CSE291-14: The Number Field Sieve

https://cseweb.ucsd.edu/classes/wi22/cse291-14

Emmanuel Thomé

January 27, 2022
Part 4a

Polynomial selection in NFS

Introduction

Size of the coefficients: possible and impossible things

Searching for good polynomials with base-$m$

Skewness

Non-monic linear polynomials: Kleinjung’s 2005 algorithm
Plan

Introduction

Size of the coefficients: possible and impossible things

Searching for good polynomials with base-$m$

Skewness

Non-monic linear polynomials: Kleinjung’s 2005 algorithm
Our goal: review the different methods for polynomial selection.

- Why is it important?
- What kind of game is it?
- What are the different methods?
- How do we measure the quality of the output?
The polynomial selection phase is when we choose the pair of polynomials that define both sides of the NFS diagram.

- The algebraic polynomial $f$ defines the number field.
- The rational polynomial (thus far, $x - m$) completes the picture.

In practice, this is more general

In fact, not even $x - m$ is monic. Several methods do relax this condition (but not the first ones).
In some cases, both polynomials are of degree $> 1$. 
Importance of polynomial selection

Polynomial selection is important because it determines the size of the “norms” (actually, of the integers being checked for smoothness).

- Asymptotic analysis crudely reduced polynomial selection to the choice of the pair \((D, \delta)\).
- We eventually found out that \(\delta\) was controlling the compromise between the size of \(\text{Res}(\phi, x - m)\) and of \(\text{Res}(\phi, f)\).

This general role is also true in practice

A good polynomial selection makes these “norms” small as \(\phi(x) = a - bx\) ranges over the values we explore.

- Certainly, some things can be achieved, and some can’t.
- Can we force these values to be smooth more often than on average?
A general workplan

Starting point: a method that can yield good polynomial pairs.

- Arrange so that the method has many degrees of freedom.
- Explore a huge search space to find exceptional situations.
- Find reasonable assessment criteria that make it possible to identify which are the “exceptionally good” polynomial pairs.
An extension: non-linear rational polynomial

What if we replace $x - m$ by $m_1x - m_0$?

- $\text{Res}(a - bx, x - m) = a - bm$ becomes $\text{Res}(a - bx, m_1x - m_0) = am_1 - bm_0$, which looks nicer.
- If we write $m = m_0/m_1 \in \mathbb{Q}$ and that $f(m) \equiv 0 \pmod{N}$, everything works as before.
- The condition to meet is the existence of a common root:

$$\text{Res}(f(x), m_1x - m_0) \equiv 0 \pmod{N}.$$ 

This extra degree of freedom has been part of all polynomial selection algorithms since the early 2000s.
An extension: higher degree polynomials

If a polynomial selection can find a pair of nonlinear polynomials:

- whose resultant is divisible by $N$ with multiplicity 1
- and with a known common root in $\mathbb{Z}/N\mathbb{Z}$

Then we can work exactly along the lines of NFS.

Caveat: no such thing is known in general, EXCEPT for DLP.

- NFS for DLP (discrete logs in $\mathbb{Z}/p\mathbb{Z}^\times$): $p$ replaces $N$.
- The existence of root finding mod $p$ is the key.
- In some cases (but not always), this wins.
Traditionally, notations are as follows:

- $f$ is the algebraic polynomial.
  The coefficients are named $f_0, \ldots, f_d$, or $a_0, \ldots, a_d$.
- $g$ is the linear polynomial.

Often, to highlight the symmetric roles played by the two sides:

- $f_0$ is “the polynomial on side 0” (typically $\deg f_0 = 1$).
- $f_1$ is “the polynomial on side 1”.

But this messes with the per-coefficient notations. Notations $a_0, \ldots, a_d$ are preferred for coefficients of the nonlinear polynomial in this case.

Implementations such as Cado-NFS are mostly agnostic w.r.t side numbering.
Plan

Introduction

Size of the coefficients: possible and impossible things

Searching for good polynomials with base-\(m\)

Skewness

Non-monic linear polynomials: Kleinjung’s 2005 algorithm
Our first approach consists in searching for “small” polynomial pairs.

