Exact Solution to Large Sparse Integer Linear Systems

J. G. Dumas, Lmc Imag, Grenoble
W. Turner, NCSU, Raleigh
Z. Wan, U of Delaware, Newark

May 4, 2003

Abstract

Large scale linear algebra computations to obtain exact and symbolic results are now possible. Algorithmic developments in the last decade or two together with implementation in the software library LinBox [2] are part of the reason this is so. In this poster we illustrate the current state of the art by discussing three approaches to a challenging linear problem.

Trefethen has posed a "Hundred Dollar, Hundred Digit Challenge," [4]. Aimed at numerical analysts, the Challenge consists of 10 problems which have real number solutions. Ten digits of accuracy are asked in the answer to each. All of them are numerically difficult. Problem #7 is the computation of the $(1,1)$ entry of A^{-1}, where A is the 20000×20000 matrix whose entries are zero everywhere except for the primes $2, 3, 5, 7, \ldots, 224737$ along the main diagonal and the number 1 in the positions a_{ij} where $|i-j|$ is a power of 2. We took the task of producing the *exact* solution to this problem as a challenge to illustrate the capabilities of LinBox.

Three algorithmic approaches to the problem were implemented and tested. First, one may compute the exact value as a quotient of determinants:

$$A^{-1}_{1,1} = \frac{det(A_{2..20000,2..20000})}{det(A)}$$

We compute these two determinants by blackbox methods, specifically by using Wiedemann's algorithm [6, 3]. Compute the characteristic polynomial modulo primes p and apply Chinese remaindering to obtain the integer determinants.

Alternatively one may solve the system $Ax = e_1$ and extract the first entry. We implemented two variants of this approach, based on Dixon's [1] scheme of inverting the matrix modulo a prime p and then using Hensel lifting to obtain the rational solution. The lifting stage requires repeated solution of systems of the form $Ax = b \pmod{p}$ for various right hand side vectors b. If $A^{-1} \pmod{p}$ is computed these systems are straightforwardly solved as $x = A^{-1}b$. However, the inverse of the matrix modulo a word size prime p is quite large, occupying 20000*20000 words or 3.2

gigabytes when a 64 bit prime p is used. One of our implementations was run on a large memory machine and was able to use explicit creation and storage of this inverse matrix. The other implementation computes and stores only the minimal polynomial of A and uses computations with it to solve the necessary systems during lifting.

The solution turns out to be a quotient of integers of about 100,000 digits each. This problem is just on the edge of feasibility on current machines. A problem of size a small multiple of this would not be solvable due to time and/or memory resource limitations. Comparison of the three approaches to this problem is a good illustration of the capabilites of blackbox methods and of tradeoffs between memory and time demands of different approaches.

The determinant approach required computation modulo about 10000 primes. We made some experiments with Wiedemann's algorithm and found that it took approximatively 30 minutes to compute one minimal polynomial over a word-size prime field on a 1GHz PC. This means a total of about 450 CPU days were required for the complete computation. However, this took us only 4 days elapsed time of computation, using 182 processors (96 Intel 735MHz pentium-III, 6 Intel 1GHz pentium-III, and 20 sun ultra-450 at 4×250MHz. All the minimal polynomials are of full degree, therefore the modular answers are deterministic, and so is our integer computation.

The Dixon approach, when used with a similar size prime, requires about 20000 lifting steps. The blackbox approach (no explicit inverse) was not run to completion, but sufficiently many steps were run to project the total time. This implementation uses the LinBox field $UnparametricField < NTL :: zz_P >$ for a 30 bit prime and $UnparametricField < NTL :: ZZ_P >$ for the larger primes.

bits	k	times		
		minpolyA	one Hensel lift	k Hensel lifts
30	21569	17 min[1]	15 min[1]	224 days
120	5393	4 hr 15 min[1]	3 hr 8 min[1]	704 days
150	4314	4 hr 37 min[1]	3 hr 14 min[1]	581 days
240	2697	4 hr 37 min[2]	3 hr 10 min[2]	355 days
270	2397	5 hr 10 min[2]	3 hr 30 min[2]	349 days

This method as stated is not a parallel algorithm. However, it can be parallelized on n machines by choosing n primes $p_1, \ldots, p_n$. The i-th machine then computes ${A^{-1}}_{1,1} \pmod{p_i^{k_i-1}}$. The n solutions are combined using the Chinese Remainder algorithm to compute ${A^{-1}}_{1,1} \pmod{\prod_{i=1}^{n} p_i^{k_i-1}}$ and the solution is computed via continued fractions.

Finally the Dixon approach was tried with an explicitly computed inverse matrix modulo the 64 bit prime 1125899906842597. Then 13000 lifting steps were performed to produce a value modulo a 200,000 digit product of primes. From this the rational solution is easily obtained as described in [5]. This computation was run on a 750MHz SUN Ultrasparc with 8GB of main memory. It ran to completion in 12.5 days time. The computation of the inverse matrix required about 5.5 days and the lifting

[1] 750 MHz processor
[2] 1 GHz processor

steps about 7 days, which is less than one minute per lifting step. This method could also be parallelized albeit at a finer granularity. We have not yet implemented a parallel version.

More details on each method are in the poster. For the low memory demand, blackbox methods we see rough parity between use of one prime or many (CRA or lifting). The multiple prime approach is more easily parallelized. In this problem, however, the third, large memory method is considerably faster in CPU time.

References

[1] J.D. Dixon, *Exact solution of linear equations using p-adic expansions*, Numer. Math. **40** (1982), 137–141.

[2] Jean-Guillaume Dumas, Thierry Gautier, Mark Giesbrecht, Pascal Giorgi, Bradford Hovinen, Erich Kaltofen, B. David Saunders, Will J. Turner, and Gilles Villard. LinBox: A generic library for exact linear algebra. In *Proceedings of the 2002 International Congress of Mathematical Software, Beijing, China*. World Scientific Pub, August 2002.

[3] E. Kaltofen and B.D. Saunders. On Wiedemann's method of solving sparse linear systems. In *Proc. AAECC-9*, LNCS 539, Springer, pages 29–38, 1991.

[4] N. Trefethen. A Hundred-dollar, Hundred-digit Challenge. *SIAM News*, 35(1), 2002.

[5] J. v. Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, New York, NY, USA, 1999.

[6] D. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transf. Inform. Theory*, IT-32:54–62, 1986.