Large Numerical Linear Algebra in 1994: The Continuing Influence of Parallel Computing

Alan Edelman
Department of Mathematics
Room 2-380
Massachusetts Institute of Technology
Cambridge, MA 02139
edelman@math.mit.edu

Abstract

This note covers two aspects of the state of the art of large numerical linear algebra problems. Firstly, we look at the current records for sparse and dense linear systems and eigenvalue problems on a variety of machines. Our second subject matter is perhaps more of a question than an answer. Here we explore why network topologies of a parallel machine are hardly ever used in ways that perhaps a graph theorist might envision, especially given that linear algebra, particularly dense linear algebra, consists of many very regular predictable operations.

1 Introduction

This work represents in part a continuation of previous linear algebra surveys [4, 5]. My hope is to heighten awareness of relevant issues associated with numerical linear algebra that might be lost in the rush to write fast programs. Perhaps like the sports sections of any newspaper, I research the linear algebra records, discuss important issues associated with modern computing, but do not use this space to teach you how to play the game itself.

Section 2 of this note tabulates and discusses the latest records for large linear algebra problems. Section 3 asks why it is that some graph theorists would not recognize how machine topologies are used. A graph theorist who had never heard of a parallel machine would very likely devise a natural model of such a machine that would not be very applicable in modern parallel computing.

2 Linear Algebra Records

I have often expressed the feeling that nature does not seem to throw n^2 numbers at us haphazardly. Therefore sparse methods should be considered even for very large dense problems. Though this may seem contradictory, colleagues, students, and I are nevertheless interested in knowing the latest records for dense linear algebra problems. Though imperfect, these records do measure some notion of progress.

Furthermore, dense methods even for large problems continue to be used. Their value is comfortable reliability and predictability. Sparse methods for dense problems require research and expertise that may seem a luxury when the speed of execution is a lesser priority than having the code up and running. Of course for intermediate or small sized problems, it may well not be worth the trouble to switch to sparse approaches.

2.1 Dense Linear Systems

Curiously, in an era when laptops are obsolete one month after they are purchased, the record for the largest linear system by dense methods has hardly changed much in two years [5]. "Gigaflops" are now common on supercomputers, "teraflops" seem close. However, much to my surprise, I have no information that the world has seen the solution of a dense linear system of 100,000 equations in as many unknowns.

All of the systems below involve double precision complex numbers. Out of core methods use disk space as auxiliary memory. If your conception of a huge matrix is what a supercomputer can hold in internal memory, it may stretch your imagination to realize that the out of core problems described below only work on subblocks at a time.

Records for Various Machines

IBM (RS 6000)	20k $\times$ 20k	out of core
Intel (Paragon)	71k $\times$ 71k	out of core
Intel (iPSC/860)	75.2k $\times$ 75.2k	out of core
KSR	16k $\times$ 16k	in core
MasPar	43k $\times$ 43k	out of core
TMC (CM5)	76.8k $\times$ 76.8k	out of core

The problem solved on the Intel iPSC/860 was the record announced in [5]. That there is little improvement since then may just be a small blip in the overall trend, but this state of affairs begs to be explained. One might ask whether the state of knowledge of iterative methods is now so good that everyone has abandoned the dense approach for huge problems. This would represent a true milestone in numerical linear algebra, but I do not believe this has happened, and perhaps never will. The two explanations most worthy of consideration are the secrecy associated with why large dense systems are being solved and the limited disk memory available to those who responded to my electronic and telephone queries.

Large dense linear systems are being solved in secret. This secrecy might mean that larger problems are being solved, but the success is not publicized. The most predominant use has military applications. Further words about these and other application are discussed in [5]. The biggest consumers of large dense linear system cycles solve an integral formulation of Maxwell's equations exterior to a domain for the purpose of understanding radar cross sections. Alex Woo of NASA Ames has pointed out that for the practical problems that people wish to solve, the relevant equations are known as the Stratton-Chu equations. This can reduce to the Helmholtz equation in special cases that are useful for physical insight, but can represent an oversimplification that may not be correct. The method itself is known as the *method of moments.* Texts often refer to the EFIE and the MFIE, i.e. the electric and magnetic field integral equations, and the interest here is in solutions external to a region. Textbook discussions may be found in [12], Chapter 4 of [10] written by Poggio and Miller, and Chapters 5 and 6 of [13].

Those who did respond stated that the limitation has been disk memory. They did not have access to enough external memory to store bigger matrices. Thus the bottleneck is certainly not the algorithm and maybe it is not even the speed of the machine.