- Eventually, one of our guides will be the size of the integers we will try to factor.
- Given the power dependency in the degrees of the polynomials, we have only a few possible choices for the degree.
- Given a choice for $d = \deg f_1$, can we obtain polynomials $f_0$ and $f_1$ with small coefficients?
A good question to ask

In order to reach all integers in a range \([M, 2M]\), how large do we have to choose the coefficients of \(f_0\) and \(f_1\)?

Let \(M^{c_0}\) be a bound on the coefficients of \(f_0\) (likewise \(M^{c_1}\) for \(f_1\)).

- **First constraint:** to reach \(M\) different values with the degrees of freedom that we have:

\[
c_0 \cdot (d_0 + 1) + c_1 \cdot (d_1 + 1) \geq 1.
\]

- **Second constraint:** \(\text{Res}(f_0, f_1)\) must be at least \(M\).

Since \(\text{Res}(f_0, f_1) = M^{o(1)} \| f_0 \|_{\deg f_1} \| f_1 \|_{\deg f_0}\), we must have:

\[
c_0 \cdot d_1 + c_1 \cdot d_0 \geq 1.
\]
Two different constraints

Note that the constraints are of different nature.

1. \( c_0 \cdot (d_0 + 1) + c_1 \cdot (d_1 + 1) \geq 1 \).
   Pairs not meeting this constraint may exist, but such a family cannot reach all integers.

2. \( c_0 \cdot d_1 + c_1 \cdot d_0 \geq 1 \).
   It is outright impossible for pairs to not meet this constraint, and be useful for NFS.
Important example #1

Take what the naive polynomial selection method gives: \( d_0 = 1, \ d_1 = d, \ c_0 = c_1 = \frac{1}{d+1}. \)

- \( c_0 \cdot (d_0 + 1) + c_1 \cdot (d_1 + 1) = \frac{2}{d+1} + \frac{d+1}{d+1} \geq 1. \)
- \( c_0 \cdot d_1 + c_1 \cdot d_0 = \frac{d}{d+1} + \frac{1}{d+1} = 1. \)

Put otherwise, the resultant bound is tight, but there is immense legroom in the choice of \( f_0. \)
Important example #2

What can we obtain with $c_1 = 0$?
i.e., a family of algebraic polynomials with coefficients bounded by a constant.

The remaining constraint rewrites simply as

$$c_0 d_1 = 1,$$

which does not say much.

**Does this do anything?**

If we have access to a fictitious oracle that outputs such a polynomial $f_1$, what does it give?
SNFS: polynomial selection with an oracle

If we have access to a fictitious oracle that outputs such a polynomial $f_1$, what does it give?

- We can do the entire NFS analysis based on that.
- The algebraic norm can be rewritten as $L_N[1/3, \sqrt{\frac{\alpha}{\delta}} + \alpha \delta]$.
- This changes the optimum $\delta$ from $\sqrt{2/\alpha}$ to $\sqrt{1/\alpha}$.
- Eventually, we end up with $L_N[1/3, (32/9)^{1/3} + o(1)]$.

This is called SNFS.

- The “special” integers are those that are precisely reached by this “ideal” choice.
- By extension, the SNFS term is also used for anything that is reached by a non-general polynomial selection.
The constraint space

Example for \( d_0 = 1 \) and \( d_1 = 6 \).

Note that \( c_0 + c_1 \) appears in the smoothness probability.

\[
c_0 + c_1 = \log(\|f_0\| \cdot \|f_1\|)/\log N.
\]

\( c_0 + c_1 \) measures the polynomial-dependent part of the maximum size of the integers which are checked for smoothness.

Thus the intersection point \( P \) is “ideal”. Alas, moving towards \( P \) is expensive.
Where are we?

- Base-$m$ polynomial selection is a starting point.
- We have an argument that explains that it is not “optimal”.
- SNFS numbers are really special.
Plan

Introduction

Size of the coefficients: possible and impossible things

Searching for good polynomials with base-$m$

Skewness

Non-monic linear polynomials: Kleinjung’s 2005 algorithm
What can base-\(m\) do?

Notations: \((f_0, f_1)\), \(\deg f_1 = d\), \(f_1 = \sum a_i x^i\).

Recall that the simplistic base-\(m\) method chooses \(m \approx N^{1/(d+1)}\).

