SUPERCOMPUTING AND NUMERICAL RELATIVITY: A LOOK AT THE PAST, PRESENT AND FUTURE

David W. Hobill and Larry L. Smarr
National Center for Supercomputing Applications
605, East Springfield Avenue
Champaign, IL 61820

Abstract. Reviewing the issues in supercomputing applications that are relevant to numerical relativity, we discuss the present state of the art of computing solutions of the Einstein equations. In particular we look at how recent developments in hardware and software have influenced the maturation of numerical relativity. Special emphasis is placed upon the contributions that are being made by symbolic manipulation and scientific visualization to the theory of general relativity.

I. INTRODUCTION

In the past few years, numerical relativity has witnessed a rapid growth not only in the number of researchers in the field, but also in the scale and difficulty of the problems being attacked. As analytic methods for studying the Einstein equations become exhausted, numerical methods will likely increase in importance, particularly as observational general relativity reaches its maturity and measurements of the relativistic gravitational field become possible.

Numerical relativity, and computational physics in general, can be considered as new mode of scientific inquiry, augmenting the already well established modes of theory and observation/experimentation. It is computation that takes a theory, replaces the spacetime continuum with a finite lattice, calculates derivatives with finite differences and, using numerical methods, simulates a phenomenon that represents some aspect of physical reality. As in any scientific endeavor, measurements and observations are still needed to verify the underlying theory.

The new synergism that exists between theory, computation and experimentation, is the basis of a number of important collaborations. In general relativity, perhaps the best known collaboration exists in the analysis of gravitational radiation. Theorists have long been interested in understanding the generation, propagation, and detection of gravitational waves. However their work has generally been based upon simple idealized models of sources and detectors. Numerical models of realistic gravitational wave sources and detection devices will provide the experimentalists with more precise information concerning the energy and signatures of the gravitational waves that reach an earth-based observer from violent astrophysical events. Similar collaborations also exist among cosmologists who studying the physical processes that occurred during the first moments in the life of our universe try to determine how they may be able to explain the observations that are made today.

Phase 1. In spite of the fact that such collaborations amongst relativists have been formed, they have not reached the same point of maturity occurring in some other realms of physics such as fluid dynamics or condensed matter physics. Numerical relativity is still at an early stage in its development compared to these fields. Long before

electronic computers could realistically attack problems in numerical relativity, a few far-seeing individuals realized that numerical methods would be useful for solving the Einstein equations. This prophetic age, which began with the work of Lichnerowicz in 1944 and continued with further expositions by DeWitt (1957) and Misner and Wheeler (1957), established that the structure of the Einstein equations was amenable to computational methods.

Phase 2. Somewhat later a second stage of development began with pioneering efforts dedicated to attacking problems that at one time were considered intractable. Often problems are attempted long before the technology and methodologies are capable of providing the solutions to those problems and numerical relativity found itself in this situation at its birth. In general relativity, this heroic period began in the 1960's with the attempts by Hahn and Lindquist (1964), Cadez (1971) (the two black hole collision problem), May and White (1966) (spherical collapse) and Pachner (1973) (rotating collapse). Only the May and White code succeeded in obtaining the goals it had set out to reach. Later, successes were met in two dimensional vacuum solutions by Smarr (1979) and Eppley (1979) who completed the evolution of two colliding black holes and calculated the dynamics of Brill waves. At about the same time, one dimensional problems were extended to cylindrical [Piran (1979)] and planar [Centrella (1979)] symmetry, and 2-D hydrodynamic solutions were obtained by Wilson (1979). All of these codes were either highly idealized (1-D) or run on crudely zoned grids (2-D) since the computational power available at that time was insufficient to perform large-scale computations. Subsequently, other researchers entered the field and efforts to push the frontiers of numerical relativity further have continued to the point that now three-dimensional problems are being considered (see for example the articles by Nakamura and Stark in this volume).

Phase 3. While all of these ambitious explorations of unknown territory were being carried out, the third phase of numerical relativity began. Its purpose is to construct more accurate and stable codes capable of re-attacking those problems that now seem less intimidating. In some cases we have seen a return to one dimensional problems which can be performed with such an increased accuracy that these codes nearly rival the accuracy of codes designed to solve ordinary differential equations. In addition, analytic methods play an important role in this third phase. Not only are they being used to test and verify the results of a numerical calculation, but they are also being directly used in conjunction with the numerical calculations to produce self-consistent results. This third phase of computation is more than just a development of algorithms aimed at refining the results of previous work. It also is an exploration of some fundamental physical concepts and theories. For example in this volume we see one dimensional codes leading to new a understanding of the behavior of inhomogeneous cosmologies (Anninos et. al. and Kurki-Suonio), cosmic censorship (Shapiro & Teukolsky and Choptuik), and gravitational collapse to form star clusters and black holes (Shapiro & Teukolsky). Since each of these subjects requires high accuracy in order to precisely measure subtle effects that often escaped earlier investigators, a more careful treatment of these problems with better numerical methods is necessary.

As more sophisticated methods become commonplace the emphasis of the third phase will be focused on the solution of realistic problems with a complexity so great that computer methods are the only means of linking the theory to experiment. The field of hydrodynamics provides a good example of how complex physical reality can be, and no less will be expected of numerical relativity. General relativity has yet to reach the stage where it can simulate a large variety of complex phenomena and then correlate its conclusions either with measurements made in the laboratory or with observation.

Experimental relativity is presently undergoing its own parallel evolution. By the 1990's it is expected that these two modes should reach their maturity almost simultaneously, thereby ushering forth a new era of increased interest in general relativity.

In this paper we will outline some of the developments in computational relativity that have occurred as a result of the introduction of supercomputers. The term supercomputer in this article will always refer to those machines with the largest memories and fastest processors available. These machines are capable of solving problems that were once considered intractable on any previously available computer. We shall also discuss certain issues that are now being addressed by numerical relativists in order to ensure the continued growth of the field. While some examples of progress that have been made in our discipline will be presented, these proceedings speak for themselves concerning the present and future frontiers of numerical relativity.

II. SUPERCOMPUTERS AND THEIR IMPACT ON NUMERICAL RELATIVITY

1960's. In 1966 May and White performed the first successful 1-D general relativistic evolution on a computer (mostly on a CDC 3600). This was a spherically symmetric Lagrangian code that computed the gravitational collapse of a fluid sphere. The choice of time gauge was such that it could not avoid the central singularity that formed during a collapse to a black hole and therefore the code was limited to the hydrodynamics involved in neutron star formation.

