1987

Supercomputing about Physical Objects

John R. Rice
Purdue University, jrr@cs.purdue.edu

Report Number:
87-708

SUPERCOMPUTING ABOUT PHYSICAL OBJECTS

John R. Rice*

Computer Science Department
Purdue University
West Lafayette, IN 47907

CSD-TR-708
September 1987

Abstract

Scientific and technological advances in the next 5 to 10 years will make it feasible to create an integrated, interactive system for the design, manipulation and analysis of collections of physical objects. These advances will come in computing power through the mechanism of *parallel computation*, in *algorithms for geometry*, in *problem solving systems* to provide very high level user interfaces and in *graphics* to allow direct visualization of the behavior of the physical objects. In this paper we describe the project *Computing about Physical Objects* which is to explore the associated technical problems and to build prototypes of such systems. The focus here is upon the role of supercomputers in this area and, especially, their application to solving the partial differential equations that model many physical phenomena.

1. INTRODUCTION

Scientific and technological advances in the next 5 to 10 years will make it feasible to create an integrated, interactive system for the design, manipulation and analysis of collections of physical objects. These advances will come in *computing power* through the mechanism of parallel computation, in *algorithms for geometry* computations and

* Research supported in part by Strategic Defense Initiative grant ARO DAA929-83-K-0026.

manipulations, in *problem solving systems* to provide very high level user interfaces and in *graphics* to allow direct visualization of the behavior of the physical objects. In this paper we describe the project *Computing about Physical Objects* at Purdue University which is to explore the technical problems arising from creating these systems and to build prototypes of them. The focus is upon the role of supercomputers in this area and, especially, their application to solving the partial differential equations (PDEs) that model many physical phenomena.

The capabilities that are required for these systems fall into four categories:

Physical phenomena models. One must have accurate models and reliable methods to solve the resulting equations for things like heat flow, mechanics, combustion, structural analysis, fluid flow, etc.

Geometric designs and models. One must be create shapes easily and with great versatility. Most details of the creation must be automated and the results must be displayed in an informative, realistic fashion. Further, manipulation must be easy and allow for dynamic shape changes, motions, interactions, etc.

Software systems. Very high level, applications oriented user interfaces must be provided with the bulk of the problem solving automated. Large and diverse software systems must be integrated at a high level.

Computing resources. The machines to support this system will require gigaflops of power, many megawords of memory with many gigawords of auxiliary storage and communication bandwidths of many megabytes/sec. to support 3D color movies.

2. COMPUTING ABOUT PHYSICAL OBJECTS

We are developing the tools for computing with models of physical objects. In the course of this work, we face problems of how to represent objects by suitable models, how to manipulate and edit these models, how to analyze and simulate the behavior of modeled objects. To see the varied nature of the work, and to gain a first impression the project consider designing a small water cooled piston engine diagramed in Figure 1.

Imagine we are at a point where the piston and its linkage to the crank shaft have been designed. Now we wish to design the block such that the engine is kept cool, strong, and light. So, the exterior geometry of the block must be designed, the shape and location of the water cooling lines must be determined, and a suitable material for the block must be selected.

Having chosen the shapes, assisted by a geometric modeling interface, the user must solve a system of partial differential equations (PDEs) to analyze the heat flow and stresses associated with this geometry and choice of block material. He may wish to simulate running the engine at different speeds which changes the strength of the heat source and the stresses on the moving parts. Systems of ordinary differential equations (ODEs) describe the acceleration and constraint forces on the piston linkage. With access to sufficient computing power, a user can quickly explore a wide range of shapes and materials preliminary to a refined, optimized final design.

Figure 1. Cross section of a piston Engine. The source of heat and force is at H, the coolant within the block is shown with bubbles. The housing of the linkage L to the piston P is not shown.

What is involved in creating and simulating this scenario? First, complex object models must be created and coordinated. The geometric modeling subproject provides tools for this. The models are created through a user interface and require much automatic design support from the system. For example, the geometric design of piston and linkage requires a sophisticated solid modeling system. The ODEs describing the dynamic behavior of the piston linkage can then be derived automatically by the system for the geometry and material composition of the links. A subproject, called Project Newton, builds such a system. PDEs model the physical behavior such as heat flow, stresses and strains. They are identified and numerical methods are selected that are well suited to the problem's nature and the desired accuracy. A subproject, called Mathematical Software Systems, does this. The user should be allowed to specify a sacrifice some accuracy for the sake of speed or economy, the system can then allocate the computation among the available resources to achieve this objective - without detailed intervention by the user. The parallel processing subproject provides tools for this.

All aspects of this work require very high powered workstations and sophisticated graphics backed up by supercomputer power. Because of the scope of the effort, all projects use a very high level approach to software development and integration that must be backed up by powerful computing resources. The relationship of the principal subprojects is shown in Figure 2. We summarize each here.

Project Newton. The goal of this research is to develop a highly modularized and extensive system that duplicates the precise behavior of physical objects from their models. The work is part of a consortium effort involving groups both at Cornell and Purdue University.

