1995

//ELLPACK: A System for Simulating Partial Differential Equations

Sanjiva Weerawarana

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

Ann C. Catlin

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

Report Number:
95-043

//ELLPACK: A System for Simulating Partial Differential Equations

S. Weerawarana, Elias N. Houstis
Ann C. Catlin, and John R. Rice
Computer Sciences Department
Purdue University
West Lafayette, IN 47907

CSD-TR-95-043
June, 1995

//ELLPACK: A System for Simulating Partial Differential Equations *

Sanjiva Weerawarana, Elias N. Houstis, Ann C. Catlin and John R. Rice
Department of Computer Sciences
Purdue University
1398 Computer Science Building
West Lafayette, IN 47907–1398, USA.
E–mail: {saw,enh,acc,jrr}@cs.purdue.edu

Abstract

The rapid evolution of high performance computing (HPC) platforms has vastly improved the computing environment available for numerical simulation. With the advances in hardware technology, the complexity of writing software that makes effective use of the available hardware has also increased dramatically. Our research is aimed at solving this problem for application domains modeled by partial differential equations, by developing easy to use, high level software environments called problem solving environment that allow users to quickly make efficient use of available HPC platforms. This paper reports on //ELLPACK[1], a problem solving environment for the numerical simulation of PDE models on high performance computing platforms.

//ELLPACK allows the user to (symbolically) specify partial differential equation problems, specify the solution algorithms to be applied, solve the problem and finally analyze the results produced. The problem and solution algorithm are specified in a custom high level language through a complete graphical editing environment. A knowledge based environment assists users in selecting an optimal solution strategy for a given problem. Problems can be solved using any of the built–in solvers, or the user can introduce a new solver into the system and leverage from the infrastructure in the system. Problems may be solved on a parallel machine (or a network of workstations) using built–in domain decomposition based solvers.

Keywords: Problem solving environments, parallel computing, PDEs.

1 Introduction

Partial differential equations are important and commonly arising mathematical models for modeling physical phenomena. Due to mathematical complexity, most partial differential equation problems cannot, however, be solved analytically. They are commonly solved by finding a (discrete) approximate solution using some numerical solution scheme and then interpolating the solution over the entire problem domain. A large collection of software to numerically solve various PDE problems on a variety of machines exists today, both in the public domain and commercially.

A Problem Solving Environment (PSE) is a computer system that provides all the computational facilities necessary to solve a target class of problems [1]. These features include advanced solution methods, automatic or semiautomatic selection of solution methods, and ways to easily incorporate novel solution methods. Moreover, PSEs use the language of the target class of problems and provide a "natural" user interface, so users can run them without specialized knowledge of the underlying computer hardware or software. By exploiting modern technologies such as interactive color graphics, powerful processors, and networks of specialized services, PSEs can track extended problem solving tasks and allow users to review them easily. Overall, they create a framework that is all things to all people: they solve simple or complex problems, support rapid prototyping or detailed analysis, and can be used in introductory education or at the frontiers of science. PSEs will play a vital role in making HPC based scientific computing systems accessible to application scientists who are not experts in computer science.

//ELLPACK is a problem solving environment for solving PDE problems on high performance computing platforms as well as a development environment for building new PDE solvers or PDE solver components. The numerical solvers currently available in //ELL-

*Work reported herein was supported in part by NSF awards ASC 9404859 and CCR 9202536, AFOSR award F49620-92-J-0069, ARPA ARO award DAAH04-04-G-0010 and an award from the Intel Foundation.

[1] "//ELLPACK" is read as "parallel ELLPACK."

PACK include not only solvers that we have developed, but also a large collection of solvers developed by others. As a problem solving environment, //ELLPACK has a "natural" language for specifying PDE problems and their solution schemes. A complete graphical user interface assists users in specifying the PDE problem and the solution algorithm and also for analyzing computed solutions. Problems may be solved sequentially or in parallel on a variety of supported parallel platforms.

Given the variability of PDE models, there is, as one might expect, a plethora of PDE solving systems. The types of systems available range from highly flexible solver libraries (for example, [2, 4, 9]) to more rigid problem solving environments for "standard" PDE models (for example, ABAQUS, ANSYS and NASTRAN). The //ELLPACK system is aimed at providing users with the same level of flexibility available to users of low-level PDE libraries, but with the same level of convenience available in problem solving environments for standard PDE models. PDE/Protran [3] and Visual DEQSOL [7] are also systems with a similar view. //ELLPACK differs from these in many ways, including the internal software architecture which is based primarily on standardized interfaces and the portability across sequential and parallel machines.