1970's. Two dimensional numerical relativity met with its first successes on the CDC 6600 [Smarr (1975)] and eventually CDC 7600 [Smarr and Eppley (1979) and Wilson (1979)] machines. These were the first computers with speeds and memories capable of carrying out elementary finite differenced calculations on grids fine enough to resolve the relevant length scales in a 2-D general relativity problem. The core memories (on the order of 10's of KWords) of these machines were still small compared to what was needed in numerical relativity so ingenious operator splittings often had to be performed to take advantage of what memory was available. In addition the need to transfer data back and forth to rotating disk or magnetic tape introduced considerable overhead in the coding. Numerical relativity required an inordinate amount of patience and perseverance and only a few diehard computer specialists were willing to put up with the idiosyncrasies of the supercomputer of that era. Lack of access proved to be even a greater problem. There were few machines capable of performing numerical relativity codes and they were not readily available to academic researchers. As a result the number of practitioners of numerical relativity remained small.

1980's. Both the increased speed and memory of the Cray supercomputers, have relieved many of the frustrations associated with having to deal with time and memory constraints. For those codes that still require more memory and must use external storage devices, the new solid state disks can transfer data at a rate of 125 times faster than a rotating disk and can be treated almost as an extension of main memory. Computers with memories of hundreds of millions of 64-bit words are now available making older problems more tractable and solutions to new and more difficult problems possible. Table 1 presents the core memory and speed of those machines that have contributed to the development of numerical relativity in the past two decades.

Supercomputer Capabilities. In general the evolutionary pattern has been to simultaneously increase both speed and memory. The scalar speed up of the fast

processors in the CDC 6600 meant that the results could be produced in a reasonable amount of time. The vector processing of the CDC 7600 with even faster CPU's made the speed up factor even greater. However vectorization required some changes in the way codes were written. The Cray-1 continued this trend and in addition made more memory available. Multiple processors became available on the Cray-XMP and the YMP will extend this number even further. Finally the Cray-2, while not significantly increasing the speed and number of processors compared to the Cray-XMP, will by virtue of its large memory open the door to more detailed calculations in numerical relativity. Along with the availability of powerful workstations that act as an intelligent interface between the user and the supercomputer, many of the earlier limitations to large-scale computing have been eliminated and new researchers are now entering the field.

date first used	computer	core memory (Words)	Peak Mflop rating	estimated time to complete a 2-D calculation
1964	CDC 6600	40KW	7M	1 week
1969	CDC 7600	65KW	30M	4 days
1978	Cray-1	2MW	160M	10 hours
1984	Cray-XMP	8MW	200M*	7.5 hours
1987	Cray-2	128MW	2000M**	1 hour

Table 1. A comparison of different supercomputers used for numerical relativity codes. The peak megaflop ratings are obtained from the literature supplied by different vendors and may vary depending upon the codes used. * Most XMP's are used in single processor mode. **Assuming code uses four processors.

The computationalist must still deal with the peculiar characteristics of particular hardware and software implementations, especially if they are dependent upon the latest developments in technology and methodologies. Like an experimentalist, the computationalist has an apparatus with which he/she must interact and continually coax to perform in the manner intended. It is unlikely that the forefront of numerical relativity will ever be 'user friendly', but advances are always being made to ensure that it will not be entirely user abusive.

Increased Complexity. Supercomputers present the researcher with the opportunity to increase the complexity of the problems that are now being attacked. In general relativity this complexity can occur in many ways. The most obvious among these are the use of locally finer grids and spatially larger domains, the extension of a particular model to higher dimensions and the addition of more realistic physics in the set of equations or initial data. The increased demands on computer memory and execution time increase geometrically with an increase in any one of these realms.

Grid Refinement. Augmentation of both the local spatial and temporal resolution of an existing code can lead to a better correspondence with analytic behavior. If for example a particular second order accurate finite difference method is employed to approximate the Einstein equations, then decreasing the grid size by a factor of two increases the accuracy by a factor of four. This will often (but not always) be enough to reproduce either the behavior of an analytic solution or the results from a simpler numerical test-bed calculation. An example of how a simple increase in the fineness of the numerical grid proceeds to correctly reproduce the known analytic results is presented in these proceedings by Bernstein, et. al. Unfortunately, for a given N-dimensional time-explicit evolution, a grid refinement by a factor of two requires a 2^{N+1} increase in computational time.

Domain Enlargement. The spatial extent and length of time over which the evolution is calculated is also an important factor in increasing the accuracy of a calculation. Outer boundary conditions for asymptotically flat spacetimes will always be more accurate if they can be posed at large distances (ideally, infinitely far) from compact sources with relativistic motions and/or regions of strong gravitational fields. With an increase in the spatial extent of the problem, longer time evolutions are necessary. Since one is not refining the Courant limit zone, the increase in computational time is less than that discussed above.

Higher Dimensions. The complexity that arises in the passage to a higher spatial dimension from a lower one has even a more profound effect. First, there is the obvious increase in the number of grid points that are the consequence of going to higher dimensions. Second, the increase in dimensions is accompanied by a larger number of non-zero metric coefficients which translates into a need to calculate and store more variables at each grid point. The equations and expressions appearing in them are more complex simply due to the increase in non-zero variables. While in principle the Einstein equations for a completely general metric can written on a finite number of pages, the computer used to construct a numerical solution may be unable to store all of the variables at enough grid points to make such a calculation meaningful. Finally, physics in three dimensions is significantly more complex than it is in fewer spatial dimensions and finer grids may be necessary in order to resolve some of this complexity. Some problems still need to be delayed until technological gains (primarily associated with available memory) can be made.

Variable Increase. An increase in variables is a result not only of increasing the number of spatial dimensions, but is also a consequence of the decision to employ auxiliary variables and more complicated gauge choices. Until recently, most researchers have followed the Wilson (1979) suggestion of keeping the number of variables in the three metric to a minimum through choices in the shift vector. However more geometric and natural choices of shift vectors do exist [Smarr and York (1978)], but they increase the number of elliptic equations that need to be solved at each time step and do not eliminate metric variables. The freedom to use alternative gauges has yet to be explored and it is quite likely that in the future, gauge choices will be made not for computational ease but for more fundamental theoretical reasons. There is still much work to be done in this relatively untouched area of research.

