Solving parametric linear systems using sparse rational function interpolation

Ayoola Jinadu (Joint work with Michael Monagan)

Department of Mathematics

August 29,2023

SIMON FRASER UNIVERSITY

Problem Setup

Consider a parametric linear system

$$Ax = b$$

such that $A \in \mathbb{Z}[y_1, y_2, \ldots, y_m]^{n\times n}$, $\mathrm{rank}(A) = n$ and $b \in \mathbb{Z}[y_1, y_2, \ldots, y_m]^n$.

Goal: Interpolate the unique vector

$$x = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^T = \begin{bmatrix} \frac{f_1}{g_1} & \frac{f_2}{g_2} & \cdots & \frac{f_n}{g_n} \end{bmatrix}^T \quad (1)$$

such that for $f_k, g_k \in \mathbb{Z}[y_1, y_2, \ldots, y_m]$,

- $g_k \neq 0$, $g_k \,|\, \det(A)$, and
- $\gcd(f_k, g_k) = 1$ for $1 \leq k \leq n$.

Applications: engineering, computer vision, computer graphics.

- Using Cramer's rule,

$$x_i = \frac{\det(A^i)}{\det(A)} \in \mathbb{Z}(y_1, y_2, \ldots, y_m)$$

where A^i is the matrix obtained by replacing the i-th column of A with b.

- Let $\tilde{x}_i := \det(A^i) = x_i \det(A) \in \mathbb{Z}[y_1, y_2, \ldots, y_m]$.

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

$$B_{i,j} := \frac{B_{k,k}B_{i,j} - B_{i,k}B_{k,j}}{B_{k-1,k-1}};$$

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

$$\tilde{x}_i := \frac{N_i}{D_i};$$

end do

Problems with the Bareiss/Edmonds/Lipson algorithm

Expression swell occurs at the **final step**, when $k = n - 1$, where

$$B_{n,n} = \frac{B_{n-1,n-1}B_{n,n} - B_{n,n-1}B_{n-1,n}}{B_{n-2,n-2}} = \det(A) \in \mathbb{Z}[y_1, y_2, \ldots, y_m]$$

1. The same situation also holds
$$\tilde{x}_i := \frac{N_i}{D_i}$$
where $N_i = B_{i,n+1}B_{n,n} - \sum_{j=i+1}^{n} B_{i,j}\tilde{x}_j$; and $D_i = B_{i,i}$.
2. To compute the unique vector x in simplest terms, we have to compute
$$h_i = \gcd(\tilde{x}_i, \det(A))$$
which may be expensive.

Consider the following real linear system of 21 equations in variables $x_1, x_2, \ldots, x_{21}$ and parameters $y_1, y_2, \ldots, y_5$:

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

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

Using the Bareiss/Edmonds/Lipson algorithm, we determined that

- $\#B_{n,n} = \#\det(A) = 1033$,
- $\#B_{n-2,n-2} = 672$ and
- $\#B_{n,n}B_{n-2,n-2} = 14348$, so an expression swell factor of $14348/1033 = 14$.

i	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

i	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: Number of polynomial terms in $\tilde{x}_i = N_i/D_i$ and $x_i = f_i/g_i$ and expression swell factor for computing $\tilde{x}_i$

Methods that avoid expressison swell

1. Using lazy polynomial arithmetic approach [**Monagan and Vrbik**, 2009] : They compute

$$B_{i,j} := \frac{B_{k,k}B_{i,j} - B_{i,k}B_{k,j}}{B_{k-1,k-1}};$$

and

$$\tilde{x}_i := \frac{N_i}{D_i}$$

where $N_i = B_{i,n+1}B_{n,n} - \sum_{j=i+1}^{n} B_{i,j}\tilde{x}_j$; and $D_i = B_{i,i}$.

2. We can also use sparse polynomial interpolation algorithms to interpolate $\tilde{x}$ and $\det(A)$.

However, we still have to simplify the solutions (computing $\gcd(\det(A), \tilde{x}_i)$).

Using Gentleman & Johnson minor expansion algorithm

The Gentleman & Johnson minor expansion algorithm can also be used to compute

$$x_i = \frac{\det(A^i)}{\det(A)}$$

where A^i is obtained by replacing the i-th column of A with b.

Again, we still have to simplify the solutions (computing $\gcd(\det(A), \det(A^i))$).

