1990

//ELLPACK: A Numerical Simulation Programming Environment for Parallel MIMD Machines

Elias N. Houstis
Purdue University, enh@cs.purdue.edu

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

N. P. Chrisochoides

H. C. Karathanasis

P. N. Papochiou

See next page for additional authors

Report Number:
90-949

//ELLPACK; A NUMERICAL SIMULATION PROGRAMMING ENVIRONMENT FOR PARALLEL MIMD MACHINES

E. N. Houstis
J. R. Rice
N. P. Chrisochoides
H. C. Karathanasis
P. N. Pappachiou
M. K. Samartzis
E. A. Vavalis
Ko Yang Wang

CSD-TR-949
January 1990

//ELLPACK: A NUMERICAL SIMULATION PROGRAMMING ENVIRONMENT FOR PARALLEL MIMD MACHINES*

E.N. Houstis, J.R. Rice, N.P. Chrisochoides,
H.C. Karathanasis, P.N. Papachiou,
M.K. Samartzis, E.A. Vavalis, and Ko Yang Wang

Computer Sciences Department
Purdue University
Technical Report CSD-TR-949
CAPO Report CER-90-7
January, 1990

* This work was supported in part by AFOSR 88-0243, ARO grant DAAG29-83-K-0026 and NSF grant CCF-8619817.

1 Introduction

In this paper we present the implementation of an "intelligent" mathematical software system for the parallel processing of second order elliptic partial differential equations (PDE) and describe its software components. The system is referred throughout with the acronym //ELLPACK since it is a superset of the well known ELLPACK system [Rice 85]. The design of //ELLPACK is based on a scenario for future numerical simulation systems which are capable of accommodating users with different computational objectives and implemented on a distributed hardware facility involving high power parallel machines. Its design objective is to provide a uniform programming environment for implementing parallel MIMD PDE solvers, automatic partitioning and allocation of the PDE computation, a very high level problem specification language, an interactive high level environment for grid selection, a domain partitioning and mapping facility, a uniform environment for obtaining software engineering measurements, and a graphical display of the solution output. The //ELLPACK system is implemented on a hardware facility consisting of graphics workstations supporting the X11 window system and connected to NCUBE, ALLIANT and SEQUENT machines through a wide bandwidth local network. The software infrastructure of //ELLPACK includes i) a man machine interface consisting of a PDE problem oriented language and X11 facilities for composing, editing and executing a //ELLPACK program, and geometry tools for specifying the PDE domain and its boundary conditions, ii) a PDE solution preprocessing subsystem capable of automatically generating orthogonal meshes, a domain decomposition tool for partitioning and allocation of the specified computations and a PDE solution specification/selection tool, iii) the //ELLPACK libraries for each target parallel machine built assuming a hierarchical structure of PDE solvers with fixed interfaces and iv) a PDE postprocessing subsystem consisting of facilities to collect, analyze and visualize performance data and tools for visualizing the computed solution.

This paper is organized as follows: Sections 2 and 3 describe our scenario for future numerical simulation systems and hardware facilities. The software infrastructure of //ELLPACK is presented in Section 4. The description of the various software components is described in Sections 5 to 8. The final version of this paper will include performance data for the NCUBE hardware.

2 A scenario for future numerical simulation methods

It has been predicted in [Noor 83], [Rice 88], and [Hous 89a] that in the 1990's we will see the widespread use of distributed computer facilities organized hierarchically with respect to their computational power and connected with appropriate netware. They will consist of powerful graphics workstations, parallel MIMD systems with tens of processors in the billion instruction per second range, parallel MIMD systems with hundreds of processors in several million instructions per second and SIMD machines with several thousands (maybe close to a million) processors. In the meantime, the necessity of closing the gap between hardware and software technology has been recognized by many as one of the fundamental problems of parallel computation. The future hardware facilities will require "intelligent" software tools capable of exploiting the enormous power of the cooperating computational engines, while making the idiosyncrasies of such facilities transparent to the application user. The parallel (//)ELLPACK group at Purdue University has established a research program to develop a programming environment and software tools that attempt to reduce the parallel computation overhead for certain applications governed by partial differential equations (PDEs), while allowing the PDE algorithm designer to specify them in a reasonable time with good implementation mappings to the future hardware facilities.

It is clear that the current monolithic designs of numerical simulation systems are not flexible enough to be mapped efficiently on the future hardware facilities. Furthermore, the new generation of numerical solution systems will be characterized by interactiveness at many levels, decision making and feed back. Figure 1 displays the facets of future numerical simulation systems. It includes high level user interfaces for specifying the component of PDE problem in textural graphical formats, tools for modeling and manipulating the geometry of the problem domain, facilities for mapping the underlying computations to the selected machines, "intelligent" components for selecting the efficient method/machine pairs for the specified problem, performance evaluation and I/O data visualization tools. The main objective of the //ELLPACK project is to study the requirements of such systems, develop appropriate infrastructure and realize an instance of this scenario that attempts to address many of the issues related to the parallel processing of PDEs.

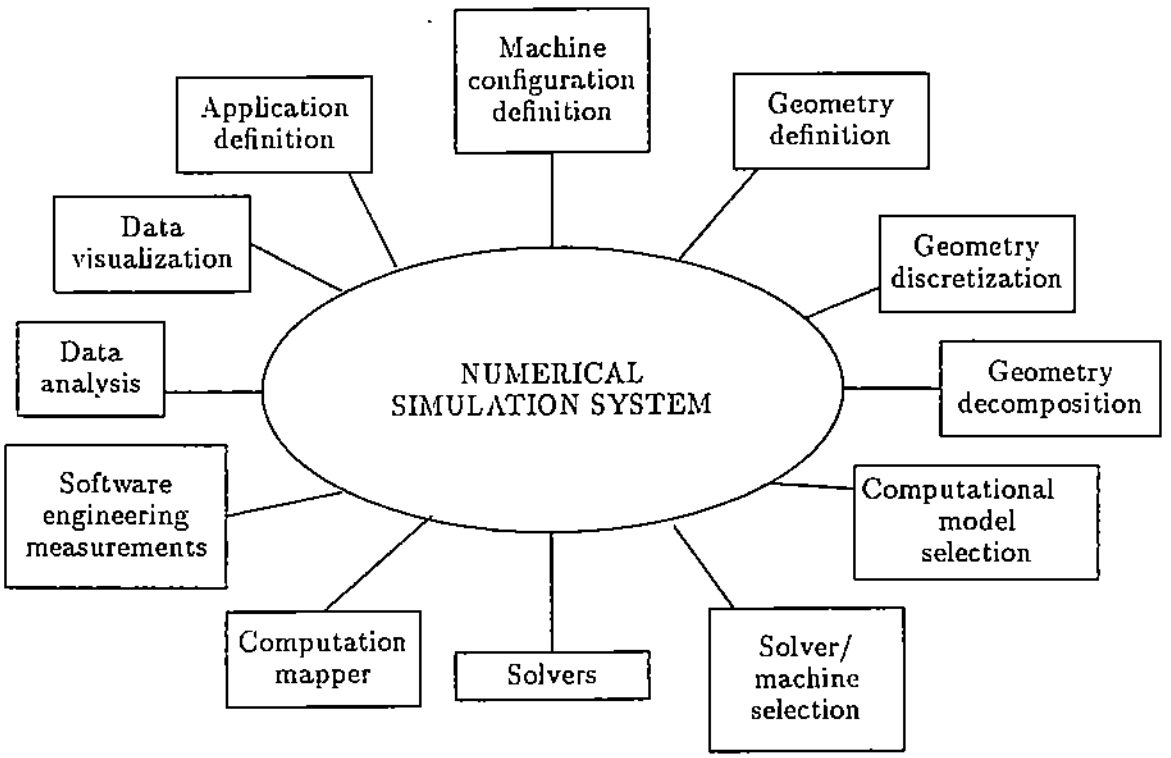

Figure 1: The facets of future numerical simulation systems.

