

# *//ELLPACK: A Problem Solving Environment for PDE Based Applications*

This book presents the software architecture and implementation of the problem solving environment *//ELLPACK* (also known as *PELLPACK*) for modeling physical objects described by partial differential equations (PDEs). The scope of this problem solving environment (PSE) is broad as *PELLPACK* incorporates many PDE solving systems and some of these, in turn, include several specific PDE solving methods. Its coverage for 1-D, 2-D and 3-D elliptic or parabolic problems is quite broad, and it handles some hyperbolic problems. Since a PSE should provide complete support for the problem solving process, *PELLPACK* also contains a large amount of code to support graphical user interfaces, analytic tools, user help, domain or mesh partitioning, machine and data selection, visualization, and various other tasks. Its total size is well over 1 million lines of code.

The *PELLPACK* open-ended software architecture consists of several software layers. The top layer is an interactive graphical interface for specifying the PDE model and its solution framework. This interface saves the results of the user specification in the form of a very high level PDE language which is an alternative interface to the *PELLPACK* system. This language also allows a user to specify the PDE problem and

its solution framework textually in a natural form. The PELLPACK language preprocessor generates a Fortran control program with the data structure interfaces, calls to specified components and libraries of the PDE solution framework, and functions defining the PDE problem. The PELLPACK program execution is supported by a high level tool where the virtual parallel system is defined, where the execution mode, file system, and hardware resources are selected, and where the compilation, loading, and execution are controlled. Finally, the PELLPACK PSE integrates several PDE libraries and PDE systems available in the public domain. The system employs several parallel reuse methodologies based on the decomposition of discrete geometric data to map sparse PDE computations to parallel machines.

## INTRODUCTION

The concept of a mathematical software library was introduced in the early 70s [41] to support the reuse of high quality software. In addition, special journals, conferences, public domain software repositories (e.g., ACM, Netlib), and commercial libraries (i.e., IMSL, NAG) have been established to support this concept. Similar efforts can be found in engineering software, particularly in the areas of structural and fluid mechanics. The increasing number, size, and complexity of mathematical software libraries necessitated the development of a classification and indexing of existing and future software modules. This software is currently organized in terms of the mathematical models involved. A significant effort in this direction is the GAMS on-line catalog and advisory system [5] which has become a standard framework for indexing mathematical software. Information about engineering software can be found in several handbooks which usually describe the applicability and functionality of existing packages. The advances in desktop software/hardware, workstation clustering and distributed computing technologies, and the ease of access to supercomputing facilities have made computational prototyping a new, cost





ation libraries. The size of the parallel library is 128,000 lines of Fortran code for each implementation and consists of finite element and difference modules for discretizing elliptic PDEs, a parallelization of the ITPACK library [28], [30], [32] and the MP-PCG (parallel preconditioning conjugate gradient) package [44]. The parallel library is based on the discrete domain decomposition approach and it is implemented in both the host-node and hostless programming paradigms. A number of tools and libraries exist to support the domain decomposition methodology and estimate (specify) its parameters. For the reuse of existing “legacy” sequential PDE software we have implemented two domain decomposition based reuse methodologies described in [33].

The book is organized in two parts. Part I describes the system design and architecture. Part II is the User Guide for PELLPACK.

Part I consists of 4 chapters. The remainder of this chapter describes the exact applicability of the system in terms of the existing PDE libraries and pre-defined frameworks. We list several standard solution frameworks for various PDE models, and we describe the frameworks needed to use one of the integrated “foreign” systems. In addition we describe parallel re-use frameworks for steady-state PDE software. The multi-level PELLPACK architecture and the three level programming environment is described in Chapter 2. The PELLPACK PSE allows the user to execute programs in a variety of physical and virtual parallel architectures. Chapter 3 describes a visual scripting execution environment that allows the user to select the computers and to direct the running of computations and the visualizing of results. The PELLPACK library [25] and its highlights are presented in Chapter 4, which also presents two scenarios that demonstrate the PELLPACK design objective of reuse of high quality mathematical software, the facility for development of new PDE software, and the integration of “foreign” software.

All references are listed at the end of Chapter 4.