Geometric Modeling. Geometric and solid modeling has reached a plateau that cannot

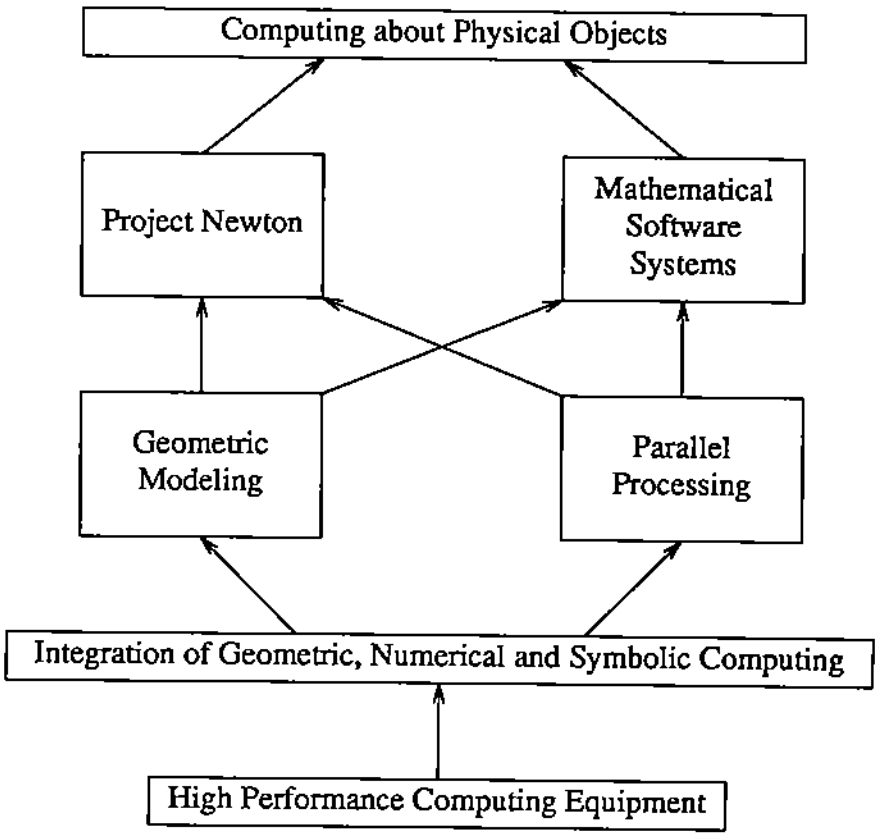


Figure 2. The relationship of the subprojects of Computing about Physical Objects.

be elevated unless a number of basic computational problems in mathematics are solved efficiently. This subproject focuses on these problems. Its results, as they mature, will impact the other projects significantly.

Mathematical Software Systems. A high level mathematical software system targeted toward elliptic partial differential equations is already operational. Its principal components are an interactive, mathematically based, graphically oriented interface and a broad range of problem solving modules. This 100,000+ lines of code system is built using various software tools so that it can readily evolve and be enhanced. It is an ideal vehicle to test approaches to future systems for computations about physical objects.

Parallel Processing. Computing with physical models requires enormous computing power which will come mostly from massive parallelism. Our work concentrates on three of the many aspects of this problem area. First, we will be heavily involved in the algorithmic infrastructure both for numeric and geometric computation. Second, we will study how to create an intelligent system for resource allocation without burdensome user input or intervention. We will consider both tightly coupled computing

environments where algorithm specific, synchronous approaches seem to be most promising, and loosely coupled environments where we will concentrate on very general, heuristic approaches to resource allocation. Finally, we will monitor performance issues continually.

3. GEOMETRY BASED APPROACH

Our approach is based on the ELLPACK system described in [Rice and Borsvert, 1985] and [Rice, 1987]. The philosophy and many of the details of our approach is given there so instead of describing ELLPACK, we present our approach to the organization of mathematical software systems for partial differential equations. Two hierarchical views are given in Figures 3 and 4. Figure 3 shows five levels of software separating the user from the computing facility, levels that we believe will be present in most large scale scientific systems of the next decade.

Figure 4 shows the mathematical software system for this project. The workstation environment and application oriented expert system of Figure 3 is shown as the *user interface* here. It must communicate with the user in his own terms, thus if one is designing engine cylinders, then this interface must have a relevant vocabulary (e.g., piston, valve, block, ...) and operators (e.g., run engine, open valve, increase bore, ...). The mathematical software infrastructure of Figure 3 is decomposed into four layers in Figure 4. The top is the *physical objects system* which is application independent but which provides all the facilities to create, manipulate, simulate and analyze collection of physical objects.

The *physical objects* are self contained objects (in the sense of object oriented programming) that represent a specific physical object in the computational model. A physical object is partitioned into a collection of computational domains. Three reasons for this are: a) to separate physical phenomena. For example, the stresses and heat flows in a cylinder wall might be computed by unrelated software packages using completely different methods; b) iterative and time marching methods sometimes need two or three "copies" of the physical object; c) to divide the work. For example, if 100 processors were applied to computing the heat distribution in an engine block, then the block could be partitioned into 100 subdomains.

The *computational tools* in Figure 4 are the nuts and bolts of the software infrastructure. These include libraries of numerical methods, packages to display functions and shapes, procedures to move or modify basic geometric objects and so forth. There may well be a time where the computing facility contains hardware implementations of some of these tools. It will then be the task of the computational resources manager (see Figure 3) to access this hardware using the network operating system.

