Solving Parametric Linear Systems using Sparse Rational Function Interpolation

Ayoola Jinadu and Michael Monagan

Department of Mathematics, Simon Fraser University
Burnaby, British Columbia, V5A 1S6, Canada
{ajinadu, mmonagan}@sfu.ca

Abstract. Let $Ax = b$ be a parametric linear system where the entries of the matrix A and vector b are polynomials in m parameters with integer coefficients and A be of full rank n. The solutions x_i will be rational functions in the parameters. We present a new algorithm for computing x that uses our sparse rational function interpolation which was presented at CASC 2022. It modifies Cuyt and Lee's sparse rational function interpolation algorithm to use a Kronecker substitution on the parameters. A failure probability analysis and complexity analysis for our new algorithm is presented. We have implemented our algorithm in Maple and C. We present timing results comparing our implementation with a Maple implementation of Bareiss/Edmonds/Lipson fraction free Gaussian elimination and three other algorithms in Maple for solving $Ax = b$.

Keywords: Parametric Linear Systems, Sparse Rational Function Interpolation, Kronecker Substitution, Failure Probability, Black Box.

1 Introduction

Consider the parametric linear system $Ax = b$ where the coefficient matrix $A \in \mathbb{Z}[y_1, y_2, \cdots, y_m]^{n\times n}$ is of full rank n and $b \in \mathbb{Z}[y_1, y_2, \cdots, y_m]^n$ is the right hand side column vector such that the number of terms in the entries of A and b denoted by $\#A_{ij}, \#b_i \leq t$ and $\deg(A_{ij}), \deg(b_i) \leq d$. It is well know that the solution x is unique since $\operatorname{rank}(A) = n$. In this paper we aim to compute the solution vector of rational functions

$$x = \begin{bmatrix} x_1 \; x_2 \; \cdots \; x_n \end{bmatrix}^T = \begin{bmatrix} \frac{f_1}{g_1} \; \frac{f_2}{g_2} \cdots \frac{f_n}{g_n} \end{bmatrix}^T \tag{1}$$

such that for $f_k, g_k \in \mathbb{Z}[y_1, y_2, \cdots, y_m]$, $g_k \neq 0$, $g_k | \det(A)$ and $\gcd(f_k, g_k) = 1$ for $1 \leq k \leq n$. Using Cramer's rule, the solutions of $Ax = b$ are given by

$$x_i = \frac{\det(A^i)}{\det(A)} \in \mathbb{Z}(y_1, \cdots, y_m) \tag{2}$$

where A^i is the matrix obtained by replacing the i-th column of A with the right hand side column vector b and $\det(A)$ is a polynomial in $\mathbb{Z}[y_1, y_2, \cdots, y_m]$. Let $\tilde{x}_i = x_i \det(A)$ be a polynomial in $\mathbb{Z}[y_1, y_2, \cdots, y_m]$.

Maple and other computer algebra systems such as Magma have an implementation of the Bareiss/Edmonds one step fraction free Gaussian elimination algorithm [2, 5] which triangularizes an augmented matrix $B = [A|b]$ to obtain $\det(A)$ as a polynomial in $\mathbb{Z}[y_1, y_2, \cdots, y_m]$ and then solves for the polynomials $\tilde{x}_i$ via back substitution using Lipson's fraction free back formula [8]. Ignoring pivoting, the following pseudo-code of the Bareiss/Edmonds algorithm and Lipson's fraction free back substitution formula solves $Ax = b$:

$B := [A|b\,]$; $B_{0,0} := 1$;
// fraction free triangularization begins
for $k = 1, 2, \cdots, n-1$ **do**
  for $i = k+1, k+2, \cdots, n$ **do**
    for $j = k+1, k+2, \cdots, n+1$ **do**

$$B_{i,j} := (B_{k,k}B_{i,j} - B_{i,k}B_{k,j}) \text{ quo } B_{k-1,k-1} \tag{3}$$

    end do
    $B_{i,k} := 0$;
  end do
end do
// fraction free back substitution begins
$\tilde{x}_n := B[n, n+1]$;
for $i = n-1, n-2, \cdots, 2, 1$ **do**
  $N_i := B_{i,n+1}B_{n,n} - \sum_{j=i+1}^{n} B_{i,j}\tilde{x}_j$;
  $D_i := B_{i,i}$;

$$\tilde{x}_i := N_i \text{ quo } D_i; \tag{4}$$

end do
// simplification begins
for $i = 1, 2, \cdots, n$ **do**
  $h_i = \gcd(\tilde{x}_i, B_{n,n})$;
  $f_i := \tilde{x}_i$ quo h_i; $g_i := B_{n,n}$ quo h_i;
  $x_i := \dfrac{f_i}{g_i}$;
end do

Note that the divisions indicated by the quotient operator `quo` are exact in $\mathbb{Z}[y_1, y_2, \cdots, y_m]$ and $B_{k,k}$ is the determinant of the principle k by k submatrix of A. However there is an expression swell because at the last major step of triangularizing B when $k = n-1$ where it computes

$$B_{n,n} = \frac{B_{n-1,n-1}B_{n,n} - B_{n,n-1}B_{n-1,n}}{B_{n-2,n-2}} = \det(A), \tag{5}$$

the numerator polynomial in (5) is the product of determinants

$$B_{n,n}B_{n-2,n-2} \in \mathbb{Z}[y_1, y_2, \cdots, y_m]. \tag{6}$$

If the original entries $B_{i,j}$ from B are sparse polynomials in many parameters then the numerator polynomial in (5) may be 100 times or more larger than $\det(A)$. The same situation also holds for the polynomials $\tilde{x}_i$.

One approach to avoid this expression swell tried by Monagan and Vrbik [15] computes the quotients of (3) and (4) directly using lazy polynomial arithmetic. Another approach is to interpolate the polynomials $\tilde{x}_i$ and $\det(A)$ directly from points using sparse polynomial interpolation algorithms [3, 17] and Chinese remaindering when needed. This approach is described briefly as follows. Pick an evaluation point $\alpha \in \mathbb{Z}_p^m$ and solve $A(\alpha)x(\alpha) = b(\alpha) \mod p$ for $\tilde{x}(\alpha)$ using Gaussian elimination over $\mathbb{Z}_p$ and also compute $\det(A(\alpha))$ at the same time. Then $\tilde{x}_i(\alpha)$ is given by $x_i(\alpha) \times \det(A(\alpha))$. Thus we have images of $\tilde{x}_i$ and $\det(A)$ so we can interpolate them.

To compute the solution vector x in simplest terms that we compute the $h_i = \gcd(\tilde{x}_i, \det(A))$ for $1 \le i \le n$ and cancel them from $\frac{\tilde{x}_i}{\det(A)}$ to simplify the solutions. However, in practice there may be a large cancellation in $\frac{\tilde{x}_i}{\det(A)}$. That is, h_i may be a large factor so that the final solution $x_i = \frac{\tilde{x}_i/h_i}{\det(A)/h_i}$ may be small. Our new algorithm will interpolate x_i directly thus avoiding any gcd computations which may be expensive.

Example 1 *Consider the following linear system of* 21 *equations in variables* $x_1, x_2, \cdots, x_{21}$ *and parameters* $y_1, y_2, \cdots, y_5$

$$
\begin{aligned}
&x_7 + x_{12} = 1,\ x_8 + x_{13} = 1,\ x_{21} + x_6 + x_{11} = 1,\ x_1y_1 + x_1 - x_2 = 0\\
&x_3y_2 + x_3 - x_4 = 0,\ x_{11}y_3 + x_{11} - x_{12} = 0,\ x_{16}y_5 - x_{17}y_5 - x_{17} = 0\\
&y_3(-x_{20} + x_{21}) + x_{21} = 0,\ y_3(-x_5 + x_6) + x_6 - x_7 = 0,\ -x_8y_4 + x_9y_3 + x_9 = 0\\
&y_2(-x_{10} + x_{18}) + x_{18} - x_{19} = 0,\ y_4(x_{14} - x_{13}) + x_{14} - x_{15} = 0\\
&2x_3(y_2^2 - 1) + 4x_4 - 2x_5 = 0,\ 2y_1^2(x_1 - 1) - 2x_{10} + 4x_2 = 0\\
&2y_3^2(x_{19} - 2x_{20} + x_{21}) - 2x_{21} = 0,\ 2y_4^2(x_7 - 2x_8 + x_9) - 2x_9 = 0\\
&2x_{11}(y_3^2 - 1) + 4x_{12} - 2x_{13} = 0,\ 2y_4^2(x_{12} - 2x_{13} + x_{14}) - 2x_{14} + 4x_{15} - 2x_{16} = 0\\
&2y_3^2(x_4 - 2x_5 + x_6) - 2x_6 + 4x_7 - 2x_8 = 0,\ 2y_5^2(x_{15} - 2x_{16} + x_{17}) - 2x_{17} = 0\\
&2y_2^2(-2x_{10} - x_{18} - x_2) - 2x_{18} + 4x_{19} - 2x_{20} = 0
\end{aligned}
$$

where the solution of the above system defines a general cubic Beta-Splines in the study of modelling curves in Computer Graphics.

Using the Bareiss/Edmonds/Lipson algorithm on page 2, we find that $\#B[n,n] = \det(A) = 1033$, $\#B[n-2,n-2] = 672$ and $\#B[n,n] \times B[n-2,n-2] = 14348$, so an expression swell factor of $14348/1033 = 14$. Furthermore, we obtain $\#\tilde{x}_i, \#x_i$ and the expression swell factor labelled **swell** for computing $\tilde{x}_i$ in Table 1.

The Gentleman & Johnson minor expansion algorithm [7] can also be used to compute the solutions x_i by computing $n+1$ determinants, namely, the numerators $\det(A^i)$ for $1 \le i \le n$ (A^i is as defined in (2)) and the denominator $\det(A)$ only once. But then we still have to compute $g_i = \gcd(\det(A^i), \det(A))$ to simplify the solutions x_i which is not cheap.