Regarding iterative methods for dense problems. The biggest success story may be Alex Yeremin who has solved 75k $\times$ 75k systems on a Cray Y-MP. He too ran up against difficulties with external storage. Also encouraging is work by Kane, Keyes, and Prasad [9] comparing iterative approaches to dense approaches for the boundary element problem. They report that preconditioned iterative approaches outperform dense approaches, but they did not test this result on huge problems. Their application is solid mechanics.

Ronald Klees of the GeoForschungsZentrum in Potsdam, Germany reported solving a 20,000 $\times$ 20,000 system in several minutes on a Fujitsu S600/20. He is also solving a boundary integral equation, though his field of application is Physical Geodesy.

Andrew Odlyzko has a record of his own. He will be solving a dense system of size 200,000 $\times$ 200,000. Why is this not the record? His matrices only have 0's and 1's, the system is solved modulo 2. When he is done he will be able to factor the 129 digit cryptography challenge integer.

Lastly, there are a number of users interested in large dense problems with applications to scattering. This is probably an imperfect understanding, but I think that these problems are dense because theoretically input from any direction may be scattered into all other directions. Jussi Rahold from Espoo, Finland mentioned the scattering of light by dust particles. The matrix was of size 412,128 using conjugate gradient. Alfons Hoekstra from Amsterdam solved a problem of size just over 100,000 in order to understand elastic light scattering. He also used conjugate gradient. Tim Kelley of North Carolina State sent me a paper where the solution of a dense problem of size up to 3,100,000 using GMRES and multilevel approaches. His paper concerns transport problems in one space dimension. I believe that this also may be thought of as a scattering problem, but I have not verified this. An earlier dense scattering problem is described in my previous survey [5].

2.2 Dense Eigenvalue Problems

I fear that I have less information on the dense eigenvalue problem. So far as I am aware, the records have not increased for the dense eigenvalue problem either. The record for the symmetric dense problem including eigenvectors must still be 27,000, while for the unsymmetric problem it is 10,000.

The largest symmetric eigenvalue problem computed on the CM-5 was performed by Jean-Philippe Brunet of Thinking Machines Corporation on a 16k $\times$ 16k matrix. Furthermore, he is interested in computing the eigenvalues of a $3k \times 3k$ double precision complex non-Hermitian eigenvalue problem that arises from the discretization of the Dirac equation. His near term goal is to extend this to order 15k. The matrices are actually sparse, but to his knowledge, sparse methods were only successful in extracting the extreme eigenvalues. These eigenvalues will be used to construct preconditioners for a large linear system that arises in computational Quantum Chromodynamics.

Anna Tsao at the Supercomputing Research Center in Bowie, Maryland is interested in symmetric dense eigenvalue problems of size bigger than 10,000, but has not yet performed any computations on problems of that size.

2.3 Sparse Linear Systems

This is the first time I have begun asking about sparse systems so the information included here is probably less complete than for dense problems. Nevertheless, I wanted to get some feeling for whether the largest problems were being solved by direct methods, iterative methods, or some kind of mixture.

2.3.1 Direct Methods

My information on the record holders for sparse direct methods is mostly anecdotal evidence. The general opinion among experts is that problems of size 300,000 are certainly being solved, and probably much larger. As we will see in the next subsection, probably the problems that people are solving using iterative methods are at least three times larger. Given that my information is incomplete, and also that different problems require different solution methods, it would be inappropriate to make too hasty judgment about which methods are "better."

Steve Zitney at Cray Research reported his interest in plantwide dynamic simulations of a distillation facility. He hopes to be solving simulation matrices involving "hundreds of thousands of equations" very soon.

2.3.2 Iterative Methods

Steve Ashby at Lawrence Livermore Laboratories reports solving sparse systems with one million unknowns using preconditioned conjugate gradient. He is currently working on using the multigrid algorithm as the preconditioner. He claims that one million is fairly routine. The application is three dimensional modeling of groundwater flow through heterogeneous porous media. His current goal is to solve problems with 100,000,000 unknowns.

Though this is not the solution of a linear system, Louis Howell at Lawrence Livermore told me about Paul Woodward at the University of Minnesota who demonstrated a showpiece fluids simulation on a cluster of SGI workstations. What is impressive is that the calculation was on a $1024 \times 1024 \times 1024$ box, i.e. over a billion variables. Howell concludes, correctly I believe, that elliptic problems (i.e. solutions of linear systems) can not be far behind.

2.4 Sparse Eigenvalue Problems

The ultimate record in this entire paper is the computation of the smallest eigenvalue and the smallest eigenvector of a billion by billion matrix performed by Simons, Jorgensen, and Olson. This was reported to me by Frank Jensen in Denmark. The method used was a variant of the Davidson Method. The application area was the Configuration Interaction Method for electronic structure computations.

