Solving Linear Systems of Equations Over Cyclotomic Fields

Liang Chen   Michael Monagan

Computational Algebra Group
Centre for Experimental and Constructive Mathematics
Department of Mathematics
Simon Fraser University

We want to solve large linear systems involving roots of unity arising from a problem in computational group theory. For example, the complex number i satisfies $i^4 = 1$. It is a primitive 4^{th} root of unity. A primitive k^{th} root of unity is a root of the cyclotomic polynomial $m_k(z)$. For example, i is a root of $m_4(z) = z^2 + 1$. The cyclotomic polynomials are of special interest because there are lots of primes for which $m_k(z)$ factors into distinct linear factors modulo p. For example

$$m_5(z) = z^4 + z^3 + z^2 + z + 1 = (z-3)(z-4)(z-5)(z-9) \bmod 11.$$

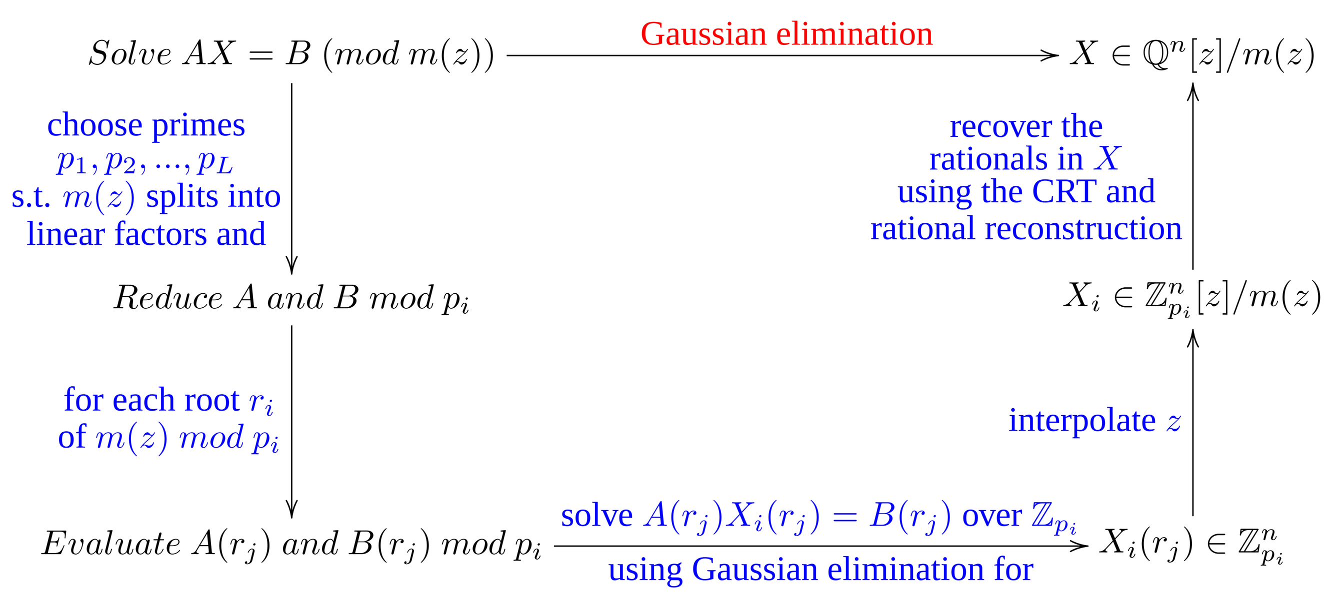
The following lemma tells us how to find such primes and how to factor $m_k(z)$.

Lemma: If p is a prime and $k|(p-1)$, then $m_k(z)$ has $d = \deg\ m(z)$ roots in $\mathbb{Z}_p$. Moreover, if ω is a primitive k^{th} root of unity $mod\ p$, then

$$\{\omega^i : 1 \le i \le k \text{ and } gcd(i,k) = 1\}$$

are roots of $m_k(z)$.

We exploit this to design two efficient algorithms for solving a linear system $AX = B$ involving roots of unity. The following figure describes the first algorithm:



Algorithm 1: Chinese Remaindering with Rational Reconstruction

Input: Matrix $A \in \mathbb{Q}^{n\times n}[z]$, vector $B \in \mathbb{Q}^n[z]$, polynomial $m(z) \in \mathbb{Z}[z]$.
Output: Vector $X \in \mathbb{Q}^n[z]$ which satisfies $AX = B\ (mod\ m(z))$.

1. Clear fractions in A and B and set $k := 1$.
2. **Loop**
   (a) Pick a new prime p_k which splits $m(z)$.
   (b) Find all roots $r_1, \ldots, r_d$ of $m(z)$ mod p_k.
   (c) **for** $j = 1$ **to** d **do**
      Solve $A(r_j)X_k(r_j) \equiv B(r_j)$ mod p_k for $X_k(r_j) \in \mathbb{Z}^n_{p_k}$.
      If there is no solution then go back to step $2(a)$.
   (d) Interpolate $X_k(z) \in \mathbb{Z}^n_{p_k}[z]$ from points r_j's and $X_k(r_j)$'s.
   (e) Apply Chinese remaindering to recover X mod $p_1 \times p_2 \times \cdots \times p_k$.
   (f) Apply rational reconstruction to recover $X \in \mathbb{Q}^n[z]/m(z)$.
   (g) If $m(z)|AX - B$ then output X.
   (h) Set $k := k + 1$.