	1	2	3	4	5	6	7	8	9	10	11
$\#N_i$	586	1,172	1,197	1,827	2,142	1,666	2,072	1,320	1,320	2,650	2,543
$\#D_i$	2	3	6	9	9	9	9	9	18	18	27
$\#\tilde{x}_i$	293	586	504	693	882	686	840	536	424	879	638
swell	2	2	3	3	3	3	3	3	3	3	4
$\#\ f_i$	1	2	4	4	4	19	16	8	8	8	2
$\#\ g_i$	5	3	10	7	4	22	16	16	26	12	3

	12	13	14	15	16	17	18	19	20	21
$\#N_i$	3,490	3,971	5,675	7,410	4,940	7,072	11,793	12,802	11,211	9,620
$\#D_i$	36	36	117	153	153	432	672	672	672	672
$\#\tilde{x}_i$	834	1,033	871	1044	696	348	690	836	693	528
swell	4	4	7	7	7	20	17	15	16	18
$\#\ f_i$	1	1	1	1	1	2	14	4	1	1
$\#\ g_i$	3	3	5	5	3	3	23	7	4	7

Table 1: Number of terms in $\tilde{x}_i$ and x_i and expression swell factor for computing $\tilde{x}_i$

In this work, we interpolate the simplified solutions $x_i = f_i/g_i$ directly using sparse rational function interpolation. We use a black box representation to denote any given parametric linear system. That is, a black box **BB** representing $Ax = b$ denoted by $\mathbf{BB}:\ \mathbb{Z}_p^m \to \mathbb{Z}_p^n$ is a computer program that takes a prime p and an evaluation point $\alpha \in \mathbb{Z}_p^m$ as inputs and outputs $x(\alpha) = A^{-1}(\alpha)b(\alpha) \in \mathbb{Z}_p^n$. The implication of the black box representation of $Ax = b$ is that important properties of x such as $\#f_k, \#g_k$ and their variable degrees are unknown so we have to find them by interpolation.

Our first contribution is a new algorithm that probes a given black box **BB** and uses sparse multivariate rational function interpolation to interpolate the rational function entries in x modulo primes and then uses Chinese remaindering and rational number reconstruction to recover its integer coefficients.

Our algorithm for solving $Ax = b$ follows the work of Jinadu and Monagan in [10] where they modified Cuyt and Lee's sparse rational function interpolation algorithm to use the Ben-Or/Twari interpolation algorithm and Kronecker substitution on the parameters in order to solve parametric polynomial systems by computing its Dixon resultant.

Our second contribution is a hybrid Maple + C implementation of our algorithm for solving parametric linear systems and it can be downloaded for use from the web at:

http://www.cecm.sfu.ca/personal/monaganm/code/ParamLinSolve/.

Our third contribution is the failure probability analysis and complexity analysis of our algorithm in terms of number of black box probes required. The analysis in this paper follows [12].

This paper is organized as follows. In section 2, we review the sparse multivariate rational function algorithm of Cuyt and Lee and we describe how it should be modified with the use of a Kronecker substitution on the parameters. Our algorithms are presented in Section 3. Section 4 contains the failure probability analysis and complexity analysis of our algorithm. In section 5, we present timing results comparing a hybrid Maple+C implementation of our algorithm with a Maple implementation of the Bareiss/EdmondsLipson fraction free Gaussian elimination algorithm with three other algorithms for solving $Ax = b$.

2 Sparse Multivariate Rational Function Interpolation

2.1 Cuyt and Lee's algorithm

Let $\mathbb{K}$ be a field and let $f/g \in \mathbb{K}(y_1, \cdots, y_m)$ be a rational function such that $\gcd(f, g) = 1$. Cuyt and Lee's algorithm [4] to interpolate f/g must be combined with a sparse polynomial interpolation to interpolate f and g.

The first step in their algorithm is to introduce a homogenizing variable z to form the auxiliary rational function

$$\frac{f(y_1z, \cdots, y_mz)}{g(y_1z, \cdots, y_mz)} = \frac{f_0 + f_1(y_1, \cdots, y_m)z + \cdots + f_{\deg(f)}(y_1, \cdots, y_m)z^{\deg(f)}}{g_0 + g_1(y_1, \cdots, y_m)z + \cdots + g_{\deg(g)}(y_1, \cdots, y_m)z^{\deg(g)}}$$

and then normalize it using either constant terms $f_0 \neq 0$ or $g_0 \neq 0$. However it is not uncommon to have $f_0 = g_0 = 0$. Thus in the case when both constant terms g_0 and f_0 are zero, one has to pick a basis shift $\beta \in (\mathbb{K} \setminus \{0\})^m$ and form the auxiliary rational function as

$$\frac{\hat{f}(y_1z, \cdots, y_mz)}{\hat{g}(y_1z, \cdots, y_mz)} := \frac{f(y_1z + \beta_1, \cdots, y_mz + \beta_m)}{g(y_1z + \beta_1, \cdots, y_mz + \beta_m)} = \frac{\sum_{j=0}^{\deg(f)} \hat{f}_j(y_1, \cdots, y_m)z^j}{\sum_{j=0}^{\deg(g)} \hat{g}_j(y_1, \cdots, y_m)z^j}.$$

The introduction of the basis shift β forces the production of a constant term in $\hat{f}/\hat{g}$ so that we can normalize it using either $\hat{f}_0$ or $\hat{g}_0$. Thus we can write

$$\frac{\hat{f}(y_1z, \cdots, y_mz)}{\hat{g}(y_1z, \cdots, y_mz)} = \frac{\sum_{j=0}^{\deg(f)} \frac{\hat{f}_j(y_1, \cdots, y_m)z^j}{\hat{g}_0}}{1 + \sum_{j=1}^{\deg(g)} \frac{\hat{g}_j(y_1, \cdots, y_m)z^j}{\hat{g}_0}}.$$

Note that $\hat{g}_0 = \tilde{c} \times g(\beta_1, \beta_2, \cdots, \beta_m) \neq 0$ for some $\tilde{c} \in \mathbb{K}$. If a rational function f/g is represented by a a black box, we can recover it by densely interpolating univariate auxiliary rational functions

$$\hat{A}(\alpha^j, z) = \frac{\frac{f_0}{g_0} + \frac{f_1(\alpha^j)}{g_0}z + \cdots + \frac{f_{\deg(f)}(\alpha)}{g_0}z^{\deg(f)}}{1 + \frac{g_1(\alpha^j)}{g_0}z + \cdots + \frac{g_{\deg(g)}(\alpha^j)}{g_0}z^{\deg(g)}} \in \mathbb{Z}_p(z) \text{ for } j = 0, 1, 2, \cdots$$

for $\alpha \in \mathbb{Z}_p^m$ from the black box and then use the coefficients of $\hat{A}(\alpha^j, z)$ via sparse interpolation to recover f/g. In order to densely interpolate $\hat{A}(\alpha^j, z)$, we use the Maximal Quotient Rational Function Reconstruction algorithm (MQRFR) [14] which requires $\deg(f) + \deg(g) + 2$ black box probes on z.

Note that the use of a basis shift in the formation of the auxiliary rational function destroys the sparsity of f/g, so its effect has to be removed before f/g

can be recovered. Cuyt and Lee remove the effect of this basis shift by adjusting the coefficients of the lower degree terms in the numerator and denominator of $\hat{A}(\alpha^j, z)$ by the contributions from the higher degree terms before the sparse interpolation step is performed. We show how to do this in Subroutine 6.

2.2 Using a Kronecker Substitution on the parameters

In this work, the Ben-Or/Tiwari algorithm is the preferred sparse polynomial algorithm for the Cuyt and Lee's algorithm because it requires the fewest number of black box probes. However, in order to interpolate a polynomial $f \neq 0$ using the Ben-Or/Twari interpolation algorithm over $\mathbb{Z}_p$, the working prime p is required to be at least p_n^d where p_n is the m-th prime and $d = \deg(f)$. Unfortunately, such a prime p may be too large for machine arithmetic if the number of parameters m or the total degree d is large. This is the main drawback of using the BenOr/Tiwari algorithm. Here we review the idea of Jinadu and Monagan from [10] where they formulated how to use a Kronecker substitution to combat the large prime problem posed by using the Ben-Or/Twari algorithm in Cuyt and Lee's method.

Definition 2 *Let $\mathbb{K}$ be an integral domain and let $f/g \in \mathbb{K}(y_1, \cdots, y_m)$. Let $r = (r_1, r_2, \cdots, r_{m-1}) \in \mathbb{Z}^{m-1}$ with $r_i > 0$. Let $K_r : \mathbb{K}(y_1, \cdots, y_m) \to \mathbb{K}(y)$ be the Kronecker substitution*

$$K_r(f/g) = \frac{f(y, y^{r_1}, y^{r_1 r_2}, \cdots, y^{r_1 r_2 \cdots r_{m-1}})}{g(y, y^{r_1}, y^{r_1 r_2}, \cdots, y^{r_1 r_2 \cdots r_{m-1}})} \in \mathbb{K}(y).$$

Let $d_i = \max\{(\deg f, y_i), \deg(g, y_i)\}$ for $1 \leq i \leq m$. Provided we choose $r_i > d_i$ for $1 \leq i \leq m-1$ then K_r is invertible, $g \neq 0$ and $K_r(f/g) = 0 \iff f = 0$.

Unfortunately, we cannot use the original definition of auxiliary rational function given by Cuyt and Lee that we reviewed in Subsection 2.1 to interpolate the univariate mapped function $K_r(f/g)$. Thus we need a new definition for how to compute the corresponding auxiliary rational function relative to the mapped univariate function $K_r(f/g)$, and not the original function f/g itself. Thus using a homogenizing variable z we define auxiliary rational function

$$F(y, z) = \frac{f(zy, zy^{r_1}, \cdots, zy^{r_1 r_2 \cdots r_{m-1}})}{g(zy, zy^{r_1}, \cdots, zy^{r_1 r_2 \cdots r_{m-1}})} \in \mathbb{K}[y](z). \tag{7}$$

As before, the existence of a constant term in the denominator of $F(y, z)$ must be guaranteed, so we use a basis shift $\beta \in (\mathbb{K} \setminus \{0\})^m$ and instead formally define an auxiliary rational function with Kronecker substitution as follows.