This paper is organized as follows: Section 2 provides a brief overview of the //ELLPACK system. Section 3 discusses the software architecture of //ELLPACK and Section 4 discusses the parallel computing methodologies used. Finally, Section 5 makes some concluding remarks.

2 Overview

The main design objective of //ELLPACK is to create an intelligent software environment where both sequential and parallel PDE solvers can be implemented in a reasonable time. //ELLPACK presents application users with a high level environment to abstractly specify PDE problems and build solvers for them using intrinsic solver components. "Knowledgeable" users apply //ELLPACK's solver development facilities to build new solvers which are then available as intrinsic solver components for application users.

The //ELLPACK system evolved from the well-known ELLPACK system [5]. //ELLPACK has evolved ELLPACK in many ways, including the addition of parallel solution facilities, a complete graphical user interface, symbolic manipulation of PDE problems and finite element solution methods. The basic infrastructure of //ELLPACK described below thus extends the infrastructure present in ELLPACK to support the added functionality.

The usage model adopted by //ELLPACK is based on decomposing the PDE problem and the solution process to its natural constituent parts. In this view, a PDE problem is said to consist of a partial differential equation, the domain over which this equation holds, the conditions (equations) that hold at the boundary of the domain and the condition that holds at the beginning of the simulation (for time-dependent problems). For a large class of numerical solution techniques, these components allow the solution process to be decomposed to domain discretization (generating the discrete domain over which the numerical solution is computed), operator discretization (generating the linear or nonlinear algebraic equations that need to be solved) and solving the resulting system of equations. Other solution schemes combine all of these steps into one composite step that results in the final PDE solution. For parallel solutions, any or all of these steps may be executed on a parallel machine.

The original ELLPACK system was designed to support the specification and solution of 2nd order, 1-, 2- and 3-dimensional linear elliptic PDE problems, using primarily finite difference solution techniques. The //ELLPACK system still uses this same basic infrastructure, but has added to it to better support nonlinear and time dependent problems, as well as using both finite difference and finite element techniques. A new system we are currently developing, PDELab [8], introduces a completely new infrastructure that intrinsically supports general PDE operators, and solves them using many different solution techniques.

When solving a problem with //ELLPACK, one must first specify the PDE problem, the solution algorithm to be applied and the desired outputs using the high level PDE language and/or the graphical tools. This specification is compiled by translating it to a FORTRAN driver program and then linking it with the appropriate libraries. After the resulting executable is run, the computed data may be imported into the output environment for visualization and analysis. The entire process is managed via the top level graphical environment of //ELLPACK.

In the rest of this section, we provide very brief overviews of the PDE language, the graphical user interface and the solver libraries in //ELLPACK.

PDE Language. The //ELLPACK language is an extension of the ELLPACK language and includes extensions for dealing with domain-decomposition based parallel solvers and finite element methods. The ELLPACK language is viewed as an extension of FORTRAN and allows experienced users to insert arbitrary FORTRAN code to create complex control structures for ELLPACK computations. The ELLPACK model views the PDE problem in terms of its intrinsic components (as described earlier) and decomposes the PDE solution process into the distinct steps of domain discretization, operator discretization, linear system reordering, linear system solution and output. Users are

```
options.
        time
equation.
        uxx + uyy = 0
boundary.
        uy = 0 on y = 0
               on y = 1
        u = 5  on x = 0
               on x = 1
grid.
        10 x points $ 10 y points
discretization.
        5 point star
solution.
        jacobi cg
output.
        table(u)
end.
```

Figure 1: An example //ELLPACK program to solve the Laplace equation on the unit rectangle using 5–point–star finite differences and the Jacobi CG linear solver on a 10 x 10 grid.

allowed to select appropriate modules for each such phase, leading to literally hundreds of possible solution paths for a given problem. Standard interfaces are defined between these modules with the goal of being a repository for contributor provided modules. The language is implemented as a translator that generates a FORTRAN program which is them linked with pre–written libraries to solve the problem at hand. Figure 1 shows an example //ELLPACK program.