Theorem 1: The running time of above algorithm is $O(n^2dLc + n^2d^2L + n^3dL + ndL^2)$ where $L \in O(ndc)$ is the number of primes needed, $n = \dim\ A$, $d = \deg\ m(z)$, $c = \max(\log ||A||_\infty, \log ||B||_\infty)$.

The above running time does not include the trial division in step $2(g)$ since this step may be avoided if we use sufficiently many primes. Solving a linear system $mod\ p_k$ using Gaussian elimination brings a factor of n^3, and rational reconstruction and Chinese remaindering brings a factor of L^2 which are the two main costs of the algorithm. An improvement can be made by doing rational reconstruction and trial division only after $1, 2, 4, 8, 16, \ldots$ primes.

Algorithm 2: p-adic Lifting with Rational Reconstruction

Input: Matrix $A \in \mathbb{Q}^{n\times n}[z]$, vector $B \in \mathbb{Q}^n[z]$, polynomial $m(z) \in \mathbb{Z}[z]$.
Output: Vector $X \in \mathbb{Q}^n[z]$ which satisfies $AX = B\ (mod\ m(z))$.

1. Clear fractions in A and B.
2. Pick a prime p which splits $m(z)$ and find all roots $r_1, \ldots, r_d$ of $m(z)$ mod p.
3. Set $k := 1$, $error := B$ and compute $A^{-1}(r_i) \bmod p$ for $i = 1, 2, \ldots, d$.
   If $A(r_j)$ is not invertible (mod p) then go back to step 2 and pick a new prime.
4. **Loop**
   (a) **for** $j = 1$ **to** d **set** $X_{k-1}(r_j) = A^{-1}(r_j)error(r_j) \bmod p$.
   (b) Interpolate $X_{k-1}(z) \in \mathbb{Z}^n_p[z]$ from r_i's and $X_{k-1}(r_j)$'s.
   (c) Compute $error := (error - AX_{k-1})/p$.
   (d) Obtain $X^{(k)} := X^{(k-1)} + X_{k-1}p^{k-1} = X_0 + X_1 \times p + X_2 \times p^2 + \cdots + X_{k-1} \times p^{k-1}$.
   (e) Apply rational reconstruction to recover $X \in \mathbb{Q}^n[z]/m(z)$.
   (f) If $m(z)|AX - B$ then output X.

Theorem 2: The running time of above algorithm is $O(n^3d + n^2d^2c^2 + nd^2cL + n^2dcL + ndc^2L + ndL^2)$ where $L \in O(ndc)$ is the number of lifting iterations needed, $n = \dim\ A$, $c = \max(\log ||A||_\infty, \log ||B||_\infty, \log n, d \log(||m||_\infty + 1))$, $d = \deg\ m(z)$.

The same remarks made about trial division apply here. The most costly part of this algorithm is updating the error in step $4(c)$. We implemented two variations which reduce the cost. (See **Lift 1** and **Lift 2** below) Also, this algorithm does d Gauss eliminations in step 3 whereas the first does Ld in step $2(c)$ which is why it is faster for large n.

Timings (in CPU seconds) for random systems.

n	Coefficient Length c							Remark
	2 digits	4 digits	8 digits	16 digits	32 digits	64 digits	128 digits	
5	.303	.321	.340	.390	.472	.700	1.558	GE
	.019	.029	.069	.136	.312	.643	1.412	CRT
	.028	.027	.049	.102	.245	.631	1.797	Lift 1
10	1.947	2.185	2.375	2.744	3.623	6.210	15.317	GE
	.050	.097	.183	.418	1.019	2.359	5.685	CRT
	.058	.091	.152	.309	.803	2.084	6.384	Lift 1
20	16.041	17.927	20.759	26.141	37.817	71.288	186	GE
	.167	.347	.727	1.616	4.759	12.149	30.983	CRT
	.158	.276	.521	1.054	3.005	8.219	26.581	Lift 1
40	148	181	207	291	476	1033	2829	GE
	.797	1.795	3.899	8.756	31.120	85.780	234	CRT
	.500	.973	1.932	3.998	11.891	33.412	113	Lift 1
$m(z) := z^6 + z^5 + z^4 + z^3 + z^2 + z + 1,\ d = 6$								

Timings (in CPU seconds) for Dabbaghian's systems.

file	sys49	sys100	sys100b	sys144	sys196	sys225	sys256	sys576	sys900	sys900b
$deg_z(m)$	4	8	4	2	2	4	4	6	8	2
k	5	24	8	4	3	5	12	7	24	4
$\|\|A\|\|_\infty$	10	5	2	4	11	2	3	3	2	5
$\|\|x\|\|_\infty$	45	14	1	1	229	875	2	1	2	1
CRT	.144	.788	.029	.036	3.344	3.056	.155	.842	2.358	1.458
Lift 1	.109	.443	.030	.029	1.183	2.374	.174	.612	2.761	.462
Lift 2	.111	.294	.100	.163	1.973	1.678	.640	3.022	7.627	5.711
GE	109	3080	30.15	10.49	4419	769	848	2055	2265	1195
# primes	4	1	1	1	9	36	1	1	1	1
Det	.293	4.159	.305	.147	6.206	4.644	3.748	53.69	338	25.74

Maplesoft command the brilliance
MITACS
NSERC CRSNG