Definition 3 *Let $\mathbb{K}$ be a field and let $f/g \in \mathbb{K}(y_1, \cdots, y_m)$ such that $\gcd(f, g) = 1$. Let z be the homogenizing variable and let $r = (r_1, \cdots, r_{m-1})$ with $r_i > d_i = \max\{(\deg f, y_i), \deg(g, y_i)\}$. Let K_r be the Kronecker substitution. We define*

$$F(y, z, \beta) := \frac{f^\beta(y, z)}{g^\beta(y, z)} = \frac{f(zy + \beta_1, zy^{r_1} + \beta_2, \cdots, zy^{r_1 r_2 \cdots r_{m-1}} + \beta_m)}{g(zy + \beta_1, zy^{r_1} + \beta_2, \cdots, zy^{r_1 r_2 \cdots r_{m-1}} + \beta_m)} \in \mathbb{K}[y](z)$$

as an auxiliary rational function with Kronecker substitution K_r.

Notice in the above definition that

$$F(y,1,0) = \frac{f_k^0(y,1)}{g_k^0(y,1)} = K_r(f/g).$$

Thus $K_r(f_k/g_k)$ can be recovered using the coefficients of $F(\alpha^i, z, \beta)$ for some evaluation point $\alpha \in \mathbb{Z}_p^m$ and $i \geq 0$. If g has a constant term, then one can use $\beta = (0, \cdots, 0)$. Also observe that $\deg(K_r(f/g))$ is exponential in y but $\deg(F(y,z,\beta), z)$ through which $K_r(f/g)$ is interpolated remains the same and the number of terms and the number of probes needed to interpolate f/g are the same. To recover the exponents in y we require our input prime $p > \prod_{i=1}^m r_i$.

3 The Algorithm

Let the polynomials f_k and g_k of the entries $x_k = \frac{f_k}{g_k}$ be viewed as

$$f_k = \sum_{i=0}^{\deg(f)} f_{i,k}(y_1, y_2, \cdots, y_m) \text{ and } g_k = \sum_{i=0}^{\deg(g)} g_{i,k}(y_1, y_2, \cdots, y_m) \tag{8}$$

such that $\deg(f_{i,k}) = i$ and $\deg(g_{i,k}) = i$. Given a black box **BB** representing $Ax = b$, we divide the steps to recover x by our algorithm (Algorithm 1) into six main steps.

The first step in our algorithm is to obtain the degrees needed to interpolate x. These include the total degrees $\deg(f_k), \deg(g_k)$ for $1 \leq k \leq n$, which are needed to densely interpolate the univariate auxiliary rational functions, the maximum partial degrees $\max\left(\max_{k=1}^n(\deg(f_k, y_i), \deg(g_k, y_i))\right)$ for $1 \leq i \leq m$, which are needed to apply Kronecker substitution and the total degrees of the polynomials $f_{i,k}$ and $g_{i,k}$ which helps avoid doing unnecessary work when the effect of the basis shift is removed in Subroutine 6 (See Lines 1-5 of Algorithm 1). With high probability, we describe how to discover these degrees as follows.

Let p be a large prime. First, pick $\alpha, \beta \in (\mathbb{Z}_p \setminus \{0\})^m$ at random, and use enough distinct points for z selected at random from $\mathbb{Z}_p \setminus \{0\}$ to compute

$$h_k(z) = \frac{N_k(z)}{D_k(z)} = \frac{f_k(\alpha_1 z + \beta_1, \cdots, \alpha_m z + \beta_m)}{g_k(\alpha_1 z + \beta_1, \cdots, \alpha_m z + \beta_m)} \in \mathbb{Z}_p(z),$$

so that $\deg(f_k) = \deg(N_k)$ and $\deg(g_k) - \deg(D_k)$ with high probability.

Next, pick $\alpha \in (\mathbb{Z}_p \setminus \{0\})^{m-1}$, $\beta, \theta \in \mathbb{Z}_p$ at random and compute

$$H_i(z) := \frac{H_{f_i}}{H_{g_i}} = \frac{f_k(\alpha_1, \cdots, \alpha_{i-1}, \theta z + \beta, \alpha_{i+1}, \cdots, \alpha_n)}{g_k(\alpha_1, \cdots, \alpha_{i-1}, \theta z + \beta, \alpha_{i+1}, \cdots, \alpha_m)} \in \mathbb{Z}_p(z)$$

using enough distinct random points for z from $\mathbb{Z}_p \setminus \{0\}$. With high probability $\deg(f_k, y_i) = \deg(H_{f_i}, z)$ and $\deg(g_k, y_i) = \deg(H_{g_i}, z)$ for $1 \leq i \leq m$.

Finally, suppose we have obtained $\deg(f_k), \deg(g_k)$ correctly for $1 \le k \le n$. Then pick $\alpha \in (\mathbb{Z}_p \setminus \{0\})^m$ at random and use enough random distinct points for z selected from $\mathbb{Z}_p \setminus \{0\}$ to compute the univariate rational function

$$W_k(z) = \frac{\overline{N}_k}{\overline{D}_k} = \frac{\sum_{j=0}^{d_{f_k}} \bar{N}_{i,k}(z)}{\sum_{i=0}^{d_{g_k}} \bar{D}_{i,k}(z)} = \frac{f_k(\alpha_1 z, \cdots, \alpha_m z)}{g_k(\alpha_1 z, \cdots, \alpha_m z)}.$$

Now if $\deg(f_k) = d_{f_k}$ and $\deg(g_k) = d_{g_k}$ then $\deg(f_{i,k}) = \deg(\bar{N}_{i,k})$ and $\deg(g_{i,k}) = \deg(\bar{D}_{i,k})$ with high probability. But, if there is no constant term in f_k and g_k then $\deg(f_k) \neq d_{f_k}$ and $\deg(g_k) \neq d_{g_k}$ because $e_k = \deg(\gcd(\overline{N}_k, \overline{D}_k)) > 0$. Since we do not know what e_k is, then it follows that if $e_k = \deg(f_k) - d_{f_k} = \deg(g_k) - d_{g_k}$ with high probability then $\deg(f_{i,k}) = \deg(\bar{N}_{j,k}) + e_k$ and $\deg(g_{i,k}) = \deg(\bar{D}_{i,k}) + e_k$ with high probability.

Example 4 *Let* $\dfrac{f_1}{g_1} = \dfrac{y_1^3 + y_1 y_2}{y_2^2 + y_3}$ *where* $f_{3,1} = y_1^3,\ f_{2,1} = y_1 y_2,\ g_{2,1} = y_2^2$ *and* $g_{1,1} = 1$. *Then* $W_1(z) = \dfrac{f_1(\alpha_1 z, \alpha_2 z, \alpha_3 z)}{g_1(\alpha_1 z, \alpha_2 z, \alpha_3 z)} = \dfrac{\alpha_1^3 z^2 + \alpha_2 z}{\alpha_2 z + \alpha_3}$. *Thus* $\deg(f_1) = 3 \neq d_{f_1} = 2$ *and* $\deg(g_1) = 2 \neq d_{g_1} = 1$. *Hence* $\deg(f_{3,1}) = 2+3-2 = 3, \deg(f_{2,1}) = 1+3-2 = 2, \deg(g_{2,1}) = 1+2-1 = 2, \deg(g_{2,1}) = 0+2-1 = 2.$

The second step in our algorithm is to probe the black box **BB** with $\alpha \in \mathbb{Z}_p^m$ as input evaluation point to obtain $x(\alpha) = A^{-1}(\alpha)b(\alpha) \in \mathbb{Z}_p^n$ (See Line 17-18). The third step is to perform dense interpolation of auxiliary univariate rational functions using the images $x(\alpha) = A^{-1}(\alpha)b(\alpha) \in \mathbb{Z}_p^n$ (See Lines 23-25). By design, the fourth step is to determine the number of terms in the leading term polynomials $f_{\deg(f_k),k}$ and $g_{\deg(f_k),k}$ and interpolate them via calls to Subroutine BMStep in Lines 29-30. Next $\#f_{i,k}$ and $\#g_{i,k}$ as defined in (8) are determined by calls Subroutine RemoveShift in Lines 33-34 where the effect of the basis shift $\beta \neq 0$ is removed and the coefficients of the auxiliary rational functions in variable are adjusted in order to interpolate $f_{i,k}$ and $g_{i,k}$

Note that for each $i, \#f_{i,k}$ (or $\#g_{i,k}$) is obtained when $\deg(\lambda, z) < \frac{\#f_{i,k}}{2}$ for some feedback polynomial $\lambda \in \mathbb{Z}_p[z]$ produced by the Berlekamp-Massey algorithm in Subroutine BMStep. Once $f_{i,k}, g_{i,k}$ modulo a prime have been interpolated, the sixth step in our algorithm is to apply rational number reconstruction (RNR) on the assembled vector $\overline{X} = [\frac{f_k}{g_k} \mod p, 1 \le k \le n]$ to get x in Line 41. If RNR process fails then more primes and images of x are needed to interpolate x. The final step is to call Algorithm 2, a similar to Algorithm 1, except that $\#f_{i,k}$ and $g_{i,k}$ are now known, and it uses Chinese remaindering to get the solution x.

Algorithm 1: ParamLinSolve

Input: A black box $\mathbf{BB} : \mathbb{Z}_p^m \to \mathbb{Z}_p^n$ with $m \geq 1$.

Output: Vectors $x \in \mathbb{Z}(y_1, \cdots, y_m)^n$ or **FAIL**.

Compute total degrees $(\deg(f_k), \deg(g_k))$ for $1 \leq k \leq n$

$e_k \leftarrow \deg(f_k) + \deg(g_k) + 2$.

$e_{\max} \leftarrow \max_{k=1}^{n} \{e_k\}$

Compute (E_{f_k}, E_{g_k}) where E_{f_k} and E_{g_k} denote the lists of the total degrees of the polynomials f_{ik} and g_{ik} in f_k and g_k respectively as defined in (8)

$D_{y_i} \leftarrow \max\left(\max_{k=1}^{n}(\deg(f_k, y_i), \deg(g_k, y_i))\right)$ for $1 \leq i \leq m$.

Initialize $r_i = D_{y_i} + 1$ for $1 \leq i \leq m$ and let $r = (r_1, r_2, \cdots, r_{m-1})$.

Pick a prime p such that $p > \prod_{j=1}^{m} r_i$ and a basis shift $\beta \neq 0 \in \mathbb{Z}_p^m$ at random.