3 //ELLPACK: a realization of a future numerical simulation system

In this section we describe the design of a PDE solving system capable of supporting the parallel processing of elliptic PDEs on MIMD machines and which incorporates many of the facets of the next generation numerical simulation systems depicted in Figure 1. The system //ELLPACK can be considered a superset of ELLPACK with new facilities for determining parameters of certain parallel solvers, modified module interfaces and a new man-machine interface. A preliminary design of the system was reported in [Hous 89a]. In this paper and the technical report [Hous 89b] we discuss the detailed structure and functionality of its current implementation. The software infrastructure of //ELLPACK is described in Figure 2 and can be grouped into six subsystems: *the user interface, the PDE problem specification, PDE solution preprocessing, PDE solution. run time support* and *PDE postprocessing*. The components of these subsystems and their interactions are indicated in Figures 3 and 4.

4 Man-machine interface

The //ELLPACK man-machine interface consists of a X11-window subsystem that allows the user to compose and execute //ELLPACK programs using state-of-the-art text and graphical tools. Figure 5 depicts the layout of the //ELLPACK control window. This subsystem features an interactive editor for composing or modifying //ELLPACK programs textually, a geometry specification tool to specify the PDE domain and boundary conditions. a geometry discretization tool to generate and manipulate meshes, a solver specification tool to choose appropriate solution paths and parameters, an execution control tool to select and configure the target machine and run the program, a domain decomposition tool to partition and map the underlying computation to the selected configuration of the target machine, a performance monitoring tool to analyze the performance of the system, and a data and output visualization tool to visualize various data structures or the results of the computation. An expert system interface for tool interaction plus algorithm and machine selection is under investigation. Following we will describe these software tools in detail.

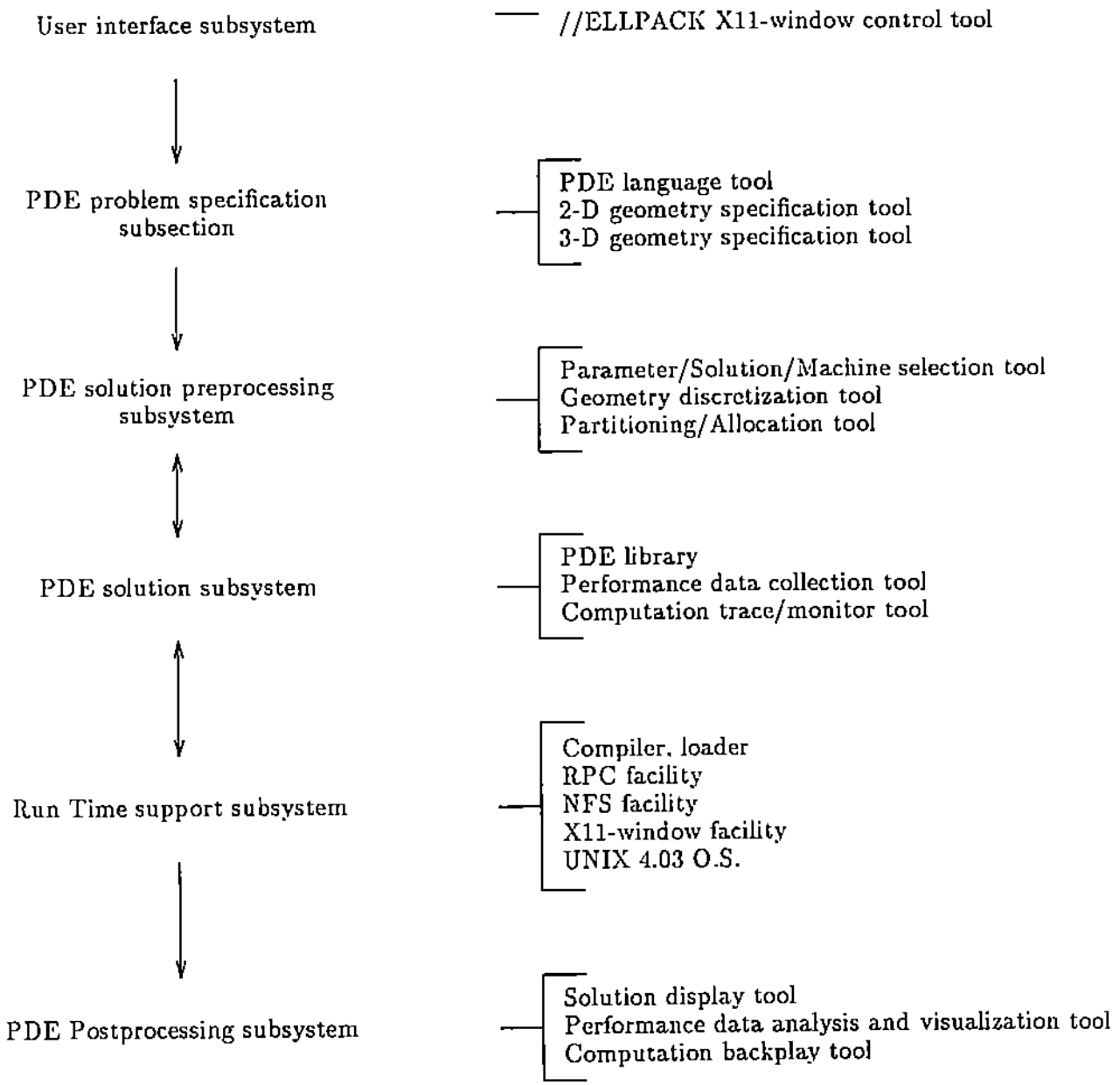

Figure 2: The software infrastructure of //ELLPACK.

