Computing and Using Minimal Polynomials

Presentation · June 2018

CITATIONS 0

READS 72

4 authors:

John Abbott
Università degli Studi di Genova
85 PUBLICATIONS 452 CITATIONS

Anna Maria Bigatti
Università degli Studi di Genova
99 PUBLICATIONS 727 CITATIONS

Elisa Palezzato
Università degli Studi di Genova
8 PUBLICATIONS 10 CITATIONS

Lorenzo Robbiano
Università degli Studi di Genova
133 PUBLICATIONS 2,545 CITATIONS

Some of the authors of this publication are also working on these related projects:

Teaching View project

CoCoA and CoCoALib: software presentations, tutorials, demos View project

All content following this page was uploaded by Anna Maria Bigatti on 21 June 2018.

The user has requested enhancement of the downloaded file.
Computing and Using Minimal Polynomials

J. Abbott
A.M. Bigatti
E. Palezzato
L. Robbiano

Anna Maria Bigatti
Università di Genova

SC²: H2020-FETOPEN-2016-2017-CSA project 712689 www.sc-square.org
Definition

K a field, \( P = K[x_1, \ldots, x_n] \), \( I \) zero-dimensional ideal in \( P \).

The **minimal polynomial** of a polynomial \( f \) modulo \( I \), \( \mu_{f,I}(z) \in K[z] \), is the monic polynomial in \( K[z] \) of minimum degree such that

\[
\mu_{f,I}(f) \in I \quad \text{or equiv.} \quad \mu_{f,I}(\bar{f}) = \bar{0} \text{ in } P/I
\]

```plaintext
/***/ I := ideal(x^2, y^2);
/***/ MinPolyQuot(x+y,I, t);
t^3         --------> (x+y)^3 is in I
/***/ f := x^2 - 3*x*y +1;
/***/ MinPolyQuot(f,I, t);
t^2 -2*t +1  ----> f^2 -2*f +1 is in I
```
Remark

If $x_i$ an indeterminate in $P = K[x_1, \ldots, x_n]$

$\mu_{x_i,I}(x_i)$ is the lowest degree $x_i$-univariate polynomial in $I$

i.e. $I \cap K[x_i] = \langle \mu_{x_i,I}(x_i) \rangle$.

```c++
/**/ I := IdealOfPoints(P, mat([[1,2], [3,2], [5,4]]));
/**/ MinPolyQuot(x,I, x);
x^3 -9*x^2 +23* x -15 -----> (x -1)(x -3)(x -5)
/**/ MinPolyQuot(y,I, y);
y^2 -6*y +8 -----> (y -2)(y -4)
```

Remark

For a CAS like CoCoA $\rightarrow$ Gröbner Bases $\rightarrow$ elimination:

well known solution, simple and elegant

... but slow and memory hungry

$\rightarrow$ worth implementing a dedicated algorithm
“by definition” -> Linear algebra

Algorithm \textbf{MinPolyQuotDef} \quad P = K[x_1, \ldots, x_n], \text{ term-ordering } \sigma

Input: \( I \) a zero-dimensional ideal in \( P \), \( f \) polynomial in \( P \)

- compute \( GB \), the \( \sigma \)-Gröbner basis for \( I \)
- from \( GB \) compute \( QB \), the monomial quotient basis of \( P/ I \)
- let \( r_0 = f^0 (= 1) \)
- \textbf{Main Loop:} for \( i = 1, 2, \ldots, \text{len}(QB) \) do
  - compute \( r_i = \text{NF}(f^i) \) \([= \text{NF}(f \cdot r_{i-1})]\)
  - if there is a linear dependency \( r_i = \sum_{j=0}^{i-1} c_j r_j \) with coefficients \( c_j \in K \)
    return \( z^i - \sum_{j=0}^{i-1} c_j z^j \)

Output: \( \mu_{f, I}(z) \in K[z] \)