Algorithm Development. Algorithm development often leads to increased complexity in a numerical calculation. Numerical hydrodynamicists in their search for greater accuracy and stability, have been led to the construction of more complex and more elaborate numerical methods that yield more accurate solutions of the partial differential equations that govern their simulations [Woodward (1988)]. Often new and more complex algorithms for calculating a solution at a fixed order of accuracy will result in more stable methods that allow for longer time evolutions to be calculated. Rather than provide specific examples of algorithm development here, we shall consider this issue within the context of a general discussion on software.

Figure 1 provides an example of the progress that is a consequence of using finer grids, larger spatial domains and more accurate and stable algorithms. In this graph the value of the lapse function at the inner most grid point is measured for maximally sliced spherically symmetric black holes. Analytically, the lapse function at the throat of such a black hole has a time dependent behavior that falls off exponentially ("collapse of the lapse") with an e-folding time of 1.82M where M is the mass of the black hole [Smarr &

York (1978)]. The values of the lapse shown in Figure 1 represent the largest absolute value of the logarithm of the lapse that still obeys the analytic behavior. Without an increase in computing power some of these developments could not have occurred.

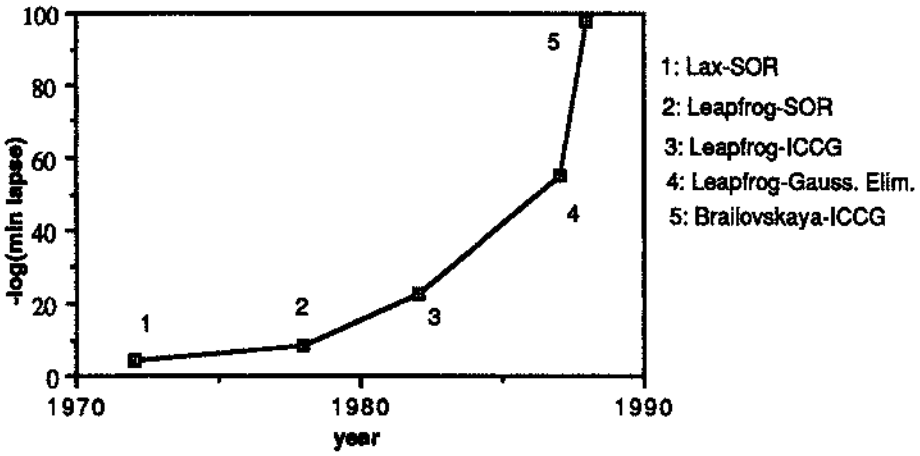


Figure 1. A graph of the increase in accuracy obtained with increased computing power and algorithmic development. The data points were obtained from the following papers: 1; Estabrook, et. al. (1973), 2; Smarr & York (1978), 3; Evans (1985), 4,5; Bernstein, et al. (this volume). The first two calculations were performed on a CDC 7600 and all subsequent results were obtained on the Cray X-MP.

More Physics. Nature itself presents us with an increase in complexity that is far more interesting but also more difficult to understand. Thus far we have discussed the more mathematically quantifiable or predictable increases in complexity (increases in memory requirements) but the introduction of a realistic source to the right hand side of the Einstein equations will produce consequences that should not be underestimated. Already we have seen from computations made in the simpler newtonian or special relativistic regimes, that hydrodynamics and many-body problems are subject to complicated and often unexpected behavior. If these systems act as sources and are coupled to nonlinear gravitational fields, it can be expected that the general relativistic systems will be no less complex. New methods of studying the behavior of such a complex physical system with many degrees of freedom will have to be developed in the future.

II. HARDWARE ISSUES

Supercomputing Demands. From the point of view of computational science as a whole,the hardware requirements of numerical relativity has yet to make a major impact in the field. Its computational demands are relatively small compared to computational fluid dynamics, computational chemistry and lattice gauge theory to name a few other disciplines that utilize the same resources. The largest two dimensional relativity code (written by Roger Ove) requires 4 million (64 bit) words of Cray-XMP memory and an additional 17 million words on the SSD (solid state disk). This is a fine resolution (200x500) code and requires between 2 and 3 CPU hours to run about 2000 time steps. Nakamura's three-dimensional code with 80^3 grid points requires 25 (64 bit) Million words of core memory. These examples are to be compared to some stellar evolution codes that need hundreds of millions of words of memory.

In a list of all of the 37 accounts on the NCSA Cray that have used 300 hours or more of computer time, numerical relativity ranks 25th and 29th in overall usage. Comparing the account with the largest usage (lattice gauge theory with 11,433 hrs) to the largest numerical relativity account (388 hrs), provides yet another demonstration of numerical relativity's relative rank as a consumer of supercomputer cycles. Numerical relativity is is still a young field, but it is expected that the 3-D codes will eventually place heavy demands upon computer power, especially when one considers the large number of variables (50 to 100) that need to be stored on a numerical grid fine enough to resolve dynamics at widely differing length scales.

An example of how important supercomputers are to numerical relativity in terms of productivity is presented in Table 2 where the execution times for different machines is compared. The code used to provide the benchmark was a one-dimensional predictor-corrector code for a maximally sliced spherically symmetric black hole (see Bernstein et. al.) The radial grid contains 1000 nodes and 20000 time steps were computed during the entire evolution. The code was written to be entirely portable so that system dependent instructions were eliminated from the very start. This allows coarse grid runs and code editing to be performed on local workstations leaving only the fine grid production runs to be performed on supercomputers. From the timing results, it is clear that if computational science is to be considered as 'experimental theory' then the benefits of fast machines (and turn-around) are obvious.

Computer	Processors	Mode	CPU Time (sec)	Ratio
Sun 3	1*	Scalar	586800	3738
Vax11-785	1	Scalar	317580	2023
Alliant FX/8	8	Vector/Parallel	5152.9	32.8
Cray-XMP	1	Vector	483.3	3.1
Cray-2	1	Vector	520.5	3.3
Cray-2	4	Vector/Parallel	157.6	1.0

Table 2. A comparison of CPU times to calculate the maximal time slices to 100M for the Schwarzschild solution (20000 time steps) with 1000 spatial nodes using a Brailovskaya finite differencing scheme to compute the evolution equations and shooting to solve the two-point boundary condition problem for the lapse function. Memory requirement: 1.1MWords. *Without floating point accelerator.