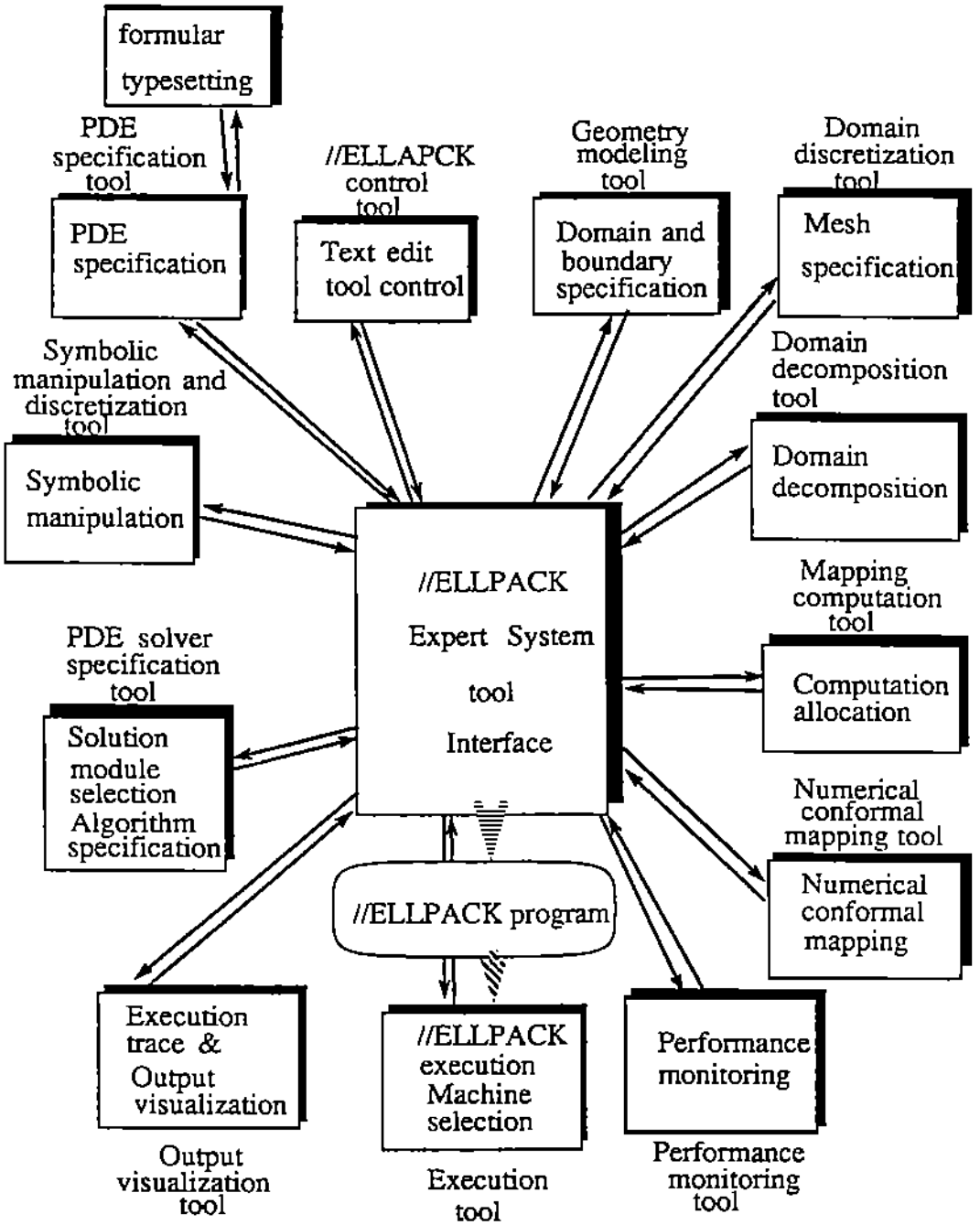

Figure 3: The software organization of //ELLPACK.

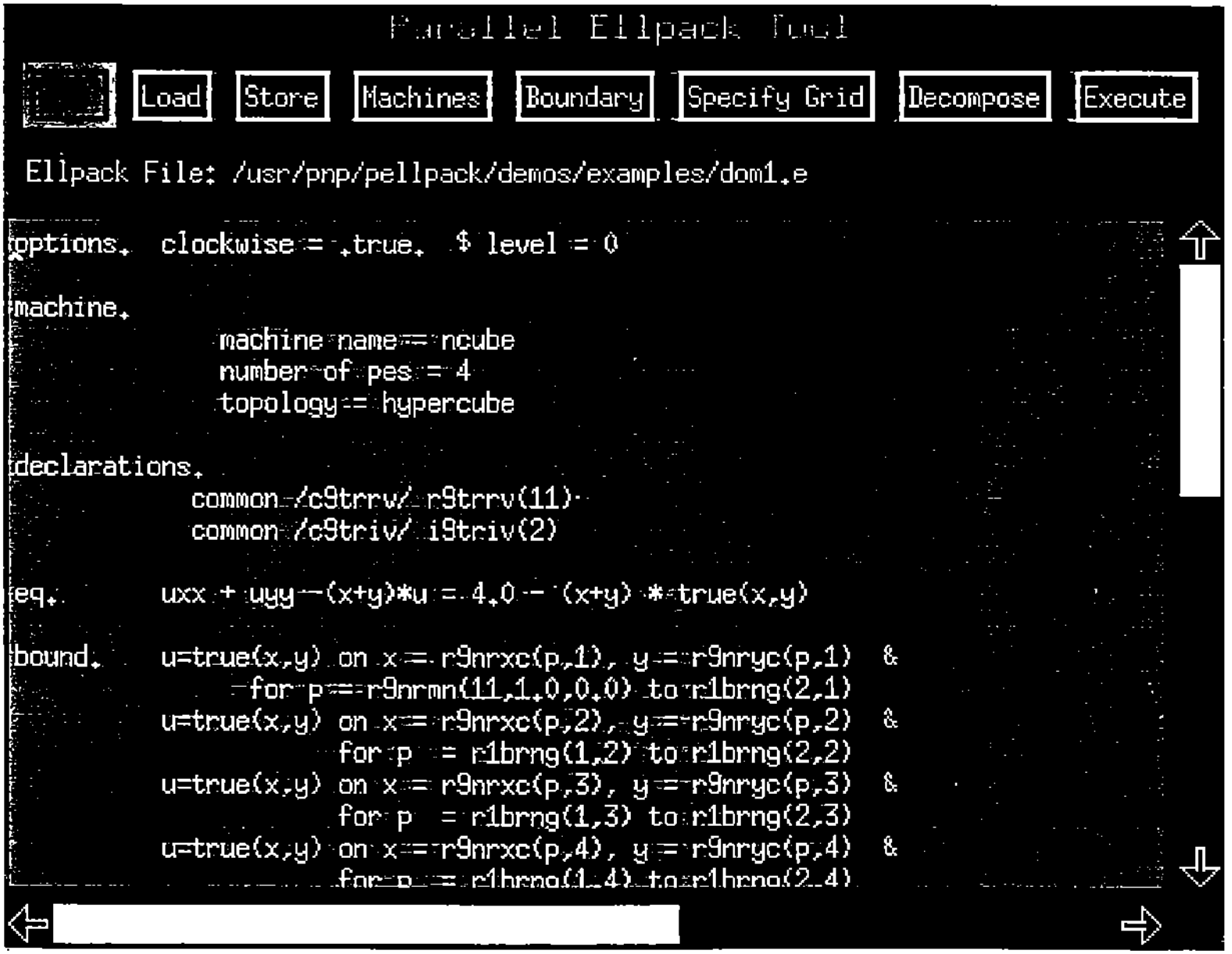

Figure 4: The control X11-window for the user interface to //ELLPACK.

4.1 //ELLPACK PDE language

//ELLPACK provides a very high level PDE problem/solution statement language, it supports facilities specifying PDE equations and domains, defining domain decomposition methods, and selecting solution algorithms. It can generate FORTRAN code for multiple target machines including parallel and sequantial ones for which an appropriate //ELLPACK library exists. The syntax of //ELLPACK language is described in [Hous 89b]. Figure 4 presents an example of a //ELLPACK program to be executed on a four processor NCUBE machine using domain decomposition, 5-point star discretization, and a Jacobi SI iteration method. The //ELLPACK preprocessor currently can generate code for NCUBE and INTEL hypercubes. Its modification is under way for the SEQUENT, ALLIANT and SUPRENUM MIMD machines. We are designing a general PDE specification language that can handle different types of PDE problems, 1-D, 2-D or 3-D domains, regular or irregular boundaries, linear or non-linear parameters, and time dependent or time independent problems. Furthermore, we are adding a new facility that will allow foreign systems to be invoked and obtain input information from the //ELLPACK environment.

