//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, Ko Yang Wang and S. Weerawarana

Department of Computer Science
Purdue University
West Lafayette, IN 47907, U.S.A.

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 parallel (//) 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 powered 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 and finite element 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: Section 2 describes //ELLPACK as a realization of the future scenario for numerical simulation systems. Sections 3 to 6 present the description of its various software components.

2 //ELLPACK: A Realization of a Future Numerical Simulation System

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

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.
© 1990 ACM 089791-369-8/90/0006/0096...$1.50

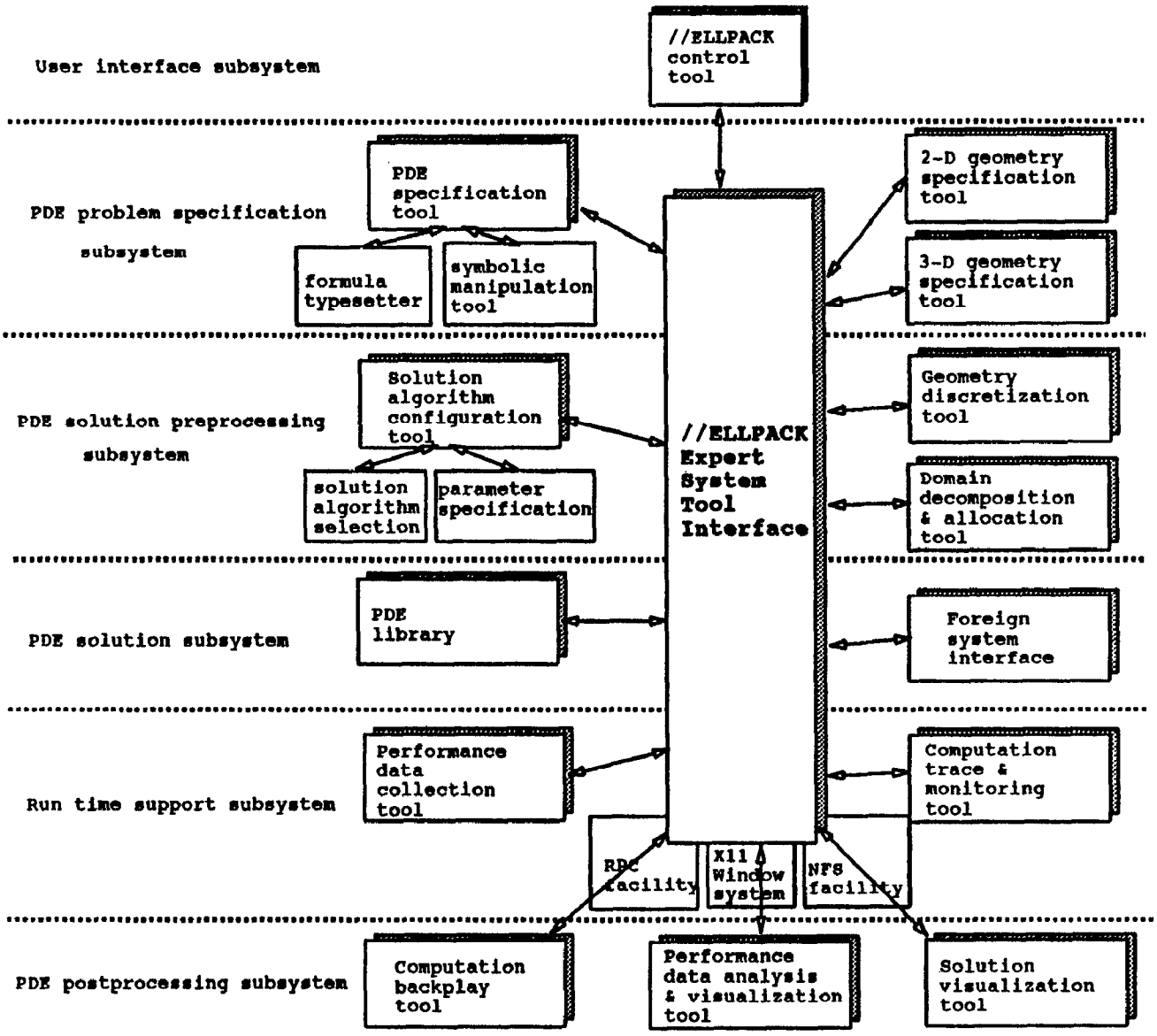

Figure 1: The software infrastructure of //ELLPACK.

