A Maple Implementation of F4

Roman Pearce, Michael Monagan

CECM / Simon Fraser University

supported by NSERC and MITACS NCE of Canada

March 1st, 2006

Why is Maple so Slow ?!

- whenever you modify an object, the system makes a copy
- $f \leftarrow f - \mathsf{lt}(f)$ makes a copy of f
- $f \leftarrow f - q\,G_i$ makes a copy of f
- conventional wisdom: all these copies mean you are doomed
- my experience: for the Bucberger algorithm you really are doomed

Is this necessarily true in general though ?

Reductions in the Buchberger Algorithm

Example

Divide $x^2y + y^3$ by $G = [x^2 + y, xy^2 - xy, y^3 - 1]$ (graded lex $x > y$)

$$\begin{array}{rcll} x^2y + y^3 & \to & \boxed{x^2y - y\,G_1} + y^3 & = y^3 - y^2 \\ & \to & \boxed{y^3 - G_3} - y^2 & = -y^2 + 1 \end{array}$$

- equivalent to a matrix triangularization

$$\begin{array}{c|cccc} & x^2y & y^3 & y^2 & 1 \\ \hline S_{12} & 1 & 1 & 0 & 0 \\ -yG_1 & 1 & 0 & 1 & 0 \\ -G_3 & 0 & 1 & 0 & \text{-}1 \end{array} \quad \longrightarrow \quad \begin{array}{c|cccc} & x^2y & y^3 & y^2 & 1 \\ \hline & 1 & 1 & 0 & 0 \\ & 0 & 1 & \text{-}1 & 0 \\ & 0 & 0 & 1 & \text{-}1 \end{array}$$

The F4 Algorithm

- put multiple syzygies into one matrix
- marginal cost of each reduction drops by an order of magnitude
- exploit strategies for sparse linear algebra
- modular method: reduce mod p, reconstruct useful rows

The F4 Algorithm

- put multiple syzygies into one matrix
- marginal cost of each reduction drops by an order of magnitude
- exploit strategies for sparse linear algebra
- modular method: reduce mod p, reconstruct useful rows
- **difficult to express Gröbner basis in terms of the generators**
- **in Ore algebras the number of columns blows up**
- **parameters require evaluation/interpolation**

Improved F4

Conversion to Linear System:

$$
\begin{array}{c|cccc}
 & x^2y & y^3 & y^2 & 1 \\
\hline
S_{12} & 1 & 1 & 0 & 0 \\
-yG_1 & 1 & 0 & 1 & 0 \\
-G_3 & 0 & 1 & 0 & -1
\end{array}
\longrightarrow
\begin{array}{c|cccc}
 & x^2y & y^3 & y^2 & 1 \\
\hline
 & 1 & 1 & 0 & 0 \\
 & 0 & 1 & -1 & 0 \\
 & 0 & 0 & \boxed{1} & -1
\end{array}
$$

- row reduce mod p to determine useful columns (ie: y^2) put them on the right hand side and solve

$$
\begin{array}{c|ccc|c}
 & x^2y & y^3 & 1 & y^2 \\
\hline
S_{12} & 1 & 1 & 0 & 0 \\
-yG_1 & 1 & 0 & 0 & 1 \\
-G_3 & 0 & 1 & -1 & 0
\end{array}
\longrightarrow
X = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}
$$

- solution is a linear combination of rows producing a new polynomial

- ie: $\begin{bmatrix} S_{12} & -yG_1 & -G_3 \end{bmatrix} \cdot X$ gives $-y^2 + 1$

Improved F4

Linear System Method:

- allows you to use p-adic lifting
- the solutions are syzygies: can express GB in terms of input
- parameters: evaluation/interpolation on syzygies, not the result
- **you still have to reduce the full matrix mod p**
- **Ore algebras produce still column blowup (in initial matrix)**

Overall this is still a big improvement.

Sparse Strategies for Linear Systems

Structured Gaussian Elimination:

- solve columns with one element, and remove corresponding rows
- declare some columns "heavy", and allow them to fill in
- use rows with one light element to eliminate columns
- extract dense rows (forward substitute and solve at the end)

result: big sparse system $\rightarrow$ small dense system

- use fast dense (modular) method to solve, back substitute solutions

Normal Form Computation

Conversion to Nullspace:

	x^2y	y^3	y^2	1
S_{12}	1	1	0	0
$-yG_1$	1	0	1	0
$-G_3$	0	1	0	-1

$\longrightarrow$

	$-yG_1$	$-G_3$	S_{12}
x^2y	1	0	1
y^3	0	1	1

- write polynomial in RHS vector, GB multiples as LHS columns
- include only monomials reducible by GB
- solution X is a syzygy which cancels all reducible terms
  ie: $S_{12} - (\begin{bmatrix} -yG_1 & -G_3 \end{bmatrix} \cdot X)$ gives $-y^2 + 1$
- matrix obviously much smaller
- efficiently compute normal forms of several polynomials at once