4.2 Geometry specification tool

According to ELLPACK [Rice 85] the boundary of a 2-D PDE domain with or without holes is specified piecewise in terms of parametric representation of each boundary piece. This tool allows the user to specify each boundary piece graphically using a cursor driver device (currently a mouse) and input the corresponding boundary conditions. Currently the specification of each boundary piece is done through a set of points that can be considered as the control points to Bernstein polynomials [Klin 90] or the interpolating points of a spline. Figure 5 and 6 depict the layout of this tool and elements specified using using the above approaches. For 3-D domains, we will interface the PROTOSOLID system [Vane 89] in collaboration with the CAPO geometry group. Figure 6 depicts an instance of this tool.

5 PDE solution preprocessing subsystem

The design objectives of this subsystem includes (a) the selection of grid/ configuration and method/machine pairs based on specified accuracy/ performance requirements and (b) the partitioning/allocation of the underlying

```
option. level = 1
machine.  machine name = ncube $ number of pes = 4
equation. Uxx + Uyy -(x+y) * U = 4.0 - (x+y) * true(x,y)
boundary. U = true(x,y) on x = 4. + .1 * P * (P-4.5)**2,&
                          y = -.5 + P                    FOR P = 0. TO 4.5
          U = true(x,y) on x= 5-P, y = 4.                FOR P = 1. TO 4.
          U = true(x,y) on x= 1., y = 4.5-P              FOR P = .5 TO 4.
          U = true(x,y) on x= 1. + 3*P, y = .5 - P       FOR P = 0. TO 1.

grid.   21 x POINTS 1. TO 5.5  $  21 y POINTS -.5 TO 4.

mesh.   finite element  (type = rectangle)

decomposition.  domains(numdomains=4)
subdomains.
      domain 1: (xi .le. 10) .and. (yi .le. 10)
      domain 1: (xi .eq. 11) .and. (yi .le. 10) .and. (yi .ge. 5)
      domain 2: (11,1), (11,2), (11,3), (11,4)
      domain 2: (xi .ge. 12) .and. (yi .le. 9)
      domain 2: (xi .ge. 12) .and. (xi .le. 16) .and. (yi .eq. 10)
      domain 3: (xi .eq. 9)  .and. (yi .ge. 11) .and. (yi .le. 18)
      domain 3: (xi .ge. 10) .and. (yi .ge. 11) .or. &
                (xi .ge. 17) .and. (xi .le. 19) .and. (yi .eq. 10)
      domain 4: ((xi .le. 8) .and. (yi .ge. 11))
      domain 4: (9,19), (9,20)
interfaces.
      domain 1: (xi .ge. 2) .and. (xi .le. 11) .and. (yi .eq. 11)
      domain 2: (11,3), (11,4), (11,5)
      domain 2: (xi .eq. 12) .and. (yi .ge. 5)  .and. (yi .le. 10)
      domain 3: (xi .ge. 12) .and. (xi .le. 17) .and. (yi .eq. 11)
      domain 3: (17,10), (18,10), (19,10)
      domain 4: (xi .eq. 9)  .and. (yi .ge. 12) .and. (yi .le. 19)
      domain 4: (10,19), (10,20)

discretization. 5 point star

solution. jacobi si.

fortran. +node.
      call q9asth
fortran. +host.
      call q9gsfn

visualization. display data with x11(i9nsub,21,21)
visualization. display functions with x11(u, error)

subprogram.  +both.
      function true(x,y)
      true = x*x + y*y
      return
      end
end.
```

Figure 5: An example //ELLPACK program.

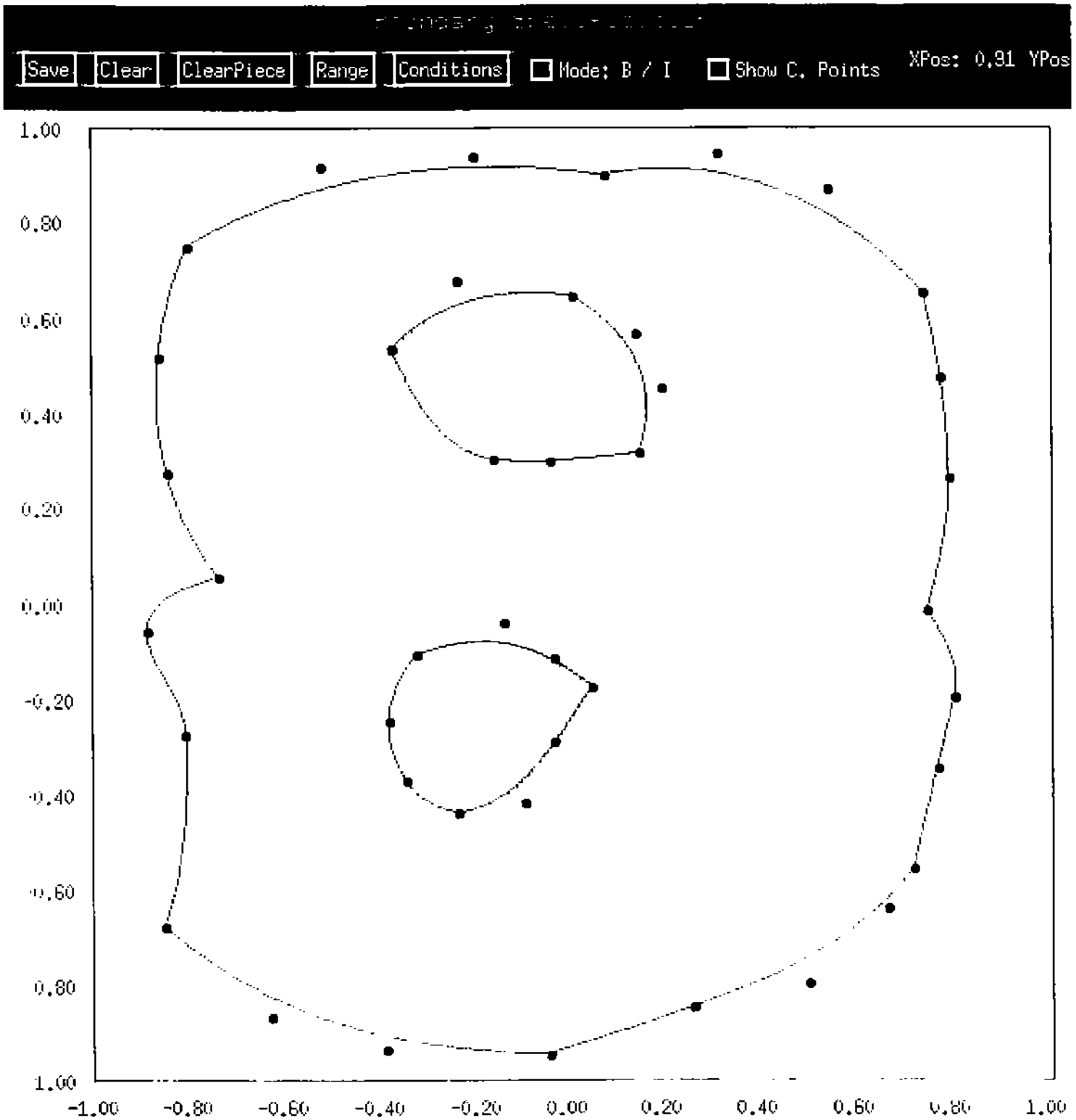

Figure 6: Example window from the //ELLPACK geometry modeling tool for 2-D domains showing the control buttons and the piecewise definition of a domain.

computation into the selected machine. For the implementation of (a) objective, we are developing an expert system [Hous 90]. Currently the grid can be specified through an interactive tool and methods can be selected from the displayed options. For the implementation of the (b) objective, we have developed two separate tools corresponding to partitioning procedures based on geometry mapping strategies.