Let $K_r : \mathbb{Z}_p(y_1, y_2, \cdots, y_m) \to \mathbb{Z}_p(y)$ be the Kronecker substitution $K_r(f_k/g_k)$

Pick a random shift $\hat{s} \in [1, p-2]$ and any generator α for $\mathbb{Z}_p^*$.

Let z be the homogenizing variable

Pick $\theta \in \mathbb{Z}_p^{e_{\max}}$ at random with $\theta_i \neq \theta_j$ for $i \neq j$.

$M \leftarrow \prod_{i=1}^{e_{\max}} (z - \theta_i) \in \mathbb{Z}_p[z]$; .... $O(e_{\max}^2)$

$k \leftarrow 1$

for $i = 1, 2, \cdots$ **while** $k \leq n$ **do**

  $\hat{Y}_i \leftarrow (\alpha^{\hat{s}+i-1}, \alpha^{(\hat{s}+i-1)r_1}, \cdots \alpha^{(\hat{s}+i-1)(r_1 r_2 \cdots r_{m-1})})$.

  for $j = 1, 2, \ldots, e_{max}$ **do**

    $Z_j \leftarrow \hat{Y}_i \theta_j + \beta \in \mathbb{Z}_p^m$

    $v_j \leftarrow \mathbf{BB}(Z_j)$ // Here $v_j = A^{-1}(Z_j) b(Z_j) \in \mathbb{Z}_p^n$

    if $v_j =$ **FAIL then** return **FAIL end** // $\mathrm{rank}(A(Z_j)) < n$.

  end

  if $i \notin \{2, 4, 8, 16, 32, \cdots\}$ **then** next **end**

  for $j = 1, 2, \ldots, i$ **do**

    Interpolate $U \in \mathbb{Z}_p[z]$ using points $(\theta_i, v_{kj} : 1 \leq j \leq e_k)$; .... $O(e_k^2)$

    $A_j(z) \leftarrow \mathrm{MQRFR}(M, U, p)$; .... $O(e_k^2)$

    Let $A_j(z) = \frac{N_j(z)}{\hat{N}_j(z)} \in \mathbb{Z}_p(z)$ // These are the auxiliary functions in z.

    if $\deg(N_j) \neq \deg(f_k)$ or $\deg(\hat{N}_j) \neq \deg(g_k)$ return **FAIL end**

    Normalize $A_j(z)$ such that $\hat{N}_j(z) = 1 + \sum_{i=1}^{\deg(\hat{N})} a_i z^i$.

  end

  $F_k \leftarrow \mathrm{BMStep}([\mathrm{coeff}(N_j, z^{\deg(f_k)}) : 1 \leq j \leq i], \alpha, \hat{s}, r)$; $O(i^2 + \#F_k^2 \log p)$

  $G_k \leftarrow \mathrm{BMStep}([\mathrm{coeff}(\hat{N}_j, z^{\deg(g_k)}) : 1 \leq j \leq i], \alpha, \hat{s}, r)$; $O(i^2 + \#G_k^2 \log p)$

  // Here $F_k = f_{\deg(f_k),k} \mod p$ and $G_k = g_{\deg(g_k),k} \mod p$

  if $F_k \neq$ **FAIL** *and* $G_k \neq$ **FAIL then**

    $f_k \leftarrow \mathrm{RemoveShift}(F_k, [\hat{Y}_1, \cdots, \hat{Y}_i], [N_1, \cdots, N_i], \alpha, \hat{s}, \beta, r, E_{f_k})$

    $g_k \leftarrow \mathrm{RemoveShift}(G_k, [\hat{Y}_1, \cdots, \hat{Y}_i], [\hat{N}_1, \cdots, \hat{N}_i], \alpha, \hat{s}, \beta, r, E_{g_k})$

    if $f_k \neq$ **FAIL** *and* $g_k \neq$ **FAIL then**

      $k \leftarrow k + 1$ // we have interpolated $x_k \mod p$

    end

  end

end

$\overline{X} \leftarrow [\frac{f_k}{g_k}, 1 \leq k \leq n]$ // Here $\overline{X} = x \mod p$

Apply rational number reconstruction on the coefficients of $\overline{X} \mod p$ to get x

if $x \neq$ FAIL and $x \mod p = \overline{X}$ **then** return x **end**

return $\mathrm{MorePrimes}(\mathbf{BB}, \overline{X}, ((\deg(f_k), \deg(g_k)) : 1 \leq k \leq n))$

Algorithm 2: MorePrimes

Input: Black box $\mathbf{BB} : \mathbb{Z}_q^m \rightarrow \mathbb{Z}_q^n$ with $m \geq 1$.
Input: Degrees $\{(\deg(f_k), \deg(g_k)) : 1 \leq k \leq n\}$
Output: Vectors $x \in \mathbb{Z}(y_1, \cdots, y_m)^n$ or **FAIL**.

Let $e_k = \deg(f_k) + \deg(g_k) + 2$ for $1 \leq k \leq n$ and let $e_{\max} = \max e_k$.
Let $B_1 = \{f_{\deg(f_k)-1,k}, \cdots, f_{0,k}\}$ and $B_2 = \{g_{\deg(g_k)-1,k}, \cdots, g_{0,k}\}$ where $f_{i,k}, g_{i,k}$ are as in (8) and
$P \leftarrow p$.
Let $N_{\max} = \max_{k=1}^{n} \left\{ \max_{i=0}^{\deg(f_k)} \{\#f_{i,k}\}, \max_{i=0}^{\deg(g_k)} \{\#g_{i,k}\} \right\}$.
do
    Get a new 62 bit prime $q > p$.
    Pick $\alpha, \beta \in (\mathbb{Z}_q \setminus \{0\})^m, \theta \in \mathbb{Z}_q^{e_{\max}}$ and shift $\hat{s} \in [1, q-2]$ at random.
    for $i = 1, 2, \cdots, N_{\max}$ **do**
        $\hat{Y}_i \leftarrow (\alpha_1^{\hat{s}+i-1}, \alpha_2^{\hat{s}+i-1} \cdots, \alpha_m^{\hat{s}+i-1})$.
        for $j = 1, 2, \ldots, e_{max}$ **do**
            $Z_j \leftarrow \hat{Y}_i \theta_j + \beta \in \mathbb{Z}_p^m$
            $v_j \leftarrow \mathbf{BB}(Z_j)$ // Here $v_j = A^{-1}(Z_j) b(Z_j) \in \mathbb{Z}_p^n$
            if $v_j =$ **FAIL then** return **FAIL end** // $\mathrm{rank}(A(Z_j)) < n$).
        end
    end
    for $k = 1, 2, \cdots, n$ **do**
        $(\hat{n}, \hat{M}) \leftarrow (\#f_{\deg(f_k),k}, \mathrm{supp}(f_{\deg(f_k),k}))$ // supp means support.
        $(\bar{n}, \bar{M}) \leftarrow (\#g_{\deg(g_k),k}, \mathrm{supp}(g_{\deg(g_k),k}))$
        $(\hat{m}, \bar{m}) \leftarrow ([\hat{M}_i(\alpha) : 1 \leq i \leq \hat{n}], [\bar{M}_i(\alpha) : 1 \leq i \leq \bar{n}])$; ..... $O(m(\hat{n} + \bar{n}))$
        if the evaluations $\hat{m}_i = \hat{m}_j$ or $\bar{m}_i = \bar{m}_j$ then return **FAIL end.**
        $M \leftarrow \prod_{i=1}^{e_k} (z - \theta_i) \in \mathbb{Z}_q[z]$; .... $O(e_k^2)$
        for $j = 1, 2, \cdots, N_{\max}$ **do**
            Interpolate $U \in \mathbb{Z}_p[z]$ using points $(\theta_i, v_{kj} : 1 \leq j \leq e_k)$; ..... $O(e_k^2)$
            $B_j \leftarrow \mathrm{MQRFR}(M, U, p) // B_j = N_j(z)/\hat{N}_j(z) \in \mathbb{Z}_q(z)$ .... $O(e_k^2)$
            Normalize $B_j(z)$ s.t. $\hat{N}_j(z) = 1 + \sum_{i=1}^{\deg(\hat{N})} b_i z^i$.
            if $\deg(N_j) \neq \deg(f_k)$ or $\deg(\hat{N}_j) \neq \deg(g_k)$ return **FAIL**
        end
        Let $a_i = \mathrm{LC}(N_j, z)$ and let $b_i = \mathrm{LC}(\hat{N}_j, z)$ for $1 \leq i \leq N_{\max}$.
        $F_k \leftarrow$VandermondeSolver$(\hat{m}, [a_1, \cdots, a_{\hat{n}}], \hat{s}, \hat{M})$; .... $O(\hat{n}^2)$
        $G_k \leftarrow$VandermondeSolver$(\bar{m}, [b_1, \cdots, b_{\bar{n}}], \hat{s}, \bar{M})$; .... $O(\bar{n}^2)$
        $F_k \leftarrow$GetTerms$(F_k, [\hat{Y}_1, \cdots, \hat{Y}_{N_{\max}}], [N_1, \cdots, N_{N_{\max}}], \hat{s}, \alpha, \beta, B_1)$
        $G_k \leftarrow$GetTerms$(G_k, [\hat{Y}_1, \cdots, \hat{Y}_{N_{\max}}], [\hat{N}_1, \cdots, \hat{N}_{N_{\max}}], \hat{s}, \alpha, \beta, B_2)$
        if $F_k =$ **FAIL** or $G_k =$ **FAIL** then return **FAIL end**
    end
    $\hat{X} \leftarrow [\frac{F_k}{G_k}, 1 \leq k \leq n]$ // Here $\hat{X} = x \mod q$
    Solve $\hat{F} \equiv \overline{X} \mod P$ and $\hat{F} \equiv \hat{X} \mod q$ using the Chinese remainder theorem
    $P \leftarrow P \times q$.
    Apply rational number reconstruction on coefficients of $\hat{F}$ mod P to get x
    if $x \neq$ **FAIL** and $x \mod q = \hat{F}$ **then** return $\overline{F}$ **else** $(\overline{X}, p) \leftarrow (\hat{F}, q)$ **end**
end

Subroutine 3: GetTerms

Input: $F_k \in \mathbb{Z}_q[y_1, \cdots, y_m]$, $\alpha \in (\mathbb{Z}_q \setminus \{0\})^m$, $\beta \in \mathbb{Z}_q^m$, $\hat{s} \in [1, q-2]$, and list of lower total degree polynomials $B_1 = \{f_{\deg(f_k)-1,k}, \cdots, f_{0,k}\}$, points $[\hat{Y}_j \in \mathbb{Z}_q^m : 1 \leq j \leq N_{\max}]$ and $[N_j \in \mathbb{Z}_q[z] : 1 \leq j \leq N_{\max}]$.