\footnotesize

```*/
QuotientBasis(I); --------- > [1, y, x]
y^0; \quad \rightarrow 1 [1, 0, 0]
NF(y^1, I); \rightarrow y [0, 1, 0]
NF(y^2, I); \rightarrow 6*y -8 [-8, 6, 0]
*/```
**Timings over \( \mathbb{F}_p \)**

MinPolyQuotDef carefully optimized →

<table>
<thead>
<tr>
<th>Example</th>
<th>GB</th>
<th>MinPoly</th>
<th>MinPoly</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Def</td>
<td>Mat</td>
</tr>
<tr>
<td>charp-deg500 ( f_1 )</td>
<td>0.38</td>
<td>4.10</td>
<td>7.06</td>
</tr>
<tr>
<td>charp-deg500 ( f_2 )</td>
<td>0.38</td>
<td>5.77</td>
<td>9.14</td>
</tr>
<tr>
<td>charp-split6</td>
<td>0.00</td>
<td>2.43</td>
<td>12.29</td>
</tr>
<tr>
<td>10000000007-randomp</td>
<td>0.17</td>
<td>4.43</td>
<td>9.02</td>
</tr>
<tr>
<td>23largeCI</td>
<td>0.00</td>
<td>1.06</td>
<td>20.68</td>
</tr>
</tbody>
</table>

**Def**: “by definition”

**Mat**: by multiplication matrix

**Elim**: by elimination

... and with rational coefficients? 😐
Computing Minimal Polynomials

Rational coefficients: modular methods

Definition $\pi_p$: reduction modulo $p$

Let $\delta \in \mathbb{N}_+$ and $p$ a prime not dividing $\delta$.

\[
\begin{align*}
\pi_p &: \mathbb{Z}_\delta &\longrightarrow & \mathbb{F}_p \\
\pi_p &: \mathbb{Z}_\delta[x_1, \ldots, x_n] &\longrightarrow & \mathbb{F}_p[x_1, \ldots, x_n] \quad \sum_t c_t t &\mapsto & \sum_t \pi_p(c_t)t
\end{align*}
\]

But how can we define the reduction modulo $p$ of an ideal?

Theorem (Reduction modulo $p$ of Gröbner Bases)

Let $I$ be a non-zero ideal in $\mathbb{Q}[x_1, \ldots, x_n]$, $GB$ its reduced $\sigma$-Gröbner basis. Let $p$ be any prime not dividing $\text{den}(GB)$.

1. The reduced $\sigma$-Gröbner basis of $\langle \pi_p(GB) \rangle$ is $\pi_p(GB)$

2. $f$ such that $p \nmid \text{den}(f) \longrightarrow$ the NF of $\pi_p(f)$ is $\pi_p(\text{NF}_{\sigma,I}(f))$

$\longrightarrow I_{(p,\sigma)} = \langle \pi_p(G) \rangle$ More in Abbott, Bigatti, Robbiano: “Ideals mod $p$”
Computing Minimal Polynomials

Module methods for minimal polynomials

Let \( I \) be a zero-dimensional ideal, \( f \) a polynomial in \( \mathbb{Q}[x_1, \ldots, x_n] \).

**Proposition**

\[ \delta = \text{den}(f) \cdot \text{den}(\text{GB}_\sigma(I)) \]

then \( \mu_{f,I}(z) \) has all coefficients in \( \mathbb{Z}_\delta \).

**Example 1**

\( P = \mathbb{Q}[x, y] \) and \( I = \langle 2x + 3y, y^2 - 4 \rangle \).

Two possible Gröbner bases: \( \{ x + \frac{3}{2} y, y^2 - 4 \} \) and \( \{ y + \frac{2}{3} x, x^2 - 9 \} \).

\( f = 23x + 17y \) then \( \mu_{f,I}(z) \) has integer coefficients \((= z^2 - 1225)\).

**Theorem (Bad primes)**

1. There are only finitely many bad primes.
2. \( \pi_p(\mu_{f,I}(z)) \) is a multiple of \( \mu_{\pi_p(f), I(p, \sigma)}(z) \).

\[ \rightarrow \text{detect bad primes} \]
### Computing Minimal Polynomials

### Timings over \(\mathbb{Q}\)

Modular computation + CRT + rational reconstruction

<table>
<thead>
<tr>
<th>Example</th>
<th>GB</th>
<th>(\mathbb{Q})</th>
<th>MinPoly</th>
<th>coeff</th>
<th>deg</th>
</tr>
</thead>
<tbody>
<tr>
<td>QQ-rand</td>
<td></td>
<td>time</td>
<td>time verified</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QQ-Cl1 (\ell_1)</td>
<td>0.00</td>
<td>47.86</td>
<td>0.39</td>
<td>12</td>
<td>1093, 100</td>
</tr>
<tr>
<td>QQ-Cl1 (\ell_2)</td>
<td>0.00</td>
<td>226.34</td>
<td>1.31</td>
<td>25</td>
<td>10^{210}, 10^0</td>
</tr>
<tr>
<td>QQ-Cl2</td>
<td>0.00</td>
<td>∞</td>
<td>3.77</td>
<td>38</td>
<td>10^{330}, 10^0</td>
</tr>
<tr>
<td>QQ-split5</td>
<td>0.00</td>
<td>∞</td>
<td>0.67</td>