- There is an immense degree of freedom in the choice of \(m\).
- Can we do many trials and hope for something nice to happen?

Opportunities for improvement:

- It is not a very big deal if \(\|f_0\|\) (max coefficient of the linear polynomial) increases by a tiny bit.
- Can this be compensated by a larger decrease of \(\|f_1\|\)?
Base-\(m\), revisited

Instead of picking \(m\) first, and then the coefficients of \(f\):

- Choose \(a_d\) first, slightly smaller than \(N^{1/(d+1)}\).
- Then choose \(m\), and deduce the rest of the coefficients.

Our game: correlation between effort and yield

Ultimately, we want to answer the question:

“If we generate \(C\) polynomial pairs, what is the best we can obtain, as a function of \(C\)?”

Can also be phrased as: if \(a_d \approx N^{1/(d+1)}/c\), how many trials does it take to have all coefficients of \(f\) close to \(a_d\)?
Base-\(m\), revisited

Let \(c\) be an arbitrary number.

- Choose \(a_d \approx N^{1/(d+1)}/c\) (many possible choices!)
- Let \(m = \lfloor (N/a_d)^{1/d} \rfloor = (N/a_d)^{1/d} + \mu\) with \(|\mu| \leq 1\).

**Lemma:** \(a_{d-1} \approx a_d\)

\[
|a_{d-1}| = \left| \frac{N - a_d m^d}{m^{d-1}} \right| = a_d \left| \frac{(m - \mu)^d - m^d}{m^{d-1}} \right|
\]

\[
\leq a_d m \left| (1 - \mu/m)^d - 1 \right|
\]

\[
\leq da_d \times \text{small constant bound}.
\]

And \(d\) is small as well, so we expect \(a_{d-1}\) to have roughly as many bits as \(a_d\).
Other coefficients

The \(d - 1\) coefficients \(a_0\) to \(a_{d-2}\) are a priori close to \(m\), with:

\[
m \approx \left(\frac{N}{a_d}\right)^{1/d} \approx \left(\frac{N^{1-1/(d+1)}}{c}\right)^{1/d} \approx N^{1/(d+1)}c^{1/d}.
\]

Heuristic: \(a_0\) to \(a_{d-2}\) behave like random integers. With probability

\[
\left(\frac{a_d}{m}\right)^{d-1} \approx \left(c^{-\left(1/(d+1)\right)}\right)^{d-1} = c^{-\left(d^2-1\right)/d},\ all\ are\ \leq\ a_d.
\]

Conclusion

By trying \(c^{(d^2-1)/d}\) values \(a_d\), we expect to:

- change \(\|f_1\|\) to \(N^{1/(d+1)}/c\).
- change \(\|f_0\|\) to \(N^{1/(d+1)} \times c^{1/d}\).
Conclusion (rewrite)

By trying $c^{(d^2-1)/d}$ values $a_d$, we expect to:

- change $\| f_1 \|$ to $N^{1/(d+1)}/c$.
- change $\| f_0 \|$ to $N^{1/(d+1)} \times c^{1/d}$.

We can also write: by trying $C$ values $a_d$, we expect to:

- change $\| f_1 \|$ to $N^{1/(d+1)}/C^{d/(d^2-1)}$.
- change $\| f_0 \|$ to $N^{1/(d+1)} \times C^{1/(d^2-1)}$.
- change $\| f_0 \| \| f_1 \|$ to $N^{2/(d+1)}/C^{1/(d+1)}$.

This moves in the right direction!

- More work leads to smaller polynomials.
- This is woefully exponential, of course.
Plan

Introduction

Size of the coefficients: possible and impossible things

Searching for good polynomials with base-$m$

Skewness

Non-monic linear polynomials: Kleinjung’s 2005 algorithm
Skewness

Notations: \((f_0, f_1), \deg f_1 = d, f_1 = \sum a_i x^i, \phi(x) = u - vx.\)

Skewness is a way to add more flexibility to the polynomial selection process.