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. They will include 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 second order PDEs used to model "field" problems.

The developing system referred as //ELLPACK can be considered as 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 1 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 the same figure.

3 The Man-Machine Interface

The //ELLPACK man-machine interface consists of four X-11 window subsystems referred throughout as tools, which are used to specify the components of an elliptic PDE problem (operator, domain, boundary conditions) symbolically, textually and/or graphically. Figure 2 depicts the layout of the PDE and geometry specification tools. Their generated output is saved and displayed in the //ELLPACK control tool in the form of a very high level language program. This tool is presented in Figure 2 which in addition features an interactive editor for modifying or composing a //ELLPACK program textually and facilitates the control of the //ELLPACK system by allowing the activation/deactivation of the various subsystems.

The //ELLPACK language is a very high level PDE problem/solution statement language. It supports facilities specifying PDE equations and domains, defining domain decompositions, and selecting solution algorithms. It can generate FORTRAN code for multiple target machines including parallel and sequential ones for which an appropriate //ELLPACK library exists. The syntax of //ELLPACK language is described in [Hous 89b]. The //ELLPACK preprocessor currently can generate code for a NCUBE hypercube. Its modification is under way for the SEQUENT, ALLIANT, SUPRENUM and CEDAR MIMD machines. Furthermore, we are adding a new facility that will allow "foreign" simulation systems to be invoked, obtain input information from the //ELLPACK environment and display output using the //ELLPACK visualization tools.

The PDE specification tool supports an interface to *Maxima* and *Fortran*, while it permits the specification of the PDE coefficients and the corresponding right side in a template form. The output of this tool is saved as ELLPACK equation coefficients or Fortran segments.

The geometry specification tool allows the graphical representation of 2-D domain boundary. Following 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 interactively. 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 cubic spline. These points can be moved or developed interactively. For 3-D domains,

