G_2-G_3-G_4$ und $G_3+G_4$. Wenn wir nun davon die Zeilen-Stufen-Form berechnen, so erhalten wir $G_1+G_2$ und $G_3+G_4$ als kanonische Basis für den Teilraum. Dies entspricht dann $g_3g_4 = x^3 - 5x + 13$. Wir rechnen nach das dies ein Teiler ist. Ebenso $g_1g_2 = x^2 + 6x - 11$ ist der andere Teiler.

Was fehlt jetzt noch?

- Ein formaler Beweis, dass, wenn $k$ groß genug ist, die einzigen kleinen Elemente in $\mathbb{Z}[x]$ liegen.
- Eine formale Abschätzung für $k$
- Bessere Strategien um kleinere Gitter zu haben.
- Bessere Strategien.
- Die Komplexitätsanalyse.

Zum Beispiel, wenn wir die Faktorisierung modulo 13, 17 vergleichen, so sehen wir bereits, dass $f$ wahrscheinlich quadratisch, kubisch zerfällt. Noch besser: modulo 23 erhalten wir sofort die richtigen Grade, damit liefert dann das Lifting bereits die Faktorisierung!

Also sollten wir für ein „paar“ Primzahlen testen und dann die beste(?) benutzen. Ebenso können wir statt $\langle G_1, \dots, G_l\rangle$ Teilgitter betrachten $L_i := \langle G_1, \dots, G_i\rangle$. Wenn wir hier Teiler finden, so haben wir nur kleinere Gitter betrachtet. Ebenso können wir versuchen nich alle Koeffizienten von $\phi(g)$ zu benutzen, .... Es gibt viele Möglichkeiten zu optimieren!

## 5. Primzahltest

Hier wollen wir anfangen Faktorisierung in $\mathbb{Z}$ anzusehen. Schritt eins sind hier Primzahlen. Fangen wir mit der Beobachtung an, dass es „einfach“ ist, alle Primzahlen zu berechnen:

Bemerkung 4.30: Um alle Primzahlen $p \le n$ zu berechnen brauchen wir lediglich $O(n \log n)$ Bit-operationen - und $O(n)$ Speicher.

Beweis. Mit dem Sieb des Eratosthenes: wir legen eine Tabelle an: $a_i := 0$ für $2 \le i \le n$. Nun suchen wir den kleinsten Index mit $a_l = 0$ und setzen $a_l = 2$ und $a_{kl} = 1$ für $l < kl \le n$. Dies wird iteriert bis kein $l \le \sqrt{n}$ mehr gefunden wird. Die $l$ mit $a_l = 2$ sind dann genau die Primzahlen. Operationen: für jedes $l$ benötigen wird maximal $n/l$ viele Durchläufe. Insgesamt also $< \sum_{l\le\sqrt{n}} n/l \le n \log n$ da $\sum_{i=1}^{k} 1/i = O(\log n)$ gilt. □

Eigentlich geht die Summe nur über Primzahlen $\le \sqrt{n}$, so dass wir die Abschätzung sogar auf $O(n \log\log n)$ verbesser können wenn wir $\sum_{p\le n} 1/p = O(\log\log n)$ benutzen.

Wir erhalten auch einen ersten, naiven Algorithmus:

ALGORITHMUS 4.31: Input: $n \geq 2$
Output: `true` falls $n$ Prim ist, sonst `false`
1: Für $q = 2$ bis $\sqrt{n}$ tue
2: Falls $q|n$ dann return `false`
3: return `true`

Dieser Algorithmus benötigt $O(\sqrt{n})$ viele Divisionen, hat also eine Laufzeit die exponentiell in der Zahl der Stellen ist. Falls wir Tabellen mit Primzahlen haben, so können wir den Algorithmus beschleunigen. Wir merken auch an, dass dieser Algorithmus im Prinzip sogar mehr leisten kann: er liefert bei Bedarf, eine vollständige Faktorisierung.

**5.1. Pseudoprimzahlen.** Um den Primzahltest zu beschleunigen werden wir versuchen aus der Definition notwendige Bedingungen zu erhalten. Die Hoffnung ist, dass diese notwendigen Bedingungen einfach zu testen sind (und die nicht Primzahlen erkennen) und ggf. sogar gebündelt, hinreichend sind!

DEFINITION 4.32: Ein Primzahltest ist ein Algorithmus mit der folgenen Eigenschaft: falls der Algorithmus `true` liefert, so ist die Zahl Prim.
Ein Zusammengesetztheitstest ist, analog, ein Verfahren das, falls es `true` liefert beweist, dass $n$ keine Primzahl ist.

Also: Falls ein Zusammengesetztheitstest `true` liefert, so wissen wir das $n$ *nicht* Prim ist, sonst wissen wir nichts.

DEFINITION 4.33: Sei $n > 1$. Dann heißt $\phi(n) := |(\mathbb{Z}/n\mathbb{Z})^*|$ die Eulersche-$\phi$-Funktion und $\lambda(n) := \exp(\mathbb{Z}/n\mathbb{Z})^*$ heißt der Exponent von $(\mathbb{Z}/n\mathbb{Z})^*$.

FOLGERUNG 4.34: Sei $n = \prod p_i^{n_i}$

(1) $\phi(p^r) = (p-1)p^{r-1}$
(2) $\phi(n) = \prod \phi(p_i^{n_i})$
(3) $\lambda(p^r) = \phi(p^r)$ für $p \neq 2$
(4) $\lambda(2^r) = 1/2\phi(2^r)$ für $r \geq 3$
(5) $\lambda(n) = \text{kgV } \lambda(p_i^{n_i})$

(Folge aus der EZT) Beachte: $(\mathbb{Z}/p^n\mathbb{Z})^*$ ist zyklisch für $p \neq 2$.

Ebenso aus der EZT/ AGS wissen wir:

SATZ 4.35: Fur $a \in (\mathbb{Z}/n\mathbb{Z})^*$ gilt $a^{\phi(n)} \equiv 1 \bmod n$.
Ist $n$ eine Primzzahl, so gilt $a^{n-1} \equiv 1 \bmod n$ für jedes $1 \leq a < n$ und $a^n \equiv a \bmod n$ für jedes $a$.

Dies liefert den ersten Pseudoprimzahltest: Teste $a^{n-1} \equiv 1 \bmod n$. Falls hier etwas anderes herauskommt, so ist $n$ nicht Prim. Wir haben also eine Zusammengesetztheitstest. Als nächstes müssen wir nun untersuchen wie gut der Test ist.

BEISPIEL 4.36: $n = 91$, $a = 2$. Wir müssen $2^{90} \bmod 91$ berechnen. Es gilt $90 = 64 + 16 + 8 + 2 = (((4+1)2+1)4)+1)2$, also $a^5 = 32 \bmod 91$, $a^{10} \equiv 32^2 = 23$, $a^{10+1} \equiv 23 \cdot 2 = 46$, $()^4 \equiv (46^2)^2 \equiv 23^2 \equiv 74$, $((\,) \cdot a)^2 \equiv (74 \cdot 2)^2 \equiv 57^2 \equiv 64 \not\equiv 1 \bmod 91$ Also ist dies ein Beweis dafür, dass 91 nicht prim ist. Die Kosten dies zu verifizieren sind $O(\log n)$ Multiplikationen in $\mathbb{Z}/n\mathbb{Z}$, also klassisch $O(\log^3 n)$ bit-Operationen.

Wie gut ist dieser Test?

DEFINITION 4.37: $n \in \mathbb{N} \setminus \mathbb{P}$ heißt (Fermatsche) Pseudoprimzahl zur Basis $a$ falls $a^{n-1} \equiv 1 \bmod n$ gilt. $a \in \mathbb{N}$, $\mathrm{ggT}(a,n) = 1$ heißt Zeuge für die Zerlegbarkeit von $n$ falls $a^{n-1} \not\equiv 1 \bmod n$ gilt.
Eine Zahl $n \in \mathbb{N} \setminus \mathbb{P}$ heißt Carmichael-Zahl falls $n$ eine Pseudoprimzahl ist für jedes $a$ mit $\mathrm{ggT}(a,n) = 1$

Also: wenn es Carmichael Zahlen gibt, so kann unsere naiver Test nie richtig funktionieren.

BEMERKUNG 4.38: $n \in \mathbb{N}$ ist genau dann Carmichael, wenn $n = \prod_{i=1}^{r} p_i$ mit $r \geq 2$ und paarweise verschiedenen ungeraden Prmzahlen $p_i$ mit $p_i - 1 | n - 1$ gilt.

BEWEIS. Sei $n$ eine Carmichael Zahl, $p^m | n$, $p$ Prim. Dann ist $\mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/p^m\mathbb{Z}$ wohldefiniert und wir sehen, $\lambda(p^m) | \lambda(n)$. Wegen $n$ Carmichael, so gilt $\lambda(n) | n-1$. Mit $\lambda(p^m) = (p-1)p^{m-1}$ und $p \not| n-1$ folgt dann $p-1 | n-1$ und $m = 1$.