A key design concept in this project is to make the geometry the basic data structure. Equations, arrays, grids, properties, etc., are then attached to this data structure to create the full description (data structure) of the physical object. This is illustrated in Figure 5 where a computational domain is shown along with types of associated information. Note that object interfaces (*faces*) have separate and complete data structures of the same nature as the domain's. A hypothetical example of the information about the two

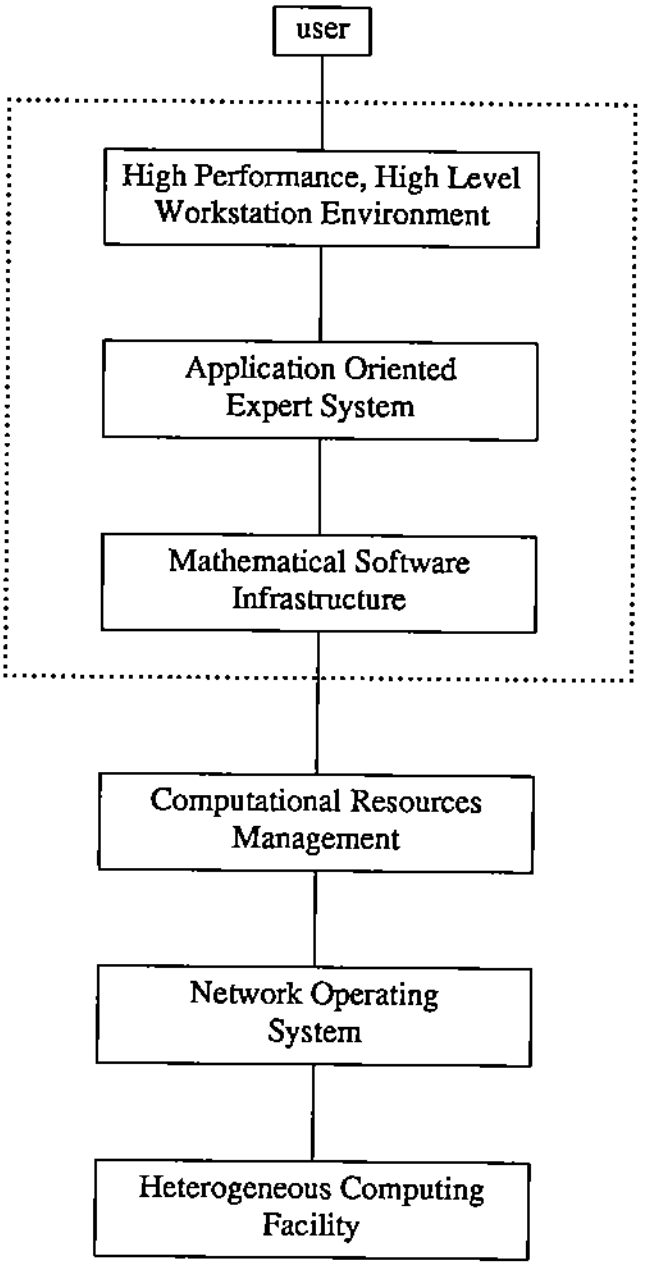
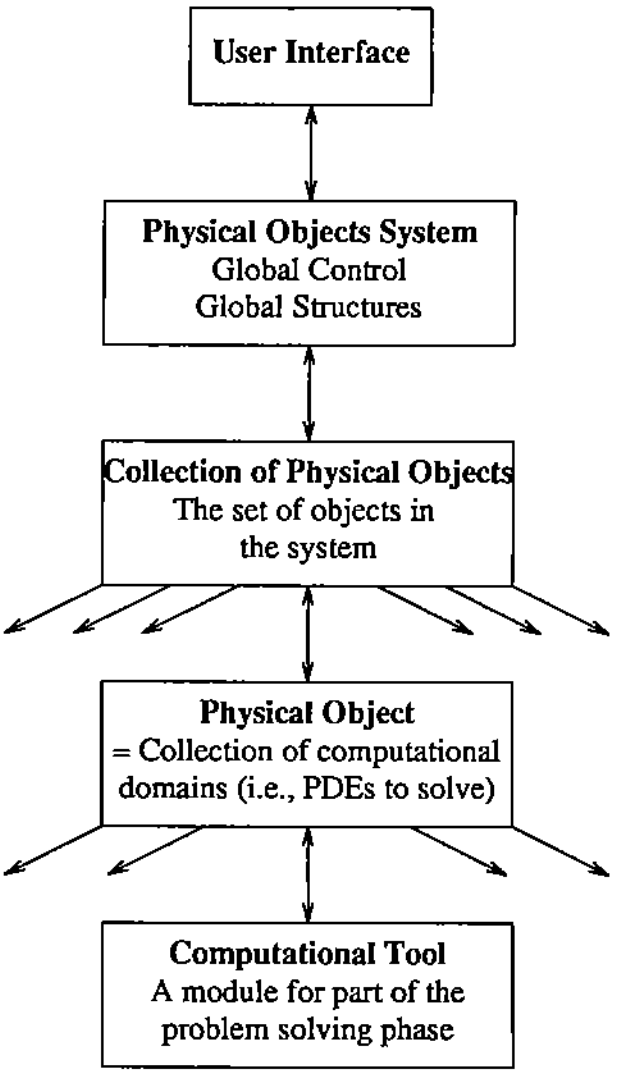


Figure 3. Schematic of the future organization of scientific computing. The mathematical software systems area is indicated by the dotted box.

objects in Figure 5 follows.

1. Domain: **Lefthub -u**
   A. Faces (Top 1, Left 1, Left 2, Bot 2), Clockwise
   B. Variable = u



Complex system using multiple resources. Very specific to applications area.

Complex system of a more deterministic nature, programmable.

