1994

A Software Platform for Integrating Symbolic Computation with a PDE Solving Environment

Sanjiva Weerawarana

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

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

Report Number:
94-031

A SOFTWARE PLATFORM FOR INTEGRATING SYMBOLIC COMPUTATION WITH A PDE SOLVING ENVIRONMENT

Sanjiva Weerawarana
Elias N. Houstis
John R. Rice

CSD TR-94-031
May 1994

A SOFTWARE PLATFORM FOR INTEGRATING SYMBOLIC COMPUTATION WITH A PDE SOLVING ENVIRONMENT

Sanjiva Weerawarana, Elias N. Houstis and John R. Rice [1]
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907-1398, USA.
Tel: +1 317 494 9992, Fax: +1 317 494 0739, E-mail: {saw,enh,jrr}@cs.purdue.edu

1. INTRODUCTION

It has been widely recognized that integrating symbolic computation with numeric computation is beneficial to the process of solving partial differential equations (PDEs). In spite of this, most PDE solving software does not apply symbolic computation as an integral part of the solution process. The primary reason for this lack of interaction between these two computing methodologies is the difficulty in achieving this interaction in practice. In this report, we describe our experience with developing a software environment that supports the convenient and ubiquitous interaction of these two (and other) computing methodologies. The next section provides a broad overview of the PDELab project [1], the larger context to which this work belongs. Section 3 highlights the integrated use of symbolic computation in PDELab and the following section describes the software environment that facilitates this.

2. THE PDELAB PROJECT

The objective of the PDELab project is to design a framework for building software environments that provide all the computational facilities needed to solve target classes of problems "quickly", by communicating in the user's terms. We refer to these environments as *Problem Solving Environments* (PSEs) and the application specific PSEs as workbenches. PDELab focuses on the development of PSEs for scientific applications where the underlying phenomena are modeled by PDEs.

In general, PSE technology is expected to reduce the time between an idea and validation of the discovery, to get a "quick" answer to almost any question that has a readily computable answer, to support programming-in-the-large, to provide "knowbots" (intelligent agents) that implement various scientific problem solving processes and to allow easy prototyping. PDELab is a software platform which supports the development of PSEs for PDE based applications and it realizes, to a degree, many of the above expectations.

Software Architecture. The software architecture adopted for PDELab is characterized by the software independence of its parts. It is based on "clean layering" and object-oriented methodologies. PDELab consists of three layers. The lowest layer of the PDELab architecture consists of the libraries, knowledge bases and other similar computational agents that drive the simulation process. For PDE computing, these components manipulate a certain collection of meta-objects (consisting of code and knowledge) that are involved in PDE computations, including PDE equations, geometric domains, boundary and initial conditions, grids and meshes of the PDE geometric regions, matrix equations, and discrete functions. The software architecture of this layer is therefore based on the structure and assumed interactions of these PDE objects.

The software infrastructure layer is the "middleware" that facilitates much of the functionality of PDELab. This layer is responsible for the integration of various software components to form integrated workbenches and facilitate the development of these components as well. The communication fabric that supports component integration is based on the *software bus* model [2]. The software bus concept is an attempt to emulate the hardware bus mechanism that provides a standard hardware interface to attach additional capabilities to a machine. In a hardware bus, new units describe their capabilities to the bus controller, which then passes the information along to other units in the bus. In the PDELab software bus, PDEBus, software components register their "exported" services with the software bus and rely on the software bus to invoke these services when

1. Work reported herein was supported in part by AFOSR grant F49620-92-J-0069, NSF grants 9202536-CCR and 9123502-CDA and Purdue University.

requested by interested clients. The software bus is responsible for the application of any representation translators as required for the valid invocation of the service. Thus, the software bus provides a mechanism where two tools can interoperate without having explicit knowledge about each other.

The upper layer of the PDELab architecture provides application PSE developers with a collection of tools and services required to build such PSEs. The tools at this level include visual programming tools that support *programming-in-the-large* or *megaprogramming* using PDELab components, computational skeletons for template based programming, tools for editing various PDE objects and support for building custom tools that deal with application dependent aspects of a PSE. These tools are organized into a toolbox available to the application PSE developer and appear as a collection of building tools integrated by the software bus.

This three-layered architecture is realized by seven components: PDEKit, PDEVpe, PDESpec, PDEBus, PDEPack, PDEView and PYTHIA. In the remainder of this section, we briefly describe these components.