Ron Shepard also uses the Davidson Method to solve problems of size up to 10,000,000 and has mentioned benchmark calculations of size 100,000,000 performed by other researchers on an Intel Touchstone.

Tucker Carrington of the University of Montreal reports that he has solved a one million by one million problem using the Lanczos method in a chemistry application.

Albert Galick of Computational Electronics has solved a nonsymmetric eigenvalue problem for six eigenvalues. His largest problem was of size around 500,000 using an iterative Chebyshev-preconditioned Arnoldi method. This took several days on an Ardent Titan.

Also noteworthy is the nice eigenvalue survey by Bai [1] which concentrates more on sparse problems and smaller dense problems than I have been considering in my surveys. His survey is worth reading, but I have not been able to make use of it because he does not explicitly mention that any dense applications require more than 10,000 variables or that the sparse applications require significantly more than 100,000 variables.

3 The BLAS, the BLACS, and the outlawing of graph theory

I now turn to the issue of communication on the networks of a multicomputer. The issue that I wish to raise here is what constitutes an aesthetically elegant use of network topologies, and by comparison what are the realities of current supercomputing and their implications for linear algebra and more general computations. Perhaps it is best to say that I suffer from an inner conflict between what the mathematician inside of me feels ought to be the way parallel machines should be, and what my engineering colleagues tell me how parallel machines must be, at least for now.

3.1 Mathematical elegance, couch potato messages, and linear algebra libraries

My notion of what a parallel machine for numerical linear algebra (and other disciplines) ought to be is based on 1) mathematically natural notions, 2) the history of the Basic Linear Algebra Subroutines, and 3) experiences with the CM-2 supercomputer that were never publicized. What each of these three aspects have in common is a reasonably clean abstract world where the user who wishes to use his rational faculties to directly improve the speed of algorithms is allowed to do so with a reasonable ability to predict the fruits of his/her labors. The real world of computing, by contrast, is not so tidy and many experts say that it must not be.

What constitutes natural to a mathematician is easy to explain:

Definition 3.1 *The* graph theorist's natural model *of a supercomputer network allows you to imagine that every time you snap your fingers, a fixed packet of data is allowed to cross exactly one link, and all the different links may be used simultaneously. What matters is the amount of data and the number of hops the data must make.*

This is the kind of network on which you can do precise mathematics and prove exact theorems. Indeed much very sophisticated work has gone on in this area.

In addition to the graph theorist's natural model, I would also like the ability for the network to allow *couch potato messages:*

Definition 3.2 *A network allows for* couch potato messages *if neither processor address information nor any pointer to a program residing in memory need be sent with the message.*

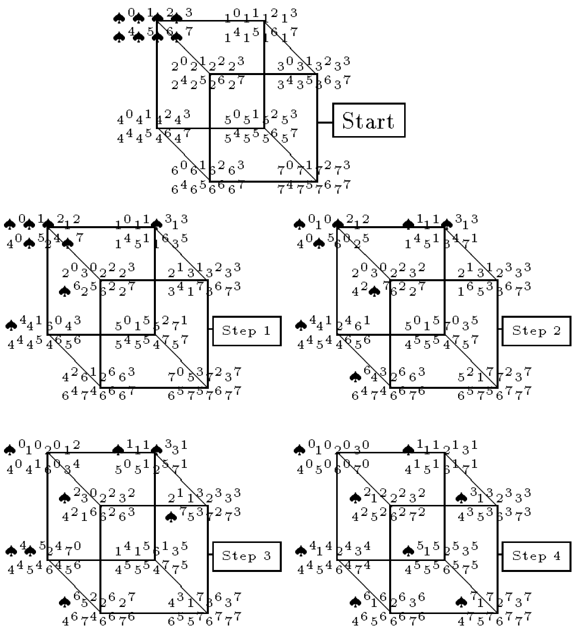
Most current networks send destination addresses along with messages. The active messages approach devised by Dave Culler of UC Berkeley suggests sending a pointer to a program location as well. The couch potato messages proposed here differ in that they are pushed around from point to point. Rather than actively knowing where they are going or what is going to happen to them when they arrive at their destination, handlers at intermediate nodes and destination nodes are ready for them and tell them what to do. One important aspect of this approach, is that messages may be scheduled aand communication costs need not be increased by sending address bits.