Observation: the polynomial \(x - m\) is unbalanced. So is the expression \(\text{Res}(u - vx, x - m) = u - vm.\)

- Can we work with larger \(u\) and smaller \(v\)?
Skewed polynomials

\[ \text{Res}(u - vx, f_1) = F(u, v) = (u^d a_d + \cdots + u^i v^{d-i} a_i + \cdots + v^d a_0). \]

If coefficients of \( f_1 \) have roughly the same size and both coefficients of \( \phi(x) \) are bounded by \( A \), then all \( a_i u^i v^{d-i} \) have the same size.

\[
\begin{align*}
\text{Res}(u - vx, f_1) &= F(u, v) = (u^d a_d + \cdots + u^i v^{d-i} a_i + \cdots + v^d a_0). \\
&= (u^d a_d + \cdots + a_i u^i v^{d-i} + \cdots + v^d a_0).
\end{align*}
\]
Skewed polynomials

\[ \text{Res}(u - vx, f_1) = F(u, v) = (u^d a_d + \cdots + u^i v^{d-i} a_i + \cdots + v^d a_0). \]

- If coefficients of \( f_1 \) have roughly the same size and both coefficients of \( \phi(x) \) are bounded by \( A \), then all \( a_i u^i v^{d-i} \) have the same size.

- If the \( a_i \) are unbalanced, say \( \frac{a_i}{a_{i+1}} \approx S > 1 \), then with \( |u| < A\sqrt{S} \) and \( |v| < A/\sqrt{S} \), all \( a_i u^i v^{d-i} \) have the same size.

The ratio \( S \) is called the \textbf{skewness}; the polynomials are skewed.
Skewed polynomials

\[
\operatorname{Res}(u - vx, f_1) = F(u, v) = (u^d a_d + \cdots + u^i v^{d-i} a_i + \cdots + v^d a_0).
\]

- If coefficients of \( f_1 \) have roughly the same size and both coefficients of \( \phi(x) \) are bounded by \( A \), then all \( a_i u^i v^{d-i} \) have the same size.

- If the \( a_i \) are unbalanced, say \( \frac{a_i}{a_{i+1}} \approx S > 1 \), then with \( |u| < A\sqrt{S} \) and \( |v| < A/\sqrt{S} \), all \( a_i u^i v^{d-i} \) have the same size.

The ratio \( S \) is called the skewness; the polynomials are skewed.
Skewed norm

**Definition**

Given \( P = \sum p_i x^i \in \mathbb{R}[x] \), the \( S \)-skewed (infinity) norm of \( P \) is:

\[
\| P \|_S = \| P \|_{\infty, S} = \max_{0 \leq i \leq \deg P} |p_i S^{i-d/2}|.
\]

All polynomials above have the same \( S \)-skewed norms (with their respective \( S \)). If \( \| P \| = \| Q \|_S \), then

\[
\max\{\text{Res}(u - vx, P), (u, v) \in [0, A]^2\} = \max\{\text{Res}(u - vx, Q), (u, v) \in [0, A\sqrt{S}] \times [0, A/\sqrt{S}]\}.
\]
How do we find skewed polynomials?

When we revisited base-$m$, we chose $a_d$ first, and then $m$. This gave rise to:

- $a_d \approx a_{d-1} \approx N^{1/(d+1)}/c$.
- $m = \frac{d}{\sqrt{N/a_d}} \geq N^{1/(d+1)}$ = the textbook base-$m$.

This polynomial pair is already somewhat skewed, we may turn it to our advantage.
How do we find skewed polynomials?

When we revisited base-$m$, we chose $a_d$ first, and then $m$. This gave rise to:

- $a_d \approx a_{d-1} \approx N^{1/(d+1)}/c$.
- $m = \sqrt{N/a_d} \geq N^{1/(d+1)} = \text{the textbook base-}m$.

This polynomial pair is already somewhat skewed, we may turn it to our advantage.