Andererseits, $n = \prod p_i$ und $p_i - 1 | n - 1$, dann gilt $\lambda(n) = \mathrm{kgV}\,(p_i - 1 : 1 \leq i \leq r)$. Wegen $p_i - 1 | n - 1$ gilt dies dann für das kgV , also ist $n$ Carmichael. □

Damit erhalten wir $561 = 3 \cdot 11 \cdot 17$ ist die kleinste Carmichael Zahl. Noch schlimmer:

$$|\{n \in \mathbb{N} | n \leq x \text{ ist Carmichael}\}| \geq x^{2/7}$$

für $x >> 0$ nach einem Paper von Ahlford, Granville und Pomeranz in '94. Es gibt also unendlich viele Carmichael Zahlen.

BEMERKUNG 4.39: Falls wir $n-1$ faktorisieren können, so können wir dies zum Testen der Primzahleigenschaft nutzen: Angenommen, wir haben $a \in \mathbb{Z}$, $\mathrm{ggT}(a,n) = 1$, $a^{n-1} \equiv 1 \bmod n$ und $a^{(n-1)/p} \not\equiv 1 \bmod n$ für alle $p | n-1$ so ist $n$ Prim.

BEWEIS. Sei $e := \mathrm{ord}(a + n\mathbb{Z}) = \min\{r : a^r \equiv 1 \bmod n\}$, dann gilt $e | n-1$. Nch unseren Vorraussetzungen sogar $e = n-1$, also ist $n - 1 | \phi(n) \leq n - 1$, also ist $n$ Prim. □

Damit können wir so vorgehen: Wir suchen nach der kleinsten Pseudoprimzahl $C$ zu einer Basis $a \leq B$ für ein kleines $B$. Damit kann dann für jedes $n < C$ sehr schnell getestet werden ob $n$ Prim ist. Nachteil: das Suchen nach der kleinsten Pseudoprimzahl ist sehr aufwendig. Vorteil: wenn wir nur an Zahlen beschränkter Größe interessiert sind, so ist dies sehr schnell.

Für spezielle $n$, z.B. $n = 2^k + 1$ kann die Bemerkung benutzt werden. Im Allgemeinen klappt dies nicht, da die Faktorisierung nicht erhalten werden kann.

SATZ 4.40 (Pocklington): Seien $a$, $n \in \mathbb{N}$, $\mathrm{ggT}(a,n) = 1$ mit $a^{n-1} \equiv 1 \bmod n$. Ferner sei $d | n-1$ mit $\mathrm{ggT}(d, (n-1)/d) = 1$ so, daß für alle Primteiler $q | d$ gilt $\mathrm{ggT}(a^{(n-1)/q} - 1, n) = 1$. Dann gilt $q \equiv 1 \bmod d$ für alle Primteiler $p | n$. Speziell folgt für $d > \sqrt{n} - 1$, dass $n$ selber Prim ist.

BEWEIS. Sei $p|n$ und $q|d$, setze $e := \max\{m \in \mathbb{N}|q^m|d\}$, also $q^e\|d$, damit folgt dann auch $q^e\|n-1$ (wegen $\text{ggT}(d,(n-1)/d) = 1$). Mit $b := q^{(n-1)/q^e} \bmod n$ folgt $b^{q^{e-1}} \equiv a^{(n-1)/q} \not\equiv 1$ (wegen $\text{ggT}(a^{(n-1)/q} - 1, n) = 1$), damit dann auch $b^{q^{e-1}} \not\equiv 1 \bmod p$. Also ist $q^e$ die genaue Ordnung von $b$ modulo $p$, damit $q^e|\lambda(p) = p-1$. Dies gilt für alle Primteiler von $d$, also $d|p-1$ oder $p \equiv 1 \bmod d$ wie Behauptet. Falls nun $d > \sqrt{n} - 1$ ist, so folgt für jeden Primteiler von $n$, dass $p > \sqrt{n}$ gilt, also ist $n$ Prim. □

Auch dieser Satz benötigt eine Faktorisierung, ist jedoch die Grundidee zu ECPP was der zur Zeit schnellste und meist benutzte Primzahltest ist. (Elliptic Curve Primality Proving). Dort wird die selbe Idee wie im Satz benutzt, jedoch wird statt in $(\mathbb{Z}/n\mathbb{Z})^*$ in einer anderen Gruppe gearbeitet. Die Idee (die wir auch hier versuchen könnten) ist einfach: Wir suchen alle *kleinen* Primteiler von $n-1$ (das geht schnell) und versuchen dann zu zeigen, dass der Rest ($d$) selber Prim ist - in dem der selbe Algorithmus dann auf $d$ angewendet wird. $d$ wird maximal $(n-1)/2$ sein, also haben wir das Problem um eine (binär) Stelle vereinfacht. Hier klappt das nicht, da $d$ nicht Prim sein wird, bei den Elliptischen Kurven funktioniert dies seht gut, da wir dann die Gruppe wechseln können.

**5.2. Miller-Rabin.** Wir versuchen immer noch die Carmichael Zahlen zu umgehen. Die neue Idee die wir jetzt einsetzen werden ist die: falls $n$ Prim ist, so ist $\mathbb{Z}/n\mathbb{Z}$ ein Körper. Im einem Körper hat die Gleichung $x^2 - 1 = 0$ maximal (genau) zwei Lösungen: $\pm 1$. Wir werden jetzt den Fermat-Test dazu benutzen Lösungen dieser Gleichung zu suchen. Wenn wir etwas $\neq \pm 1$ finden, so beweist dies, dass $n$ nicht Prim ist.

DEFINITION 4.41: $n \in \mathbb{N}$ ungerade, $n - 1 = 2^t u$ mit $u$ ungerade. $u$ heißt strenge Pseudoprimzahl zur Basis $a$, falls entweder $a^u \equiv 1 \bmod n$ gilt oder ein $0 \leq s < t-1$ mit $a^{2^s u} \equiv -1 \bmod n$ existiert.
$a$ heißt Zeuge für die Zerlegbarkeit von $n$, falls $a^u \not\equiv 1 \bmod n$ und $a^{2^s u} \not\equiv -1 \bmod n$ für alle $0 \leq s < t$ gilt.

BEMERKUNG 4.42: Besitzt $n$ einen Zeugen zur Zerlegbarkeit $a$, so ist $n$ keine Primzahl.

BEWEIS. Sei $a$ der Zeuge. Setze $m := \max\{0 \leq s | a^{2^s u} \not\equiv 1\}$, dies existiert da 0 in der Menge ist. Angenommen, $n$ ist Prim, so folgt $a^{n-1} \equiv 1 \bmod n$, also $m < t$. Für $b := a^{2^m u}$ gilt nun $b^2 \equiv 1 \bmod n$, $b \not\equiv 1$, also $b \equiv -1$ da $b$ Nullstelle von $x^2 - 1$ in dem Körper $\mathbb{F}_n$ ist. Damit kann $a$ aber kein Zeuge sein, also haben wir einen Widerspruch, und $n$ ist nicht Prim. □

Dies kann jetzt zu einem sehr starken Test für kleine Primzahlen benutzt werden:

| $a$ | $B$ |
|---|---|
| $2$ | $n < 2047$ |
| $2, 3$ | $n < 1.373.653$ |
| $2, 3, 5$ | $n < 25.326.001$ |
| $2, 3, 5, 7$ | $n < 3.215.031.751$ |

Also: wenn $n$ eine strenge Pseudoprimzahl zur Basis 2 und 3 ist, und $n < 1.373.653$ gilt, so ist $n$ Prim.

Wenn wir den Test systematisch nutzen wollen, so müssen wir feststellen wie viele Zeugen es gibt. Dies werden wir im weiteren nun angreifen.

LEMMA 4.43: Sei $C_n$ eine zyklische Gruppe der Ordnung $n$, dann gibt es $\operatorname{ggT}(n,m)$ viele Elemente der Ordnung $m$.

BEWEIS. Wir haben $C_n = \{1, g, g^2, \ldots, g^{n-1}\}$. Setze $d := \operatorname{ggT}(n,m)$. Sei nun $x = inC_n$ mit $\operatorname{ord} x = m$ gegeben. Dann ex. ein $e$ mit $x = g^e$, also $x^m = 1 = g^{em}$ genau dann, wenn $n|em$ gilt. Wir erhalten nun Äquivalenzen:

$$n|em \iff \frac{n}{d}d|ed\frac{m}{d} \iff \frac{n}{d}|e\frac{m}{d} \iff \frac{n}{d}|e \iff e = a\frac{n}{d}$$

für ein geeignetes $a$. □

LEMMA 4.44: Sei $2 \neq p \in \mathbb{P}$, $q = p^e$, $\phi(q) = 2^t u$ mit $u$ ungerade. Ist nun $r$ ungerade, so gilt:

$$|\{x \in (\mathbb{Z}/q\mathbb{Z})^* | x^{2^s r} \equiv -1\}| = \begin{cases} 2^s \operatorname{ggT}(u,r) & s < t \\ 0 & s \geq t \end{cases}$$

BEWEIS. $(\mathbb{Z}/q\mathbb{Z})^* = \langle g \rangle$ ist zyklisch der Ordnung $\phi(q) = (p-1)p^{e-1}$, da $q$ ungerade ist. Daher ex. zu jedem $x \in (\mathbb{Z}/q\mathbb{Z})^*$ ein $1 \leq f \leq \phi(q)$ mit $x = g^f$. Damit gilt $x^{2^s r} \equiv -1$ genau dann, wenn $g^{e2^s r} \equiv -1$ gilt. Da $\operatorname{ord} -1 = 2$ gilt, so folgt $g^{2^{t-1}u} \equiv -1$. Wir unterscheiden nun 2 Fälle: $s > t-1$, dann hat $2^s er \equiv 2^{t-1}u \bmod \phi(q)$ keine Lösung, es kann solche $x$ nicht geben.