Physical objects include:
Geometry definition
Physical characteristics
Interface specifications

Purposes of computational domains:
separate physical phenomena
iterative methods
divide the work

Example tools:
Linear equation solvers
Function plotters/viewers
Geometry manipulation
Physics discretization

Figure 4. Hierarchical view of the mathematical software system to be used in the Computing about Physical Objects project.

$$u_{xx} + \sigma(x,y,\mu) * u_{yy} - cos(x*v)*u = L(x,y,\tau)$$

C. Solid brass, heat flow

D. Subframe (4,7)
   Faces: Top 1, Left 1, Left 2, Bot 2
   Copy: Lefthub -v
   Map: Lefthub -u.mp.1, MAP.Lefthub-u.1 (x, y)

E. A(627,83), A.low(627,83), A.up(627,83)
   x.grid(32), y.grid(21), grid.pts(32,21)

2. Face: **Top1 -u**
   A. Parametrized, Ends (p, 2.2, 3.7)
   B. B. u_x + *coeff* 4* (u −w) = coef 5(v)
      Ends: (0.7, 8.2), (1.2, 9.6)
      Copy: Top1 -v, Botleft -w
      Domains: Lefthub -u
      Map: Top1 -u.mp.1, MAP.Lefthub -u.1 (x, y)
   C. Ellipse, continuous convection
   D. Subframe (3,7)
   E. xvals(12), yvals(12), params(12), type(12)
      coeff5.vals(12), W.vals(12), BC.array(12,3)

Figure 5. View of a computation domain with interior and faces. The types of information in the associated data structures are listed.

4. ORGANIZATION OF THE COMPUTATIONAL SYSTEM

Figure 6 shows a high level block diagram of the computational system for Computing about Physical Objects. Note that the user appears twice, once at the top while the problem is being formulated and analyzed and at the bottom while the problem is being solved. The "expert" access to the system (appear right) is where information about the

performance of software modules and machine is entered into the system. Due to space limitations, we do not discuss in detail most of this diagram, see the references for more information. We do discuss the parts most relevant to supercomputing: machine selection and problem partition.

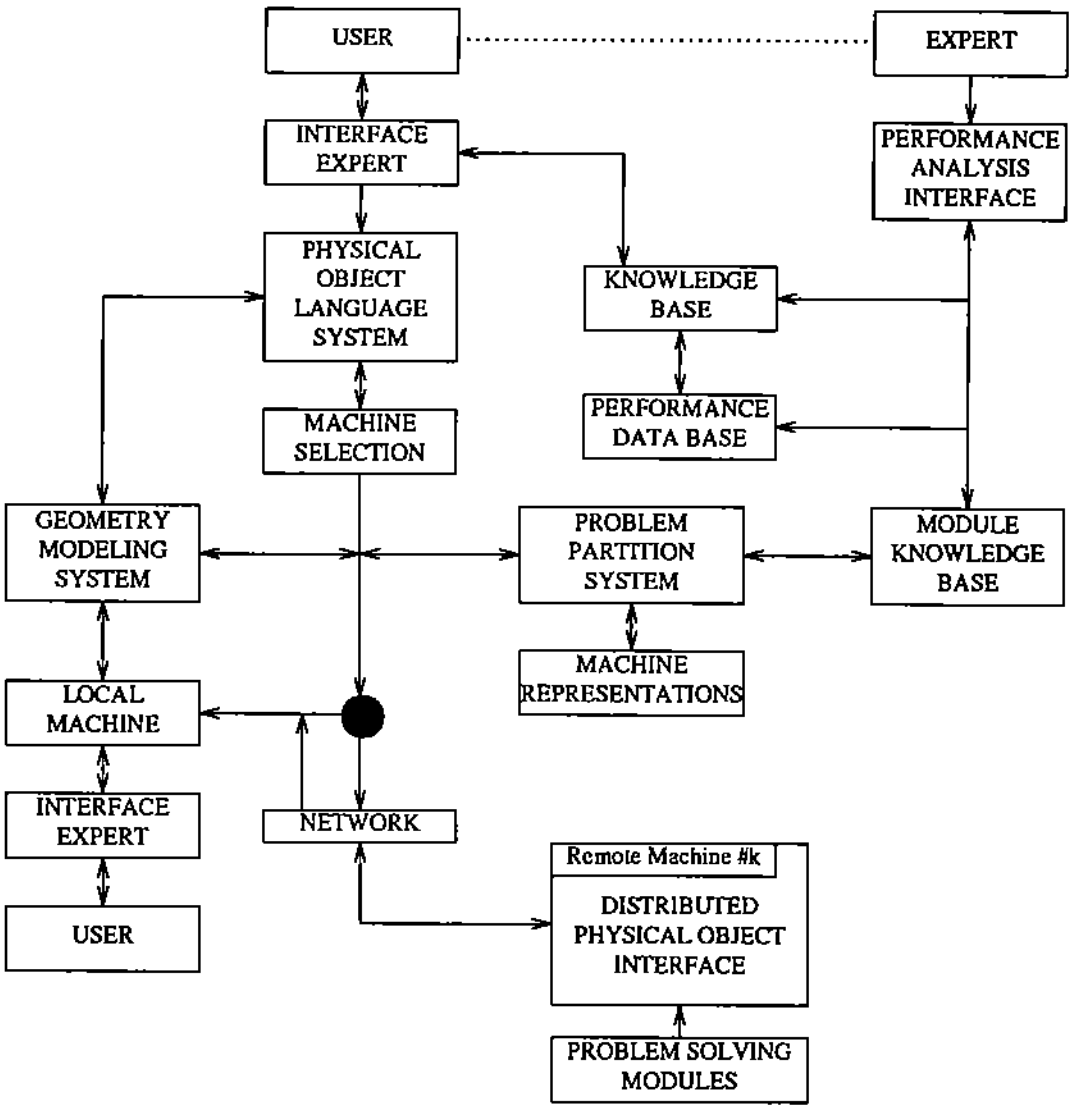


Figure 6. General high level block diagram of the computational system for the Computing about Physical Objects project.