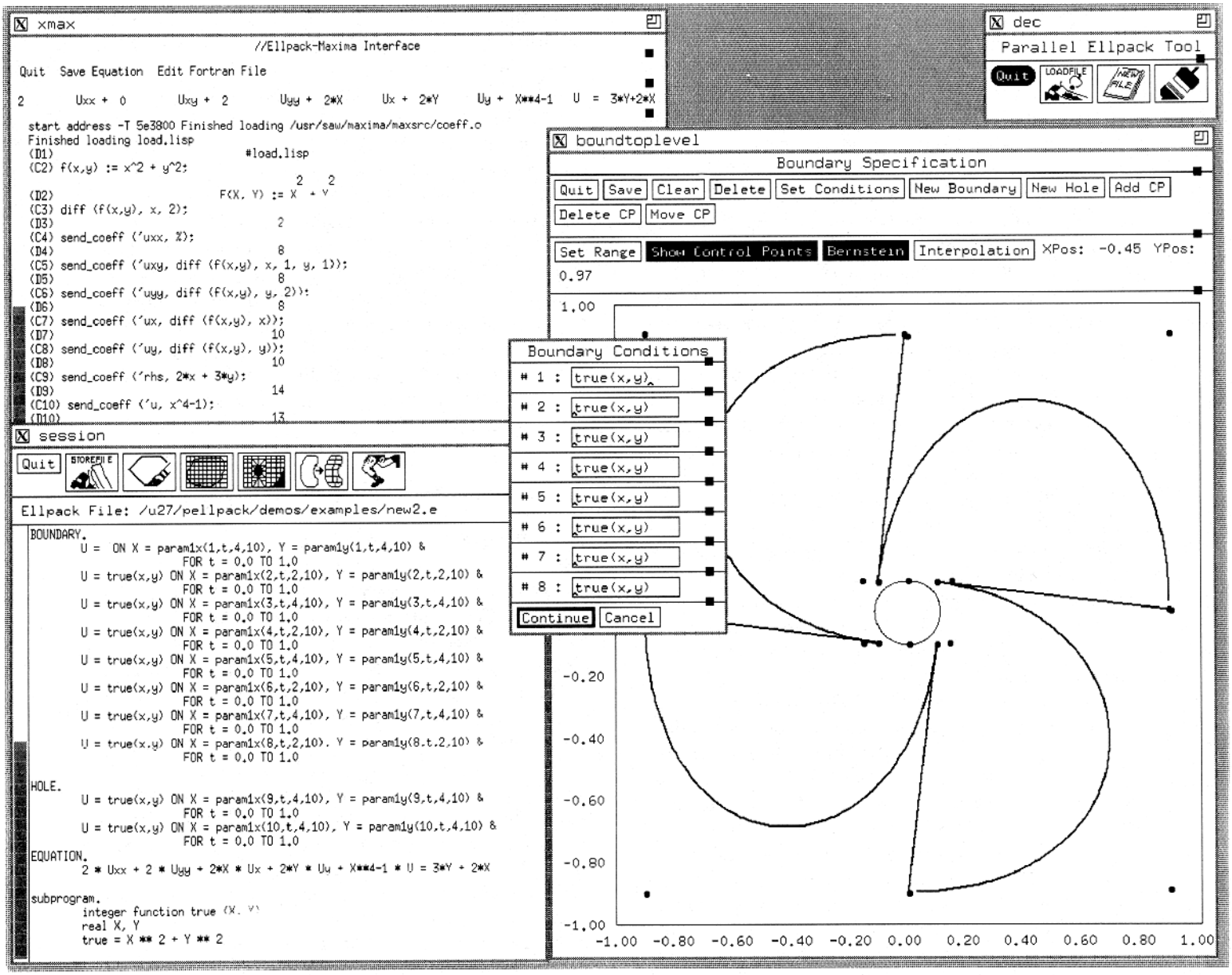

Figure 2: The X11-windows for PDE and geometry specification and //ELLPACK control tools.

we are interfacing the ProtoSolid system [Vane 89] in collaboration with the CAPO geometry group.

4 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 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 a list of displayed options. For the implementation of the (b) objective, we have developed two separate tools corresponding to partitioning procedures based on geometry mapping strategies [Chri 90]. Following we give brief descriptions of these tools.

The Geometry discretization tool is used to display and manipulate 2-D orthogonal and finite element (FEM) meshes on 2-D grids generated by the ELLPACK's 2-D domain processor [Rice 85]. Figure 3 depicts instances of such meshes and the functionality supported by the tool. In case of orthogonal meshes, lines can be moved, removed or added, while for FEM meshes, the nodes can be moved without guaranteeing regularity. For 3-D polyhedra domains, we will use the ProtoSolid system with its own orthogonal mesh generator [Vane 89].

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 on the continuous or discrete geometric data, the algebraic data or at the data flow graph of the computation. We have developed and implemented several partitioning strategies based on finite element or difference meshes referred as *domain decomposition* techniques. The goal of these techniques is to subdivide the domains in load balanced subdomains with minimum interface length. For the partitioning of PDE computations, we have developed a software tool called *domain decomposer* supported by different domain decomposition strategies [Chri 89]. This tool has its own user interface and provides different heuristic solutions to the *continuous* or *discrete* 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 4 depicts the layout of this tool, while its functionality and performance are presented in [Chri 89], [Chri 90].

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

5 PDE Solution Subsystem

The software architecture of the current //ELLPACK library is influenced significantly by the structure of the geometry decomposition solvers. Its design is based on the requirements of the so-called "*continuous*" and "*discrete*" geometry decomposition techniques. In the first case, a partitioning/mapping of the continuous PDE domain is obtained, using a global course mesh and a sequential ELLPACK program is defined over each subdomain and executed in each processor with user defined continuous interface conditions. The "small" PDE problems are solved repeatedly until convergence to the global solution is reached. The discrete domain decomposition solvers are based on a partitioning of a global mesh of the PDE domain and usually 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 these solvers, continuous or discrete data are distributed in all processors. Each individual processor