5.1 Geometry discretization tool

For the generation of orthogonal meshes, we are using the ELLPACK's 2-D domain processor [Rice 85]. For the display and modification of such meshes (moving,removing,adding mesh lines) we have developed the tool depicted in Figure 7. For 3-D polyhedra domains, we will use the PROTOSOLID system with its own orthogonal mesh generator [Vane 89]. Figure 8 depicts the 3-D geometry tool. We are currently evaluating various finite element mesh generators to be incorporated in the //ELLPACK environment.

5.2 Domain decomposition tool

The parallel processing of PDE computations requires the partitioning and allocation of the underlying computations to fit the targeted architecture. This problem can be formulated and solved from the continuous or discrete geometric data, the algebraic data or at the data flow graph of the computation. We have implemented a partitioning strategy based on finite element or difference meshes referred as *domain decomposition*. The goal of this strategy is to subdivide the domains in load balanced subdomains with minimum interface length. The domain decomposition tool has its own user interface and supports different heuristic methods [Chri 89] for the partitioning problem. These heuristics differ in the degree of optimality with respect to the interface length, topology and algorithm complexity. The user can modify an automatically obtained decomposition interactively or define one manually. Currently, the allocation is implemented by the solvers based on the information provided by the domain decomposer. Figure 9 depicts the layout of this tool, while its functionality and performance are presented in [Chri 89].

5.3 PDE solver specification tool

In the //ELLPACK environment, there is more than one solution path (sequence of methods to be applied) for a given problem. The selection of

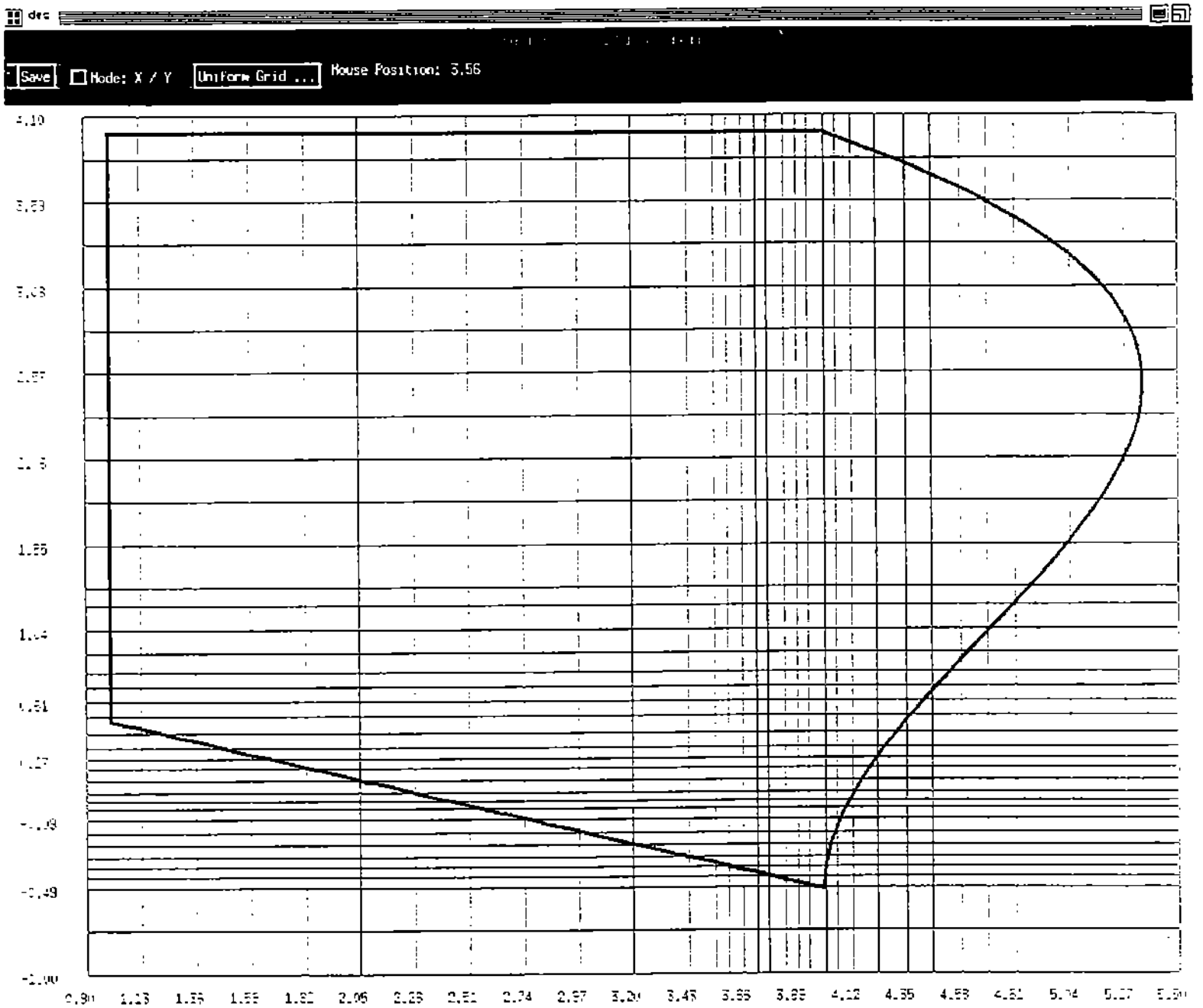

Figure 7: Example window from the 2-D geometry discretization tool.