Figure 7 shows the structure of the *machine selection* component of the system. We note at the top that a lot of information about the problem to be solved is collected by the user interface system. The software modules are to be analyzed in advance to obtain formulas to estimate computational requirements using this data. This allows concrete estimates to be made for the machines available and predicted performances are produced

(they may well be reviewed by the user) which are the basis for the machine selection. At this point one can incorporate work load information about the available machines, we already have an experimental system for this. The selections are made on the basis of various values and these may be applied either statically (the computation is scheduled once and for all) or dynamically. Note a large and complex computation may be developed in this environment which has many different modules and phases. Thus we will systematically explore techniques to distribute different parts of the computation among different machines.

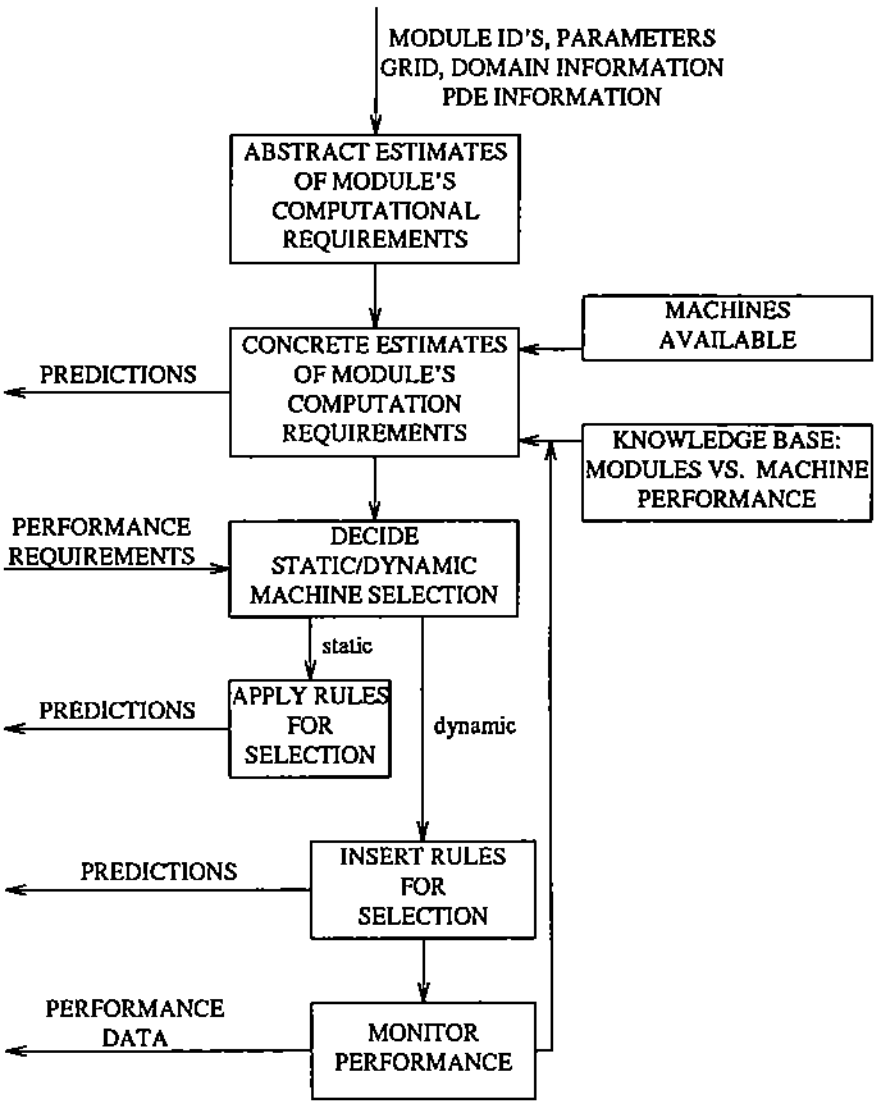


Figure 7. Schematic of the machine selection subsystem.

The computing facility to be used at Purdue initially includes common workstations, an Alliant FX/1 as a workstation, LISP machines, a Cyber 205 (2 pipe, 2 megawords), a FLEX/32 (7 processors) and an NCUBE (128 processors). There are also