Our sparse multivariate rational function interpolation method from CASC 2022

Suppose $A = f/g$ such $f, g \in \mathbb{Q}[y_1, y_2, \ldots, y_m]$ is represented by a "modular" black box.

- Our method is a modification of the Cuyt and Lee's method + the Ben-Or/Tiwari algorithm.

Two main problems posed when the Ben-Or/Tiwari algorithm is used :

- The points $\{(2^i, 3^i, \ldots, p_m^i) : i \geq 0\}$ can cause unlucky evaluation points problem.
- The working prime $p > p_m^{deg(f)}$ may be too large for machine arithmetic use.

Our new sparse rational function interpolation algorithm uses

1. A Kronecker substitution K_r: smaller primes are needed
   - We interpolate $K_r(A) = A(y, y^{r_1}, y^{r_1 r_2}, \ldots y^{\prod_{j=1}^{m-1} r_i})$ instead of $A = f/g$
   - Our new working prime must satisfy $p > \prod_{j=1}^{m} r_i$ where $r_i > \max(\deg(f, y_i), \deg(g, y_i))$.
2. A new set of randomized evaluation points: we use $\left\{y = \alpha^{\hat{s}+j} : j = 0, 1, 2, \ldots\right\}$ where $\hat{s} \in [0, p-2]$ is a random shift and α is a generator for $\mathbb{Z}_p^*$.

Our method requires the interpolation of auxiliary rational functions

$$F(\alpha^{\hat{s}+i}, z, \beta) = A(z\alpha^{\hat{s}+i} + \beta_1, z\alpha^{(\hat{s}+i)r_1} + \beta_2, \ldots, z\alpha^{(\hat{s}+i)\prod_{j=1}^{m-1} r_i} + \beta_m) \in \mathbb{Z}_p(z)$$

via calls to the black box, normalize them and then use their coefficients to recover $A = f/g$.

Our new black box algorithm for solving $Ax = b$

Let

$$f_k = \sum_{i=0}^{\deg(f_k)} f_{i,k}(y_1, y_2, \ldots, y_m) \text{ and } g_k = \sum_{j=0}^{\deg(g_k)} g_{j,k}(y_1, y_2, \ldots, y_m)$$

such that $f_{i,k}$ and $g_{j,k}$ are homogeneous polynomials of degree i and j respectively

Goal : to avoid gcd computations by interpolating $x_i = f_i/g_i$ directly using sparse rational function interpolation

Our new approach:

1. We use a "modular" black box **BB** : $\mathbb{Z}_p^m \to \mathbb{Z}_p^n$ for $B = [A|b]$
   - It accepts accepts an evaluation point α and a prime p to first compute $B(\alpha) \mod p$
   - then it solves $x(\alpha) = A^{-1}(\alpha)b(\alpha) \in \mathbb{Z}_p^n$ using Gaussian elimination over $\mathbb{Z}_p$.
2. We pre-compute all the needed degree bounds : we need
   - total degrees $\deg(f_k), \deg(g_k)$ $1 \leq k \leq n$.
   - maximum partial degrees $\max(\deg(f_k, y_i), \deg(g_k, y_i))$ for $1 \leq i \leq m$.
   - total degrees $\deg(f_{i,k}), \deg(g_{i,k})$
3. We interpolate x from the points $x(\alpha)$ using our sparse multivariate rational function interpolation algorithm
   - we interpolate $f_{\deg(f_k),k}$ and $g_{\deg(g_k),k}$ first then $f_{\deg(f_k)-1,k}$ and $g_{\deg(g_k)-1,k}, \ldots, f_{0,k}$ and $g_{0,k}$.
4. We use rational number construction and Chinese remaindering if needed.

Implementation and comparison to other algorithms

- We have implemented our algorithm for solving $Ax = b$ in Maple with some parts coded in C for efficiency.
- Maple's in built commands : using LinearSolve and ReducedRowEchelon
- a Maple implementation of the Gentleman & Johnson algorithm
- a Maple implementation of the Bareiss/Edmonds/Lipson algorithm

Benchmarks 1 (Artificial parametric linear systems)

We created a linear system $Wx^* = c$ which is equivalent to $Ax = b$ such that

- $W = DA$ and $c = Db$ for A is a diagonal matrix and $\mathrm{rank}(D) = n$
- The polynomial entries of D and A are small (it involves 10 parameters).
- the solutions of $Wx^* = c$ is much smaller than the determinants of the matrices involved.