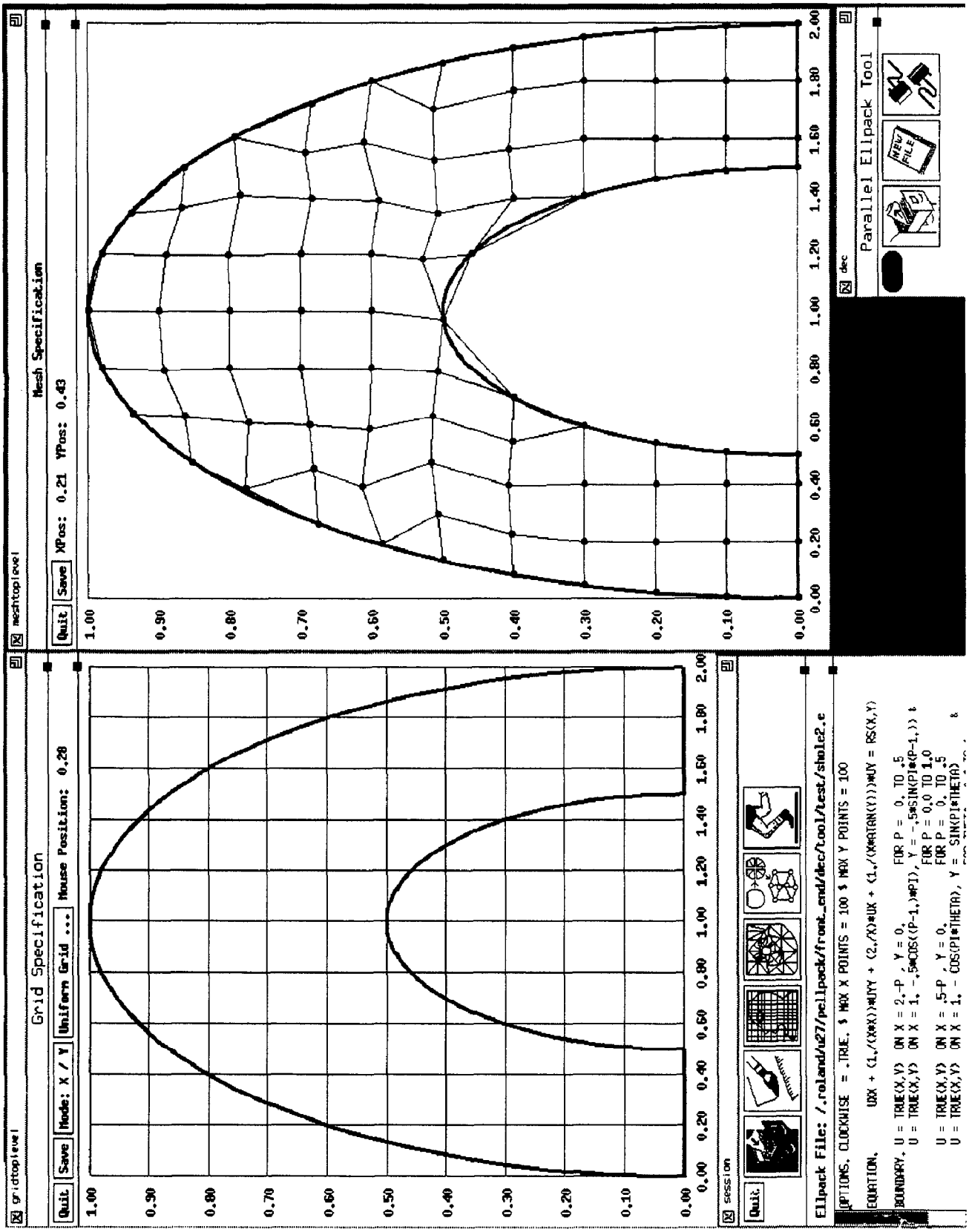

Figure 3: 2-D geometry discretization tool.

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 unknowns. In the case of multi-segment ELLPACK PDE solvers the solution is obtained in four steps or phases.

[Phase 1] deals with domain discretization, and partitioning/allocation of the underlying computation. Currently this phase takes place in the front-end or host/node of the parallel system. Its parallel implementation is underway.

[Phase 2] generates and stores the distributed data in the local memories of the available processors according to the partitioning/allocation 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 discrete domain decomposition solvers.

6 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].

6.1 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 collection 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 collection is generated automatically by the //ELLPACK preprocessor when 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 4.

6.2 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 has been built. The current implementation uses the ATHENA X-11 toolkit. 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 asso-

Figure 4: Domain decomposition tool interface.

Table 1: The //ELLPACK-NCUBE discretization modules.