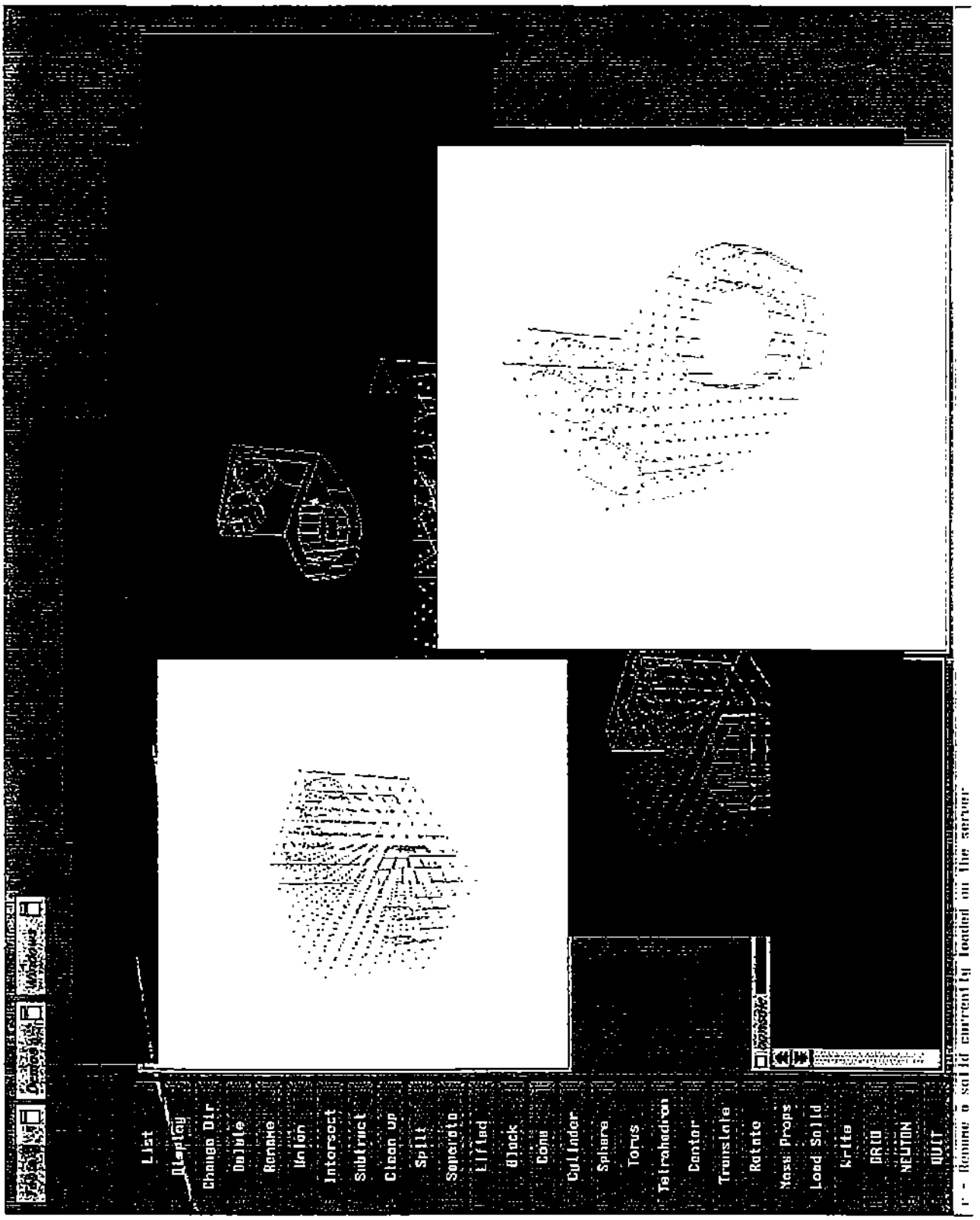

Figure 8: Example window from the prototype of the 3-D geometry modeling/discretization tool.

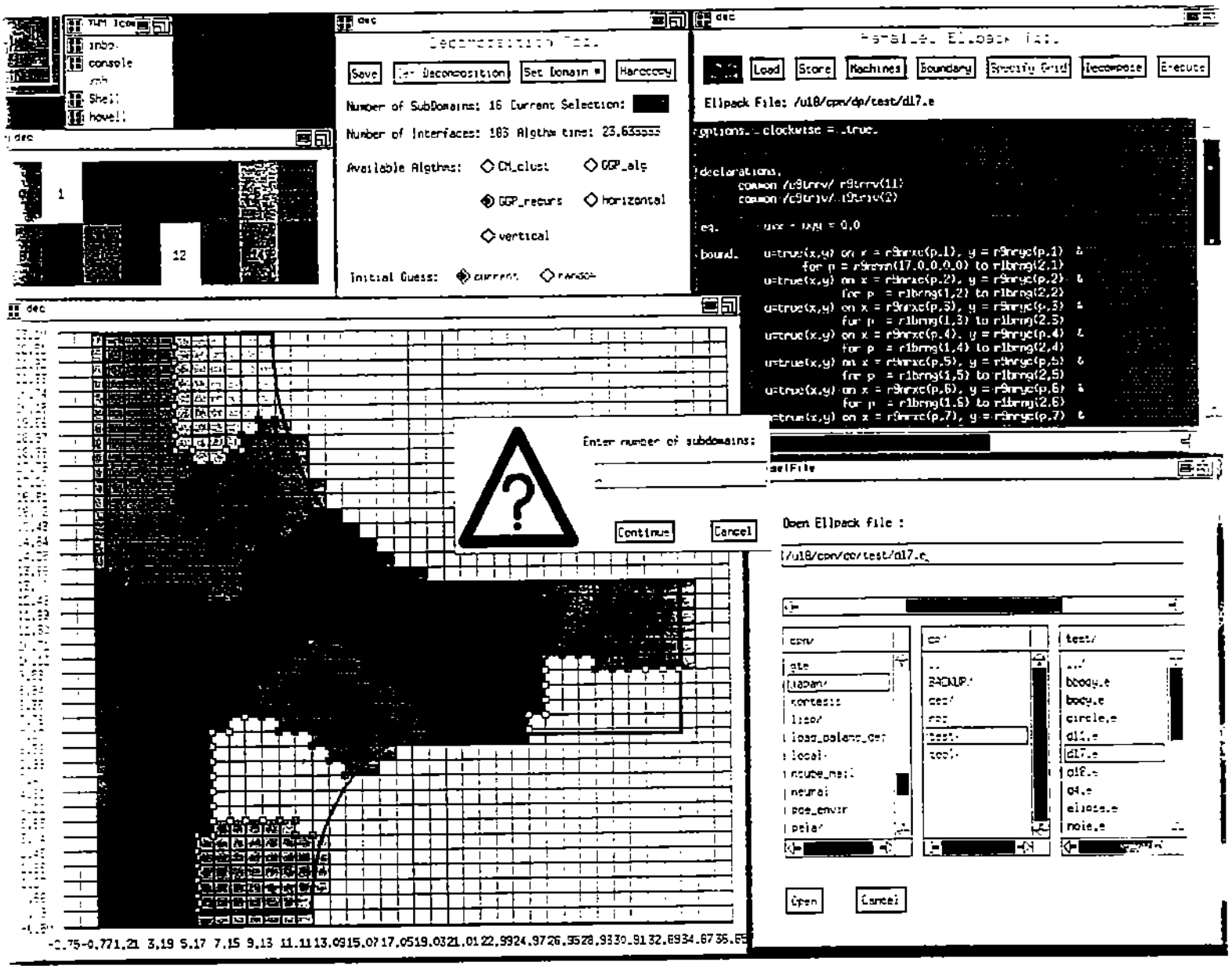

Figure 9: Example window showning the layout and functionality of the domain decomposition tool.

the path can be done in three levels. First, the user can specify a solution path *manually* by consulting the //ELLPACK manual and using the computation segments of the //ELLPACK language. Second, an *advisor system* will be available which will indicate to the user the possible applicable solution paths to the given problem. This facility eliminates the reference to the //ELLPACK manual, but the final decision is still left to the user. Finally, an *expert system* will be available capable of choosing the final solution path for the user. In the case of sequential ELLPACK, we will use the advisor/expert system developed by Dyksen and Gritter [Dyks 90]. For the //ELLPACK modules, a similar system is under development [Hous 90]. Figure 10 presents the layout of this tool. Most of its buttons or menus correspond to the //ELLPACK language segments.