Supercomputer Upgrades. At the present time, the eight million word Cray XMP-48 (and SSD with its 128Mword memory) provides enough computing power and memory to perform most 2-D calculations. The Cray-2 will have a maximum memory of 512 million words (the NCSA Cray-2 will be smaller with 128Mwords) and this increase by a factor of sixty-four will make three-dimensional relativity a real possibility. One Giga-word machines will be available by 1990 (the Cray-YMP will have a 1Gword SSD by late 1989). If one extrapolates the 1-D results of Bernstein, et. al., then in order to obtain 3-D results with an accuracy of less than one per cent, a machine with approximately 20Gwords of memory will be needed. A computer with a memory of this order of magnitude may not be available until about 1995. It will be needed, however, if experimental and numerical relativity are to simultaneously open the door to the new field of gravity wave astronomy.

Parallel Supercomputing. Another critical hardware issue is that of parallelism. Almost all of the numerical relativity codes to date have run on single processor computers. However during the next five years a wide variety of parallel machines will become available and their use will be essential to achieve the speed-ups needed for frontier numerical relativity codes. This will require a whole new generation of codes that take

advantage of parallelism. At the present time most parallel supercomputers use a small number of processors (<16) with a shared memory. Programming these machines is still an inherently serial process using loops to sequentially calculate portions of a system that have been split off from the whole. This is contrary to the idea that in a 3+1 decomposition of spacetime an entire spacelike hypersurface is updated simultaneously from local information at each grid point. Most of the equations of theoretical physics are partial differential equations that lead to explicit finite differenced equations with nearest neighbor interactions. Consequently, few if any data dependencies need appear in the algorithms that exploit the locality inherent in physical systems.

With a small number of processors, the operations occurring at each grid point cannot be calculated simultaneously. The operations must be distributed among the available processors until the entire loop is completed. If, on the other hand, there are as many processors as nodes in a finite differenced version of a PDE (each with their own local memory and capable of communicating with its nearest neighbors), then the loop structure and operator splitting is superfluous since each node performs its operations using information obtained from its neighboring nodes and itself.

The Connection Machine with its 65,536 processors pursues this philosophy with its 'news' (North-East-West-South) grid, and the get_from_news operation that in assembly language (Paris: an acronym for parallel instruction set) passes information from one node to another. Unfortunately in the connection machine the processors at each node only have 8Kbytes of memory, but it does not take a great deal of imagination to consider the possibilities that can arise from the use of processors with a larger memory at each node. One step in this direction is the MYRIAS machine that will use 1024 Motorola 68020 chips each with 4Mbytes of memory. However this still only allows a 30x30 2-D grid if each processor acts as a grid point. It is likely that this interim solution will be superceded when the price of high-speed large-memory processors is low enough that tens of thousands of such chips will be available in one machine.

Networking. Access to remote computers has been, and will remain in the near future one of the biggest problems in supercomputing. High speed T-1 (1.5Mbytes/sec transfer rates) networks are currently becoming available to researchers. Figure 2 shows a map of the upgraded NSFnet which consists of a T-1 backbone connecting the national supercomputer centers and other major universities in the United States. The network through-put to an individual is hampered by the fact that many regional networks which feed the T-1 backbone have lower bandwidths.

Networks not only provide the link to the supercomputer upon which the computationalist depends, they also act as a means of communication between researchers. In some cases it may necessary to exchange data and images quickly and computer networks are the only means of providing that service on time scales of hours or even minutes. New software is needed to allow the user to take advantage of multiple computer environments. NCSA's popular Telnet software uses TCP/IP protocols to link a researcher's workstation and multiple remote hosts. Its ability to transfer files, provide interactive computing capabilities on different machines and emulate a number of different terminal types (VT100, Tektronix, Raster, etc.) makes it the basis for a distributed computing environment and enhances the productivity of the individual scientist.

Local Workstations. Regardless of the capabilities of the network, it is always in the interest of the individual to perform as many tasks as possible in a local environment. Intelligent workstations are necessary in order to execute various chores

such file editing, collection and analysis of data, routing of electronic mail, development and testing of prototypical codes, compilation of codes, symbolic calculation, and manipulation of visual images. The individual researcher is able to customize his/her workstation to the particular needs of the science being done through choices in hardware and software. As workstations increase in power to the point that the entire code development phase can be carried out in the office of the individual scientist, the supercomputers will need only be used for the purpose of 'number crunching'. The goal of network software development is to make the entire process of code development, file transfer, simulation and analysis of the results completely transparent to the scientist at his or her workstation.

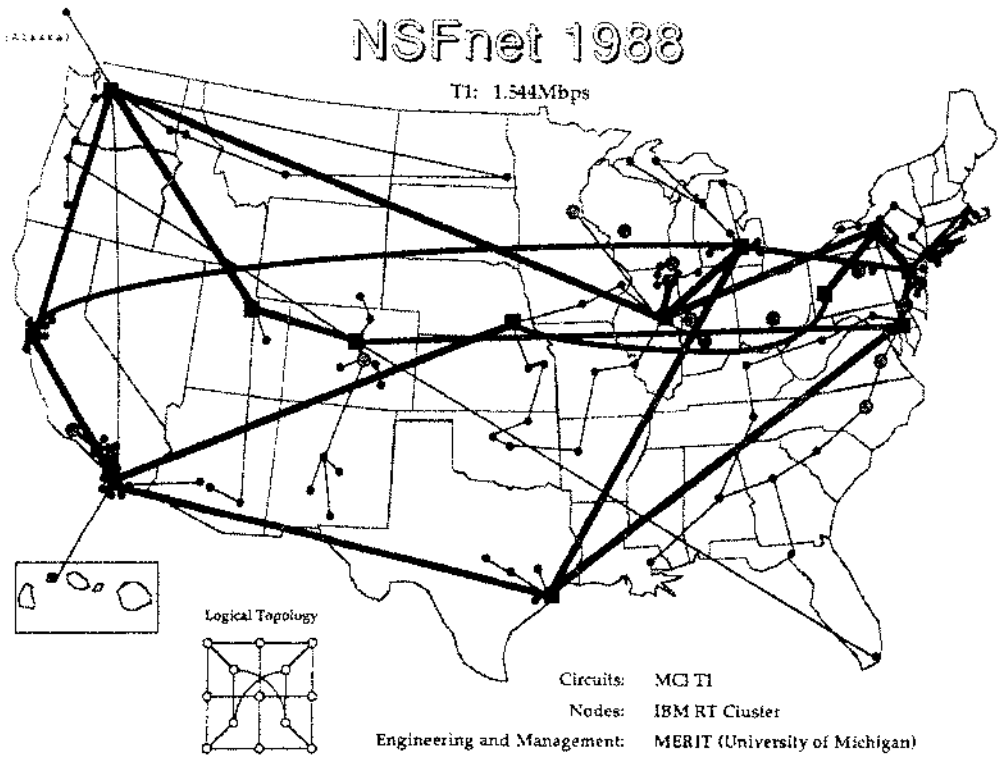


Figure 2. A map of the NSFNet used to connect the national supercomputer centers with smaller regional networks. The backbone that connects the major centers (thick lines) has recently been upgraded to T-1 status.