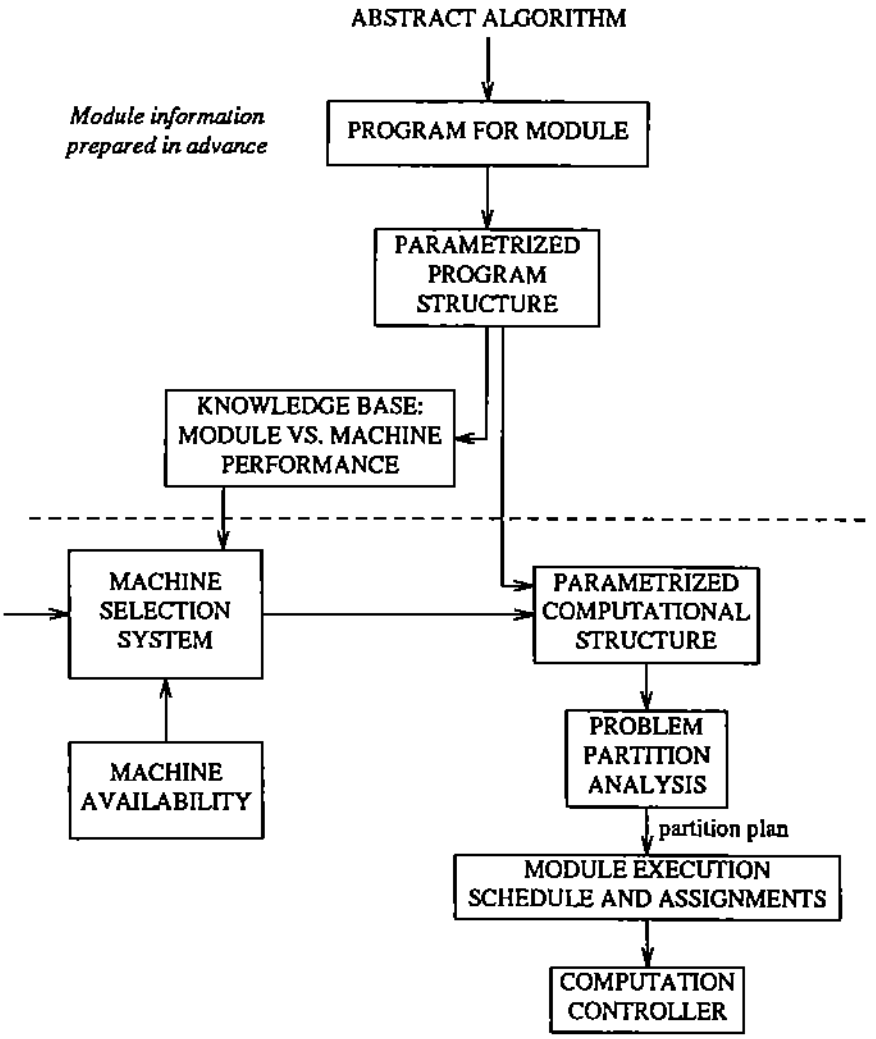


Figure 8. Schematic of the problem partition subsystem. The dotted line separates the advance software structure setup from the actual partition actions.

numerous mini-computers (VAX 11/780, 11/785, 8600) on the network.

If multiprocessors are available (as they will be for this project) then one must consider partitioning computations assigned to them. We envisage using precompiled libraries for problem solving so that any vectorization or parallelization is done independently of the specific problem to be solved. However, many numerical algorithms can be parametrized nicely in various ways and we plan to creat a partitioning environment where we can vary the amount of partitioning. For example, a "small" computation might be created by using 32 subdomains plus library modules with 32 as a parameter. A larger computation might use 256 subdomains on 256 processors plus the same library modules with 32 changed to 256. This approach is illustrated schematically in Figure 8 where the dotted line separates the advance software structure setup from the actual problem partition step.

5. THE PRINCIPAL TECHNICAL PROBLEMS AREAS

We now list the seven principal technical problem areas along with references to recent Purdue work in each area. One should also consult [Bajaj et. al., 1987] for an overview of the project. We see that the group at Purdue has already invested a large effort building up to this project, over 50 papers published in these seven areas and there are many unpublished technical reports. Thus we have a good start on the project but there is still an enormous amount of basic research, experimentation and development to be done to create these systems.

AREA 1: PDE Solvers. These software modules must be general, robust, efficient and parallelizable. The special emphasis at Purdue has been in collocation methods (using splines or piecewise polynomials), high order finite difference methods and adaptive methods. [Birkhoff and Lynch, 1985], [E. Houstis et. al., 1987], [E. Houstis et. al., 1988], [Lynch and Rice, 1978], [Rice, 1986a], [Rice, 1987a], [Rice, 1987c]

AREA 2: Partitioning of Parallel Computations. The partitioning (or program restructuring) can occur at compile time, load time or run time. It must be automatic but user controllable and be effective. [C. Houstis et. al., 1987a]

AREA 3: Actual Software System. It must be modular, hierarchically structured, efficient and machine adaptable. The Purdue group has built one PDE solving system with about 120,000 lines of Fortran code. [E. Houstis et. al., 1985a], [E. Houstis et. al., 1985b], [E. Houstis et. al., 1985c], [Rice, 1987b], [Rice and Boisvert, 1985], [Rice et. al., 1986]

AREA 4: Performance Analysis and Control. One must be able to predict with reasonable confidence both execution times and accuracy achieved. This involves benchmarking both hardware and software speeds, parametrizing software modules appropriately and, most difficult of all, assessing the actual accuracy of numerical methods. Extensive evaluation of PDE software performance has been done at Purdue. [Boisvert et. al., 1979], [Dyksen et. al., 1984], [C. Houstis et. al., 1987b], [Ribbens and Rice, 1986], [Rice, 1986b], [Rice, 1986c], [Rice et. al., 1981]

AREA 5: Computation Control. One must make reasonable resource use estimates in a distributed computation, synchronize the computations plus estimate accuracies and efficiencies being obtained. Purdue work is in the study of synchronous versus asynchronous approaches. [C. Houstis et. al., 1987a], [Marinescu and Rice, 1987a], [Marinescu and Rice, 1987b]

AREA 6: Geometric Computation. One must be able to create, modify, move and reshape collections of geometric domains. Multiple representations are probably required as well as basically new techniques. Purdue work here is in the areas of algebraic geometry algorithms, general 2D and 3D domain processing, domain mapping techniques and motion planning. [Ahbyankar, 1983], [Ahbyankar and Bajaj, 1987a], [Ahbyankar and Bajaj, 1987b], [Ahbyankar and Bajaj, 1987c], [Atallah and Bajaj, 1987], [Bajaj, 1985], [Bajaj, 1986], [Bajaj, 1987a], [Bajaj, 1987b], [Bajaj and Kim, 1987a],

[Bajaj and Kim, 1987b], [Bajaj and Kim, 1987c], [Bajaj and Kim 1987d], [Bajaj and Kim 1987e], [Bajaj and Kim 1987f], [Bajaj and Moh 1987], [Bajaj, Hoffmann and Hopcroft 1987], [Bajaj, Liu and Wu 1987], [Hoffmann and Hopcroft, 1985], [Hoffmann and Hopcroft, 1986], [Hoffmann and Hopcroft, 1987a], [Hoffmann and Hopcroft, 1987b], [Hoffmann and Hopcroft, 1987], [Hoffmann et. al., 1986], [Ribbens, 1986], [Rice, 1984a], Rice, 1984b].