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

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	NA
# det(W)	5,000	209,960	9,741,747	NA	NA	NA	NA	NA
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	!	!	!	!	!

! = out of memory and NA means Not Attempted

Benchmarks 2 (Real parametric linear systems)

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
Caglar	12	56	23072	1685.57s	NA	1232.40s	15480.35s	NA	15744
Sys66a	66	34	145744	665507.32s	!	!	!	!	NA
Sys66b	66	31	107468	255819.27s	!	!	!	!	NA

! = out of memory and NA means Not Attempted

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

	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 %

Theorem

- *Let* $\deg(b_j), \deg(A_{ij}), \deg(f_i), \deg(g_i) \leq d$.
- *Let* $\#A_{ij}, \#b_j, \#f_i, \#g_i \leq t$ *and let* $\|A_{ij}\|_\infty, \|b_j\|_\infty \leq h$.
- *Let* N_a *be greater than the required number of auxiliary rational function needed to interpolate* x.
- *Let* e *be the Euler number where* $e = 2.718$.
- *Suppose all the precomputed degree bounds obtained to interpolate* x *are correct.*
- *Suppose our new black box algorithm for solving* $Ax = b$ *only needs one prime to interpolate* x.

If prime p *is chosen at random from the list of* N *primes* $P = \{p_1, p_2, \ldots, p_N\}$ *such that* $p_{\min} = \min(P)$ *then the probability that our new black box algorithm returns FAIL is at most*

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

Theorem

- *Let* $\deg(b_j), \deg(A_{ij}), \deg(f_i), \deg(g_i) \leq d$.
- *Let* $\#A_{ij}, \#b_j, \#f_i, \#g_i \leq t$ *and let* $\|A_{ij}\|_\infty, \|b_j\|_\infty \leq h$.
- *Let* N_a *be greater than the required number of auxiliary rational functions needed to interpolate* x.
- *Let* $e = 2.718$ *be the Euler number.*

Suppose our new black box algorithm for solving $Ax = b$ *gets the support of the* x_i *but it needs more primes to recover the coefficients.*

If our algorithm selects a new prime at random from the list of N *primes* $P = \{p_1, p_2, \ldots, p_N\}$ *such that* $p_{\min} = \min(P)$ *to reconstruct the coefficients of* x *using rational number reconstruction*

Then probability that our new black box algorithm for solving $Ax = b$ *returns FAIL*

$$\leq \frac{6N_a n^2 d \left(\log_{p_{\min}}(th\sqrt{n})\right) + 2N_a n^2 md \log_{p_{\min}}(\mathrm{e})}{N} + \frac{7n^2d^2N_a + 4nd^2t^2}{p_{\min} - 1}.$$

Complexity analysis (in terms of the number of black box probes used)

Theorem

Suppose

$$f_k = \sum_{i=0}^{\deg(f_k)} f_{i,k}(y_1, y_2, \ldots, y_m) \text{ and } g_k = \sum_{j=0}^{\deg(g_k)} g_{j,k}(y_1, y_2, \ldots, y_m)$$

such that $f_{i,k}$ *and* $g_{j,k}$ *are homogeneous polynomials of degree* i *and* j *respectively*

- *Let* $\hat{N}_{\max} = \max_{k=1}^{n}(\max_{i=0}^{\deg(f_k)}\{\# f_{i,k}\}, \max_{j=0}^{\deg(g_k)}\{\# g_{i,k}\})$
- *Let* $e_{\max} = 2 + \max_{k=1}^{n}\{\deg(f_k) + \deg(g_k)\}$ *(#points needed for univariate rational function interpolation)*
- *Let* $H = \max_k(\|f_k\|_\infty, \|g_k\|_\infty)$

The number of black box probes required by our algorithm to interpolate the solution vector x *is*

$$O(e_{\max}\hat{N}_{\max} \log H).$$

Conclusion

1. A new black box algorithm to solve parametric linear systems that uses sparse rational function interpolation.
2. Implementation done in Maple with several parts coded in C for efficiency.
3. A detailed failure probability & complexity analysis in terms of number of black box probes used.