6 PDE solution subsystem

The architecture of the //ELLPACK library is based on the assumption that an MIMD PDE solver consists of seven distinct tasks which communicate with each other through fixed interfaces. These tasks correspond to mathematical steps needed to obtain a PDE solution in a parallel MIMD computational environment. These steps are: a) PDE domain decomposition, b) Partitioning of the underlying computation into parallel parts, c) Mapping of the underlying computation onto the target machines, d) PDE equation discretization, e) Preprocessing of the solution data structures, f) Discrete equations solution, and g) Postprocessing of the output data.

In our first implementation of this library, we have assumed that the partition and allocation of the computation is determined on the domain discretization level using the geometry decomposer whose data are distributed in all processors. In order to take advantage of the existing discretization modules, we perform global indexing of the unknowns and equations at the front end or host of the parallel machine and broadcast it to individual processors. Each individual processor generates only the equations associated with its subdomains. The algebraic equations data are stored locally in sparse mode using the same data structures as ELLPACK with some additional structures for the interface or overlapping data assuming either *block, substructuring* or *wrap-around* ordering schemes. In the case of multisegment ELLPACK PDE solvers the solution is obtained in four steps or phases.

Phase 1 deals with domain discretization, and partitioning of the under-

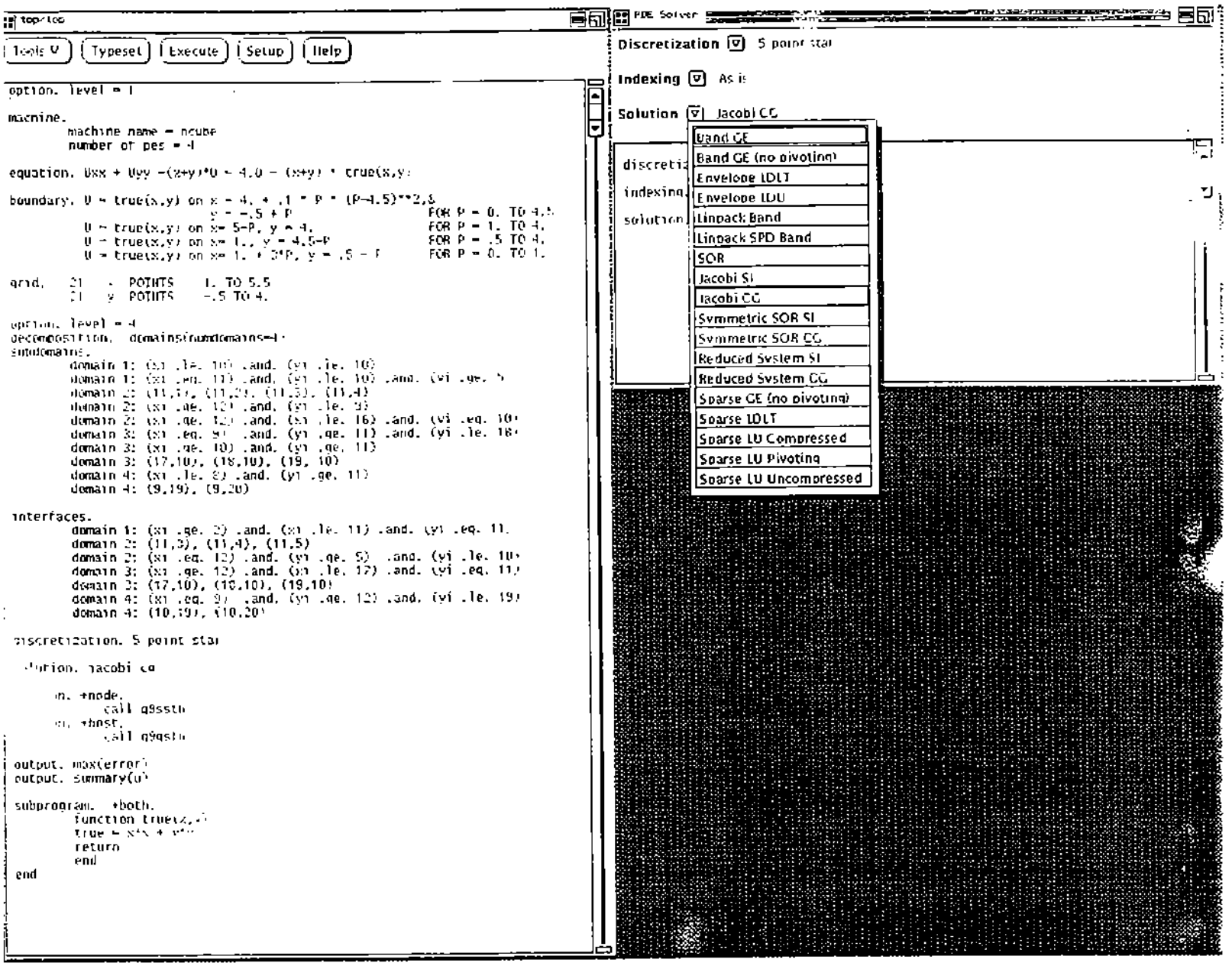

Figure 10: Example window showing the layout of the PDE solver specification tool where the //ELLPACK program is displayed (left) as it is created by selecting numerical modules (right).

lying computation. This phase takes place in the front-end or host of the parallel system.

Phase 2 generates and stores the distributed data in the local memories of the available processors in the ordering scheme defined in Phase 1.

Phase 3 carries out the solution of the discretized PDE using the algebraic data structure defined in Phase 2.

Phase 4 deals with postprocessing of the solution and its derivatives.

The current implementation of these phases are described in [Hous 89b]. Most of the modules used to implement the above phases are parallelized versions of the corresponding sequential ELLPACK ones. The interested reader should refer to [Rice 85] for their complete description. Tables 1, 2 and 3 present a brief view of the existing //ELLPACK library, the compatibility of the modules and their applicability. It is worth noticing that the *direct solvers* use wrap-around or sparse matrix indexing, while the *iterative solvers* use block or substructuring indexing of the algebraic equations.

7 PDE postprocessing subsystem

The responsibilities of this subsystem are: (a) collect the data for performance analysis and visualization, (b) monitor of the computation, and (c) display the computed solution in various forms. We have implemented two tools to support (a) and (c), while (b) will be implemented using the TRIPLEX software system [Krum 89].

7.1 Performance evaluation facility

The //ELLPACK performance evaluation facility consists of three parts - the collection, analysis and visualization of the //ELLPACK program performance data. The performance data collecting facility provides a common base for both basic program performance measurement and sophisticated performance evaluation of different numerical algorithms and their implementations. Performance data can be collected at different granularity levels of the program, ranging from a single numerical module to the whole program. Primitives are also provided to collect data for arbitrary program blocks, such as each iteration of an iterative method. Both communication and computation time data are collected. The code for performance data

Table 1: The current //ELLPACK-NCUBE library.