III. ALGORITHMS AND SOFTWARE

Operating Systems. The emergence of a single consistent operating system running on all of the links of a network from the researcher's workstation to the

supercomputer will be of great help to the numerical relativist. Computing on different machines (with different operating systems) that form a distributed computing system has always been a programming nightmare. The passage from one system to another calls for different commands, protocols, and utilities, many of which are not compatible with each other. Added to the difficulties of having to learn the fundamentals of different operating systems in order to effectively perform interactive tasks, codes which have built in system dependencies often require a large expenditure of effort during the process of porting them from one machine to another.

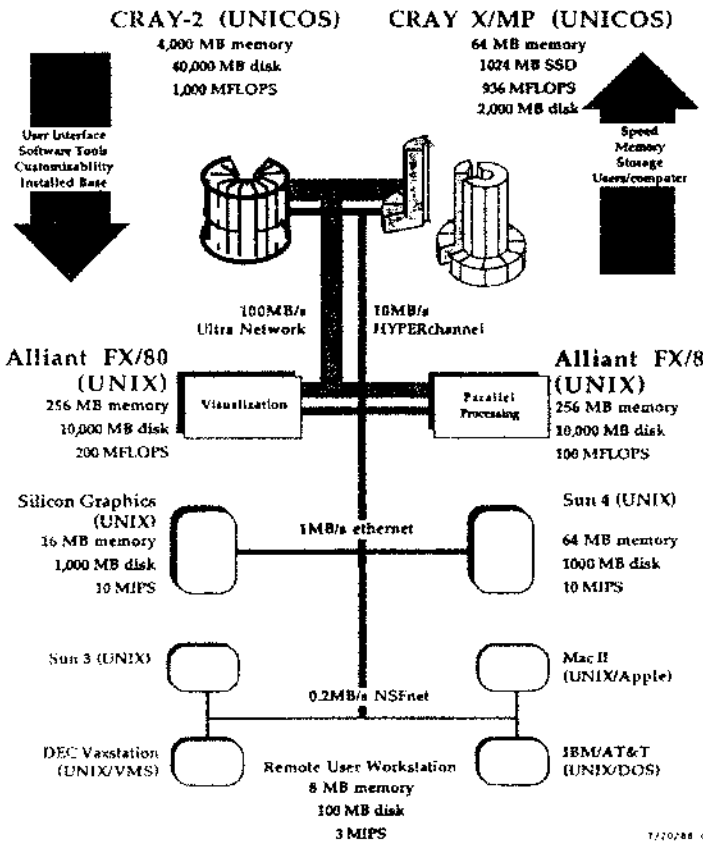


Figure 3. The NCSA Unix pathway showing the different computers and their connecting networks that use Unix or Unix-like operating systems. The user of such a system need only learn one set of commands, utilities, protocols, etc., thereby placing him/her in an efficient computing environment.

At the present time it appears that Unix in some guise or another will, at least for the near future, become the operating system of choice among computational scientists. This has been heavily influenced by the importance of scientific workstations (most of which run Unix or Unix-like operating systems) in computational research. The fact that this operating system may now be found on the most powerful supercomputers as well as widely available micro's is restructuring the way one works with computers. The step toward a complete Unix environment will undoubtedly accelerate the development of distributed computing. Figure 3 provides an example of a distributed Unix environment

linking the supercomputers at NCSA to the workstations and personal computers used by researchers.

Programming Languages. Along with the shift to Unix, there has been a growing popularity of the C programming language, even among scientific programmers. Indeed we have seen the rapid publication of a C version of the popular *Numerical Recipes* book of Press et. al. (1988). At the University of Illinois, one computational physics course (taught by Stephen Wolfram) has as a prerequisite, the ability to program in C. It is not yet clear what the impact of C will be in scientific computing. While C gives the programmer a more intimate relationship with memory allocations and registers, numerical calculations are often carried out using a Fortran like syntax. Proposals for Fortran-8X include extensions that are aimed specifically at vector and parallel computing. Rather than be restricted to standard serial computations, the array extensions allow data to be operated on all at once. Many manufacturers of newer parallel and vector machines have already implemented these extensions. Fortran is not dead as far as the computational scientist is concerned and reports of its demise like that of Mark Twain's have been greatly exaggerated.

Compilers. Luckily compilers are becoming more intelligent with increasing version numbers. This is particularly important in vector and parallel processing. Scientists do not want to, nor should they have to, worry about adding compiler directives to every portion of code that can be vectorized or run concurrently. The bulk of this work should be recognized by the compiler and the listing files should provide information on where optimization, vectorization and concurrency are hindered or difficult to recognize. Expert compilers are being developed that are able to recognize vectorizable and parallelizable codes. Some of these even enter into a dialog with the user to ensure that the compiler and scientist mutually agree on the procedures to be executed. This important development allows the researcher to concentrate more on the physics of a particular simulation leaving it to the compiler to keep track of the various tasks being performed on the computer.

Applications Software. The issues surrounding software tools depend heavily upon the type of calculations that ones needs to perform. In computational engineering and chemistry, a number of general applications packages have been developed for use by a large majority of researchers in their respective fields. Nothing like this has occurred in general relativity (except perhaps for algebraic computing). Most numerical relativists still write their own code. This is largely due to the fact that many algorithms are still being explored. Relativity is not alone in these circumstances, hydrodynamicists who already have a great deal more experience in computation are still developing better algorithms to calculate the complex nonlinear behavior of fluids.

At the present time, numerical relativity lacks a nucleus of community software capable of performing the common tasks that each researcher encounters. The development of code generators, elliptic equation solvers, partial derivative to finite difference translators, and visualization software specific to the needs of numerical relativists will require many man-years of effort. Applications software for numerical relativity will only be produced from a group made up of a critical mass of computational relativists capable of collaborating with specialists in computer graphics, numerical analysis, and symbolic manipulation. The potential for such a situation exists currently at the national supercomputing centers, but requires a concerted effort by the numerical relativity community to pool its resources and expertise.