- Aim at the same skew-norm, starting from a smaller $a_d$: bits we still have to cancel (■), bits we no longer care about (■), new bits to cancel (■), a moderately larger rational norm because $m$ got larger (■).
Analysis of skewed base-\(m\)

Analysis is a bit painful, but the outcome is quite clear:

With the same number of trials, we can expect to find smaller skewed-norms that in the non-skewed base-\(m\).
\(C^{1/(d+1)}\) is replaced by a mildly larger number

Refinements:

- Do not optimize \(a_0\).
- Rationale: this makes it possible to form many linear combinations like \(f_0 + tf_1\) and choose the best one. We’ll get to that with root properties and root sieving.
Plan

Introduction

Size of the coefficients: possible and impossible things

Searching for good polynomials with base-$m$

Skewness

Non-monic linear polynomials: Kleinjung’s 2005 algorithm
Better polynomials: non-monic $f_0$

In fact, $f_0$ can be non-monic:

$$f_0 = m_1 x - m_0.$$

Then, the common root modulo $N$ must be $m = m_0 / m_1$ and

$$\text{Res}(f_1, m_1 x - m_0) = a_d m_0^d + a_{d-1} m_1 m_0^{d-1} + \cdots + a_0 m_1^d.$$

**Remark:** if the latter is equal to $N$, it implies

$$a_d m_0^d \equiv N \mod m_1.$$

First ingredient of Kleinjung’s algorithms (2006 and 2008) is called Kleinjung “Lemma 2.1”. It computes a reasonably good $f_1$ from a fixed choice of $N, d, m_1, m_0$ and $a_d$. 
Kleinjung “Lemma 2.1”

**Input:** $N, d, m_1, m_0$, and some fixed coefficients $[a_j, \ldots, a_d]$

**Output:** A polynomial $f_1$ such that $\text{Res}(f_1, m_1x - m_0) = N$

First, compute

$$r_j = \frac{N - \sum_{i=j+1}^{d} a_i m_0 m_1^{d-i}}{m_1^{d-j}}$$

Then, for $i = j - 1, j - 2, \ldots, 0$, compute:

- $r_i = \frac{r_{i+1} - a_{i+1} m_0^{i+1}}{m_1}$
- $a_i = \frac{r_i + t_i m_1}{m^i_0}$, where $t_i$ is an integer such that

$$-\frac{m^i_0}{2} \leq t_i < \frac{m^i_0}{2} \quad \text{and} \quad t_i \equiv -\frac{r_i}{m_1} \mod m^i_0$$

The output is $f_1 = a_d x^d + a_{d-1} x^{d-1} + \cdots + a_1 x + a_0$. 
Kleinjung’s “Lemma 2.1” algorithm applied to RSA-155 with $d = 5$ and

$$a_5 = 358870426380$$
$$m_0 = 31392776870911769459515309198$$
$$m_1 = 823916492006383$$

gives:

$$f_1 = 358870426380x^5$$
$$+ 428308592054328x^4$$
$$- 16336877672072510723154851996x^3$$
$$- 12601611387318107328006122118x^2$$
$$- 19621855499511523845845304751x$$
$$- 836976378549595985304502899$$
If we apply Kleinjung “Lemma 2.1” with only the leading coefficient $a_d$ fixed and with $m_0$ close to $\tilde{m}_0 = \sqrt{\frac{N}{a_d}}$, the algorithm yields:

- $a_{d-1}$ rather small: $|a_{d-1}| < m_1 + da_d \frac{m_0 - \tilde{m}_0}{m_1}$.
- Other $a_i$’s satisfy $|a_i| < m_1 + m_0$. 

Our goal, and how we reach it

Have coefficient sizes which are a good match to some skewness.

Find smart way to make $a_d - 2$ small.

Rely on (mild) luck to make $a_d - 3$ small.
If we apply Kleinjung “Lemma 2.1” with only the leading coefficient $a_d$ fixed and with $m_0$ close to $\tilde{m}_0 = \sqrt[d]{\frac{N}{a_d}}$, the algorithm yields:

- $a_{d-1}$ rather small: $|a_{d-1}| < m_1 + da_d \frac{m_0 - \tilde{m}_0}{m_1}$.
- Other $a_i$’s satisfy $|a_i| < m_1 + m_0$.

Our goal, and how we reach it

Have coefficient sizes which are a good match to some skewness.
If we apply Kleinjung “Lemma 2.1” with only the leading coefficient $a_d$ fixed and with $m_0$ close to $\tilde{m}_0 = \sqrt{\frac{N}{ad}}$, the algorithm yields:

- $a_{d-1}$ rather small: $|a_{d-1}| < m_1 + da_d \frac{m_0 - \tilde{m}_0}{m_1}$.
- Other $a_i$’s satisfy $|a_i| < m_1 + m_0$.