AREA 7: User Interface. It must natural, easy to use and highly graphics oriented. Purdue has built interactive, graphics, expert PDE systems interfaces. [Dyksen and Ribbens, 1987], [McFaddin and Rice, 1987], [Rice, 1985], [Rice, 1987d]

6. REFERENCES

Abhyankar, S. S., (1983), Desingularization of plane curves. In *Proc. Symp. in Pure Mathematics*, 40, 1-45.

Abhyankar, S. S., and C. Bajaj, (1987a), "Automatic rational parameterization of curves and surfaces I: Conics and conicoids", *Computer Aided Design*, 19,1, 11-14.

Abhyankar, S. S., and C. Bajaj, (1987b), "Automatic rational parameterization of curves and surfaces II: Cubics and cubicoids", *Computer Aided Design*, to appear.

Abhyankar, S. S., and C. Bajaj, (1987c), "Automatic parameterization of rational curves and surfaces III: Algebraic plane curves", CSD-TR-619, Computer Science, Purdue University.

Atallah, M., and C. Bajaj, (1987), "Efficient algorithms for common transversals", *Information Processing Letters*, 25, 2, 87-91.

Bajaj, C., (1985), "The Algebraic complexity of shortest paths in polyhedral spaces". In *Proc. 23rd Annual Allerton Conference on Communication, Control and Computing*, Univ. of Illinois, 510-517.

Bajaj, C., (1986), "An efficient parallel solution for shortest paths in 3-dimensions". In *Proc. 1986 IEEE International Conference on Robotics and Automation*, San Fransisco, 1897-1900.

Bajaj, C., (1987a), "Exact and approximate shortest path planning". In *Path Planning*, R. Franklin, ed., SIAM, to appear.

Bajaj, C., (1987b), "On algorithmic implicitization of rational algebraic curves and surfaces", CSD-TR-681, Computer Science, Purdue University.

Bajaj, C. and M. Kim, (1987a), "Generation of configuration space obstacles I: The case of a moving sphere", *IEEE J. of Robotics and Automation*, to appear.

Bajaj, C. and M. Kim, (1987b), "Generation of configuration space obstacles II: The case of moving algebraic surfaces", CSD-TR-586, Computer Science, Purdue University.

Bajaj, C. and M. Kim, (1987c), "Generation of configuration space obstacles III: The case of moving algebraic curves", *Algorithmica*, to appear.

Bajaj, C., and M. Kim, (1987d), "Compliant motion planning with geometric models", *Proc. of 3rd ACM Symposium on Computation Geometry*, 171-180.

Bajaj,C., and M. Kim, (1987e), "Convex decomposition of objects bounded by algebraic curves", CSD-TR-677, Computer Science, Purdue University.

Bajaj, C., and M. Kim, (1987f), " Convex hull of objects bounded by algebraic curves'', CSD-TR-697, Computer Science, Purdue University.

Bajaj, C., C. Hoffmann and J. Hopcroft, (1987), ''Tracing algebraic curves: Plane curves'', CSD-TR-637, Computer Science, Purdue University.

Bajaj, C., C. Hoffmann, E. Houstis, J. Korb and J. Rice, (1987), ''Computing about physical objects'', CSD-TR-696, Computer Science, Purdue University.

Bajaj, C., C. Liu, and M. Wu, (1987), ''A face area evaluation algorithm for solids in CSG representation'', CSD-TR-682, Computer Science, Purdue University.

Bajaj, C., and T. Moh, (1987), ''Generalized unfoldings for shortest paths'', *Intl. J. of Robotics Research*, to appear.

Birkhoff, G. and R.E. Lynch, (1985), ''Numerical solutions of elliptic problems'', *SIAM Publications*, Philadelphia.

Boisvert, R.F., E.N. Houstis, and J.R. Rice, (1979), ''A system for performance evaluation of partial differential equations software''. *IEEE Trans. Software Engineering*, **5**, 418–425.

Dyksen, W.R., R.E. Lynch, J.R. Rice and E.N. Houstis, (1984), ''The performance of the collocation and Galerkin methods with Hermite bi-cubics,'' *SIAM J. Numer. Anal.*, **21**, 695–715.

Dyksen, W.R. and C.J. Ribbens, (1987), ''Interactive ELLPACK: An interactive problem solving environment for elliptic partial differential equations'', *ACM Trans. Math. Software*, **13**, to appear.

Hoffmann, C. and J. Hopcroft, (1985), ''Automatic surface generation in computer aided design'', *The Visual Computer*, **1**, 92-100.

Hoffmann, C. and J. Hopcroft, (1986), ''Quadratic blending surfaces'', *Comp. Aided Design*, 18, 301-306.

Hoffmann, C. and J. Hopcroft, (1987a), ''Geometric ambiguities in boundary representations'', *Comp. Aided Design*, 19, 141-147.

Hoffmann, C. and J. Hopcroft, (1987b), ''The potential method for blending surfaces and corners'', in *Geometric Modeling*, G. Farin, ed., SIAM, 347-366.