Module Reference	Method
5-point star	Finite difference on general 2-D domains
P2C1COL	Quad. spline collocation for rectangular domains
P3C1COL	Hermite collocation for general 2-D domains
P3C2COL	Cubic spline collocation for rectangular domains
Finite Element	Linear quad. elements on general domains

Table 2: The solution modules of //ELLPACK-NCUBE library

Module	Ordering Scheme	Method
Jacobi-CG	block, arrow-head	Jacobi conjugate gradient
Jacobi-SI	block, arrow-head	Jacobi with Chebyshev acceleration
SOR	block, arrow-head	Successive over relaxation
SSOR CG	block, arrow-head	Symmetric SOR conjugation gradient
SSOR SI	block, arrow-head	Symmetric SOR with Chebyshev acceleration
Jacobi Schwarz	block	Schwarz splitting with Jacobi iterations
GS Schwarz	block	Schwarz splitting with Gauss-Seidel iterations
Band Gauss Elimination	wrap-around	Gauss elimination
Multilevel Elimination	block	Gauss elimination
Parallel Sparse	sparse	Gauss elimination

Table 3: The "triple" modules of //ELLPACK-NCUBE library.

Module Reference	Assumptions
5-point star/Jacobi-SI	Strip domain partitioning
5-point star/Jacobi-CG	Strip domain partitioning
5-point star/Schwarz	Tensor product rectangular splitting
P3C2 spline/Schwarz	Tensor product rectangular partitioning

the appropriate option is set or primitive invoked. This subsystem records the timing of each of the modules plus other important context information such as names and types of the modules and the order that the modules appeared in the //ELLPACK program. These data can be stored permanently in a database so that systematic performance evaluation of numerical algorithms is possible. This subsystem is a separate entity from the //ELLPACK program and can be used to do several kinds of performance evaluation and to support the development of "expert" systems for the grid/configuration and method/machine selection problems. It also includes tools for the user to select, compose and compare performance data of different numerical algorithms. Primitives for data comparison include computing mean values, variances, maximum, minimum, communication/computation ratios, communication hot spots location, and other performance indicators. The performance data visualization tool allows the user to select combinations of different categories of performance data and to visualize them in different graphical forms. Currently the visualization of this data is done in bar chart format, more sophisticate graphic representation is under development. The layout of this tool is illustrated in Figure 11.

7.2 Data and output visualization facility

The //ELLPACK data and output visualization facility is intended to provide a graphical representation of the data structures and PDE solutions obtained. The //ELLPACK visualization and output statements direct the //ELLPACK preprocessor to generate code for preparing data for visualization. Statements exist to visualize 2-D, 3-D data structures or results of 2-D, 3-D problem domains for the specified functions.

To visualize 2-D data structures or producing 3-D plots of functions on 2-D domains, a X-11 visualization tool is built. The current implementation uses an "alpha" release of the XView toolkit from Sun. The routines for drawing 3D surfaces can be easily adapted to other X11 toolkit or non-toolkit applications because they only use Xlib calls. The user is able to select the function to be graphed from a menu of available functions. After that a window with the 3D-plot of the requested function is brought up. The plot window(s) can be resized and a number of the parameters associated with the plot changed interactively. These are the horizontal and vertical rotation angles, the resolution of the graph, the z-intercept and the type of the graph. The window is then updated to reflect the new changes. The current version of this tool is based on an early implementation of 3D-

Figure 11: Example screen using the performance evaluation tool.

plot facility in Interactive ELLPACK and XELLPACK [Bono 90]. Most of the code has been migrated from FORTRAN to C and adapted to the requirements of an interactive X application. Figure 12 shows an instance of this tool.

The 3-D plots of a function on 2-D domain utilize both height and colors to achieve the 3-D visual effect on a 2-D display monitor. Unfortunately, this is not possible for functions on 3-D domain, only the color can be used to distinguish the values of elements in the 3-D domain. For a color representation of a 3-D volume, one needs the ability to "peel" part of the volume away to see the actual values inside the volume. For this purpose, we are currently using the "NCSA X Data Slice"(XDS) [NCSA 89] from the University of Illinois to view slices of the 3-D data volume. The //ELLPACK preprocessor generates code to translate the data values into the format that XDS requires and then XDS is invoked to view data slices interactively. Other methods to visualize 3-D data are under investigation. The 3-D data visualization tool can be used to visualize any data structures or functions on the 3-D domains including the decomposition and mapping of 3-D domains and the 3-D meshes.

Figure 12: Example screen showing the output display tool usage.

References

[Bono 90] Bonomo, J. and W.R. Dyksen, *XELLPACK: An Interactive Problem-Solving Environment for Elliptic Partial Differential Equations*, in Intelligent Mathematical Software, (Editors: E. N. Houstis, J. R. Rice and R. Vichnevetsky), Elsevier, to appear, 1990.

[Chri 89] Chrisochoides, N.P., C.E. Houstis, E.N. Houstis, S.M. Kortesis, and J. Rice, *Automatic load balanced partitioning strategies for PDE computations*, in Intern. Conf. on Supercomputing, 1989, pp. 99-107.

[Dyks 90] Dyksen, W.R. and C. Gritter, *Expert system for the solution of elliptic partial differential equations* Technical report. Purdue University, West Lafayette, IN, 1990, in preparation.

[Klin 90] Klinkner, S., *Graphical Editing of Algebraic Surfaces Models - A Toolkit*, Technical report, Purdue University, West Lafayette, IN, 1990, in preparation.

[Hous 89a] Houstis, E.N., T.S. Papatheodorou, and J.R. Rice. *Parallel ELLPACK: An expert system for the parallel processing of partial differential equations.* Math. Comp. Simul., **31**, 1989, pp. 497-508.

[Hous 89b] Houstis, E.N., J.R. Rice, N.P. Chrisochoides, H.C. Karathanasis, P.N. Papachiou, M.K.Samartzis, E.A. Vavalis, and K. Wang, *Parallel (///) ELLPACK PDE solving system* CAPO technical report CER-89-20, Purdue University, West Lafayette, IN, 1989.

[Hous 90] Houstis, E.N., C.E. Houstis, J.R. Rice and P. Varodoglou, *ATHENA: An Expert System for //ELLPACK*, submitted to Second International Conference of Expert Systems for Numerical Computing.

[Krum 89] Krumme, D.W., A.L. Couch, B.R. House and Jon Cox, *The Triplex Tool Set for the NCUBE Multiprocessor*, Technical Report, Tufts University, June 27, 1989, 112 pages.

[NCSA 89] NCSA group, *NCSA X Data Slice for the X Window System*, Technical report, Nat. Ctr. Supercomputer Appl., University of Illinois at Urbana-Champaign, Sept., 1989.

[Noor 83] Noor, A.K. (Editor), *State of the art surveys on finite element technology*, The American Society of Mechanical Engineers, 1983.

[Rice 85] Rice, J.R. and R.F. Boisvert, *Solving Elliptic Problems Using ELLPACK*, Springer-Verlag, New York, 1985.

[Rice 88] Rice, J.R., *Supercomputing About Physical Objects*, Supercomputing, (eds, E. Houstis, T.S. Papatheodorou, C.D. Polychronopoulos), Springer Verlag, New York, 1988, pp. 443-455.

[Vane 89] Vanecek, G. Jr., *PROTOSOLID: An inside look*, CAPO report CER-89-26, Department of Computer Science. Purdue University, November 1989, 36 pages.