Elliptic Equations. For numerical relativists, perhaps the closest that one comes to having applications packages is in the area of elliptic equation solvers. The simultaneous over-relaxation (SOR) methods that were used in the original pioneering efforts of numerical relativity have been replaced with preconditioned conjugate gradient (CG) methods, alternating direction implicit (ADI), multigrid (see the Choptuik and Cook articles) and other methods. These algorithms, while being more complex, are more accurate, stable and often converge a good deal faster that the older SOR method. They generally require more storage and therefore have only recently begun to fall into favor. Evans (1986) has provided a good example of how much faster and accurate the incomplete Cholesky conjugate gradient (ICCG) method is and Fig. 4 reproduces his results. Similar results, reported in this volume, have been obtained by Nakamura and his colleagues.

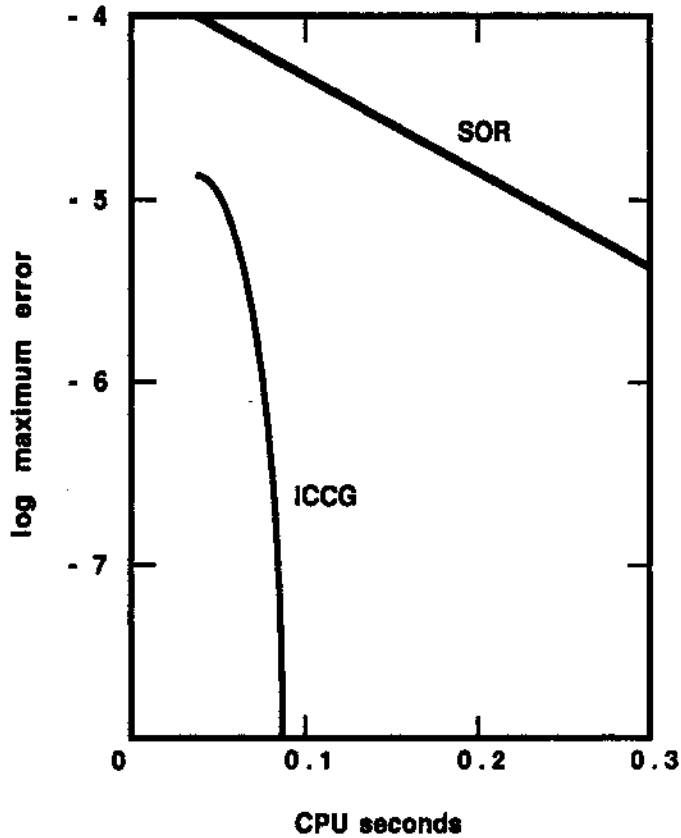


Figure 4. The rate of convergence for SOR and ICCG as a function of CPU time. These tests were run on a 900 node mesh (12x75) and solve the maximal slicing equation. [After Evans (1986)].

Outside of general relativity, there are efforts to create general purpose elliptic equation solvers. The ELLPACK project is a good example of a collaboration among computer scientists, numerical analysts and mathematicians designed to produce a software package useful for a large number of users. This package has its own programming language that sets up the equation, specifies a finite difference scheme and produces a graphical display of the solution. Unfortunately, ELLPACK has not yet reached a level of generality where it can be employed in relativity. Given the way it is set up, an entire numerical relativity code would have to be treated as subroutine called by ELLPACK. If the researcher performs the finite differencing of the elliptic equations, there is still an

inadequate supply of methods capable of solving the large sparse, nonsymmetric matrix equations that are generated. Fewer still are optimized for parallel and vector processing. This is situation is beginning to change with extensions that are now being added to the Yale Sparse Matrix routines and there is a similar project now under way at Los Alamos. Until some of these projects produce software that is truly 'all purpose', numerical relativists have little alternative than to write their own special purpose elliptic solvers.

Hyperbolic Equations. The techniques for constructing solutions to the hyperbolic evolution equations are just now being explored and there is an incredible diversity of methods available to the computational scientist. Initial gains in the field were achieved in going from first order (e.g. Lax methods) to second order methods. The standard leapfrog scheme has provided accuracy to within five or ten percent for the finite differenced version of the Einstein equations. However there are limitations to these methods and greater accuracy (to less than one percent) will require the use of more modern predictor-corrector methods (see Bernstein et. al.). These techniques can add increased stability to the system and this has the advantage of allowing longer time evolutions to be performed. It is not yet clear what the 'best' algorithm is for GR hyperbolic evolution equations, but it is likely that the current standard methods will be enhanced or superceded as numerical relativity matures.

While multigrid schemes are now being used in numerical relativity to solve the elliptic constraint equations, adaptive grid methods for the hyperbolic equations have not generally been used. The exception to this has been the work of Evans and Choptuik (this volume). Adaptive methods have proven to be successful in fluid dynamics where they are used to follow the flow of fluids. Using adaptive methods with appropriate gauge choices to follow the geometry should make significant contributions to numerical relativity. This is a subject that will require more analysis and numerical experimentation in the future.

Higher order (beyond second order) finite differencing schemes are also being used to construct numerical solutions to the Einstein equations. These will probably be necessary as long as the memory available with supercomputers is such that increased accuracy cannot be gained through a decrease in step size. The price that one pays for this in the evolution is that variables need to be stored on more time steps. In this case one trades off the need for higher grid resolution for the storage of more variables at each node.

Finally, for the hydrodynamic equations, the modern monotonicity requirements implemented in piecewise parabolic methods have contributed to important advances in this field. These methods can treat the advective part of the hyperbolic equations more accurately than straight forward finite differencing schemes, but they are inherently more complicated and require more storage [Woodward (1988)]. Smooth particle hydrodynamics is another new method that is now being applied to relativistic problems (see Kochanek and Evans in these proceedings) with some success. The algorithm depends upon a binary tree structures and this makes vectorization and parallelization difficult nor can one determine apriori how many particles are needed to reproduce the accuracy obtained with finite difference methods requiring less computational effort. In addition these proceedings demonstrate that spectral methods (Marck and Bonazzola) and finite element techniques (Mann) which have produced a number of successes in newtonian hydrodynamics are now being considered in the relativistic theory.