Hoffmann, C. and J. Hopcroft, (1987), ''Simulation of physical systems from geometric models'', special issue, *IEEE J. of Robotics and Automation*, (June).

Hoffmann, C., J. Hopcroft, and M. Karasick, (1986), ''Boolean operations on boundary representations of polyhedral objects'', in preparation.

Houstis, C.E., E.N. Houstis and J.R. Rice, (1984), ''Partitioning and allocation of PDE computations in distributed systems''. In *PDE Software: Modules, Interfaces and Systems*, (Engquist and Smedsaas, eds.), North-Holland, 67-85.

Houstis, C.E., E.N. Houstis and J.R. Rice, (1987), ''Partitioning PDE computations: Methods and performance evaluations'', *Journal Parallel Computing*, to appear.

Houstis, C.E., E.N. Houstis, J.R. Rice and M. Samartzis, ''Benchmarking of bus multiprocessor hardware for large scale scientific computing''. In *Advances in Computer Methods for Partial Differential Equations, VI*, (Stepleman and Vishnevetsky, eds), IMACS, 136-141.

Houstis, E.N., W.F. Mitchell, and J.R. Rice, (1985a), ''Collocation software for second order elliptic partial differential equations'', *ACM Trans. Math. Software*, **11**, 379–412.

Houstis, E.N., W.F. Mitchell, and J.R. Rice, (1986b), ''Algorithm 638 GENCOL:

Collocation on general domains with bicubic Hermite polynomials'', *ACM Trans. Math. Software*, **11**, 416–418.

Houstis, E.N., W.F. Mitchell and J.R. Rice, (1985c), ''Algorithm 638, INTCOL and HERMCOL: Collocation on rectangular domains with bicubic Hermite polynomials'', *ACM Trans. Math. Software*, **11**, 416-418.

Houstis, E.N., M.A. Vavalis and J.R. Rice, (1987), ''Parallelization of a new class of cubic spline collocation methods''. In *Advances in Computer Methods for Partial Differential Equations, VI*, (Stepleman and Vishnevetsky, eds), IMACS, 167-174.

Houstis, E.N., E.A. Vavalis and J.R. Rice, (1988), ''Convergence of an $O(h^4)$ cubic spline collocation method for elliptic partial differential equations'', *SIAM J. Num. Anal.*, to appear.

Lynch, R.E. and Rice, J.R., (1978), ''High accuracy finite difference approximation to solutions of elliptic partial differential equations'', *Proc. Nat. Acad. Sci.*, **75**, 2541–2544.

Marinescu, D.C. and J.R. Rice, (1987a), ''Domain oriented analysis of PDE splitting algorithms'', *J. Info. Sci.*, **42**, to appear.

Marinescu, D.C. and J.R. Rice, (1987b), ''Analysis and modeling of Schwarz splitting algorithms for elliptic PDE's''. In *Advances in Computer Methods for Partial Differential Equations, VI* (Stepleman and Vishnevetsky, eds), IMACS, 1-6.

McFaddin, H.S. and J.R. Rice, (1987), ''Parallel and vector problems on the FLEX/32'', CSD-TR-661, Computer Science, Purdue University.

Ribbens, C., (1986), ''Domain mappings: A tool for the development of vector algorithms for numerical solutions of partial differential equations'', Ph.D. Thesis, Purdue University.

Ribbens, C.J. and J.R. Rice, (1986), ''Realistic PDE solutions for nonrectangular domains'', CSD-TR-639, Computer Science, Purdue University.

Rice, J.R., (1985), ''Problems to test parallel and vector languages'', CSD-TR-516, Computer Science, Purdue University.

Rice, J.R., (1986a), ''Parallelism in solving PDEs'', *Proc. Fall Joint Compiler Conf.*, IEEE, 540-546.

Rice, J.R., (1986b), ''Multi-FLEX machines: Preliminary report'', CSD-TR-612, Computer Science, Purdue University.

Rice, J.R., (1986c), ''Design of a tensor product population of PDE problems'', CSD-TR-628, Computer Science, Purdue University.

Rice, J., (1986), ''Adaptive tensor product grids for singular problems''. In *Algorithms for the Approximation of Functions and Data*, (J. Mason, ed.), Oxford University Press.

Rice, J.R., (1987b), ''ELLPACK: An evolving problem solving environment''. In *Problem Solving Environments for Scientific Computing* (B. Ford, ed.) North-Holland, to appear.

Rice, J.R., (1987c), ''Parallel methods for partial differential equations''. In *The Characteristics of Parallel Computations*, (Jamieson, Gannon, Douglass, eds), MIT Press, Chapter 8, 209-231.

Rice, J.R., (1987d), ''Using supercomputers today and tomorrow''. In *Proc. Fourth Army Conf. Appl. Math. Computing*, 1333-1343.

Rice, J.R., and R.F. Boisvert, (1985), "Solving elliptic problems using ELLPACK", Springer Verlag.

Rice, J.R., W.R. Dyksen, E.N. Houstis, and C.J. Ribbens, (1986), "ELLPACK status report". CSD-TR 579, Computer Science, Purdue University.

Rice, J.R., Houstis, E.N. and Dyksen, W.R., (1981), "A population of linear, second order, elliptic partial differential equations on rectangular domains, Parts 1 and 2", *Math. Comp*, **36**, 475–484.

Rice, J.R., (1984a), "Numerical computation with general two dimensional domains". *ACM Trans. Math. Software*, **10**, 443–452.

Rice, J.R., (1984b), "Algorithm 624: A two dimensional domain processor". *ACM Trans. Math. Software*, **10**, 453–562.