$s \leq t-1$, setze $d := \operatorname{ggT}(u,v)$, so ist $v/de \equiv 2^{t-s-1}u/d \bmod 2^{t-2}u/d$ eindeutig nach $e$ lösbar, damit hat $2^s re \equiv 2^{t-1}u \bmod \phi(q)$ genau $2^s d$ Lösungen modulo $\phi(q) = 2^s u(2^{t-2}u/d)$ □

SATZ 4.45: Sei $9 \neq n \in \mathbb{N}$ ungerade und keine Primzahl. Dann besitzt $n$ mindestens $3/4\phi(n)$ Zeugen in $(\mathbb{Z}/n\mathbb{Z})^*$ für die Zerlegbarkeit.

BEWEIS. Schreibe $n = \prod_{i=1}^r p_i^{e_i} =: \prod_{i=1}^r q_i$, $n-1 = 2^t u$, $\phi(q_i) = 2^{t_i}u_i$, $u$, $u_i$ ungerade und $t_1 \leq \ldots \leq t_r$. Dann ist $\operatorname{ggT}(n-1, \phi(q_i)) = \operatorname{ggT}(2^t u, 2^{t_i}u_1) = 2^{t_i'}u_i'$ mit $t_i' = \min(t, t_i)$ und $u_i' = \operatorname{ggT}(u, u_i)$. Setze

$$B_n := \{a \in (\mathbb{Z}/n\mathbb{Z})^* | a^u = 1\} \dot\cup \dot\bigcup_{s=0}^{t-1} \{a | a^{2^s u} \equiv -1\} =: B_n^* \dot\cup B_n^{(s)}$$

für die Menge der Nicht-Zeugen. Die Kongruenzen werden jetzt mit dem Chinesischen Restsatz zerlegt:

$$|B_n^*| = \prod |B_{q_i}^*| = \prod \operatorname{ggT}(\phi(q_i), u) = \prod \operatorname{ggT}(u_i, u) = \prod u_i'$$

und

$$|B_n^{(s)}| = \prod |B_{q_i}^{(s)}| = \prod 2^s \operatorname{ggT}(u_i, u) = 2^{rs} \prod u_i'$$

für $s < t_1$ und 0 sonst. Damit folgt dann

$$|B_n| = (1 + \sum_{s=0}^{t_1-1} 2^{rs}) \prod u_i'$$

also

$$\frac{|B_n|}{\phi(n)} = \frac{1 + \sum_{s=0}^{t_1-1} 2^{rs}}{2^{\sum t_i}} \frac{\prod u_i'}{\prod u_i} =: f_1 f_2.$$

Wegen $\sum_{s=0}^{t_1-1} 2^{rs} = (2^{rt_1}-1)/(2^r-1)$ können wir abschätzen:

$$f_1 \leq \frac{(2^r-1)+(2^{rt_1}-1)}{(2^r-1)2^{rt_1}} = \frac{1}{2^r-1}(1+\frac{2^r-2}{2^{rt_1}}) \leq \frac{1}{2^r-1}\frac{2(2^r-1)}{2^r} = \frac{1}{2^{r-1}}$$

und $f_2 \leq 1$.

Jetzt müssen wir 3 Fälle unterscheiden:

1. Fall: $r \geq 3$, dann $|B_n| \leq \phi(n)/2^{r-1} \leq \phi(n)/4$

2. Fall: $r = 2$. Dann gilt $f_1 \leq 1/2$. Falls $u_1' < u_1$ oder $u_2' < u_2$ gilt, so ist $f_2 \leq 1/2$ und wir sind fertig. Also: $u_1 = u_1'$ und $u_2 = u_2'$ und daher $\phi(q_i) = p_i^{e_1-1}(p_i-1) = 2^{t_1}u_i$ und $p_i^{e_1-1}|u_i' = u_i|n-1$ und $p_i|n$. Damit $e_i = 1$, $q_i = p_i$. Wegen $u_i|p_1-1$, also $p \equiv 1 \bmod u_1$ ist $0 \equiv n-1 \equiv p_1p_2-1 \equiv p_2-1 \bmod u_1$, also $u_1|p_2-1$ und $u_1|u_2$. Analog folgt auch $u_2|u_1$, also $u_1 = u_2$ und somit $t_1 < t_2$ da andernfalls $p_1 = p_2$ wäre. Damit können wir die Abschätzung für $f_1$ verbessern zu

$$f_1 \leq \frac{(2^r-1)+(2^{rt_1}-1)}{(2^r-1)2\cdot 2^{rt_1})} \leq 1/4$$

(Wie oben, wo wir $2^{\sum t_i}$ durch $2^{rt_1}$ abgeschätzt haben, erhalten wir nun $\geq 2\cdot 2^{rt_1}$).

3. Fall: $r = 1$, also $n = p^e$, $e > 1$. Offensichtlich gilt $B_n \subset \{a|a^{n-1} \equiv 1 \bmod n\}$, aber dann können wir mit dem Lemma 4.43 abschätzen:

$$|B_n| \leq \mathrm{ggT}(\phi(n), n-1) = \mathrm{ggT}(p^{e-1}(p-1), p^e-1) = p-1$$

Damit folgt dann $|B_n| \leq 1/p^{e-1}\phi(n) \leq 1/4\phi(n)$ wie gefordert. für $p^e > 9$. □

Damit erhalten wir dann den Probabilistischen Primzahltest nach Miller-Rabin:

ALGORITHMUS 4.46: Input: $n \in \mathbb{N}$ ungerade, $n > 9$.
Output: `false` falls $n$ beweisbar keine Primzahl ist, `true` sonst.

- Wähle $0 < a < n$ zufällig.
- Falls $\mathrm{ggT}(a,n) > 1$ return `false`
- Berechne $a^{2^s u}$ mod $n$ wie im Test beschrieben. Falls Werte $\neq \pm 1$ herauskommen, return `false`
- return `true`

Nach dem Satz ist die Wahrscheinlichkeit, dass `true` berechnet wird, obwohl $n$ nicht Prim ist $P \leq 1/4$. Wenn wir den Test für unabhängig gewählte zufällige $a$ wiederholen, so erhalten wir für $n = 10$ Iterationen eine „Fehlerwahrscheinlichkeit“ von $4^{-10}$, für $n = 20$ erhalten wir $4^{-20} = 9.09 \cdot 10^{-13}$. Wenn wir davon ausgehen, dass ein Computer auch nie perfekt ist (1 Fehler in 20,000 Stunden z.B. (Random Rays im Speicher)), können wir einfach so oft iterieren bis die Wahrscheinlichkeit, dass der Computer falsch rechnet größer ist, als die Wahrscheinlichkeit eine zusammengesetzte Zahl nicht erkannt zu haben. Mathematisch gesehen, ist dies natürlich kein Beweis.

Wenn wir an die Riemannsche Vermutung glauben, so kann man zeigen, dass jede zusammengesetzte Zahl $n$ einen Zeugen hat mit mit $a \leq 2\log^2(n)$. Damit würden wir einen deterministischen Test erhalten. Alternativ können wir auch einfach alle $a \in \mathbb{Z}/n\mathbb{Z}$ ausprobieren...

Mit GRH ist die Laufzeit dann $O(\log^4 n \log\log n)$. Es gibt Verfahren (ECPP) die in einer Laufzeit von (möglicherweise: hier gehen mehr Vermutungen ein!) $O(\log^{4+\epsilon})$ ein Zertifikat dafür findet, dass $n$ Prim ist (einen Prim-Zeugen). Dieses kann dann schneller, d.h. in $O(\log^3 n)$ verifiziert werden.

Seit 2002 gibt es auch einen deterministischen Primzahltest (AKS) ohne GRH, leider ist die Laufzeit $O(\log^6 n)$, und damit nicht konkurenzfähig.