Our goal, and how we reach it

Have coefficient sizes which are a good match to some skewness.
If we apply Kleinjung “Lemma 2.1” with only the leading coefficient $a_d$ fixed and with $m_0$ close to $\tilde{m}_0 = \sqrt[2]{\frac{N}{a_d}}$, the algorithm yields:

- $a_{d-1}$ rather small: $|a_{d-1}| < m_1 + d a_d \frac{m_0 - \tilde{m}_0}{m_1}$.
- Other $a_i$'s satisfy $|a_i| < m_1 + m_0$.

Our goal, and how we reach it

Have coefficient sizes which are a good match to some skewness.

- Find smart way to make $a_{d-2}$ small.
If we apply Kleinjung “Lemma 2.1” with only the leading coefficient $a_d$ fixed and with $m_0$ close to $\tilde{m}_0 = d\sqrt{\frac{N}{a_d}}$, the algorithm yields:

- $a_{d-1}$ rather small: $|a_{d-1}| < m_1 + da_d \frac{m_0 - \tilde{m}_0}{m_1}$.
- Other $a_i$’s satisfy $|a_i| < m_1 + m_0$.

Our goal, and how we reach it

Have coefficient sizes which are a good match to some skewness.

- Find smart way to make $a_{d-2}$ small.
- Rely on (mild) luck to make $a_{d-3}$ small.
Making $a_{d-2}$ small

Use the equation $a_d m_0^d \equiv N \mod m_1$.

**Key idea**

Build $m_1$ as a product of small primes. Use the combination of modular information to fabricate a small $a_{d-2}$.

- Let $\mathcal{P}$ be a set of small primes $\equiv 1 \mod d$ ($m_1$ will be a product of a subset of $\mathcal{P}$).
- Pick some $a_d$ (e.g. smooth).
- Some primes $r \in Q \subset \mathcal{P}$ give $d$ solutions to $a_d x^d \equiv N \mod r$.
- Any choice of exactly one $d$-th root modulo each of those $r$’s gives a value $m_0$ defined modulo $m_1 = \prod r$ by CRT.

We may choose one which is close to $\tilde{m}_0 = \sqrt[d]{N/a_d}$. 
Many choices

Pick $\ell$ primes for which $a_d x^d \equiv N \mod r$ has $d$ solutions.

- In total, $d^\ell$ possible choices for $m_0$.
- $m_0$ is a linear combination of $\ell$ values among $d \times \ell$.
  This follows from explicit Chinese Remainder Theorem.

Expand the value $a_{d-2}/m_0$ obtained by “Lemma 2.1” from the $d$-th roots of $m_0 \mod r$ that we have chosen.

- By restricting to 1st order terms, we get a linear combination.
- If $a_{d-2}/m_0$ ends up being close to an integer $\lambda$ for some chosen $m_0$, then for $f'_1 = f_1 - \lambda(m_1x - m_0)x^{d-2}$, we have:
  - $a'_{d-2}/m_0$ close to 0,
  - $a'_{d-1}$ does not change much.
Finding combinations that are close to $\mathbb{Z}$

The problem can be reduced to the following:

- $\ell$ sets $S_1, \ldots, S_d$, each containing $d$ real numbers in $[0, 1)$.
- $d^\ell$ choices: $(s_1, \ldots, s_d)$ with each $s_i \in S_i$, and:

$$a_{d-2} \mod \mathbb{Z} \equiv \sum s_i.$$ 

- Naive complexity: $O(d^\ell)$.
- Better:
Finding combinations that are close to $\mathbb{Z}$

The problem can be reduced to the following:

- $\ell$ sets $S_1, \ldots, S_d$, each containing $d$ real numbers in $[0, 1)$.
- $d^\ell$ choices: $(s_1, \ldots, s_d)$ with each $s_i \in S_i$, and:

$$a_{d-2} \mod \mathbb{Z} \equiv \sum s_i.$$

- Naive complexity: $O(d^\ell)$.
- Better: $O(d^{\ell/2})$. 

What do small combinations give?

Algorithm has:
- $a_d$ chosen small.
- $a_{d-1}$ small by construction, $\approx m_1$.
- $a_{d-2}$ small thanks to small combinations.

With some extra luck, $a_{d-3}$ may be somewhat smaller than expected.