On Signature-based Gröbner Bases over Euclidean Rings
The following is joint work by


We are from the

University of Kaiserslautern.
Idea of signatures:

Try to detect zero reductions in advance.
Let $I = \langle g_1, g_2 \rangle \in \mathcal{R} := \mathbb{Q}[x, y, z]$ and let $<$ denote DRL where

\[
g_1 = xy - z^2, \\
g_2 = y^2 - z^2.
\]
Apply signatures in \( R^2 \):

\[
\text{sig}(g_1) = e_1, \\
\text{sig}(g_2) = e_2.
\]

Order signatures by POT (e.g. \( e_2 > x^{1000} e_1 \)).

In general:
\[
\text{sig(} \text{polynomial} \text{)} = \text{lt(} \text{module representation} \text{)}
\]

Main idea: Try to keep signatures minimal.
Generate first S-pair:

\[
g_3 \; := \; \text{sp}(g_1, g_2) = yg_1 - xg_2
\]
\[
= \; y(xy - z^2) - x(y^2 - z^2)
\]
\[
= \; xz^2 - yz^2.
\]

\[
\text{sig}(g_3) \; = \; \text{lt}(ye_1 - xe_2) = -xe_2.
\]
sp(g_3, g_2) reduces to zero:

lt(g_2) = y^2 \text{ coprime to } lt(g_3) = xz^2

(Buchberger’s Product Criterion)
\[ \text{sp}(g_3, g_2) \text{ reduces to zero (signature edition)}: \]

\[
\text{sig}(\text{sp}(g_3, g_2)) = \text{lt}(y^2(ye_1 - xe_2) - xz^2e_2) \\
= -xy^2e_2. 
\]

Use syzygy \( g_1e_2 - g_2e_1 \) with lead term \( xye_2 \)

▷ Reduce module representation

▷ Lower signature for \( \text{sp}(g_3, g_2) \)

*(Syzygy Criterion)*
What’s about $\text{sp}(g_3, g_1)$? 

Buchberger’s Product Criterion? **NO**

Buchberger’s Chain Criterion? **NO**
But \( \text{sp}(g_3, g_1) \) reduces to zero:

\[
\text{sig}(\text{sp}(g_3, g_1)) = \text{lt}(y(ye_1 - xe_2) - z^2e_1) = -xye_2.
\]

**Again:** Syzygy \( g_1e_2 - g_2e_1 \) with lead term \( xy e_2 \)

▷ Reduce module representation

▷ Lower signature for \( \text{sp}(g_3, g_1) \)

*(Syzygy Criterion)*
Precondition for this talk: Next chosen S-pair from pair set has minimal possible signature.

Note: Over fields this limitation is not required, but makes life easier.
General rule
For each signature handle exactly one element.

Sketch of proof
Take pairs by increasing signature.
Think in the module: $\alpha, \beta$ module elements.

If $\text{sig}(\alpha) = \text{sig}(\beta)$ reduce in module.
(We are not cancelling the polynomial lts!)

▷ $\text{sig}(\alpha - \beta) < \text{sig}(\alpha), \text{sig}(\beta)$.
▷ Algorithm has handled these relations already.
Let’s compute with signatures over Euclidean rings.
Problem not mentioned until now: **sig-reductions**

When reducing polynomials (**over fields**) we are not allowed to change the signature:

**Increasing signature?**
Well, we want small signatures.

**Decreasing signature?**
We can throw away the element, criteria apply.

**Over fields** $\Rightarrow$ **computation by increasing signature.**
Over Euclidean rings stuff gets more difficult.

We want strong Gröbner Bases:
For all $f \in I \setminus \{0\}$ there exists $g \in G$ s.t. $\text{lt}(g) \mid \text{lt}(f)$. ($G \subset I$ and $\text{L}(G) = \text{L}(I)$ is not enough anymore.)
Have to take care of the **coefficients**, too:

If

\[ \text{l.m.}(g) \mid \text{l.m.}(f) \quad \text{&} \]

\[ \exists \, a, b \text{ coefficients s.t.} \]

\[ \text{l.c.}(f) = a \text{l.c.}(g) + b, \; a \neq 0 \; \text{and} \; b < \text{l.c.}(f) \]

\[ \text{then compute } f - a \frac{\text{l.m.}(f)}{\text{l.m.}(g)} g. \]

*(Either smaller l.m or smaller l.c!)*
Same process generalizes concept of S-pairs:

We need to consider **GCD-pairs**.
For **efficiency** we need to relax signature handling:

Allow signature changes on the **coefficient** level.
Two main problems arise from this.
Over Euclidean rings we can no longer guarantee that the computation of signature-based Gröbner Bases is done by increasing signatures.
We need to **restrict signature-based criteria** to remove useless elements:

We can only remove elements for a given signature \( S \) if there exists a syzygy \( \pi \) s.t. \( \text{lt}(\pi) \mid S \)
Still, **signature drops** may appear.

**Idea**

▷ Stop computation at this point.
▷ Interreduce intermediate basis without considering signatures.
▷ Apply new signatures / module representations and restart.
Optimization 1: Exploit GCD-pairs

Replace $f \in G$ by $gp(f, g)$ if there exists $g \in G$ s.t.

$$gp(f, g) = (\pm 1)f + ctg \quad \& \quad \text{sig}(gp(f, g)) = (\pm 1)\text{sig}(f).$$

▷ Trying to keep coefficient growth at a low.
Optimization 2: **Optimistic sig-reductions**

- **sig**-reduce an element $f$ w.r.t. $G$.

- If there exists $g \in G$ s.t. $c t l t(g) = l t(f)$ and $s i g(f - c t g) < s i g(f)$ for some $c$ and $t$, then start **usual reduction process** (no longer taking care of signatures).

- If $f$ reduces to zero we can go on.

- Otherwise we have to restart the computation.
Restarting is a huge bottleneck in general.

But often the intermediate computed elements are quite useful for further computations.
Optimization 3: Hybrid algorithm

▷ Start with signature-based algorithm.

▷ If the signature drops, restart for a (small) number of times the signature-based algorithm.

▷ Take intermediate basis and start non-signature-based Gröbner basis computation.
<table>
<thead>
<tr>
<th>Examples</th>
<th>STD</th>
<th>HBA</th>
<th>STD/HBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.43</td>
<td>0.37</td>
<td>28.19</td>
</tr>
<tr>
<td>2</td>
<td>24.91</td>
<td>0.10</td>
<td>249.10</td>
</tr>
<tr>
<td>3</td>
<td>87.27</td>
<td>0.39</td>
<td>223.77</td>
</tr>
<tr>
<td>4</td>
<td>83.51</td>
<td>0.20</td>
<td>417.55</td>
</tr>
<tr>
<td>5</td>
<td>23,200.05</td>
<td>5,873.21</td>
<td>3.95</td>
</tr>
<tr>
<td>6</td>
<td>134.29</td>
<td>0.61</td>
<td>220.15</td>
</tr>
<tr>
<td>7</td>
<td>1,004.56</td>
<td>1,128.07</td>
<td>0.89</td>
</tr>
<tr>
<td>8</td>
<td>554.02</td>
<td>337.55</td>
<td>1.641</td>
</tr>
</tbody>
</table>
Up next

Throw some machine learning on it.
And what’s about finite rings?
Thank you for your attention.
Questions? Remarks?