**5.3. Faktorisieren in $\mathbb{Z}$.** Faktorisierung von ganzen Zahlen ist schwierig. Dies wird in der Regel mit einer langen Kette von Algorithmen angegangen. Die Struktur ist in etwa so:

ALGORITHMUS 4.47: Input: $n \in \mathbb{Z}$

1: $n := |n|$
2: $n \in \{0, 1\}$ fertig
3: ex. $k$, $r$ mit $n = r^k$, so wende Algorithmus auf $r$ an.
4: Teste ob $n$ (wahrescheinlich) Prim ist. Falls ja, beweise es und terminiere.
5: Probedivision bis $10^6$, Tabellen.
6: ja nach Größe von $n$ versuche verschiedene Algorithmen die jeweils terminieren falls 1 Faktor gefunden ist. Dann müßen die Faktoren rekursiv weiter behandelt werden. Dies ist z.B. $p-1$, Pollard-$\rho$, $ECM$, ($MPQS$ oder $NFS$)

LEMMA 4.48: Sei $1 < n \in \mathbb{N}$ ungerade, $n = \prod_{i=1}^r p_i^{e_i}$. Dann ist $|\{x \in \mathbb{Z}/n\mathbb{Z} | a^2 \equiv 1 \bmod n\}| = 2^r$

BEWEIS. Mit dem Chinesischen Restsatz folgt $a^2 \equiv 1 \bmod n$ genau dann, wenn $a^2 \equiv 1 \bmod p_i^{e_i}$. Da die $(\mathbb{Z}/p^e)^*$ für ungerade $p$ stets zyklisch der Ordnung $(p-1)p^{e-1}$ sind, so gibt es für jedes $p$ genau 2 Wurzeln: 1 und $-1 = p^e - 1$. Damit folgt dann die Behauptung aus dem Restsatz. □

Da wir im wesentlichen nur ungerade Zahlen faktorisieren werden, so können wir das Lemma einsetzen. Die Idee ist:

FOLGERUNG 4.49: Sei $\pm 1 \neq a \in (\mathbb{Z}/n\mathbb{Z})^*$ mit $a^2 = 1$. Dann ist $\mathrm{ggT}(a \pm 1, n)$ ein echter Teiler von $n$.

BEWEIS. Wegen $a \neq \pm 1 \bmod n$ ist $0 < a \pm 1 < n$. Aus $a^2 \equiv 1 \bmod n$ folgt $n|(a+1)(a-1)$, also können die ggTs nicht trivial sein. □

Dies ist, im Prinzip, die Basis für „alle“ modernen Faktorisierer. Es bleibt nur noch die Frage, wie $a$ gefunden wird.

BEISPIEL 4.50: $n = 91$, $b = 12$, dann gilt $b^{12} \equiv 1 \bmod n$ und für $b^6 = 64 =: a \bmod b$ folgt $\mathrm{ggT}(63, 91) = 7$, $\mathrm{ggT}(65, 91) = 13$.

BEMERKUNG 4.51: Sei $n \in \mathbb{N}$ ungerade, $m \in \mathbb{N}$ mit $m = 2^t u$, $u$ ungerade und $\lambda(n)|m$. Setze $A := \{a \in \mathbb{Z}/n\mathbb{Z} | \operatorname{ord}(a) \equiv 0 \bmod 2, -1 \notin \langle a \rangle\}$. Dann gilt:

(1) Für jedes $a \in A$ ex. $0 \leq s \leq t-1$ so, dass $b := a^{2^s u} \neq \pm 1$ und $b^2 \equiv 1$
(2) Ist $n \neq p^e$, so gilt $|A| \geq 1/2\phi(n)$

BEWEIS. Da die Ordnung $\operatorname{ord}(a)$ gerade ist, so gibt es Elemente der Ordnung $a$ in $\langle a\rangle$. Da $\operatorname{ord}(a)|\lambda(n)|m = 2^t u$ muss dies eine Element von der angegebenen Form sein. Es kann nicht $-1$ sein, da $-1$ nicht im Erzeugnis war.

Für den 2. Teil, sei $n = \prod_{i=1}^r p_i^{e_i}$ mit $r \geq 2$ und $2^{\tilde{t}} \| \lambda(n)$. Wegen $\lambda(n) = \operatorname{kgV}\ \lambda(p_i^{e_i})$ können wir oBdA $2^{\tilde{t}} | \lambda(p_1^{e_1}) = \phi(p_1^{e_1})$ annehmen.

Angenommen, $\langle \mathbb{Z}/n\mathbb{Z}^* \setminus A\rangle = (\mathbb{Z}/n\mathbb{Z})^*$. Dann ist $\psi : a \mapsto a^{2^{\tilde{t}-1}u}$ ein Homomorphismus mit $\psi(a) \in \langle a\rangle$ und $\psi(a)^2 \equiv 1 \bmod n$. Für $b \in \mathbb{Z}/n\mathbb{Z} \setminus A$ folgt sofort $\psi(b) = \pm 1$. Da Elemente nicht in $A$ alles erzeugen, so folgt dann $\psi(a) = \pm 1$ für alle $a$. Mit dem Chinesischen Restsatz können wir aber nun ein $b$ bauen mit $\operatorname{ord}(b) = 2^{\tilde{t}} \bmod p_1^{e_1}$ und $b \equiv 1 \bmod p_i^{e_i}$ für $i > 2$. Damit folgt dann aber $\psi(b) \neq \pm 1$, somit muss $\langle \mathbb{Z}/n\mathbb{Z}^* \setminus A\rangle$ eine echte Untergruppe sein, also $|\langle \mathbb{Z}/n\mathbb{Z}^* \setminus A\rangle| \leq 1/2\phi(n)$ und damit $|A| > 1/2\phi(n)$ □

Dies ist schwer zu benutzen, da $\lambda(n)$ oder ein Vielfaches davon nicht bestimmt werden kann.

Als nächstes sehen wir uns die $p-1$ Methode an.

LEMMA 4.52: Sei $n \in \mathbb{Z}$ ungerade, $p \in \mathbb{P}$ mit $p|n$, $m \in \mathbb{N}$ mit $p-1|m$. Dann gilt $p| \operatorname{ggT}(a^m - 1)$ für alle $a \in \mathbb{Z}/n\mathbb{Z}^*$.

BEWEIS. Fermat/ Euler: $a^{p-1} \equiv 1 \bmod p$ und damit $a^m \equiv 1 \bmod p$, also $p|a^m - 1$. Wegen $a|n$ folgt dann auch $p| \operatorname{ggT}(a^m - 1, n)$. □

Dies liefert eine (schnelle?) Methode Primteiler $q$ zu finden bei denen $q-1$ nur kleine Primteiler hat.

ALGORITHMUS 4.53 ($p-1$-Methode): Input: $n \in \mathbb{N}$, $C > 0$

1: $a := 1$
2: solange $p < C$ ist, tue
3: $\quad e := \lfloor \log_p C \rfloor$
4: $\quad a \leftarrow a^{p^e} \bmod n$.
5: Falls $\operatorname{ggT}(a-1, n)$ nicht trivial ist, gib den Faktor aus, und terminiere.

Klar, wenn $p|n$ mit $p-1 = \prod q^e$ mit $q^e \leq C$, dann wird der Algorithmus dies finden.

Es gibt Verallgeminerungen hervon die auf $p+1$ oder ähnlichen Ideen basieren.

Pollard-$\rho$

LEMMA 4.54: Sei $M$ eine endliche Menge, $|M| = m$, $f : M \to M$ eine Abbildung und $x_1 \in M$. Setze $x_{i+1} := f(x_i)$. Dann ex. $0 \leq k < m$ und $0 < l \leq m$ mit $k + l \leq m$, so dass $(x_i)_i = (x_1, \ldots, x_k, \overline{x_{k+1}, \ldots, x_{k+l}})$ gilt.

BEWEIS. Da $M$ endlich ist, muss es Wiederholungen in $(x_i)_i$ geben. Ab der 1. Wiederholung muss es dann periodisch werden. □

FOLGERUNG 4.55: Sei $n \in \mathbb{N}$ ungerade, $p \in \mathbb{P}$ mit $p|n$, $x_1 \in \mathbb{Z}$, $x_{i+1} := f(x_i)$ für $f : \mathbb{Z} \to \mathbb{Z}$ mit $f(x+p) \equiv f(x) \bmod p$. Dann ex $i < j$ mit $p| \operatorname{ggT}(x_j - x_i, n)$

BEWEIS. Nach Konstruktion ist $M := \mathbb{Z}/\mathbb{Z}$ endlich und $f : M \to M$. Nach dem Lemma gibt es daher $k$, $l$ mit $x_k \equiv x_l \bmod p$, also $p| \operatorname{ggT}(x_k - x_l, n)$ □

Da wir natürlich in der Praxis $p$ nicht kennen, so wird $f$ als Polynom gewählt, damit gilt dann für jedes $p$: $f(x+p) \equiv f(x) \bmod p$.

BEISPIEL 4.56: $n = 91$, $f = t^2+1$, $x_1 = 1$ also $(x_i)_i = (1, 2, \overline{5, 26, 40, 54})$ und $\operatorname{ggT}(26-5, 91) = 7$, $\operatorname{ggT}(40-1, 91) = 13$.

Um dies sinnvoll Einsetzen zu können müßen wir noch untersuchen, wie lange die Periode sein wird, wann und wie oft wir ggTs berechnen sollen.

LEMMA 4.57: Sei $M$ endliche Menge, $|M| = m$, $\tilde{M}^{(r)} := \{x \in M^r | x_i \neq x_j\}$. Dann gilt

$$\frac{|\tilde{M}^{(r)}|}{|M^r|} < \exp(\frac{-r(r-1)}{2m}) < \exp(-\delta)$$

für $\delta := \frac{(r-1)^2}{2m}$.