Matrix Size Improvement

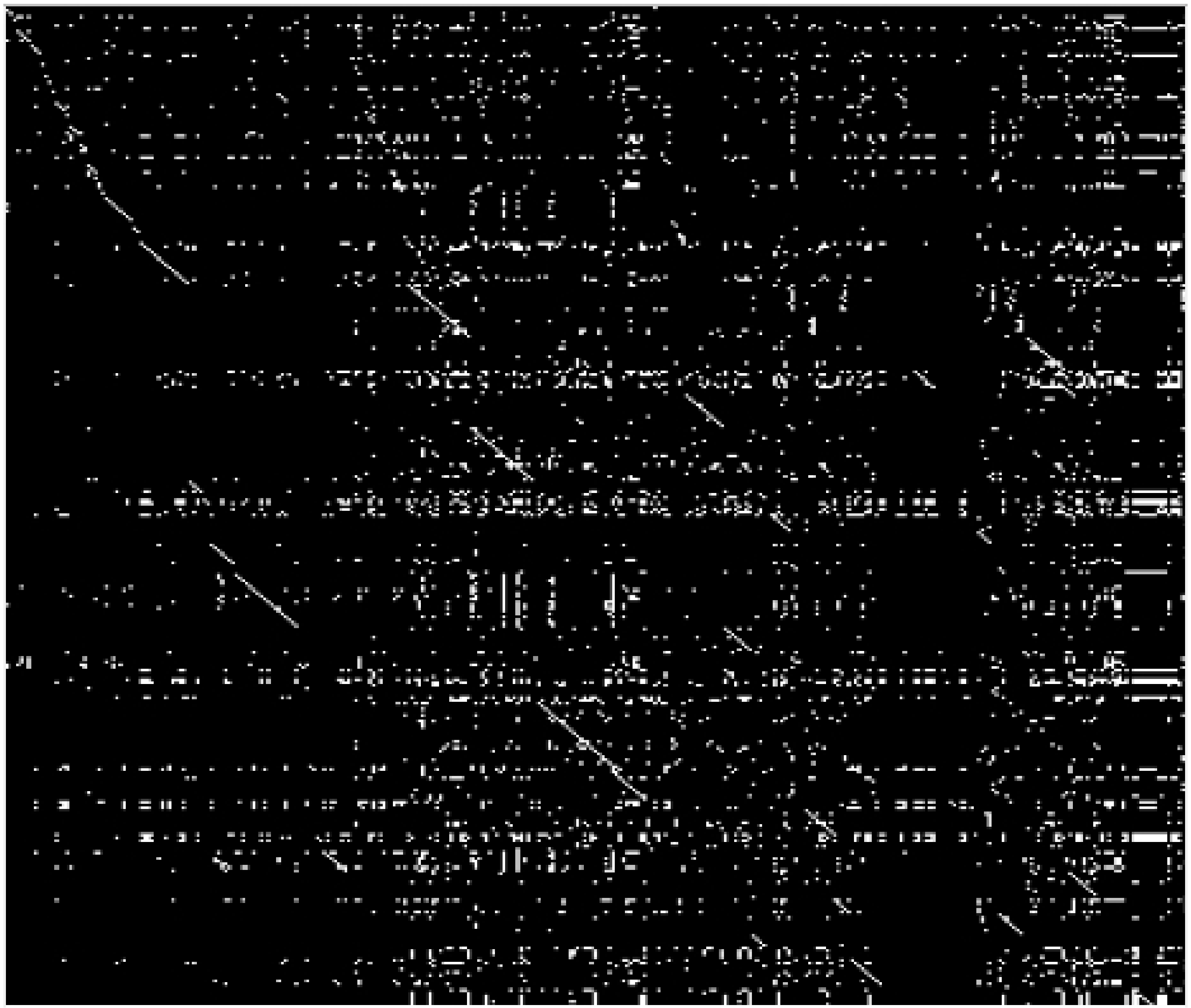
Cyclic-7

Regular F4	Improved F4	Nullspace
11 x 71	11 x 11, 1 rhs	9 x 9, 1 rhs
46 x 159	46 x 46, 2 rhs	41 x 41, 3 rhs
93 x 274	93 x 93, 5 rhs	84 x 84, 7 rhs
208 x 465	208 x 208, 11 rhs	182 x 182, 18 rhs
412 x 729	412 x 412, 21 rhs	360 x 360, 41 rhs
734 x 1074	734 x 734, 41 rhs	628 x 628, 89 rhs
1165 x 1387	1165 x 1165, 52 rhs	963 x 963, 171 rhs
1238 x 1358	1238 x 1238, 62 rhs	950 x 950, 229 rhs
	etc.	

- matrices are smaller, but with more right hand side vectors
- for improved F4 we had to do work (mod p) to choose RHS
- minimal column blowup from Ore algebras

Example Matrix (338 x 338)

15 S-polynomials divided by Gröbner basis of Katsura-6



Block Structure

- systems are **block triangular** (upper and lower blocks)
- no elimination required to solve, similar cost for any field
- some blowup during back substitution (unavoidable)
- very fast for parameters and Ore algebras

Block Structure

- systems are **block triangular** (upper and lower blocks)
- no elimination required to solve, similar cost for any field
- some blowup during back substitution (unavoidable)
- very fast for parameters and Ore algebras

Observe:

- this is a sparse strategy for polynomial division
- it is efficient for reducing many polynomials at once

Question: Can we use it to speed up F4 ?

Application to F4

Procedure:

- select a set of syzygies and reduce them using this algorithm
- add result to current basis
- watch F4 grind to a halt :(

Application to F4

Problem: the normal forms are not inter-reduced

Example: reduce $\{2x^2 + x + 1, x^2 + x\}$ by $\{x^2 + 1\}$

$$\longrightarrow \{-x + 1, x - 1\}$$

Solution:

- put normal forms into rows of a (small) matrix
- do Gaussian elimination (actually, Gauss-Jordan)
- (better) do linear system solving trick, put columns in RHS

How Much Was Gained ?

Rational Numbers:

- modular method required only for very small matrix

- extreme exploitation of sparsity

Parameters:

- evaluation required only for very small matrix

- convert to linear system $\rightarrow$ interpolate only small syzygies

Ore Algebras:

- blowup terms only appear if they are needed for the division

Current Status of the Project

Our implementation is about 70 percent complete.

Done:

- symbolic preprocessing / construct linear system
- efficient solution of block system (all domains)

Not Done:

- efficient inter-reduce (LinearAlgebra:-Modular)

Not Good Enough:

- critical pair handling