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 quadrilateral/brick 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

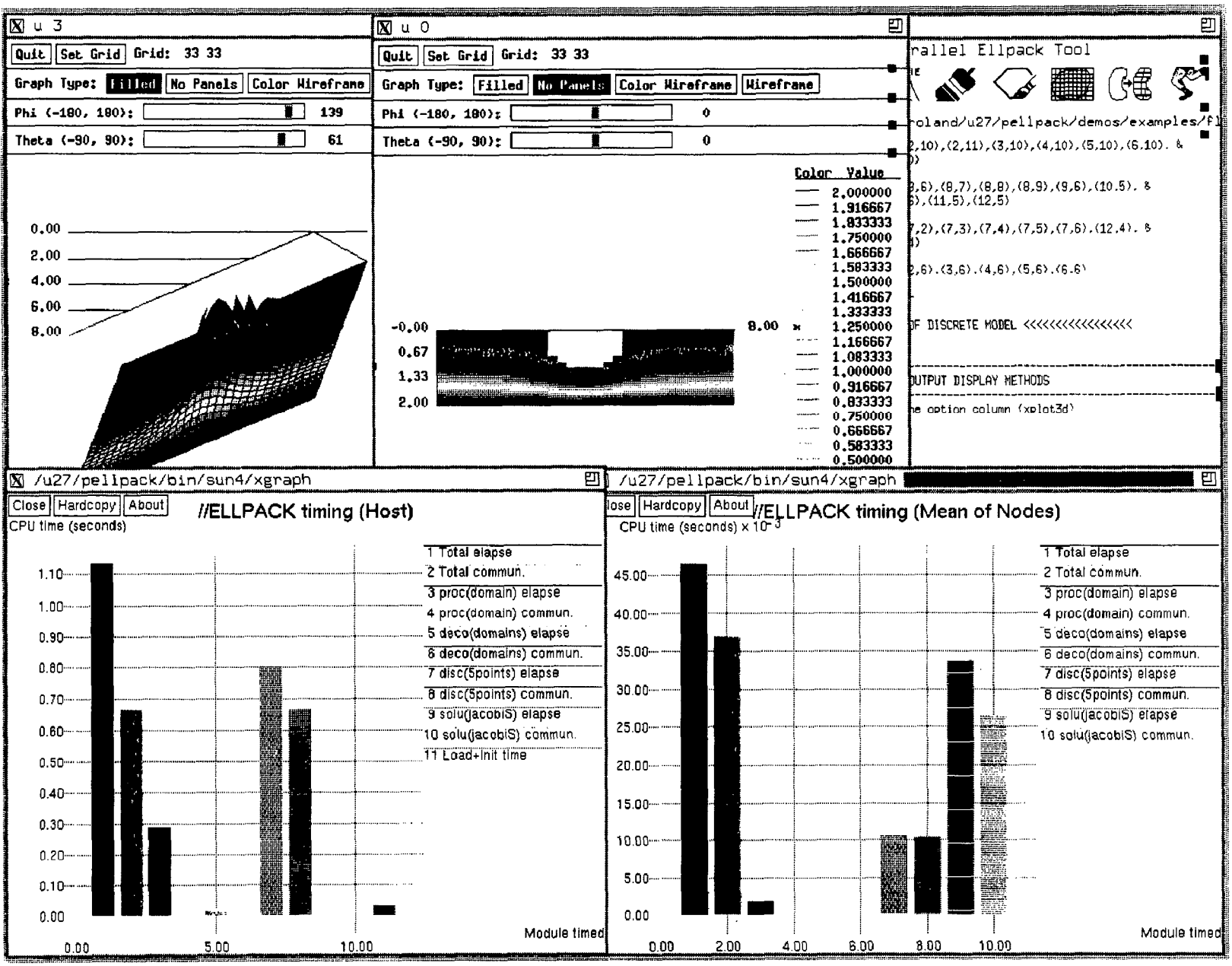

Figure 5: Solution and performance data display tools.

ciated 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-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 4 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) facility [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.

Acknowledgement

This research was supported by AFOSR 88-0234, ARO grant DAAG29-83-K-0026, NSF grant CCF-8619817. The research of the first author was also partially supported by ESPRIT GENESIS project.

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, 1990, to appear.

[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 Inter. Conf. on Supercomputing, 1989, pp. 99-107.

[Chri 90] Chrisochoides, N.P., C.E. Houstis, and E.N. Houstis, *Geometry Based Mapping Strategies for PDE Computations*, CAPO Technical Report CER-90-16, Purdue University, West Lafayette, IN, 1990.

[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] Norr, 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, West Lafayette, Indiana, November 1989, 36 pages.