Graphical User Interface. The collection of graphical tools and the structure of the //ELLPACK language directly follow the PDE problem and solution algorithm decomposition described earlier. For example, the 2-dimensional domain specification language and editing tool allows one to specify any two dimensional domain using a boundary parameterization. Similarly, an equation specification tool allows one to specify the equation that holds on the interior of the domain. The equation tool also provides some symbolic operations for manipulating the PDE operator, including operator linearization and discretization. Figure 2 illustrates some of the graphical tools.

Solver Libraries. The libraries of solvers available in //ELLPACK include both finite difference and finite element solution schemes. In addition to the solver components that we have developed locally, we have integrated several publicly available PDE codes into //ELLPACK to demonstrate its extensibility features. The systems we have thus integrated include VECFEM [2], FIDISOL [6], and PDECOL [4]. For solving the system of linear equations that arise in many solution techniques, //ELLPACK includes several of the widely used linear equation solvers including ITPACK, LINPACK, SPARSKIT and NSPCG. It also includes parallel implementations of several operator discretization schemes and Parallel ITPACK, an implementation of ITPACK for distributed memory parallel machines. The parallel libraries have been implemented for distributed memory parallel machines as well as networks of workstations. The code has been ported to and tested on an nCUBE/2, an Intel iPSC/860, an Intel Paragon and networks of Unix workstations. The parallel solvers can be used with a wide variety of communication systems, including PICL, PARMACS and MPI. The parallel computing methodology used in //ELLPACK is discussed further in Section 4.

Introducing New Solvers. It is clear that as a general problem solving environment, //ELLPACK cannot be expected a priori to solve every PDE application that is of interest. As such, //ELLPACK was designed so that users could easily integrate their own specialized solvers and solver parts into the environment. These users may then take advantage of the graphical environment along with pre– and post–processing support. In addition, the new solvers are now available for other //ELLPACK users.

Since the problem specification is defined by well–documented data structures, functions, and files formats, the integration process is straightforward. The new code must be modified to access the required data from //ELLPACK data structures, its name and any user–specifiable parameters are placed in the //ELLPACK software repository, and the modified code is installed in the //ELLPACK library system. In this way, "software parts" such as mesh generators, discretizers, and solvers can be added to the system with minimal effort. Some software parts with complex input requirements take significantly more effort to integrate. For example, some discretizers require the PDE equations and boundary conditions to be defined through FORTRAN functions. //ELLPACK can generate these functions through the graphical interface, using its symbolic manipulation capabilities which are available through the equation specification tool. Using this infrastructure, we have already integrated many varied and complex software parts.

3 Software Architecture

In order to realize the //ELLPACK computational environment, we have adopted three levels of programming with standardized data structures and interfaces among the various PDE objects involved in the solution process.

At the highest level, the graphical user interface provides application users with knowledge–based, object–oriented editors to define problem components, spec-

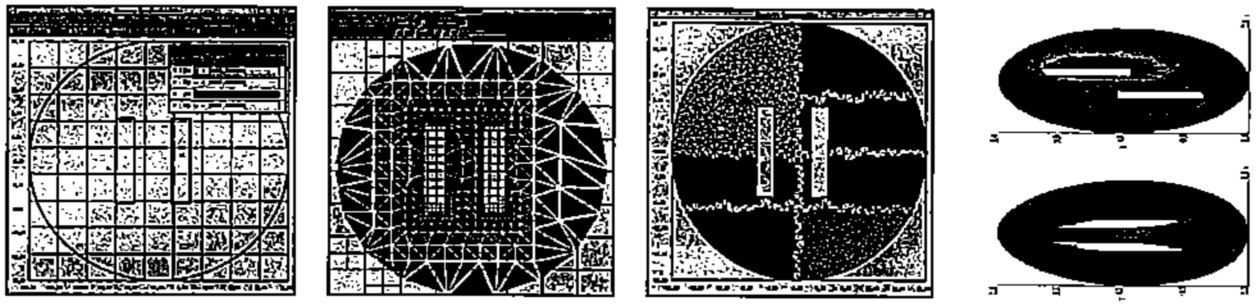

Figure 2: A sample of the graphical tools available in //ELLPACK.

ify the solution process and perform various post-processing analyses. The problem and solution specifications are expressed in terms of a high level PDE language, which is used to represent the PDE objects produced by the graphical editors. The //ELLPACK language processor is used to compile this high level problem and solution specification into a driver program that invokes various library modules to realize the user's solution process.