BEWEIS.

$$\frac{|\tilde{M}^{(r)}|}{|M|} = \frac{m(m-1)\dots(m-r+1)}{mm\dots m} = \prod_{i=1}^{r-1}(1-\frac{i}{m}) \leq \prod \exp(-i/m) = \exp(-(\sum i)/m) = \exp(-r(r-1$$

Da $1+x \leq e^x$ für alle $x$ gilt. □

FOLGERUNG 4.58: Setze $f^r := f(f^{r-1})$ die $r$-mal Iterierte Anwendung von $f$. Falls nun $f$ „hinreichend allgemein“ ist (was hier heissen soll $f^r(x)$ ist gleichverteilt), $\delta \in \mathbb{R}_{>0}$ und $r := \lfloor\sqrt{2m\delta}\rfloor + 1$. Dann gilt

$$P((x_1, f(x_1), \dots, f^{r-1}(x_1)) \text{ paarweise verschieden}) < e^{-\delta}$$

Damit sehen wir, dass die Periodenlänge etwa $\sqrt{m}$ sein wird - also sehr lang.

BEMERKUNG 4.59 (Brent): Seien $f : \mathbb{Z} \to \mathbb{Z}$, $x_1 \in \mathbb{Z}$, $x_{i+1} = f(x_i)$ wie oben. Es gelte $p | \operatorname{ggT}(x_j - x_i, n)$. Ferner sei $l(j) := \lfloor\log_2(j)\rfloor$, $\tilde{i} := 2^{l(j)+1} - 1$, $\tilde{j} := \tilde{i} + (j-i)$. Dann gelten $p | \operatorname{ggT}(x_{\tilde{j}} - x_{\tilde{i}}, n)$ und $\tilde{i} = 2^{l(\tilde{j})} - 1$

BEWEIS. Sei $2^h = lej < 2^{h+1}$, also $l(j) = h$. Dann folgt $\tilde{i} = 2^{h+1} - 1$, $\tilde{j} - \tilde{i} = j - i$. Nach Voraussetzung gilt $x_j \equiv x_i \bmod p$, also auch $x_{j+k} \equiv x_{i+k} \bmod p$ für alle $k$, also auch $x_{\tilde{j}} \equiv x_{\tilde{i}} \bmod p$. Es gilt $l(\tilde{j}) = h = l(j)$ und aus $\tilde{j} = \tilde{i} + j - i < 2^{h+1} + 2^{h+1} = 2^{h+2}$ folgt $l(\tilde{j}) = h+1 = l(j)+1 = l(\tilde{i})$ □

Damit reicht es bei der $\rho$ Methode nor $\operatorname{ggT}(x_j - x_i, n)$ für $i = 2^{l(j)} - 1$ zu testen. Speziell heißt dies auch, dass wir nur die $x_i$ für $i = 2^k - 1$ speichern müssen! In 1981 wurde damit $2^{2^8} + 1 \approx 10^{78}$ Faktorisiert. Die beiden Primteiler haben 15 bzw. 61 Stellen.

SATZ 4.60 (Brent): Seien $\delta > 0$, $n \in \mathbb{N} \setminus \mathbb{P}$ ungerade. Mit der Pollardschen $\rho$-Methode findet man einen Primteiler $p$ von $n$ nach höchstens $O(\sqrt{\delta}\sqrt[4]{n})$ Versuchen mit einer Wahrscheinlichkeit $P \geq 1 - e^{-\delta}$. Pro Versuch benötig man höchstens $O(\log^2 n)$ Bitoperationen.

BEWEIS. Mit Lemma 4.57 und Folgerung 4.58 findet man $x_j \equiv x_i \bmod p$ mit Wahrscheinlichkeit $P \geq 1 - e^{-\delta}$ nach $\sqrt{2p\delta} + 1$ Versuchen. Mit Bemerkung 4.59 folgt, dass wenn wir jedoch $4(\sqrt{2p\delta} + 1)$ Versuche erlauben, wir nur für $i = 2^{l(j)} - 1$ ggTs berechnen müssen. Da $p \leq \sqrt{n}$ ist, so folgt $r \leq 4(\sqrt{2\delta}\sqrt[4]{n} + 1)$ Versuche als ein ggT mit Kosten $\log^2 n$ □

**5.4. Quadratisches Sieb.** Das Quadratische Sieb (wie auch das NFS) versucht Zahlen $X$, $Y$ zu finden mit

$$X^2 - Y^2 \equiv 0 \bmod n$$

Dann ist (wahrscheinlich) $\mathrm{ggT}(X - Y, n)$ oder $\mathrm{ggT}(X + Y, n)$ nicht trivial. Unser Beispiel hier ist $n = 7429$.

Für $X = 227$, $Y = 210$ gilt $X^2 = 51529 \equiv 6955 \equiv 44100$ und $\mathrm{ggT}(227-210, 7429) = 17$, $\mathrm{ggT}(227 + 210, 7429) = 437$.

Schön, aber wie finden wir $X$ und $Y$?

Setze $m := \lfloor \sqrt{n} \rfloor$ und $f(t) := (t+m)^2 - n$. In unserem Beispiel: $m = 86$, $f(t) = x^2 + 172x - 33$. Wir Berechnen $f(s)$ für $|s|$ klein und zerlegen die Werte in Primfaktoren:

$$\begin{array}{lcrcrcl} f(-3) & = & 83^2 - 7429 & = & \text{-}540 & = & -2^2 \cdot 3^3 \cdot 5 \\ f(1) & = & 87^2 - 7429 & = & 140 & = & 2^2 \cdot 5 \cdot 7 \\ f(2) & = & 88^2 - 7429 & = & 315 & = & 3^2 \cdot 5 \cdot 7 \end{array}$$

Also sehen wir, dass

$$\begin{array}{lcl} 83^2 & \equiv & -2^2 \cdot 3^3 \cdot 5 \\ 87^2 & \equiv & 2^2 \cdot 5 \cdot 7 \\ 88^2 & \equiv & 3^2 \cdot 5 \cdot 7 \end{array}$$

und damit $87^2 \cdot 88^2 \equiv 2^2 \cdot 3^2 \cdot 5^2 \cdot 7^2 \bmod 7429$. Setzen wir $X := 87 \cdot 88 = 7656 \equiv 227$ und $Y := 2 \cdot 3 \cdot 5 \cdot 7 = 210$ so erhalten wir unsere Zahlen von oben.

Also, bleibt die Frage, wie dies systematisch erledigt werden kann.

Bemerkung 4.61: Sei $f \in \mathbb{Z}[t]$, $r$, $s \in \mathbb{Z}$ mit $f(r) \equiv 0 \bmod s$. Dann gilt auch $f(r + ks) \equiv 0 \bmod s$

Damit können wir die Tabelle mit Hilfe eines Siebes bestimmen. Wir berechnen einfach $f(s)$ für „kleine“ $s$, finden $s$ so, dass $f(s)$ durch 2 teilbar ist, teilen die 2 Anteile raus, und wiederholen mit 3, 5, ....

| $s$ | $f(s)$ | 2 | 3 | 5 |
|---|---|---|---|---|
| -3 | -540 | $2^2 \cdot 135$ | $2^2 \cdot 3^3 \cdot 5$ | $2^2 \cdot 3^3 \cdot 5 \cdot 1$ |
| -2 | -373 | | | |
| -1 | -204 | $2^2 \cdot 51$ | $2^2 \cdot 3 \cdot 17$ | |
| 0 | -33 | | $3 \cdot 11$ | |
| 1 | 140 | $2^2 \cdot 35$ | | $2^2 \cdot 5 \cdot 7$ |
| 2 | 315 | | $3^2 \cdot 35$ | $3^2 \cdot 5 \cdot 7$ |
| 3 | 492 | $2^2 \cdot 123$ | $2^2 \cdot 3 \cdot 41$ | |

Einige Zeilen haben jetzt eine Vollständige Zerlegung, andere nicht. Also müssen wir jetzt Festlegen,

- Welche $s$ wir in die Tabelle aufnehmen
- Welche Primzahlen wir benutzten

Die Sammlung der Primzahlen heißt Faktorbasis, die der $s$ Siebinterval. Diese können recht groß sein, diese Woche (2.7.2012) gab es eine email mit den Details für die Faktorsierung einer Zahl mit 212 Dezimalstellen. Die Faktorbasis hatte alle Primzahlen bis $25 \cdot 10^7$, also $13,6 \cdot 10^6$ viele. Das Siebintervall selber ist nicht angegeben, aber die Zeit: es wahren etwa 500 CPU-Jahre (7 Monate auf vielen Maschinen). Es wurden $1,3 \cdot 10^9$ viele *nützliche* $s$ gefunden.

Wir werden dies noch im Detail diskutieren, aber schon mal vorab:

- wenn die Faktorbasis klein ist, so werden sehr wenige $s$ übrigbleiben.
- wenn $s$ groß ist, brauchen wir wahrscheinlich große Primzahlen
- je mehr Primzahlen wir erlauben, desto mehr $s$ brauchen wir, da wir ja noch Zeilen kombinieren müssen.