Output: $\overline{f}_k \in \mathbb{Z}_q[y_1, \cdots, y_m]$

$(\overline{A}, \overline{f}_k, \hat{d}) \leftarrow (F_k, F_k, \deg(F_k))$ and set $\Gamma = (0, 0, , \cdots, 0) \in \mathbb{Z}_q^{N_{\max}}$.
$\overline{D} \leftarrow [\deg(e) : e \in B_1]$, $\quad \hat{M} \leftarrow [\operatorname{supp}(e) : e \in B_1]$ // supp means support.
for $h = 1, 2, \cdots, |\overline{D}|$ **do**
 $d \leftarrow \overline{D}_h$
 if $\beta \neq 0$ **then**
  Pick $\theta \in \mathbb{Z}_q^{\hat{d}+1}$ at random.
  for $j = 1, 2, \cdots, N_{\max}$ **do**
   $Z_{j,t} \leftarrow \overline{A}(y_1 = \hat{Y}_{j,1}\theta_t + \beta_1, \cdots, y_m = \hat{Y}_{j,m}\theta_t + \beta_m)$ for $1 \leq t \leq \hat{d} + 1$; . . . . . . $O(m\#\overline{A} + m\hat{d})$
   Interpolate $\overline{W}_j \in \mathbb{Z}_q[z]$ using $(\theta_t, Z_{j,t} : 1 \leq t \leq \hat{d} + 1)$; . . . . . . $O(\hat{d}^2)$
   $\Gamma_j \leftarrow \Gamma_j + \overline{W}_j$; . . . . . . $O(\hat{d})$
  end
 end
 if $d \neq 0$ **then**
  $P \leftarrow \left[\operatorname{coeff}(N_j, z^d) : 1 \leq j \leq N_{\max}\right]$
  if $\beta \neq 0$ **then** $P_j \leftarrow P_j - \operatorname{coeff}(\Gamma_j, z^d)$ for $1 \leq j \leq N_{\max}$ **end**
  $\hat{m} \leftarrow [\hat{M}_i(\alpha) : 1 \leq i \leq \hat{n}]$ where $\hat{n} = \#\hat{M}_h$; . . . . . . $O(m\hat{n})$
  if any monomial evaluations $\hat{m}_i = \hat{m}_j$ **then** return **FAIL end.**
  $\overline{A} \leftarrow \text{VanderSolver}(\hat{m}, P, \hat{s}, \hat{M}_h)$; . . . . . . $O(\hat{n}^2)$
 else
  $\overline{A} \leftarrow \operatorname{coeff}(N_1, z^0)$ // We use only one point to get the constant term
  if $\beta \neq 0$ **then** $\overline{A} \leftarrow \overline{A} - \operatorname{coeff}(\Gamma_1, z^0)$ **end**
  $(\overline{f}_k, \hat{d}) \leftarrow (\overline{f}_k + \overline{A}, \deg(\overline{A}) + 1)$.
 end
end
return $\overline{f}_k$.

Subroutine 4: BMStep

Input: $P = [P_j \in \mathbb{Z}_p : 1 \leq j \leq i]$, i is even, $\alpha \in \mathbb{Z}_p$, shift $\hat{s} \in [1, p-2]$ and r which defines the Kronecker substitution K_r.

Output: $\bar{F} \in \mathbb{Z}_p[y_1, y_2, \cdots, y_m]$ or **FAIL**.

Run the Berlekamp-Massey algorithm [1] on P to obtain $\lambda(z) \in \mathbb{Z}_p[z]$; . . $O(i^2)$
if $\deg(\lambda, z) = \frac{i}{2}$ **then** return **FAIL end** // More images are needed
Compute the roots of λ in $\mathbb{Z}_p[z]$ to obtain the monomial evaluations $\hat{m}_i$. Let $\hat{m} \subset \mathbb{Z}_p$ be the set of monomial evaluations $\hat{m}_i$ and let $t = |\hat{m}|$; . . . $O(t^2 \log p)$
if $t \neq \deg(\lambda, z)$ **then** return **FAIL end** // $\lambda(z)$ is wrong.
Solve $\alpha^{e_i} = \hat{m}_i$ for e_i with $e_i \in [0, p-2]$ // The exponents are found here.
Let $\hat{M} = [\, y^{e_i} : i = 1, 2 \cdots, t \,]$ // These are the monomials
$F \leftarrow$VandermondeSolver $(\hat{m}, [P_1, \cdots P_t], \hat{s}, \hat{M})$ // $F \in \mathbb{Z}_p[y]$; . . . . . . $O(t^2)$
$\bar{F} \leftarrow K_r^{-1}(F) \in \mathbb{Z}_p[y_1, \cdots, y_m]$.// Invert the Kronecker map K_r.
return $\bar{F}$

Subroutine 5: VandermondeSolver

Input: Vectors $\hat{m}, b \in \mathbb{Z}_p^t$, shift $\hat{s} \in [1, p-2]$ and monomials $[M_1, \cdots, M_t]$

Output: $F \in \mathbb{Z}_p[y_1, \cdots, y_m]$

Let $V_{ij} = \hat{m}_i^{\hat{s}+j-1}$ for $1 \leq i, j \leq t$.

Solve $Va = b$ for the coefficients a_i using Zippel's $O(t^2)$ algorithm [17].

return $F = \sum_{i=1}^{t} a_i M_i$

Subroutine 6: RemoveShift

Input: $F_k \in \mathbb{Z}_p[y_1, \cdots, y_m]$, $\beta \in \mathbb{Z}_p^m$, list of degrees E_{f_k}, random shift $\hat{s} \in [1, p-2]$, a generator α for $\mathbb{Z}_p^*$, r which defines Kronecker substitution K_r, list of vectors $[\hat{Y}_j \in \mathbb{Z}_p^m : 1 \leq j \leq i]$ and list of univariate polynomials $[N_j \in \mathbb{Z}_p[z] : 1 \leq j \leq i]$.

Output: $f_k \in \mathbb{Z}_p[y_1, \cdots, y_m]$ or **FAIL**

$(\overline{A}, f_k, d) \leftarrow (F_k, F_k, \deg(F_k))$
Initialize $\Gamma_j = 0$ for $1 \leq j \leq i$.
for $\bar{d} \in E_{f_k}$ **do**
    if $\beta \neq 0$ **then**
        Pick $\theta \in \mathbb{Z}_p^{d+1}$ at random.
        for $j = 1, 2, \cdots, i$ **do**
            Evaluate $\overline{A}$; .... $O(m\#\overline{A} + md))$.
            $Z_{j,t} \leftarrow \overline{A}(y_1 = \hat{Y}_{j,1}\theta_t + \beta_1, \cdots, y_m = \hat{Y}_{j,m}\theta_t + \beta_m)$ for $: 1 \leq t \leq d+1$.
            Interpolate $\overline{W}_j \in \mathbb{Z}_p[z]$ using $(\theta_t, Z_{j,t} : 1 \leq t \leq d+1)$; ...... $O(d^2)$
            $\Gamma_j \leftarrow \Gamma_j + \overline{W}_j$ .... $O(d)$
        end
    end
    if $\bar{d} \neq 0$ **then**
        $P \leftarrow \left[\text{coeff}(N_j, z^{\bar{d}} : 1 \leq j \leq i\right]$.
        if $\beta \neq 0$ then $P_j \leftarrow P_j - \text{coeff}(\Gamma_j, z^{\bar{d}})$ for $j = 1, 2, \cdots, i$ **end**
        // The P_j's are adjusted to correct the effect of the basis shift β.//
        if $[P_j = 0 : 1 \leq j \leq i]$ **then**
            $\overline{A} \leftarrow 0$ // There is no monomial of total degree $\hat{d}$.
        else
            $\overline{A} \leftarrow \text{BMStep}([P_1, \cdots, P_i], \alpha, \hat{s}, r)$; .... $O(i^2 + \#\overline{A}^2 \log p)$
            if $\overline{A} =$ **FAIL then** return **FAIL end** // More P_j's are needed.
        end
    else
        $\overline{A} \leftarrow \text{coeff}(N_1, z^0)$// We get the constant term.
        if $\beta \neq 0$ then $\overline{A} \leftarrow \overline{A} - \text{coeff}(\Gamma_1, z^0)$ **end**
    end
    $(f_k, d) \leftarrow (f_k + \overline{A}, \hat{d} + 1)$.
end
return f_k

4 Analysis

4.1 Failure Probability Analysis

Here we identify all the problems that can occur in our algorithm for solving parametric linear systems. The proofs in this paper require the Schwartz-Zippel Lemma [16, 17]. We state the lemma and some useful results now. The analysis in this section follows [12].

Lemma 5 (Schwartz-Zippel Lemma) *Let $\mathbb{K}$ be a field and let f be a non-zero polynomial in $\mathbb{K}[y_1, y_2, \cdots, y_m]$. If α is chosen at random from F^m with $F \subset \mathbb{K}$ then* $\operatorname{Prob}[f(\alpha) = 0] \leq \frac{\deg(f)}{|F|}$.