What ought to seem natural for linear algebra libraries requires understanding how libraries are constructed. LAPACK and its precursors make use of a set of Basic Linear Algebra Subroutines (BLAS) that may be written in a machine specific manner at the assembly language level though sample BLAS are written in FORTRAN so that untuned programs may still run on the various machines. It is well understood that the operations in dense linear algebra are so regular and predictable, that it is not unreasonable to sacrifice some people-hours to the task of fine tuning these operations for everyone's benefit. Indeed dense linear algebra is about as regular and predictable as programming can get; it is no wonder that the developers of LAPACK request that manufacturers fine tune the BLAS. The management of registers, cache, and floating point operations is often well enough understood by specialists that optimal or near-optimal management of resources may be obtained.

On a multicomputer, memory is distributed so that a communications network must pass messages among the individual processors. Following the lesson from the previous paragraph, it seems natural that there ought to be a collection of Basic Linear Algebra Communications Subroutines (BLACS). It might also seem natural, unless you know too much about current multicomputers, that these BLACS would be very much on the same level as the BLAS. They would be written in assembler by the manufacturers of a number of machines by carefully orchestrating the communication's links. One would expect that the regularity and predictability of dense linear algebra would almost demand such approaches. This was my vision of the BLACS in 1989. Yet the BLACS as they currently exist as part of the new scalable ScaLAPACK have never been written in assembler and few clever mathematical graph theory papers have proven appropriate for these BLACS on any real parallel machines. The BLACS as they exist are calls to a network that will do the work, they do not explicitly manage the network in the sense in which the BLAS manage the registers, cache, and memory. If the BLACS were on the same level as the BLAS, network resources would be carefully managed inside the BLACS. Manufacturers would fine tune the scheduling of messages according to fixed guidelines.

There are at least two ways to use the control provided by the graph theorist's natural network and also the couch potato approach to messages. There are other possibilities not explicitly discussed in the paper. One way is to orchestrate the messages yourself through the network; a second way is to give compilers the smarts to precompute a schedule.

To illustrate the first approach, orchestrating messages through the network, consider the following picture of so-called "all-to-all personalized communication" or "total-exchange" from [6] that was used by Steve Heller as a kernel in the "twuffler" project at Thinking Machines Corporation. The twuffler project began when it was realized that software was sufficient to accelerate IO communication, i.e. no new hardware was needed. Indeed plans to build new hardware were dropped. In the picture below, all the hypercube links are used all of the time to obtain maximal use of bandwidth.



The second approach, automatic scheduling by a compiler, was taken by Denny Dahl of Thinking Machines Corporation [2], Steve Hammond of NCAR [7, 8], and Shukla and Agrawal [11]. There is an important distinction that should be emphasized. The first approach described above, i.e. user orchestrated patterns, is appropriate for very regular communication patterns. Such patterns arise in dense linear algebra problems. By contrast, automatic scheduling by a compiler is what is needed for unstructured grid communication.

3.2 Misconceptions about the Connection Machine CM-2

Engineers say that what I am envisioning can not exist, or if it could exist it would not be useful. There is actual historical evidence that careful control of a machine's network can lead to improvements. Unfortunately, this evidence was never properly understood by network designers. The historical evidence is based on somewhat secret successes obtained by scheduling messages on the Connection Machine CM-2 [3].

Unless you worked closely with a very small number of people inside of Thinking Machines Corporation, you might never have known that the graph theorist's natural model of the hypercube was *not* designed into the CM-2, but we later discovered that hardware designed to accelerate data paths to the floating point units actually served remarkably well as handlers for routing. Those familiar with the distinction between the fieldwise model and the slicewise model of the CM-2 will understand that the fieldwise model was not quite as flexible for hypercube communication as was accidentally true for the slicewise model.

This discovery began with work by Steve Vavasis of Cornell University then further developed by Mark Bromley, Steve Heller, and myself, who showed that it was possible to apply the graph theory of the hypercube in the mathematically natural approach described above by using hardware for purposes other than that for which they were designed! This mathematical abstraction was very close to what the machines was really doing. Careful management of the data often lead to 100% utilization of the hypercube channels. While many people were promoting load balancing of the processors, we were pushing "All the wires, all of the time." In our work, no address bits were sent, and latency was not a big issue.

Unfortunately this lesson was lost, partially due to secrecy, partially because Thinking Machines already made the commitment to the CM-5 architecture in which control of the communications network is more limited and less tidy, and finally, maybe, it just was impossible to keep it in an asynchronous environment. Perhaps architects did not believe that some users might sometimes want this control, even if most users would not. Since I am not a computer architect, I could not tell you for sure.