Wie werden die Zeilen (mit Vollständiger Faktorisierung über unserer Faktorbasis) kombiniert? Mit linearer Algebra! Sehen wir es uns nochmal an: wir wollen Quadrate auf der rechten Seite. Was wir tun können ist geeignete Zeilen „zu multiplizieren“ da Links immer Quadrate stehen, können wir versuchen so auch Rechts welche zu erhalten.

Sei $r_1, \ldots, r_R$ die Sammlung der vollständig faktorisierten Zeilen (Relationen) genauer gesagt die Werte $f(s_i)$, $p_1, \ldots, p_B$ die Sammlung der Primzahlen schliesslich setze $M = (M_{i,j})_{i,j} \in \mathbb{Z}^{R \times B}$ mit $M_{i,j} := v_{p_j}(s_i)$ die Relationenmatrix. Dann suchen wir ein Produkt der Relationen $r_i$ das ein Quadrat ist. Die Primzerlegung des Produktes können wir aus der Matrix ablesen: wir suchen einen Exponentenvektor $\underline{e} = (e_1, \ldots, e_R) \in \{0,1\}^R$ mit $eM \in (2\mathbb{Z})^B$. Dann ist $\prod r_i^{e_i}$ ein Quadrat. Wie wir leicht sehen, können wir dies vereinfachen: wir suchen einen Vektor $e$ im Kern von $M \in \mathbb{F}_2^{R \times B}$ - was natürlich viel einfacher ist, als über $\mathbb{Z}$ zu arbeiten.

In dem großen Beispiel war die Matrix auch groß: nach ein paar Tricks, war die Matrix etwa $(89 \cdot 10^6)^2$ mit $18 \cdot 10^9$ Einträgen. Das Finden eines Kernvektor dauerte ebenfalls 3 Monate auf verschiedenen Clustern.

So, wie werden jetzt die beiden Parameter $R$ und $B$ bestimmt?

**Definition 4.62:** Sei $B > 0$. Eine Zahl $r \in \mathbb{N}$ heißt $B$-glatt, wenn aus $\mathbb{P} \ni p | r$ sofort $r \leq B$ folgt.
Für $x > 0$ sei

$$\psi(x, B) := |\{0 < r \leq x | r \in \mathbb{N}, r \text{ ist } B\text{-glatt}\}|$$

Für die Anzahl der $B$-glatten Zahlen gibt es Abschätzungen.

**Satz 4.63:** Sei $\epsilon > 0$, $x \geq 10$ und $\omega \leq (\log x)^{1-\epsilon}$. Dann gilt

$$\psi(x, x^{1/\omega}) = x\omega^{-\omega + f(x,\omega)} = x\omega^{-\omega + o(1)}$$

für eine Funktion $f$ mit $f(x,\omega)/\omega \to 0$, gleichmässig für alle $x$ und für $\omega \to \infty$.

Also gilt ungefähr $\psi(x, x^{1/\omega}) \sim x\omega^{-\omega}$.

Um weiter zu kommen, müssen wir eine häßliche Funktion einführen:

**Definition 4.64:** Seien $n$, $u$, $v \in \mathbb{R}$, mit $n > e$, also $\log n > 1$. Dann ist

$$L_n[u, v] := \exp(v(\log n)^u (\log \log n)^{1-u})$$

BEMERKUNG 4.65: Es gilt $L_n[0,v] = (\log n)^v$ und $L_n[1,v] = \exp(v \log n)$. Ein Algorithmus mit einer Laufzeit von $O(L_n[0,v])$ hat eine polynomielle Laufzeit (vom Grad $v$), während ein Algorithmus mit $(L_n[1,v])$ exponentiell ist. Ein Algorithmus mit $u \in ]0,1[$ heißt *sub-exponentiell*.

Wir wollen nun die $L_n[u,v]$ mit $\psi$ verbinden um Aussagen über die Parameter im Quadratischen Sieb zu treffen.

Zentral ist der folgende Satz:

SATZ 4.66: Seien $a$, $u$, $v$ positive reelle Zahlen. Dann gilt für $n \in \mathbb{N}$, $n \to \infty$:

$$\psi(n^a, L_n[u,v]) = n^a / L_n[1-u, (a/v)(1-u) + o(1)]$$

Der Beweis, den ich weglasse, ist einfach nur einsetzen der Definitionen und dann mit $L_n[u,v]$ rechnen.

Damit können wir jetzt wieder zurück zum Faktoriesieren gehen: Mit Hilfe des Siebes berechnen wir Zahlen

$$(x-m)^2 - n = x^2 - 2mx + m^2 - n$$

der Grösse $\sqrt{n}$ wenn wir $x$ als beschränkt (oder wenigstens klein im Vergleich zu $m$) ansehen. Wegen $n >> 0$ ist dies realistisch. Mit dem Satz angewendet auf $a = 1/2$ und eine Faktorbasis mit allen Primzahlen $< L_n[u,v]$ erhalten wir dann

$$\phi(\sqrt{n}, L_n[u,v]) = \sqrt{n} / L_n[1-u, \frac{1}{2v}(1-u) + o(1)]$$

Wenn unsere Zahlen zufällig verteilt sind, so brauchen wir daher

$$L_n[1-u, \frac{1}{2v}(1-u) + o(1)]$$

viele Versuche um eine glatte Relation zu finden. Damit unsere Matrix einen Kern hat, brauchen wir (wahrscheinlich) mindestens

$$B = L_n[u,v]$$

viele Relationen, also insgesamt

$$L_n[u,v] L_n[1-u, \frac{1}{2v}(1-u)]$$

viele Versuche. Da der $L_n[u,.]$ Term dominant ist, so ist dieser Ausdruck „minimal“ für $u = 1/2$, damit haben wir

$$L_n[1/2,v] L_n[1/2, \frac{1}{2v}(1/2)] = L_n[1/2, v + \frac{1}{4v}]$$

viele Versuche.

Halten wir also fest: unsere Faktorbasis wird alle Primzahlen $p$ mit $p \leq L_n[1/2,v]$ beinhalten. Wir suchen dann $L_n[1/2,v]$ viele Relationen mit $L_n[1/2, v + \frac{1}{4v}]$ vielen Versuchen.

Der Aufwand für das Sieben ist $O(L_n[1/2,v])$ viele Primzahlen. Für jede Primzahl brauchen wir $O(L_n[1/2, v + \frac{1}{4v}]/p)$ viele Operationen, damit erhalten wir (bis hierhin) eine Gesamtlaufzeit von $O(L_n[1/2, v + 1/(4v) + o(1)])$.

Um nun einen Kernvektor zu finden müßen wir in der Matrix der Größe $L_n[1/2, v] \times L_n[1/2, v]$ arbeiten. Naiv (Gauß) benötig dies $O(L_n[1/2, v]^3) = O(L_n[1/2, 3v])$ Bit-Operationen, hier gibt es jedoch bessere Methoden mit einer Laufzeit von „nur“ $O(L_n[1/2, 2v])$. Also ist unsere Gesamtlaufzeit

$$O(L_n[1/2, v + 1/(4v)] + L_n[1/2, 2v])$$

für $v = 1/2$ sehen wir, dass beide Terme gleich groß werden. Damit erhalten wir dann eine Gesamtlaufzeit von $O(L_n[1/2, 1 + o(1)])$.

KAPITEL 5

# Resultanten und Faktorisierung

Hier ist $R$ wieder ein kommutativer Ring mit 1, aus praktischen Gründen sollte $R$ ein Körper oder ein euklidischer Ring sein.

**Definition 5.1:** Seien $a, b \in R[x]$ für einen Integritätsring $R$ gegeben. Die Sylvestermatrix von $a$ und $b$ ist die $(m+n) \times (m+n)$-Matrix

$$S_{n,m}(a,b) := \begin{pmatrix}
a_n & & & & b_m & & & & \\
a_{n-1} & a_n & & & b_{m-1} & b_m & & & \\
\vdots & a_{n-1} & \ddots & & \vdots & b_{m-1} & \ddots & & \\
\vdots & \vdots & \ddots & a_n & b_1 & \vdots & \ddots & \ddots & \\
a_1 & \vdots & & a_{n-1} & b_0 & b_1 & & \ddots & b_m \\
a_0 & a_1 & & \vdots & & b_0 & \ddots & & b_{m-1} \\
 & a_0 & \ddots & \vdots & & & \ddots & & \vdots \\
 & & \ddots & a_1 & & & & & b_1 \\
 & & & a_0 & & & & & b_0
\end{pmatrix}$$

mit $a = \sum_{i=0}^{n} a_i x^i$ und $b = \sum_{i=0}^{m} b_i x^i$.
Die Resultante $\operatorname{Res}_{n,m}(a,b) = \det S_{n,m}(a,b) \in R$. Für $b$ nicht konstant, sei $\operatorname{Res}(0,b) = 0$ und $\operatorname{Res}(0,b) = 1$ sonst. Analog für $b = 0$.
Wir schreiben $\operatorname{Res}(a,b) = \operatorname{Res}_{\deg a, \deg b}(a,b)$.