Definition 6 *Let $f = \sum_{i=1}^{t} a_i N_i \in \mathbb{Z}[y_1, y_2, \cdots, y_m]$ where $a_i \in \mathbb{Z} \setminus \{0\}, t = \#f \geq 1$ and N_i is a monomial in variables $y_1, y_2, \cdots, y_m$. The height of f denoted by $\|f\|_\infty$ is defined as $\|f\|_\infty = \max_{i=1}^{t} |a_i|$. We also define $\|H\|_\infty =$ $\max(\|f_k\|_\infty, \|g_k\|_\infty)$ where $H = \frac{f_k(y_1, \cdot, y_m)}{g_k(y_1, \cdot, y_m)}$.*

Theorem 7 *[9, Proposition 2] Let A be a $n \times n$ matrix with $A_{ij} \in \mathbb{Z}[y_1, \cdots, y_m]$, $\#A_{ij} \leq t$ and $\|A_{ij}\|_\infty \leq h$. Then $\|\det(A)\|_\infty < n^{\frac{n}{2}} t^n h^n$.*

Lemma 8 *[6, Lemma 2, page 135] Let $f, g \in \mathbb{Z}[y_1, y_2, \cdots, y_m]$. If $g|f$ then $\|g\|_\infty \leq \mathrm{e}^{\sum_{i=1}^{m} \deg(f, y_i)} \|f\|_\infty$ where* $\mathrm{e} = 2.718$.

For the rest of this paper, let $\#A_{ij}, \#b_j, \#f_i, \#g_i \leq t$ and let $\|A_{ij}\|_\infty, \|b_j\|_\infty \leq h, \deg(b_j), \deg(A_{ij}), \deg(f_i), \deg(g_i) \leq d$. Let $P = \{p_1, p_2, \cdots, p_N\}$ be the list of machine primes to be used in our algorithm such that $p_{\min} = \min_{i=1}^{N}\{p_i\}$ and N is a large positive integer. We now estimate the height of the entries x_i.

Theorem 9 *We have $\|x_k\|_\infty \leq \mathrm{e}^{nmd} n^{\frac{n}{2}} t^n h^n$ where* $\mathrm{e} = 2.718$.

Proof. By Cramer's rule, the solutions of $Ax = b$ are given by $\dfrac{R_k}{R} = \dfrac{\det(A^k)}{\det(A)}$ where A^k denotes the matrix obtained by replacing the k-th column of the coefficient matrix A by the column vector b. Let $h_k = \gcd(R_k, R)$. Observe that

$$x_k = \frac{R_k/h_k}{R/h_k} = \frac{f_k}{g_k}$$

where $\gcd(f_k, g_k) = 1$. Therefore $f_k | R_k$ and $g_k | R$. By Lemma 8, it follows that

$$\|g_k\|_\infty \leq \mathrm{e}^{\sum_{i=1}^{m} \deg(R, y_i)} \|R\|_\infty \leq \mathrm{e}^{\sum_{i=1}^{m} nd} \|R\|_\infty \leq e^{nmd} \|R\|_\infty \tag{9}$$

and

$$\|f_k\|_\infty \leq \mathrm{e}^{nmd} \|R_k\|_\infty \tag{10}$$

since $\deg(R, y_i) \leq \deg(R) \leq n \times \max_{i=1}^{n}\{\deg(A_{ij})\} \leq nd$. Therefore

$$\|x_k\|_\infty \leq \max(\|f_k\|_\infty, \|g_k\|_\infty) \leq \mathrm{e}^{nmd} \max(\|R_k\|_\infty, \|R\|_\infty) \leq \mathrm{e}^{nmd} n^{\frac{n}{2}} t^n h^n$$

by Theorem 7. □

We remark that the above bound for the height of x_k is the worst case bound.

4.1.1 Unlucky Primes and Evaluation Points

Definition 10 *Let p be a prime. A prime p is said to be unlucky if* $p|\det(A)$.

Definition 11 *Suppose p is not an unlucky prime. Let $\alpha \in \mathbb{Z}_p^m$ be an evaluation point. We say that α unlucky if* $\det(A)(\alpha) = 0$.

Lemma 12 *Let p be a prime chosen at random from P and let* $\mathrm{e} = 2.718$. *Then*

$$\Pr[p \textit{ is unlucky}] \leq \frac{n \log_{p_{\min}}(th\sqrt{n}) + nmd \log_{p_{\min}} \mathrm{e}}{N}.$$

Proof. Let $R = \det(A)$ and let c be an integer coefficient of R. The number of primes p from P that can divide c is at most $\lfloor \log_{p_{\min}} c \rfloor$. So

$$\Pr[\, p \,|\, c\,] \leq \frac{\log_{p_{\min}} c}{N}.$$

By definition, prime p is unlucky $\iff p|R \implies p$ divides one term in R. So

$$\Pr[p \text{ is unlucky}] = \Pr[\, p \,| R\ ] \leq \Pr[\, p \text{ divides one term in } R\ ] \leq \frac{\log_{p_{\min}} \|R\|_\infty}{N}.$$

Using Theorem 9, it follows that

$$\Pr[p \text{ is unlucky}] \leq \frac{\log_{p_{\min}} \left(\mathrm{e}^{nmd} n^{\frac{n}{2}} t^n h^n\right)}{N} \leq \frac{n \log_{p_{\min}}(th\sqrt{n}) + nmd \log_{p_{\min}} \mathrm{e}}{N}.$$

$\square$

Lemma 13 *Let p be a prime chosen at random from P. Let $\alpha \in \mathbb{Z}_p^m$ be an evaluation point. Then* $\Pr[\alpha \textit{ is unlucky}] \leq \dfrac{nd}{p}$.

Proof. $\Pr[\alpha \text{ is unlucky}] = \Pr[\det(A)(\alpha) = 0] \leq \dfrac{\deg(\det(A))}{p} \leq \dfrac{nd}{p}$. $\square$

4.1.2 Bad Evaluation Points, Primes and Basis Shift

Definition 14 *We say that an evaluation point $\alpha \in \mathbb{Z}_p^m$ is a* ***bad evaluation point*** *if* $\deg(f_k^\beta(\alpha, z)) < \deg(f_k, z)$ *or* $\deg(g_k^\beta(\alpha, z)) < \deg(g_k, z)$ *for any k.*

Definition 15 *We say that $\beta \in (\mathbb{Z}_p \backslash \{0\})^m$ is a* ***bad basis shift*** *if $gcd(f_k, g_k) = 1$ but* $\deg(\gcd(f_k^\beta(\alpha, z), g_k^\beta(\alpha, z))) > 0$ *for any k.*

Definition 16 *We say a prime p is* ***bad*** *if* $p|\mathrm{LC}(f_k)$ *or* $p|\mathrm{LC}(g_k)$ *for any k.*

To avoid the occurrence of bad evaluation points with high probability in Algorithm 1, we had to interpolate $F_k(\alpha^{\hat{s}+i}, z, \beta)$ for some random point $\hat{s} \in [1, p-2]$ instead of $F_k(\alpha^i, z, \beta)$. This is labelled as A_j in Line 25. Line 26 detects the occurrence of bad evaluation points, a bad basis shift or a bad prime.

Example 17 *Let p be a sufficiently large prime and let*

$$\frac{f_1}{g_1} = \frac{y_1}{(y_1+y_3)y_2} \in \mathbb{Z}_p(y_1, y_2, y_3).$$

Observe that the partial degrees $e_i = \max\{\deg(f_1, y_i), \deg(g_1, y_i)\} = 1$ for $1 \le i \le 3$. For the Kronecker map K_r to be invertible we need $r_i > e_i$, so let $r = (2, 2)$. Thus the mapped function

$$K_r(f_1/g_1) = \frac{f(y, y^2, y^4)}{g(y, y^2, y^4)} = \frac{y}{(y+y^4)y^2} = \frac{y}{y^3+y^6}.$$

Since g_1 has no constant term, we need a basis shift $\beta \in (\mathbb{Z}_p \backslash \{0\})^3$. To interpolate $K_r(f_1/g_1)$, we need to densely interpolate $F_1(\alpha^j, z, \beta)$ for $1 \le j \le 4 = 2 \times \#g_1$. Computing $F_1(\alpha, z, \beta)$ directly yields

$$F_1(\alpha, z, \beta) = \frac{f_1^\beta(\alpha, z)}{g_1^\beta(\alpha, z)} = \frac{\alpha z + \beta_1}{(z\alpha^4 + z\alpha + \beta_1 + \beta_3)(z\alpha^2 + \beta_2)}.$$

The Sylvester resultant

$$\mathcal{R} = \mathrm{Res}(f_1^\beta(\alpha, z), g_1^\beta(\alpha, z), z) = \alpha^2(\alpha^3\beta_1 - \beta_3)(\alpha\beta_1 - \beta_2) \ne 0$$

since $\alpha \ne 0$ and $\beta = (\beta_1, \beta_2, \beta_3) \ne (0, 0, 0)$. But, if $\beta_2 = \alpha\beta_1 \ne 0$ or $\beta_3 = \alpha^3\beta_1 \ne 0$ then $\mathcal{R}(\beta) = 0$ which implies that β is a bad basis shift.

4.1.3 Main Results

Theorem 18 *Let N_a be greater than the required number of auxiliary rational function needed to interpolate the unique solution x and suppose all the degree bounds obtained in Lines 1-5 of Algorithm 1 are correct. If prime p is chosen at random from P then the probability that Algorithm 1 returns FAIL is at most*

$$\frac{6N_a n^2 d \left(\log_{p_{\min}}(th\sqrt{n}) + 2md\log_{p_{\min}} \mathrm{e}\right)}{N} + \frac{2N_a n(1+d)^m + 2nt^3d^2 + 5n^2N_a d^2}{p}.$$

Proof. First, recall that $e_{\max} = \max_{k=1}^n\{\deg(f_k) + \deg(g_k) + 2\}$. Now notice that

Pr[v_j = FAIL in Line 18] = Pr[p or evaluation point $\mathrm{Z_j}$ in Line 17 is unlucky].