From all of the efforts described above, it is clear that the third phase of numerical relativity is well underway. It is expected that the search for new algorithms will provide greater stability and accuracy to the finite differenced versions of the Einstein equations and

not too far in the distant future the Holy Grail of numerical relativity mentioned in the Drexel meeting by Shapiro and Teukolsky (1986) will be attained.

IV. SYMBOLIC MANPULATION

Symbolic algebra and calculus on the computer is now being used by a large number of relativists. In fact, it might be said that more relativists use computers for symbolic manipulation than for numerical work. More and more, one is seeing that many of the calculations one needs to perform in general relativity are being executed by general purpose symbolic manipulation software packages running on stand-alone workstations. Having a dedicated machine for one's own use not only gives the user greater flexibility, but also prevents those others who have to share a computer of limited resources with that user from exhibiting antisocial behavior.

The fastest of the symbolic manipulation programs for general relativity can still be found in SHEEP since this language was designed specifically for the calculations one must perform in general relativity. In order to make it a more general purpose application, its output is readable by REDUCE which has the capability of performing the polynomial divisions, integrations and other procedures not available in SHEEP.

All of the general purpose algebraic programs now have or will have tensor manipulation packages written on top of them. REDUCE is able to use REDTEN, developed at the University of Toronto by Charles Dyer and John Harper. MAPLE also has a number of gravity packages written for it (including a version of REDTEN called MuTENSOR) and a recent development is a symbolic Newman-Penrose package written by R. McLenaghan at the University of Waterloo. MACSYMA is probably the most well known of the general purpose symbolic manipulation software and has a standard tensor manipulation packages written for component (CTENSOR) and indicial (ITENSOR) forms. The latter, equivalent to the STENSOR package in SHEEP, has not been implemented in all versions of MACSYMA and is known to have a number of bugs. Reportedly these have been eliminated in the forthcoming Common Lisp version of MACSYMA. This new version also has the capability of calculating the various quantities from tetrad frames as well. Finally, Mathematica, Stephen Wolfram's new algebraic computing software that is capable of running on almost everything from personal computers to the Cray-2, is the basis for a new tensor manipulation package being written by Steve Christensen (NCSA) and Leonard Parker at the University of Wisconsin (Milwaukee).

While these packages perform the calculations that any relativist using paper and pencil must do, i.e. calculate the Einstein equations and various quantities associated with a particular metric or manipulate indices in a symbolic fashion, they are not set up well to calculate the Einstein equations in a form useful for 3+1 numerical relativity. It should be a trivial problem to remedy this oversight. One can simply take the Einstein equations and substitute the appropriate definitions for the extrinsic curvature, lapse, shift, 3-D Ricci tensor, etc. and obtain the 3+1 decomposition. More manipulation needs to be done in order to take advantage of such procedures as numerical regularization, operator splitting, maintenance of operators in self-adjoint form, etc. At some point, the derivation of the 3+1 form of the Einstein equations will be so complicated that we should expect to see 3+1 packages being written either from scratch or on top of a pre-existing tensor package.

A number of numerical relativists are already using symbolic manipulation to derive and generate Fortran expressions for their codes. At the present time most Fortran (and in

some cases C) code generation routines available with symbolic manipulation software is quite rudimentary, and a filter must be written that turns the analytic expressions into usable code. These proceedings are a witness to the time and effort that can be saved using such a process. For example Nakamura and his co-workers use REDUCE, Ove uses MAPLE , and Stark and Hobill use MACSYMA to aid in the process of code generation. At NCSA efforts are being made to incorporate symbolic manipulation as a 'front end' for more general numerical simulations. The present implementation uses GENTRAN, a code generation software package that presently runs on top of REDUCE and MACSYMA. While improvements are needed, it is clear that as computational physicists study more complex systems, a better interface between symbolic and numerical computations will have to be developed.

Finally, symbolic manipulation will be useful for translating PDE's into finite difference equations. Future implementations of symbolic manipulation packages will likely have libraries containing different finite difference methods that one can use to translate derivatives into finite differences on standard meshes. This feature should assist algorithm development in the future.

V. SCIENTIFIC VISUALIZATION

We are all familiar with graphs of analytic functions and how much information they can convey in a single glance. The graphical representation of three dimensional functions as geometric objects has a history that goes back well over a century. As computational science matures, graphical displays of large data sets will become increasingly important and methods of directly analyzing these images in a manner that parallels the methods of standard mathematical analysis will be developed.

Vector Graphics. Computational scientists already understand the importance of even the most rudimentary visual images for deciphering the results of a numerical calculation. Simple vector (line drawing) graphics still play an important role in isolating problems that arise during the early stages of code development. Using NCAR (National Center for Atmospheric Research) or DI-3000 (Precision Visuals Inc.) software running on a Cray or even a Sun workstation, the researcher can view 1-D (x-y plots), 2-D (contour and surface plots) or 3-D ('wireframe' plots) vector graphics. These proceedings present many examples of the use of vector graphics in 1- 2- and 3-D representations. The simplicity of these graphics routines allows them to be viewed remotely with graphics terminals (Tektronix or emulators), given a reliable network connection. If the numerical data exists in tabular form, simpler methods of producing graphical results exist in some spreadsheet software products (e.g. Apple Cricketgraph) or in symbolic manipulation packages such as Mathematica.

With NCSA Telnet, tektronix emulation includes zoom, cut and paste facilities that can be used to produce (laser printer or pen plotter) hardcopy versions for a more careful analysis or presentation. In the future, color vector graphics (with tektronix 4025 capabilities) will make this medium more useful to the scientist who must use color as another variable. Only a decade ago relativists were startled to see color movies of 'jiggling grids and contours' produced from the data created by the early numerical relativity codes. The production of time dependent vector graphic images that once required an incredible amount of human effort and computer time can, today, be re-created on an Apple Macintosh (using a combination of Apple Hypercard, NCSA Telnet, and NCAR graphics) in a matter of seconds. What once was considered the state of the art in presentation graphics is now commonly used for the day to day analysis of numerical results.

Raster Graphics. Given a data array on a 2-D grid, the floating point values are transformed into 8-bit integer format (a value between 0 to 255). This array is turned into a stream of bytes (raster file) that provides an integer value to each pixel in a raster scan. The color values of each integer are assigned by a palette that choses 256 colors (not necessarily different) out of a possible 16 million colors. A look-up table then translates the integer values into images whose colors are governed by the palette file. In many cases subtile changes in the palette can produce dramatic effects in the display of the images.