**Bemerkung 5.2:** Sei $R[x]_{<n}$ der $R$-Modul der Polynome vom Grad $< n$. Dann ist $S_{n,m}$ die Matrix bzgl. der Standartbasis $1, x, \ldots, x^{n-1}$ von

$$\phi_{a,b} : R[x]_{<n} \times R[x]_{<m} \to R[x]_{<m+n-1} : (s,t) \mapsto sa + tb$$

**Lemma 5.3:** Seien $0 \neq a, b \in R[x]$ mit $\deg a \leq n$, $\deg b \leq m$, nicht beide konstant. Ferner sei $R$ eigebettet in den Quotientenkörper $K$. Dann sind äquivalent:

(1) $\deg a < n$ und $\deg b < n$ oder $\deg \operatorname{ggT}(a,b) > 0$
(2) $\operatorname{Res}_{n,m}(a,b) = 0$
(3) Es gibt $s$, $t$ mit $0 \leq \deg s < m$, $0 \leq \deg t < n$ und $sa + tb = 0$

**Beweis.** Sei $\deg a < n$ und $\deg b < m$. Dann ist die 1. Zeile der Sylvestermatrix 0, also auch die Resultante. In diesem Fall gilt dann $ba - ab = 0$ und die Gradbedingungen sind erfüllt, also gilt in diesem Speziallfall die Aussage. Im weiteren sei deshalb oBdA $\deg a = n$ und $a \neq 0$ (ggf.muss man noch $a$ und $b$ vertauschen).

Setze $c := \operatorname{ggT}(a,b)$. Angenommen, $c$ ist nicht konstant. Dann sind $s := b/c$ und $t := a/c$ Polynome vom Grad $\deg s < \deg b \leq m$ und $\deg t < \deg a = n$ mit $0 = sa + tb$. Wegen $a \neq 0$ ist $t \neq 0$. Ist $c$ eine Konstante, so folgt aus $sa + tb = 0$

sofort $a|tb$ und $a|t$ was $t = 0$ oder $\deg t \geq \deg a = n$ impliziert. $t = 0$ liefert natürlich auch $s = 0$. Damit ist dann (1) $\iff$ (3).

(2) $\iff$ (3): Über dem Körper gilt $\mathrm{Res}_{n,m}(a, b) = 0$ genau dann, wenn $\ker \phi_{a,b} \neq 0$ gilt. Aber dann gibt es $s$, $t$ mit den geforderten Eigenschaften. □

LEMMA 5.4: Seien $a$, $b$ wie oben. Dann gibt es $s \in R[x]_{<m}$ und $t \in R[x]_{<n}$ mit

$$sa + tb = \mathrm{Res}_{n,m}(a, b)$$

BEWEIS. Für $\mathrm{Res}(a, b) = 0$ haben wir es schon gezeigt. Sonst folgt es aus der Berechnung der Inversen Matrix mit Hilfe der adjungierten Matrix (oder Cramerschen Regel): Es gibt eine Matrix $\tilde{S} \in R^{(n+m)\times(n+m)}$ mit

$$S\tilde{S} = \mathrm{Res}_{n,m}(a, b) I_{n+m}$$

Aus $\tilde{S}$ können wir dann $s$ und $t$ ablesen. □

Die Anwendung von Resultanten ist nun Polynome mit „schönen“ Eigenschaften zu bauen, dies ist ein Beispiel:

BEISPIEL 5.5: $a$, $b \in R[x]$ normiert. Über einen Zerfällungskörper gelte $a = \prod(x - \alpha_i)$ und $b = \prod(x - \beta_i)$. Setze $S := R[x][y]$ und betrachte

$$r := \mathrm{Res}_y(a(x - y), b(y)) \in R[x]$$

(Wir schreiben $\mathrm{Res}_y$ um verwirrung zu vermeinden)

Dann gilt $r = \prod(x - \alpha_i - \beta_j)$.

BEWEIS. Die Koeffizenten von $a(x - y)$ als Polynome in $x$ haben maximal Grad $\deg a$, ferner gibt es genau einen Koeffizenten vom Grad $\deg a$, alle anderen sind kleiner. Die Koeffizenten von $b$ sind Konstant. In der Sylvestermatrix gibt es genau $\deg b$ viele Spalten wo die Koeffizenten von $a$ Eingetragen werden, damit folgt aus der Definition der Determinante sofort, dass $\deg r \leq \deg a \deg b$ gilt.

Wir haben gezeigt, dass es $s$, $t \in R[x][y]$ gibt mit

$$r(x) = s(x, y)a(x - y) + t(x, y)b(y) \in R[x]$$

. Auswerten an $(\alpha_i + \beta_j, \beta_j)$ zeigt

$$r(\alpha_i + \beta_j) = s(\alpha_i - \beta_j, \beta_j)a(\alpha_i) + t(\alpha_i - \beta_j, \beta_j)b(\beta_j) = 0$$

Damit ist die Behauptung bewiesen. □

Unsere Hauptanwendung wir es sein, das Faktorisieren zu erweitern. Sei $g \in \mathbb{Z}[t]$ normiert und irreduzibel. Dann ist $K := \mathbb{Q}[t]/f$ ein Körper, ein sog. Zahlkörper. Wir wollen im weiteren Polynome in $K[x]$ zerlegen. Dies wird gelingen indem wir mit Hilfe von Resultanten das Problem auf $\mathbb{Q}[t]$ reduzieren.

LEMMA 5.6: Sei $g \in \mathbb{Z}[t]$ normiert und irreduzibel, $K := \mathbb{Q}[t]/g$ und $f \in K[x]$. Wir haben die kanonische Projektion:

$$\mathbb{Q}[t][x] \to K[x] : \sum c_i(t)x^i \mapsto \sum (c_i(t) \bmod g)x^i$$

Dann können wir ein Urbild von $f$ unter der Projektion erhalten, wir schreiben dies als $f \in \mathbb{Q}[t][x]$. Ferner, in $\mathbb{C}$ gilt $g = \prod(t - \alpha_i)$ und wir haben für jedes $\alpha \in \mathbb{C}$ mit $g(\alpha) = 0$ eine Einbettung

$$\phi_\alpha : K \to \mathbb{C} : \sum a_i t^i \bmod f \mapsto \sum a_i \alpha^i$$

(Dies sind Körpermorphismen!). Die Einbettung definiert eine Abbildung

$$\phi_\alpha : K[x] \to \mathbb{C}[x] : \sum a_i x^i \mapsto \sum \phi_\alpha(a_i)x^i$$

Dann gilt

$$N_g(f) := \mathrm{Res}_t(f, g) = \prod \phi_{\alpha_i}(f)$$

BEWEIS. Sei $\mathbb{C}[x] \ni f_i := \phi_{\alpha_i} = \prod(x - \beta_{i,j})$ und

$$N_g(f)(x) = s(t, x)f(t, x) + r(t, x)g(t)$$

Dann ist $N_g(f)(\beta_{i,j}) = s(\alpha_i, \beta_{i,j})f(\alpha_i, \beta_{i,j} + r(\alpha_i, \beta_{i,j}g(\alpha_i)$, aber $f(\alpha_i, \beta_{i,j} = \phi_{\alpha_i}(\beta_{i,j}) = 0$ also gilt $N_g(f)(\beta_{[i,j}) = 0$ für alle $i$, $j$. Aus Grad-Gründen ($\deg N_g(f) = \deg g \deg f$) muss dann die Identität gelten. □

SATZ 5.7: Sei $g \in \mathbb{Z}[t]$ normiert und irreduzibel, $K := \mathbb{Q}[t]/g$ und $f \in K[x]$. Wir haben die kanonische Projektion:

$$\mathbb{Q}[t][x] \to K[x] : \sum c_i(t)x^i \mapsto \sum (c_i(t) \bmod g)x^i$$

Dann können wir ein Urbild von $f$ unter der Projektion erhalten, wir schreiben dies als $f \in \mathbb{Q}[t][x]$. Dann gilt: Setze $r_s := N_g(f(x - st)) = \mathrm{Res}(f(x - st), g)$ für $st \in \mathbb{Z}[t]$.

(1) $r_s \in \mathbb{Q}[x]$ hat Grad $\deg g \deg_x f$
(2) Falls $f$ irreduzibel ist, so gilt $r_s = \tilde{r}^l$ für ein irreduzibles Polynom in $\mathbb{Q}[x]$.
(3) Sei $f$ quadratfrei, dann gibt es (viele) $s$ so, dass $r_s$ quadratfrei ist

BEWEIS. Angenommen, $N_g(f) = cd$ für Polynome $c$, $d \in \mathbb{Q}[x]$ mit $\mathrm{ggT}(c, d) = 1$. Sei wieder wie oben $\phi_{\alpha_i} : K[x] \to \mathbb{C}[x]$ definiert. Dann gilt $\phi_{\alpha_i}(f)|N_g(f)$ nach dem Lemma, also (wegen $\phi_{\alpha_i}(f)$ ireduzibel), entweder $\phi_{\alpha_1}(f)|c$ oder $|d$. OBdA, $\phi_{\alpha_1}(f)|c$. Die Abbildung

$$\mathbb{Q}[\alpha_1] \to \mathbb{Q}[\alpha_i] : \alpha_1 \mapsto \alpha_i$$

ist ein $\mathbb{Q}$-Körperisomorphsimus der $\mathbb{Q}$ invariant lässt. Also Folgt damit auch $\phi_{\alpha_i}(f)|c$ für alle $i$, also $N_g(f)|c$ und $d = 1$. Also kann $N_g(f)$ nicht in koprime zerlegt werden, es ist also die Potenz eines einzigen. Der Übergang von $N_g(f)$ zu $N_g(f(x - s))$ ist nur ein Umbennen, also gilt es auch hier.