By Lemma 13 and 12, Pr[Algorithm 1 returns FAIL in Line 13] is at most

$$ne_{\max}N_a\left(\frac{nd}{p} + \frac{n\left(\log_{p_{\min}}(th\sqrt{n}) + md\log_{p_{\min}} \mathrm{e}\right)}{N}\right). \tag{11}$$

There are three causes of FAIL in Line 26 of Algorithm 1. All three failure causes (bad evaluation point, bad basis shift and bad prime) are direct consequence of our attempt to interpolate auxiliary rational functions A_j in Line 25. We will handle the bad evaluation point case first. Let

$$\Delta(y) = \prod_{k=1}^n \mathrm{LC}(f_k^\beta(y, z))\mathrm{LC}(g_k^\beta(y, z)) \in \mathbb{Z}_p[y].$$

Notice that the evaluation point $\alpha^{\hat{s}+j-1}$ in Line 15 is random since $\hat{s} \in [1, p-2]$ is random and α is randomly selected in Line 9. Since a basis shift β does not affect the degree and the leading coefficients of auxiliary rational functions, we have that if $\alpha^{\hat{s}+j-1}$ is a bad then $\Delta(\alpha^{\hat{s}+j-1}) = 0$. Thus

$$\text{Prob}[\alpha^{\hat{s}+j-1} \text{ is a bad for } 0 \leq j \leq N_a - 1] \leq \frac{N_a \deg(\Delta)}{p} \leq \frac{2N_a n(1+d)^m}{p}.$$

Now suppose $\theta_j := \alpha^{\hat{s}+j-1}$ is not bad for $1 \leq j \leq N_a$. Let $w_1, w_2, \cdots w_m$ be new variables and let

$$G_{kj} = \frac{\hat{f}_{k_j}}{\hat{g}_{k_j}} = \frac{f_k(\theta_j z + w_1, \cdots, z\theta_j^{(r_1 r_2 \cdots r_{m-1})} + w_m)}{g_k(\theta_j z + w_1, \cdots, z\theta_j^{(r_1 r_2 \cdots r_{m-1})} + w_m)} \in \mathbb{Z}_p(w_1, w_2, \cdots, w_m)(z).$$

Recall that $\text{LC}(\hat{f}_{k_j})(\beta) \neq 0$ and $\text{LC}(\hat{g}_{k_j})(\beta) \neq 0$. Let $\overline{R}_{kj} = \text{Res}(\hat{f}_{k_j}, \hat{g}_{k_j}, z) \in \mathbb{Z}_p[w_1, w_2, \cdots, w_m]$ be the Sylvester resultant and let $\Delta(w_1, w_2, \cdots, w_m) = \prod_{j=1}^{N_a} \prod_{k=1}^{n} \overline{R}_{kj}$. Clearly, β picked at random in Line 7 is a bad basis shift $\iff \Delta(\beta) = 0 \iff \deg(\gcd(\hat{f}_{k_j}(z, \beta), \hat{g}_{k_j}(z, \beta)) > 0$ for any k and j. Using Bezout's bound [9, Lemma 4], we have $\deg(\overline{R}_{kj}) \leq \deg(f_k)\deg(g_k) \leq d^2$. Thus

$$\text{Prob}[\beta \text{ is a bad basis shift}] = \text{Prob}[\Delta(\beta) = 0] \leq \frac{\deg(\Delta)}{p} \leq \frac{nd^2 N_a}{p}.$$

Finally, we deal with the bad prime case. Observe that Prob[prime p is bad] $\leq$

$$\text{Prob}[\, p \text{ divides 1 term of } \text{LC}(f_k) \text{ or } \text{LC}(g_k) \text{ for } 1 \leq k \leq n] \leq \frac{n \log_{p_{\min}}(\|f_k\|_\infty \|g_k\|_\infty)}{N}.$$

Using Equations (9) and (10), we have Prob[prime p is bad $:$ $1 \leq j \leq N_a$]

$$\leq \frac{N_a n \log_{p_{\min}}(\|f_k\|_\infty \|g_k\|_\infty)}{N} \leq \frac{2N_a n^2 \left(\log_{p_{\min}}(th\sqrt{n}) + 2md \log_{p_{\min}} \mathrm{e}\right)}{N}$$

Thus Pr[Algorithm 1 returns FAIL in Line 26] is at most

$$\frac{2N_a n^2 \left(\log_{p_{\min}}(th\sqrt{n}) + 2md \log_{p_{\min}} \mathrm{e}\right)}{N} + \frac{2N_a n(1+d)^m}{p} + \frac{nd^2 N_a}{p}. \tag{12}$$

Since N_a is greater than the required number of auxiliary rational function needed by Algorithm 1 to interpolate x, then Line 2 of Subroutine 4 will never return FAIL. However the feedback polynomial $\lambda \in \mathbb{Z}_p[z]$ generated to find the number of terms in $f_{i,k}$ or $g_{i,k}$ in Line 4 of Subroutine 4 might be wrong so it will return FAIL which causes Algorithm 1 to return FAIL in either Lines 29 or 30 or 33 or 34. By [13, Theorem 3], the probability of getting the wrong $\#f_{i,k}$ or $\#g_{i,k}$ $\leq \sum_{k=1}^{n} \frac{\sum_{i=0}^{\deg(f_k)} \#f_{i,k}(\#f_{i,k}+1)(2\#f_{i,k}+1)\deg(f_{i,k}) + \sum_{i=0}^{\deg(g_k)} \#g_{i,k}(\#g_{i,k}+1)(2\#g_{i,k}+1)\deg(g_{i,k})}{6p}$. Since $\#f_{i,k}, \#g_{i,k} \leq t$ and $\deg(f_{i,k}), \deg(g_{i,k}) \leq d$, we get

$$\text{Pr[Algorithm 1 returns FAIL in Lines 29 or 30 or 33 or 34]} \leq \frac{2nt^3 d^2}{p}. \tag{13}$$

Since $e_{\max} \le 4d$, our result follows by adding (11)-(13). $\square$

Theorem 19 *Suppose additional primes are selected at random from the list of primes P to reconstruct the coefficients of x using rational number reconstruction. Let p be the first prime used by Algorithm 1. Then* $\Pr[$*Algorithm 2 returns FAIL*$]$

$$\le \frac{6N_a n^2 d\left(\log_{p_{\min}}(th\sqrt{n}) + 2md\log_{p_{\min}} \mathrm{e}\right)}{N} + \frac{2ndN_a + nd^2N_a + 4nd^2t^2}{p-1}.$$

Proof. Any prime q selected by Algorithm 2 $> p$, so $\frac{1}{q} < \frac{1}{p}$. Using (11), the probability that Algorithm 2 returns FAIL in Line 13 is at most

$$n^2 e_{\max} N_a \left(\frac{d}{p} + \frac{\left(\log_{p_{\min}}(th\sqrt{n}) + md\log_{p_{\min}} \mathrm{e}\right)}{N}\right) \tag{14}$$

If the monomial evaluations obtained in Line 20 of Algorithm 2 or the monomial evaluations obtained in Line 17 of Subroutine 3 are not distinct then

$$\begin{aligned}&\Pr[\text{Algorithm 2 returns FAIL in Line 20 or 31 or 32}]\\ &\le \sum_{k=1}^{n} \frac{\left(\sum_{i=0}^{\deg(f_k)} \binom{\#f_{i,k}}{2} \deg(f_{i,k}) + \sum_{i=0}^{\deg(g_k)} \binom{\#g_{i,k}}{2} \deg(g_{i,k})\right)}{p-1} \le \frac{4nd^2t^2}{p-1}.\end{aligned} \tag{15}$$

Notice that the functions B_j obtained in Line 24 are of the form

$$\frac{f_k^{\beta}(y_1, y_2, \cdots, y_m, z)}{g_k^{\beta}(y_1, y_2, \cdots, y_m, z)} = \frac{f_k(y_1 z + \beta_1, \cdots, y_m z + \beta_m)}{g_k(y_1 z + \beta_1, \cdots, y_m z + \beta_m)},$$

and are different from the A_j obtained in Algorithm 1 because a Kronecker map is not used. Let $\Delta = \prod_{k=1}^{n} \mathrm{LC}(f_k^{\beta})\mathrm{LC}(g_k^{\beta}) \in \mathbb{Z}_p[y_1, y_2, \cdots, y_m]$. Since $\deg(\Delta) \le 2nd$ and $N_a \ge \hat{N}_{\max}$, then $\mathrm{Prob}[\hat{Y}_j$ picked in Line 9 of Algorithm 2 is bad $: 0 \le j \le \hat{N}_{\max} - 1] \le \dfrac{2ndN_a}{p}$. Hence $\Pr[$Algorithm 2 returns FAIL in Line 26$] \le$

$$\frac{2N_a n^2\left(\log_{p_{\min}}(th\sqrt{n}) + 2md\log_{p_{\min}} \mathrm{e}\right)}{N} + \frac{2ndN_a}{p} + \frac{nd^2N_a}{p}. \tag{16}$$

Our result follows by adding (14)-(16). $\square$

4.2 Complexity Analysis

Theorem 20 *Let $B = [A|b]$ be a $n \times n{+}1$ augmented matrix such that $\#B_{ij} \le t$ and $\|B_{ij}\|_\infty \le C^T$. Let prime p chosen at random from P and $C < p < 2C$. A black box probe costs $O(n^2tT + n^2mdt + n^3)$ arithmetic operations in $\mathbb{Z}_p$.*

Proof. Let $B_{ij} = \sum_{k=1}^{t} a_k B_{ij,k}(y_1, \cdots, y_m)$. The total cost of computing B mod p is $O(n^2tT_{\max})$ since the modular reduction B_{ij} mod p costs $O(tT)$. All monomial evaluations $B_{ij_k}(\alpha)$ can be computed using $O(mdt)$ multiplications and t multiplications for the product $a_k B_{ij_k}(\alpha) \in \mathbb{Z}_p$. Hence the cost of evaluating B is $O(n^2mdt)$. The cost of solving $B(\alpha)$ over $\mathbb{Z}_p$ using Gaussian elimination is $O(n^3)$. Thus a black box probe costs $O(n^2tT + n^2mdt + n^3)$. $\square$

Theorem 21 *Let* $\hat{N}_{\max} = \max_{k=1}^{n}(\max_{i=0}^{\deg(g_k)}\{\#f_{i,k}\}, \max_{j=0}^{\deg(f_k)}\{\#g_{i,k}\})$ *where* $f_{i,k}, g_{i,k}, f_k, g_k$ *is as defined in* (8) *and let* $e_{\max} = 2+\max_{k=1}^{n}\{\deg(f_k)+\deg(g_k)\}$. *Let* H *be maximum of all the integer coefficients of all the polynomials* f_k *and* g_k. *Then the number of black box probes required by our algorithm to interpolate the solution vector* x *is* $O(e_{\max}\hat{N}_{\max}\log H)$.

