

Distributed Computation of Determinants with NTL and MPI*

Min Hur[†] (undergraduate student)
Email: hurmin@shu.edu

Manfred Minimair[†] (faculty adviser)
Email: manfred@minimair.org, WEB: <http://minimair.org>

Seton Hall University
Department of Mathematics and Computer Science
400 South Orange Avenue, South Orange, NJ 07079, USA

2/21/2005

1 Project description

The goal of this undergraduate research project is to efficiently compute determinants of square matrices with integer entries by utilizing computer networks and multi-processor computers.

As well known, determinants are fundamental in solving systems of equations. Furthermore, computing determinants is a task that is very demanding computationally. Therefore having efficient software for computing determinants is of practical importance for research and applications.

The motivation of the authors for developing efficient software for determinant computation originates in the research¹ of Manfred Minimair on efficiently computing (multivariable) resultants of polynomials with certain structures. Most current algorithms represent multivariable resultants or their multiples as determinants of matrices. Therefore efficiently computing determinants is important for efficiently computing multivariable resultants.

As a software environment for the project we chose MPI (message passing interface) with C++ and NTL². We chose MPI because it is a standard for

*Undergraduate research project

[†]Partially supported by the NSF grant CCF 0430741: RUI: Resultant Techniques for Composed Polynomials

¹See the following work and its bibliography: M. Minimair. Computing resultants of partially composed polynomials. In V. G. Ganzha, E. W. Mayr, and E. V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing. Proceedings of the CASC 2004 (St. Petersburg, Russia)*, pages 359–366. TUM München, 2004.

²V. Shoup. NTL: A Library for Doing Number Theory. Available at <http://shoup.net/ntl/>.

distributed computation available across many different platforms. We chose NTL because it is well-known for its efficiency and because its internal source code seemed very accessible.

2 Distributed Algorithm

The parallelization strategy for integer determinant computation is according to the divide-and-conquer scheme of the sequential algorithm contained in NTL. That is, integer determinants are computed modulo multiple primes and the modular determinants are incrementally lifted to the integer determinant by Chinese remaindering.

Let us elaborate on the details. In the distributed computation a master node first broadcasts the given integer matrix to various worker nodes. Then it sends distinct sets of primes to the workers. The workers compute the determinants modulo the primes they have received and send each modular determinant to the master as soon as they are finished computing it. The master collects the modular determinants and incrementally lifts them by Chinese remaindering. If the workers run out of primes the master sends more primes to the workers. The master stops the workers as soon as a termination criterion, contained in NTL, is fulfilled. The work load of the master node is insignificant as compared to the work load of the worker nodes. Therefore one can have the master node act as a worker node in order to achieve a greater speed-up. However, we also found it useful to leave the master node idle and use it to supervise and profile the distributed computation.

3 Results and future directions

We timed the program for distributed determinant computation on two different platforms, that is, a network of 8 IBM PCs and a shared memory computer with 4 CPUs. Summarizing, the speed-up performance, when compared to execution of the sequential algorithm contained in NTL, is quite substantial. For sufficiently large matrices (dimension at least 200) the speed-up is almost equal to the number of worker nodes. For example, with 7 worker nodes on our network of IBM PCs the speed-up is 6.9 for the computation of the determinant of a 2500-by-2500 matrix. The distributed program finishes in about 10.3 hours.

Future directions of this project include porting the program to other distributed platforms, such as a 16-CPU cluster computer and an 8-CPU distributed memory computer available to the project, efficiently computing determinants of sparse matrices, interfacing with current software for resultant computation³ as well as interfacing with related projects such as Prof. Arthur Chitcheba's project on interpolating parametric determinants.

³M. Minimair. MR: Macaulay resultant package for Maple. *ACM SIGSAM Bulletin*, 38(1):26–27, March 2004.