PDESpec. The PDESpec PDE language is a high-level symbolic language that can be used to specify PDE problems and solution schemes. This language is based on the PDE meta-objects described earlier and is defined in terms of these objects. Each object represents a component of the PDE problem or the solution process and is designed to be sufficiently flexible to allow the specification of a wide range of PDE problems. The syntax of PDESpec is defined using the MACSYMA computer algebra system's syntax. The implementation of PDESpec is also based on MACSYMA, allowing one to easily apply symbolic transformations at the language level.[1]

PDEKit. PDEKit is the development environment that is used to build various components of the PDELab environment and also to build application specific components when generating application PSEs with PDELab. This toolkit defines and implements representations for the PDE objects in several forms (including C, FORTRAN, LISP and XDR) and also provides representation convertors to translate between various representations of each object. A set of tools for building graphical user interfaces using the OSF/Motif widget set is also provided. Other services available in PDEKit include distributed file I/O mechanisms (implemented using the PDEBus described below) and a generic object management utility to manage collections of objects.

PDEVpe. This is a Virtual Parallel Environment for Building parallel PDE solvers. The tools and services provided by PDEVpe include parallel BLAS, machine independent communication libraries, parallel language support, template-based parallelizing methodologies, skeletons for implementing parallel methods, (geometry based) matrix partitioning tools and instrumentation and visualization tools.

PDEBus. The PDELab Software Bus is the underlying communication fabric which provides the vital glue that facilitates the operation of PDELab. PDEBus allows one to interconnect various types of clients that communicate in various communication protocols. PDEBus also provides location services, process management services and messaging services.

PDEPack. PDEPack is the collection of numerical simulation facilities that forms the backbone of PDELab. It consists of sequential and parallel equation discretizers for both finite element and finite difference methods, sequential and parallel linear system solvers, nonlinear solvers, solution evaluators and error and performance estimators. In addition, PDEPack is a framework for integrating existing PDE solvers, libraries, model-specific solvers and entire PDE solving systems.

PDEView. PDEView is the intelligent user interface environment created within the PDELab framework. PDEView is a collection of editors and tools that allow users to create, edit and manipulate PDE objects. Each editor is a separate tool that can exist either on its own, or as an integral part of a problem solving session involving a collection of tools. Each editor also functions as an integration framework for functionality related to that editor. A notebook editor serves as the central interaction environment for users; it provides mechanisms for invoking other editors and behaves as a session log. The user interfaces of all the PDEView tools support the common and consistent look and feel supported by the user interface development facilities of PDEKit.

1. While the definition of PDESpec is based on MACSYMA, the actual implementation is based on MAXIMA, an AKCL-based, publicly available version of MACSYMA.

PYTHIA. In a problem solving environment, there are many situations where intelligent reasoning rather than algorithmic logic is necessary. PYTHIA is the environment that provides these computational intelligence facilities in PDELab. PYTHIA currently provides assistance in selecting a solution scheme to solve a given PDE problem using the previous performance of various solution methods on similar problems as a basis for its reasoning.

3. SYMBOLIC COMPUTATION IN PDELAB

As mentioned above, the PDESpec language is defined using MACSYMA. The PDESpec parser that is invoked when users textually enter PDE objects is implemented within MACSYMA. Symbolic transformations on PDESpec objects (for example, linearizing a nonlinear equation or discretizing a PDE operator) are also supported. When a PDESpec program is to be executed, a translator in MACSYMA symbolically analyzes the program and generates the target code using the GENCRAY code generation system [3]. Execution time discretizations (for example, discretizing continuous boundary conditions to generate discrete values at boundary points) are done using MACSYMA as an evaluation server.

Within PDEPack, direct symbolic operations are not feasible due to the difficulty of efficiently integrating the run-time mechanisms of languages such as C and FORTRAN with that of a symbolic computing environment. Instead, PDELab allows applications to statically specify the symbolic operation to be performed, but have it invoked dynamically. The static specification allows us to allocate necessary storage and control mechanisms while the dynamic invocation provides all the necessary flexibility to applications. Jacobian generation is an example of this type of usage.

Several PDEView tools apply MACSYMA as a dynamic evaluation mechanism. For example, the domain editor uses it to generate a display representation of a boundary defined in a piecewise parametric form: the parametric expressions are evaluated at various parameter values to produce a piecewise linear representation that can be displayed. The domain editor also uses this mechanism to allow users to specify arbitrary interpolation schemes. The same mechanism is used to generate piecewise linear representations of the boundary needed by certain mesh generators. In the visualization environment interpolation, differentiation, smoothing and function overlays are supported by using MACSYMA.

PYTHIA, the computational intelligence environment uses MACSYMA to determine various properties of the equations and boundary & initial conditions in order to provide advice on solution methods. Properties derived using MACSYMA include easily identifiable properties such as coefficient properties, dimensionality, boundary condition types and also more subtle properties such as singularities within the coefficient space.

The above uses of symbolic computation are all within the PDE solution environment itself. However, symbolic computation also forms an integral part of the overall problem solving process that is supported by the problem solving environments developed using PDELab. The PDELab system also provides a general environment where one can perform symbolic manipulation independent of the PDE processes involved and incorporate these results into the notebook editor as well.