This architecture is implemented in //ELLPACK in terms of seven subsystems. These subsystems represent the solution process that application users follow. The *PDE Problem Specification Subsystem* and the *PDE Solution Specification Subsystem* provide users with graphical editors, "foreign" system templates, the //ELLPACK language and embedded FORTRAN code. The *PDE Libraries* implement sequential and parallel solver components that are available to users via the solution specification subsystem. They include the ELLPACK solver library, the //ELLPACK solver library and "foreign" solver libraries such as FIDISOL, VECFEM, PDECOL and PDEONE. The various components are interconnected using standardized interfaces which supports the "plug-and-play" approach provided by //ELLPACK. The *Knowledge Based Framework* assists users with the selection of an appropriate solution process for a given problem. The *Language Processor* uses the high level PDE language specification to generate a driver program that implements it. Furthermore, it is used to integrate new PDE solver components to the //ELLPACK system. The *Execution Subsystem* provides a framework for executing //ELLPACK programs. It helps users compile and execute programs on all the hardware and software platforms that //ELLPACK supports by managing the complexities associated with sequential and multi-platform parallel execution. The *Analysis Subsystem* provides users with graphical tools for visualizing and analyzing computed solutions and for collecting and visualizing performance data. Figure 3 illustrates this view of the //ELLPACK architecture.

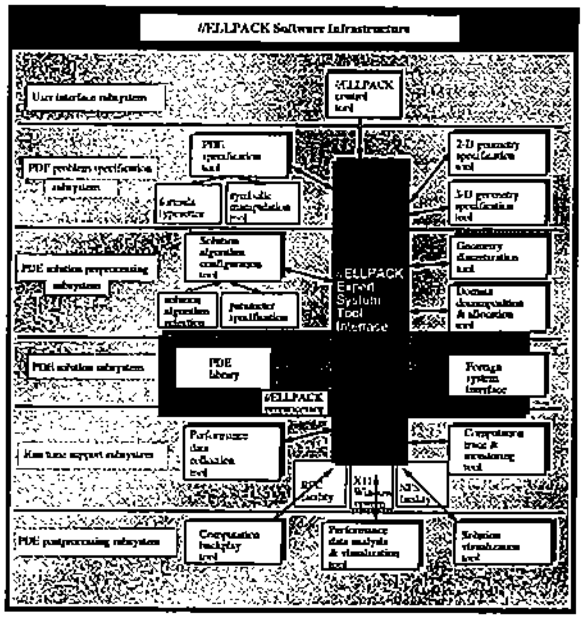
//ELLPACK Software Infrastructure
User interface subsystem
//ELLPACK control tool
PDE problem specification subsystem
PDE specification tool
formula typesetter
symbolic manipulation tool
2-D geometry specification tool
3-D geometry specification tool
PDE solution preprocessing subsystem
Solution algorithm configuration tool
solution algorithm selection
parameter specification
Geometry discretization tool
Domain decomposition & allocation tool
//ELLPACK Expert System Tool Interface
PDE solution subsystem
PDE library
Foreign system interface
//ELLPACK
Run time support subsystem
Performance data collection tool
Computation trace & monitoring tool
X11 Window system
NFS facility
PDE postprocessing subsystem
Computation backplay tool
Performance data analysis & visualization tool
Solution visualization tool

Figure 3: Subsystems view of the //ELLPACK software architecture.

4 Parallel Computing

Parallelism in //ELLPACK is achieved via domain decomposition. For PDE computations, the domain of the PDE problem is partitioned into subdomains, and computations are done on a per subdomain basis. This requires subdomain-aware numerical solver software, as well as information on what constitutes a subdomain. In order to analyze the results on the global domain, partial results are collected to generate the final global result.