The CM-5 as it currently exists contains a random number generator that decides which of two directions a path should take when going up the tree at each level other than the lowest. Thus, any opportunity to control messages is mostly lost. There are some tricks that provide some level of control, such as careful injection of messages into the network. Perhaps this is unfair, but sometimes this seems to me like controlling Boston traffic by putting traffic controls at the city limits.

3.3 What computer architects say:

I admit that I am not a specialist in computer architecture, so I asked some of my colleagues at MIT who are. I also asked for comments from leading network architects from outside of MIT, but I have yet to receive a response.

Bill Dally at MIT believes that it is not a good idea to give linear algebra library writers (or anyone else) control of the communications network. According to Bill Dally, a good network provides throughput for random traffic that is usually around 30% of peak capacity. Adaptive routing can increase this to 50% of peak capacity. The belief is that it is far better to devote resources to making a good network faster than it is to devote resources to specialize the control.

Arvind tends to agree, but he points out that exposing the underlying hardware to hard core programmers and compiler writers has often lead to better performance. He is open to the possibility that perhaps a day will come when specialized programmers would have access to the underlying communications protocols though this is not how most programmers would use the machine.

Also at MIT, Steve Ward is explicitly looking at scheduled routing in the NuMesh project.

4 Conclusions

The linear algebra records in this paper will of course change with time, and as I receive new information. Hunting down these records is not easy, but the information benefits everybody.

I also presented the view that communications networks are neither what mathematicians might expect nor what should be expected for linear algebra libraries since intrepid programmers who wish to work at the lowest level are unable to fully control the network.

At the current time the BLACS and BLAS are fundamentally different. The BLAS may be written in a careful manner to take advantage of computer memory and orchestrate the data movement between main memory, cache, and registers. The BLACS are currently requests for the network to do the work. Perhaps this is as it must be in the complicated asynchronous world of MIMD supercomputing, even in the regular context of dense linear algebra. Perhaps it must be this way but part of me wishes that it were not so. We will see what the future brings.

Acknowledgements

Many experts who are quoted here communicated their information either by electronic mail, telephone, or face to face conversation during the early part of 1994. I very much appreciate all of their information. A followup paper will be submitted to a refereed journal which will contain proper bibliographic entries, where available, for all of the information included here. Discussions with so many colleagues have very much influenced the thoughts in this paper. Of particular note are Steve Hammond at NCAR and Katherine Yelick at UC Berkeley.

References

[1] Z. Bai, Progress in the numerical solution of the nonsymmetric eigenvalue problem, *J. Numer. Lin. Alg. Appl.*, to appear.

[2] E.D.Dahl, Mapping and compiled communication on the Connection Machine system, in *Proceedings of the Fifth Distributed Memory Computer Conference, D.W. Walker and Q.F.Stoud, eds.*, IEEE Computer Society Press, Los Alamitos, CA, 1990.

[3] A. Edelman, Guess what? We have a hypercube, Thinking Machines Corporation internal report, 1989.

[4] A. Edelman, The first annual large dense linear system survey, *SIGNUM Newsletter, 26* (October 1991), 6–12.

[5] A. Edelman, "Large dense numerical linear algebra in 1993: The parallel computing influence," *Journal of Supercomputing Applications. 7* (1993), 113–128.

[6] A.Edelman, "Optimal matrix transposition and bit reversal on hypercubes: All–to–all personalized communication," *J. Parallel Dist. Comp. 11* (1991), 328–331.

[7] S.W.Hammond and T.J.Barth, An efficient massively parallel Euler solver for 2-d unstructured grids, *AIAA 30* (1992), 947–952.

[8] S.W.Hammond, *Mapping Unstructured Grid Problems to Massively Parallel Computers*, PhD dissertation, Rensselaer Polytechnic Institute, May, 1992. Also available as RIACS technical report TR92.14.

[9] J.H.Kane, D.E.Keys, and K.G.Prasad, Iterative solution techniques in boundary element analysis, *Intl. J. Num. Methods Eng. 31* (1991), 1511–1536.

[10] R.Mittra, ed., *Computer Techniques for Electromagnetics*, Pergamon Press, New York, 1973.

[11] S.B.Shukla and D.P.Agrawal, Scheduling pipelined communication in distributed memory multiprocessors for real-time applications, in *Proceedings International Symposium on Computer Architecture 1991*, Association for Computing Machinery, New York, 1991.

[12] J.A.Stratton, *Electromagnetic Theory*, New York, McGraw-Hill book company, 1941.

[13] J.J.H.Wang, *Generalized Moment Methods in Electromagnetics*, New York, Wiley, 1991.