<td>9</td>
<td>10^{64}, 10^0</td>
</tr>
<tr>
<td>QQ-split6</td>
<td>0.00</td>
<td>∞</td>
<td>175.14</td>
<td>58</td>
<td>10^{503}, 10^0</td>
</tr>
<tr>
<td>QQ-largeCI</td>
<td>0.00</td>
<td>233.24</td>
<td>0.39</td>
<td>5</td>
<td>10^{29}, 10^4</td>
</tr>
<tr>
<td>twomaxhard</td>
<td>0.42</td>
<td>∞</td>
<td>18.12</td>
<td>30</td>
<td>10^{234}, 10^{19}</td>
</tr>
<tr>
<td>twomaxsimple</td>
<td>0.33</td>
<td>5.33</td>
<td>0.67</td>
<td>15</td>
<td>10^{108}, 10^{12}</td>
</tr>
<tr>
<td>PrimaryNotMax</td>
<td>0.00</td>
<td>510.85</td>
<td>3.45</td>
<td>3</td>
<td>10^{11}, 10^0</td>
</tr>
</tbody>
</table>
Using Minimal Polynomials

### Remark

- $\ell \in K[x_1, \ldots, x_n]$ a **generic** linear form, $I$ zero-dimensional ideal
  - $I$ not radical $\implies \mu_{\ell,I}$ not square-free
  - $I$ radical $\implies \deg(\mu_{\ell,I}) = d$

If $K$ is **big enough generic** $\implies$ random

---


---

Some applications
**IsMaximal(\(I\)): practically effective NON-algorithm!**

**Algorithm** IsMaximal

**Input** \(I\), an ideal in \(P\)

**Loop:** repeat
- pick a random linear form \(\ell \in P\); compute \(\mu = \mu_\ell, I\)
- if \(\mu\) is reducible then return false
- if \(\deg(\mu) = d\) then return true

**Output** true/false indicating the maximality of \(I\).

**Remark**

IsMaximal is not an algorithm because termination is not guaranteed.

But in practice recall: if \(\ell\) a random linear form (\(K\) is big enough) then
- \(I\) not radical \(\Rightarrow\) \(\mu_\ell, I\) not square-free
- \(I\) radical \(\Rightarrow\) \(\deg(\mu_\ell, I) = d\)

This is neat and elegant, but better faster IsMaximal →
IsMaximal($I$): a very effective NON-algorithm!

[Algorithm] IsMaximal

Input $I$, an ideal in $P$

1. if $I$ is not zero-dimensional, return false
2. compute $d = \dim_K(P/I)$
3. First Loop: for each indeterminate $x_i$ do
   3.1 compute $\mu = \mu_{x_i, I}$
   3.2 if $\mu$ is reducible then return false
   3.3 if $\deg(\mu) = d$ then return true
4. if $K$ is finite then (..Frobenius space..)
5. Second Loop: repeat
   5.1 pick a random linear form $\ell \in P$; compute $\mu = \mu_{\ell, I}$
   5.2 if $\mu$ is reducible then return false
   5.3 if $\deg(\mu) = d$ then return true

Output true/false indicating the maximality of $I$. 
Algorithm Radical0Dim

Input \( I \), a zero-dimensional ideal in \( P \)

1. let \( J = I \) and compute \( d = \dim_K(P/J) \)
2. Main Loop: for each indeterminate \( x_i \) do
   2.1 compute \( \mu = \mu_{x_i}J \)
   2.2 if \( \mu \) is not square-free then
      2.2.1 let \( \mu = \text{rad}(\mu) \)
      2.2.2 let \( J = J + \langle \mu(x_i) \rangle \)
      2.2.3 compute \( d = \dim_K(P/J) \)
      (if it is worth it \( \rightarrow \) Timeout)
   2.3 if \( \deg(\mu) = d \) then return \( J \)
3. return \( J \)

Output the radical of \( I \)
Many application of minimal polynomials

/ zero-dimensional ideal

- `IsRadical(I)`
- `Radical(I)` (seen)
- `IsMaximal(I)` (seen)
- `IsPrimary(I)`: combination of `IsMaximal` and `Radical`
- `PrimaryDecomposition(I)`: combination of `MinPoly` and `IsPrimary`

and probably most of the applications found in literature which mention Lex Gröbner bases!

Thank you!!
Many application of minimal polynomials

1 zero-dimensional ideal

- `IsRadical(I)`
- `Radical(I)` (seen)
- `IsMaximal(I)` (seen)
- `IsPrimary(I)`: combination of `IsMaximal` and `Radical`
- `PrimaryDecomposition(I)`: combination of `MinPoly` and `IsPrimary`

and probably most of the applications found in literature which mention `Lex Gröbner bases`!

Thank you!!