Since significant software for the numerical solution of PDE models already exists, //ELLPACK is working toward the integration and parallelization of existing "legacy" software. We have developed three geometry based approaches to achieve this. The original approach was the development of customized parallel code. Existing sequential discretization and solution modules were parallelized by hand using the message passing model for the nCUBE 2 machine with

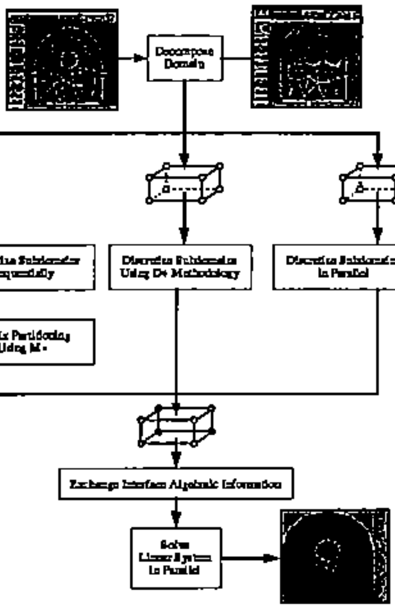
Decompose Domain
Discretize Subdomains Sequentially
Discretize Subdomains Using D+ Methodology
Discretize Subdomains in Parallel
Matrix Partitioning Using M+
Exchange Interface Algebraic Information
Solve Linear System in Parallel

Figure 4: Approaches for parallelization of PDE software.

nCUBE's native communication libraries. This code has since been migrated to several communication libraries (PICL, PARMACS, MPI), as well as to other hardware platforms (Intel Paragon, iPSC/860, and workstation network-based virtual parallel machines).

The second technique allows discretization code to run sequentially, and then passes the resulting linear system and geometry partition to a matrix partitioning module. This module sets up the solution phase, which is executed in parallel with available parallel solvers. The final technique executes sequential discretization code for each subdomain on a separate processor, and solves the linear system in parallel. Clearly, any method that attempts to reuse existing software parts is well justified. //ELLPACK uses available parallel linear solvers that have been implemented on distributed algebraic data structures obtained through "optimal" partioning of the PDE geometric data structures. Figure 4 illustrates these three approaches.

5 Conclusion

This document has provided a brief overview of the //ELLPACK system. We have used versions of this system internally at Purdue for various classes and research activities for about three years. It is currently being distributed as an alpha release and we expect to have a full public release within a short time. We hope that other researchers and academics as well as industrial users will find the system useful and will contribute to its future developments.

6 Acknowledgements

We wish to acknowledge the following people for their contribution to the realization of this system (in no particular order): Ko Yang Wang, Panos P. Papachiou, Sang-Bae Kim, Nikos P. Chrisochoides, E. A. Vavalis, Yu-Ling Lai, Cheryl L. Crabill, Margaret G. Gaitatzes, Chi Ching Chui, Shahani Markus, Amy L.S. Ng, Winnie Ng, Kostas N. Pantazopoulos, Nicholas E. Houstis, Ben Jackson, Parakevi N. Galani, Pelayia Veradogolou, M. K. Samartizs, H. C. Karanthanasis and Jian Zheng.

References

[1] E. Gallopoulos, E. Houstis, and J. Rice, *Problem-solving environments for computational science*, IEEE Computational Science and Engineering, 1 (1994), pp. 11–23.

[2] L. Gross and P. Sternecker, *The Finite Element Tool Package VECFEM*, University of Karlsruhe, 1991.

[3] IMSL, Inc., *PDE/PROTRAN User's Manual*, 1986.

[4] N. K. Madsen and R. F. Sincovec, *PDECOL: General collocation software for partial differential equations*, ACM Transactions on Mathematical Software, 5 (1979), pp. 326–351.

[5] J. R. Rice and R. F. Boisvert, *Solving Elliptic Problems Using ELLPACK*, Springer–Verlag, 1985.

[6] W. Schönauer, E. Schnepf, and M. Müller, *The FIDISOL Program Package*, University of Karlsruhe, Karlsruhe, Germany, 1985.

[7] Y. Umetani et. al., *Visual PDEQSOL: A visual and interactive environment for numerical simulation*, in Programming Environments for High-Level Scientific Problem Solving, P. W. Gaffney and E. N. Houstis, eds., North-Holland, 1992, pp. 259–269.

[8] S. Weerawarana et. al., *PDELab: An object-oriented framework for building problem solving environments for PDE based applications*, in Proceedings of the Second Annual Object-Oriented Numerics Conference, Rogue-Wave Software, Corvallis, OR, 1994, pp. 79–92.

[9] H. Wietschorke et. al., *The CADSOL Program Package*, University of Karlsruhe, Karlsruhe, Germany, 1992.