4. AN ARCHITECTURE FOR INTEGRATED NUMERIC-SYMBOLIC COMPUTATION

The abstract view of problem solving environments is as a collection of tools collaborating together to solve some problems. These tools are all connected together via the software bus. Figure 1 illustrates this conceptual view.

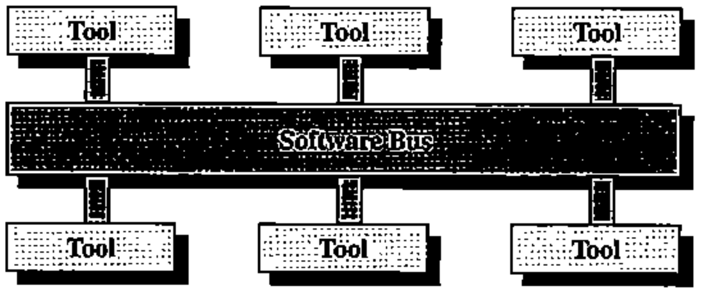

FIGURE 1. The Conceptual View of PSEs.

In PDELab, symbolic computation facilities must be available to all interested tools. To satisfy this requirement, a special tool behaves as the symbolic computation server and exports certain services that are accessed by interested clients. This tool itself is implemented by front-ending the MACSYMA computer algebra system. In addition, this tool provides program generation services using GENCRAY. Figure 2 illustrates the architecture of the symbolic tool of PDELab.

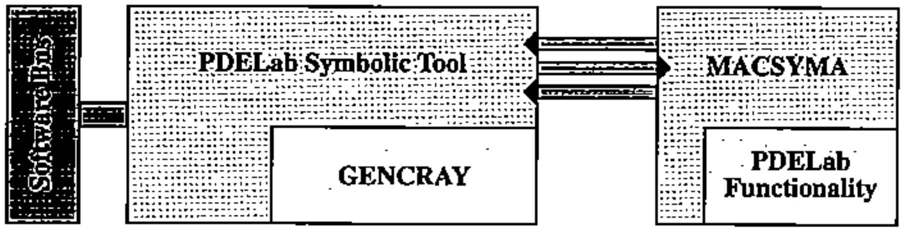

FIGURE 2. The architecture of the symbolic tool.

When a symbolic computation service is requested via the software bus, this request is transformed into one or more MACSYMA calls by the symbolic tool. The MACSYMA calls may be native calls or calls provided by the functionality specifically implemented for PDELab. Arguments to these calls may be arbitrary PDE objects and are transmitted to MACSYMA via a communication link in a textual form. The normal output produced by MACSYMA (for example, status and warning messages) are captured by the symbolic tool and displayed to the user. The communication protocol used in these input/output channels is the text command language used by MACSYMA. The results of the request, however, are sent to the symbolic tool via a special channel using the normal software bus communication protocol. The symbolic tool may further process these requests (for example, the message may be a program in an intermediate form that needs further processing by GENCRAY) or directly forward the reply to the original requestor. Program translation services of GENCRAY are provided by directly embedding GENCRAY in the symbolic tool.

To illustrate this scenario, let us consider an example. Suppose that PYTHIA wants to determine the properties of a PDE operator. PYTHIA sends the request "operator properties" to the symbolic tool with the operator object as an argument. The symbolic tool translates the operator to a text form and calls the PDELab specific function that provides this service. This function determines the properties it can determine via various symbolic transformations and sends a reply to the symbolic tool. In this case no further processing by the symbolic tool is necessary; hence it simply forwards the list of properties (in the form of an object that the requestor can understand) to the original service requestor.

5. CONCLUSION

We have described the software environment that we have developed for integrating symbolic computation and numeric computation in the context of solving partial differential equation problems. This environment uses the MACSYMA computer algebra system as the symbolic computation engine to provide the various symbolic computing services needed by PDELab.

6. REFERENCES

1. Sanjiva Weerawarana, Elias N. Houstis, John R. Rice, Ann Christine Catlin, Cheryl L. Crabill, Chi Ching Chui and Shahani Markus, "PDELab: An Object--Oriented Framework for Building Problem Solving Environments for PDE Based Applications," *Proceedings of the Second Annual Object-Oriented Numerics Conference*, Rogue-Wave Software, Corvallis, Oregon, 1994, pp. 79-92.

2. James M. Purtilo, "The Polylith Software Bus," *ACM Transactions on Programming Languages and Systems*, 16(1), 1994, pp. 151-174.

3. Sanjiva Weerawarana and Paul S. Wang, "A Portable Code Generator for Cray FORTRAN," *ACM Transactions on Mathematical Software*, 18(3), 1992, pp. 241-255.