Das Lemma gibt die Nullstellen von $N_g(f)$ explizit an. $N_g(f)$ ist quadratfrei genau dann, wenn es da keine doppelten gibt. Dies ist aber eine endliche Bedingung ($s\alpha_i + \beta_{i,j} = s\alpha_l + \beta_{l,k}$) die einfach zu vermeiden ist. □

Zum Beispiel für $f \in \mathbb{Z}[x]$ gilt $N_g(f) = f^{\deg g}$ nicht quadratfrei.

SATZ 5.8: Seien $g \in \mathbb{Z}[t]$, $f \in \mathbb{Q}[t]/f$, $f$ und $N_g(f)$ quadratfrei und $N_g(f) = \prod G_i \in \mathbb{Q}[x]$ mit $G_i$ irreduzibel. Setze $f_i(t,x) := \operatorname{ggT}(G_i(t), f(t,x))$ dann gilt

(1) $f_i \in K[x]$ ist irreduzibel
(2) $f = \prod f_i$

BEWEIS. Sei $v \in K[t]$ irreduzibel mit $v|f$. Damit folgt dann sofort $N_g(v)|N_g(f)$ und $N_g(v)$ ist eine Potenz eines irreduziblen Polynoms. Da $N_g(f)$ quadratfrei ist, muss $N_g(v)$ dann irreduzibel sein. Also gilt $N_g(v) = G_i$ für $i$ geeignet.

Aus $f = \prod \tilde{f}_i$ folgt natürlich auch $N_g(f) = \prod N_g(\tilde{f}_i)$, so dass jeder irreduzible Faktor von $N_g(f)$ die Norm eines Faktors von $f$ ist.

Noch zu zeigen ist: für $v|f$ irreduzibel in $K[x]$, $N_g(v)$ quadratfrei, gilt $v = \operatorname{ggT}(f, N_g(v))$. Angenommen, $w|\operatorname{ggT}(f, N_g(v))$, dann gilt $w|N_g(v)$ und $N_g(w)|N_g(N_g(v)) = N_g(v)^n$. Da $N_g(w)$ die Potenz eines Irreduziblen ist, und $N_g(w)|N_g(f)$, aber $N_g(f)$ quadratfrei, also ist $N_g(w)$ irreduzibel. Analog folgt natürlich $N_g(v)$ irreduzibel, also $N_g(v) = N_g(w)$. Falls jetzt $w \neq v$, dann gilt $vw|f$, also $N_g(vw) = N_g(v)^2|N_g(f)$, aber da $N_g(f)$ quadratfrei ist, kann dies nicht passieren. □

ALGORITHMUS 5.9: Input: Polynome $g \in \mathbb{Z}[t]$ normiert und irreduzibel und $f \in (\mathbb{Q}[t]/g)[x]$
Output: Faktorisierung von $f$
1: Zerlege $f = \prod f_i^i$ mit $f_i$ quadratfrei.
2: Für jedes $f_i$ tue:
3: Finde $s$ mit $n := N_g(f_i(x + st))$ quadratfrei
4: Faktorisiere $\mathbb{Q}[x] \ni n = \prod n_i$
5: Berechne $\operatorname{ggT}(f, n_i(x - st))$ um die Faktoren zu erhalten

BEISPIEL 5.10: $g := t^2 + 1$, also $K = \mathbb{Q}[i] = \mathbb{Q}[t]/g$. Als erstes Versuchen wir $f := x^4 + 1$ zu zerlegen. Wegen $f \in \mathbb{Z}[x]$ folgt $N_g(f) = f^2$, also nicht quadratfrei. Für $s = 1$, also $f(x + i)$ erhalten wir $N_g(f(x + i)) = x^8 + 4x^6 + 8x^4 - 8x^2 + 4$ was quadratfrei ist. Die Faktorisierung ist $(x^4 + 2x^2 - 4x + 2)(x^4 + 2x^2 + 4x + 2)$. Wir erhalten $\operatorname{ggT}(f, ) = x^2 \pm i$ - wie erwartet.

Dieser Algorithmus kann natürlich verbessert werden:

- Wenn $N_g(f)$ nicht quadratfrei ist, so können wir es trozdem zerlegen. Wir werden so keine vollständige Faktorisierung bekommen, aber eine Vereinfachung (kleinerer Grad)
- Wir wissen, dass in der Faktorisierung über $\mathbb{Q}$ alle Grade durch $\deg g$ teilbar sein müssen, also brauchen wir weniger Möglichkeiten zu testen
- Wir können auch im Prinzip den Hensel Algorithmus direkt verallgemeinern - wenn wir ein paar Probleme lösen können: Primzahlen werden Primideale und symmetrische Reste werden schwierig.
- Wir haben gesehen, dass ggT über $\mathbb{Q}$ schwierig und teuer ist, daher haben wir modulare Ansätze verfolgt. ggT über $K$ ist leider noch schwieriger und braucht ähnliche modulare Algorithmen.

## 1. Der Satz vom Primitiven Element

Als letzte Anwendung sehen wir un noch den Satz vom Primitiven Element an - zumindest im Speziallfall von Erweiterungen von $\mathbb{Q}$.

Sei $g \in \mathbb{Z}[t]$ normiert und irreduzibel, dann ist $K := \mathbb{Q}[t]/g$ ein (Zahl-)Körper. Ist nun $f \in K[x]$ irreduzibel, so ist $E := K[x]/f$ ebenfalls ein Körper. Was wir hier zeigen wollen ist der folgende Satz:

SATZ 5.11: Seien $f$, $g$ wie oben. Dann gibt es ein irreduzibles Polynom $h \in \mathbb{Q}[t]$ mit $F := \mathbb{Q}[t]/h$ ist isomorph (als Körper) zu $E$.

Mit der Hilfe von Resultanten und ggT werden wir dies sogar Algorithmisch lösen und den Isomorphismus explizit angeben.

Mit dem oben gezeigten gilt:

LEMMA 5.12: Es gibt ein (viele) $s \in \mathbb{Z}$ so, dass $N_g(f(x + st))$ irreduzibel ist.

BEWEIS. Wir haben gezeigt, dass es $s$ gibt, so dass $N_g(f(x + st))$ qudratfrei ist. Da aber die Anzahl der Faktoren von $N_g(f(x + st))$ in $\mathbb{Q}[t]$ und die von $f \in K[x]$ übereinstimmt, ist $N_g(f(x + st))$ irreduzibel. □

Ab jetzt nehmen wir ObDa an, dass $N_g(f)$ quadratfrei ist, also $s = 0$ erlaubt ist.

LEMMA 5.13: Unter den Voraussetzungen wie im letzten Lemma gilt $\mathbb{Q}[t]/h \sim E$ als Körper mit $h := N_g(f(x))$.

BEWEIS. Da die Grade (über $\mathbb{Q}$) gleich sind, reicht es zu zeigen, dass $h$ eine Nullstelle in $E$ hat. Aber wir wissen, $f|h$ und $f$ hat eine Nullstelle in $E$ $(x + f)$. Damit ist (in $E$), $x + f$ eine Nullstelle von $h$ und die Abbildung

$$\mathbb{Q}[t]/h \to E : t + h \to x + f$$

ist wohldefiniert, nicht trivial und damit ein Isomorphismus. □

Damit können wir jetzt Zerfällungskörper ausrechnen: Sei $f \in \mathbb{Z}[t]$. Wir faktorisieren $f$ über $\mathbb{Q}$ und erhalten irreduzible Faktoren $f_i$. Setze $K := \mathbb{Q}[t]/f_i$ für einen Faktor mit $\deg f_i > 1$. Dann können wir $f$ über $K$ zerlegen. falls es wieder Faktoren $f_i \in K[t]$ vom Grad $\deg f_i > 1$ gibt so erhalten wir eine Erweiterung von $K$. Mit den Techniken können wir daraus dann wieder eine „normale“ Erweiterung von $\mathbb{Q}$ bauen und den Algorithmus iterieren. Zum Schluss haben wir dann eine Körper $K$ gegeben durch ein irreduzibles Polynom $h \in \mathbb{Z}[t]$ so, dass über $K$ $f$ nur lineare Faktoren hat. Nach Konstruktion ist dies der kleinste Körper mit dieser Eigenschaft.

ANHANG A

# Übungszettel