NCSA Imagetool, a raster graphics package conceived by astrophysicist, Michael Norman and computer scientist, Carol Song and developed by the NCSA Software Development Group, can display, animate and analyze such images on Sun workstations and Macintosh II's. The image processing can be controlled easily through the use of a mouse and multiple palettes can be used in the analysis of the images. One-dimensional cuts can be made through any two points on the image and can be displayed as linear or logarithmic x-y plots. The color images can also be transformed into vector graphic contour or surface plots from which hard copy images can be made. The recording of the color images onto photographic slides or videotape however requires the specialized equipment and expertise of the Scientific Media Services (SMS) at NCSA. Raster graphics are becoming increasingly useful in numerical relativity and recent examples of their ability to pick out detailed information from large data files are found in the work of Evans (neutron star collisions) and Ove (T^3xR universes).

At the present time raster files must reside locally on the researchers workstation. In the future, Telnet will be able to display images from files resident on the supercomputer itself thereby saving a great deal of time and effort. Multi-windowed raster images are now being used (Ove Windows and NCSA Datascope) to represent simultaneously all ten metric coefficients (or one function at ten different times) on a workstation terminal. Besides being able to correlate data through the images, numerical integrations and differentiations can also be performed to parallel analytic methods of studying functions of two variables.

Rendered Images. A fully rendered 3-D surface that can be viewed from any position in three-dimensional space represents the next level of complexity in scientific visualization. A finely resolved wireframe surface can be thought of as made up of a large number of very small, flat 2-D polygons (simplexes would be a name closer to the spirit of general relativity) defined by the vertices formed from the wireframe. If the polygons are assigned values of color, texture, shading, etc. the original wireframe image is turned into a colored solid surface with precise reflection and/or transmission properties determined by ray tracing (geometric optics) algorithms. Transparent surfaces provide the opportunity to visualize multiple level contours or self intersecting surfaces. Viewed on a 'fast' Silicon Graphics or Raster Technologies, Inc.workstation, one can actually 'fly around' the image in real time under the control of a mouse. For complex geometrical objects, this ability can provide a great deal of information vary rapidly.

The generation of rendered images is computationally intensive and using Wavefront Technologies, Inc. software, most multiple image animations must be produced on an Alliant FX/8 minisupercomputer (and a few smaller jobs may be performed on a Silicon Graphics workstation). In this case both the production and animation of the colored images at NCSA is now left to the expertise of the the Scientific Visualization Group and SMS. An example of a rendered animation relevant to numerical relativity would be that involving a dynamic isometric embedding diagram. One can imagine a color map representing the Newman-Penrose quantity, ψ_4, laid down on a time changing surface

that is capable of studying gravitational wave generation and propagation on a dynamic spacetime background.

Volume Imaging. The imaging of functions dependent upon all four spacetime dimensions will provide the greatest challenges to scientific visualization. The problems of three-dimensional hydrodynamics being attacked by Nakamura and Stark will require the ability to see not only the surfaces of stars, but their interiors as well. These types of problems are similar to those now being attacked in geophysical, astronomical, meteorological and medical imaging. Algorithms are still being developed that will be able to visualize volumetric data. The most notable research is being done at Stellar Computer Corp. and PIXAR where methods that range from the use of multiple transparent rendered surfaces to ray-tracing through volume elements ('voxels' or extensions of polygons). Clearly the frontiers of numerical relativity have merged with the frontiers of computer graphics.

Referring to the famous quote by Richard Hamming on the purpose of computing being insight and not numbers, the report on *Visualization in Scientific Computing* [McCormick (1988)] states that "the goal of visualization is to leverage existing scientific methods by providing new scientific insight through visual methods". While it may be difficult to foretell the future, we should be prepared to meet profound and unexpected insights into general relativity through computational methods and scientific visualization.

REFERENCES

Cadez, A., (1971), Ph. D. Thesis, University of North Carolina.

Centrella, J., (1979), Ph. D. Thesis, Cambridge University.

DeWitt, B., (1957), Unpublished notes of a lecture given at GR1, Chapel Hill, North Carolina.

Eppley, K. (1979), *In Sources of Gravitational Radiation*, edited by Smarr, L., Cambridge University Press, Cambridge.

Estabrook, F., Wahlquist, H., Christensen, S., DeWitt, B., Smarr, L., and Tsiang, E., (1973) *Phys. Rev. D.*, 7, 2814, 1973.

Evans, C., (1985), In *Numerical Astrophysics*, edited by Centrella, J., LeBlanc, J., and Bowers, R., Jones and Bartlett, Boston.

Evans, C., (1986), In *Dynamical Spacetimes and Numerical Relativity*, edited by Centrella, J., Cambridge University Press, Cambridge.

Hahn, S. G. and Lindquist, R. W., (1964), *Ann. Phys.* 29, 304.

Lichnerowicz, A., (1944), *J. Math. Pures et Appl.*, 23,37.

May, M., and White, R., (1966), *Phys. Rev.*, 141, 1232.

Misner, C.W., and Wheeler, J.A.,(1957), *Ann. Phys.*, 2, 525.

McCormick, B., DeFanti, T., and Brown, M., (1988), *Computer Graphics*, 21, 1.

Pachner, J., (1973), *Can. J. Phys.*, 15, 447.

Piran, T. (1979), *In Sources of Gravitational Radiation*, edited by Smarr, L., Cambridge University Press, Cambridge.

Press, W., Flannery, B., Teukolsky, S., and Vetterling, W., (1988), *Numerical Recipes in C*, Cambridge University Press, Cambridge.

Shapiro, S. and Teukolsky, S., (1986), In *Dynamical Spacetimes and Numerical Relativity*, edited by Centrella, J., Cambridge University Press, Cambridge.

Smarr, L., (1975), Ph.D. Thesis, University of Texas at Austin.

Smarr, L. (1979), *In Sources of Gravitational Radiation*, edited by Smarr, L., Cambridge University Press, Cambridge.

Smarr, L., and York, J., (1978), *Phys. Rev. D.*, 17, 2529.

Wilson, J. (1979), *In Sources of Gravitational Radiation*, edited by Smarr, L., Cambridge University Press, Cambridge.

Woodward, P.R. (1988), In *High Speed Computing*, edited by R. Wilhelmson, Univ. of Illinois Press, Urbana.