5 Implementation and Benchmarks

We have implemented our new algorithm in Maple with some parts coded in C to improve its overall efficiency. The parts coded in C include evaluating an augmented matrix at integer points modulo prime p, solving the evaluated augmented matrix with integer entries over $\mathbb{Z}_p$ using Gaussian elimination, finding and factoring the feedback polynomial produced by the Berlekamp-Massey algorithm, solving a $t \times t$ shifted Vandermonde system and performing dense rational function interpolation using the MQRFR algorithm modulo a prime. Each probe to the black box is computed using C code and its supports primes up to 63 bits in length. We have benchmarked our code on a 24 core Intel Gold 6342 processor with 128 gigabytes of RAM using only 1 core.

To test the the performance of our algorithm, we create the following artificial problem. Let $D \in \mathbb{Z}[y_1, y_2, \cdots, y_m]^{n\times n}$ with $\mathrm{rank}(D) = n$. Let the coefficient matrix A be a diagonal matrix such that its diagonal entries are non zero polynomials $g_1, \cdots, g_n$ and let the vector $b = \begin{bmatrix} f_1 & f_2 & \cdots & f_n \end{bmatrix}^T$. Clearly the vector $x = \begin{bmatrix} \frac{f_1}{g_1} & \frac{f_2}{g_2} & \cdots & \frac{f_n}{g_n} \end{bmatrix}^T$ solves $Ax = b$. But suppose we create a new linear system $Wx^* = c$ by premultiplying $Ax = b$ by D so that $Wx^* = (DA)x^* = Db = c$. Then both parametric systems $Ax = b$ and $Wx^* = c$ are equivalent. That is,

$$x^* = W^{-1}c = \frac{\mathrm{Adj}(DA)c}{\det(DA)} = \frac{\mathrm{Adj}(A)\mathrm{Adj}(D)Db}{\det(D)\det(A)} = \frac{\mathrm{Adj}(A)b}{\det(A)} = A^{-1}b = x$$

where Adj denotes the adjoint matrix.

In Table 2 we compare our new algorithm (row ParamLinSolve) with a Maple implementation of the Bareiss/Edmonds fraction free one step Gaussian elimination method with Lipson's fraction formula for back substitution (row Bareiss), a Maple implementation of the Gentleman & Johnson minor expansion method (row Gentleman) and using Maple's commands `ReducedRowEchelonForm` (row ReducedRow) and `LinearSolve` (row LinearSolve) for solving the systems $Wx^* = c$ that were created artificially.

The two input systems solved in Table 3 are real systems (Example 1 and a system from an engineering problem) which were the motivation for this work. Note that the timings reported for the real systems in Table 3 are in the columns and not in rows as in Table 2. The notation ! indicates that Maple was unable to allocate enough memory to finish the computation and − means unknown in both Tables 2 and 3. The breakdown of the timings for all individual algorithms involved for computing the system named bigsys are reported in 4.

The artificial input systems $Wx^* = c$ were created by generating matrices D, A and column vector b randomly, with all of their entries in $\mathbb{Z}[y_1, \cdots, y_m]$ where $m = 10, \deg(D_{ij}) \leq d_T = 5, \#D_{i,j} = T \leq 2$ and $\deg(A_{ij}), \deg(b_j) \leq d = 10, \#A_{i,j}, \#b_j = t \leq 5$ and $\operatorname{rank}(A) = \operatorname{rank}(D) = n$ for $3 \leq n \leq 10$. Using the Gentleman & Johnson algorithm, we obtain $\#\det(A), \#\det(D), \#\det(W)$ (rows 2-4) and the total CPU time used to compute each of them are reported in rows 10-13. We remark that we did not compute the $\gcd(\det(A^k), \det(A))$ when the Gentleman & Johnson algorithm was used. As the reader can see from Table 2, our algorithm performed better than other algorithms for $n \geq 5$.

n	3	4	5	6	7	8	9	10
$\#\det(A)$	125	625	3,125	15,500	59,851	310,796	1,923,985	9,381,213
$\#\det(D)$	40	336	3,120	38,784	518,009	8,477,343	156,424,985	-
$\#\det(W)$	5,000	209,960	9,741,747	-	-	-	-	-
ParamLinSolve	0.079s	0.176s	0.154s	0.211s	0.220s	0.239s	0.259s	0.317s
LinearSolve	0.129s	1.26s	304.20s	124200s	!	!	!	!
ReducedRow	0.01s	0.083	11.05s	3403.2s	!	!	!	!
Bareiss	2.02s	!	!	!	!	!	!	!
Gentleman	0.040s	3.19s	239.40s	!	!	!	!	!
time-det(A)	0s	0s	0.003s	0.08s	0.898s	0.703s	17.03s	25.32s
time -det(D)	0s	0s	0.007s	1.21s	1.39s	601.8s	2893.8s	!
time-det(W)	0s	0.310s	20.44s	!	!	!	!	!

Table 2: CPU Timings for solving $Wx^* = c$ with $\#f_i, \#g_i \leq 5$ for $3 \leq n \leq 10$.

system names	n	m	max	ParamLinSolve	Gentleman	LinearSolve	ReducedRow	Bareiss	$\#\det(A)$
Bspline	21	5	26	0.220s	2623.8s	0.021s	0.026s	0.500s	1033
Bigsys	44	48	58240	7776s	!	17.85s	1.66s	!	6037416

Table 3: CPU Timings for solving two real parametric linear systems

	Time(ms)	Percentage
Matrix Evaluation	151.48s	1.9 %
Gaussian Elimination	110.71s	1.4 %
Univariate Rational Function Interpolation	706.07s	9 %
Finding $\lambda \in \mathbb{Z}_p[z]$ using the Berlekamp-Massey Algorithm	208.25s	2.6 %
Roots of λ over $\mathbb{Z}_p$	4856.96s	62 %
Solving Vandermonde systems	434.46s	5.6 %
Multiplication and Addition of Evaluation points	257.40s	3.3 %
Computing Discrete logarithms	586.64s	7.6 %
Miscellaneous	464.67s	9.4 %
Overall Time	7776s	100 %

Table 4: Breakdown of CPU timings for all individual algorithms for computing bigsys

In our experiments, the cost of evaluating augmented matrices over $\mathbb{Z}_p$ is often the most expensive part. But as the reader can see in Table 4, computing the roots of the feedback polynomial for the bigsys system is the dominating cost. This is because the number of terms in many of the polynomials f_i, g_i to be interpolated is large. In particular, it has four polynomials where $\max(\#f_i, \#g_i) > 50,000$ and our root finding algorithm for computing the roots of $\lambda(z)$ costs $O(t^2 \log p)$ where $t = \deg(\lambda)$ is the number of terms of the f_i and g_i being interpolated.

A Appendix

The systems $Wx^* = c$ for Table 2 were created using the following Maple code:

```
CreateSystem := proc(n,m,T,dT,t,d) local A, D,W,c,b,Y,i;
   Y := [ seq(y||i,i=1..m) ];
   D := Matrix(n,n, () -> randpoly( Y,terms=T, degree=dT));
   b := Vector[column](n, () -> randpoly(Y, terms = t, degree= d));
   i := [ seq( randpoly( Y, terms = t, degree= d),i=1..n) ];
   A := DiagonalMatrix(i);
   W,c := D.A, D.b; return W,c,A,D;
end:
```

References

1. Atti, N. B. and Lombardi, H. and Diaz-Toca G. M.: The Berlekamp-Massey algorithm revisited. *AAECC* ***17***, (4), pp. 75–82, 2006.
2. Bareiss, E.: Sylvester's Identity and multistep integer-preserving Gaussian elimination. *Math. Comp.* ***22***, (103), pp. 565–578, 1968.
3. Ben-Or, M., and Tiwari, P.: A Deterministic Algorithm for Sparse Multivariate Polynomial Interpolation. *Proceedings of STOC '20*, pp. 301–309, ACM, 1988.
4. Cuyt, A., and Lee, W.-S.: Sparse Interpolation of Multivariate Rational Functions. *J. Theoretical Comp. Sci.* ***412:*** pp. 1445–1456, Elsevier, 2011.
5. Edmonds, J.: Systems of Distinct Representatives and Linear Algebra. *J. Research of the National Bureau of Standards* ***718***, (4), pp. 241–245, 1967.
6. Gelfond A.: *Transcendental and Algebraic Numbers.* GITTL, Moscow, 1952; English translation by Leo F. Boron, Dover, New York, 1960
7. Gentleman, W. M., and Johnson, S. C.: The Evaluation of Determinants by Expansion by Minors and the General Problem of Substitution. *Mathematics of Computation* ***28****(126):* pp. 543–548,1974.
8. Lipson, J.: Symbolic methods for the computer solution of linear equations with applications to flow graphs. *Proceedings of SISMC '1968*, pp. 233–303, IBM, 1969.
9. Hu, J., and Monagan, M.: A fast parallel sparse polynomial GCD algorithm. *Proceedings of ISSAC '2016*, pp. 271–278, ACM, 2016.
10. Jinadu, A., and Monagan, M.: An Interpolation Algorithm for computing Dixon Resultants. *Proceedings of CASC '2022,* LNCS **13366**: pp 185-205, Springer, 2022.
11. Jinadu, A., and Monagan, M.: A new interpolation algorithm for computing Dixon Resultants. *ACM* 56 (**2**): pp 88-91, 2022.
12. Jinadu, A., and Monagan, M.: The Failure Probability and Complexity Analysis of a Dixon Resultant Interpolation Method. *Submmitted to ISSAC '2023*
13. Kaltofen, E. , and Lee, W. , and Lobo, A.: Early termination in Ben-Or/Tiwari sparse interpolation and a hybrid of Zippel's algorithm. *Proceedings of ISSAC 2000, pp. 192–201, ACM, 2000.*
14. Monagan, M.: Maximal Quotient Rational Reconstruction: An Almost Optimal Algorithm for Rational Reconstruction. *Proceedings of ISSAC '2004*, pp. 243–249, ACM, 2004.
15. Monagan, M., Vrbik, P: Lazy and Forgetful Polynomial Arithmetic and Applications. *Proceedings of CASC '2009,* LNCS **5743**: pp 226-239, Springer, 2009.
16. Schwartz, J: Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, **27**:701–717 (1980)

17. Zippel, R.: Probabilistic Algorithms for Sparse Polynomials. *Proceedings of EUROSAM '79* , pp. 216–226, (1979), Springer-Verlag, 1979.