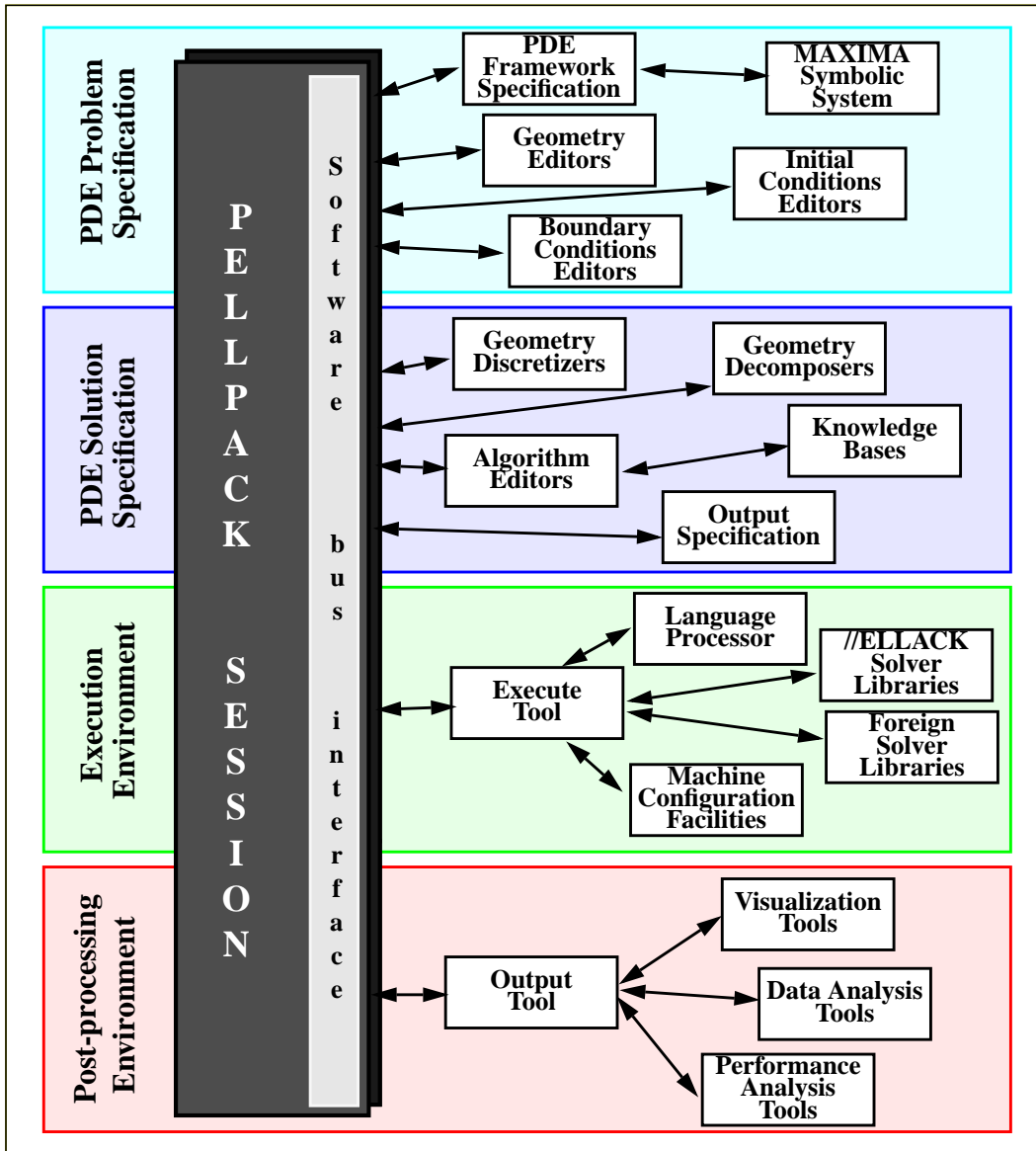


FIGURE 1. User's view of PELLPACK, depicting the tools and libraries. The diagram shows the four solution phases involved in PDE computing: problem specification, solution specification, problem execution, and solution post-processing.

## DOMAIN OF APPLICABILITY

The applicability of the PELLPACK system is defined in terms of the types of PDE software libraries integrated into the system, and the pre-defined algorithm skeletons and frameworks directly supported at the PELLPACK very high language and graphical user interface levels. An algorithm skeleton is a “solution driver”, i.e., a specification of the methods which are to be used in the solution of a PDE problem. A PELLPACK *framework* is a customized solution driver, requiring a specialized form of PDE problem and solution specification. The form of this specification is determined by the user-selected PDE software library to be used in the solution process. The framework includes the solver system selection, the mathematical representation of the PDE model (which often depends upon the selected solver), and the interfaces between the solver library and the PELLPACK runtime system. Most frameworks in PELLACK handle general (systems of) PDEs. A PELLPACK *template* is a framework for a specific PDE model, such as the Navier-Stokes equations. The PDE specification in this case is a set of parameter values.

### PDE Software Libraries

The PDE libraries currently integrated in PELLPACK are listed in Table 1. They allow the numerical solution of *field* and *flow* PDE problems in various geometric regions. The integration of these simulation libraries is done at both the PDE language and graphical interface, levels. The PELLPACK programming environment allows *differential*, *variational*, and *template* forms for specifying the PDE and auxiliary operators. The PELLPACK PDE problem specification and its “derivatives” (i.e., Jacobian, linearization transformations, forcing functions) are computed and converted symbolically to the pre-defined Fortran interface format assumed by the selected PDE library. The 3-D PDE domain geometry can be specified *only* in terms of files in well established geometry data formats (e.g., polyfile) that

PELLPACK recognizes. The system provides a 2-D geometry specification tool.

**TABLE 1. PDE systems integrated in PELLPACK, their applicability, and major characteristics**

Solver Name	PDE Model Type	Mathematical Representation and Mesh Restrictions	Dimensionality and Geometry	References
ELLPACK	single elliptic equation	Differential e.g. $u_{xx} + u_{yy} = f$	2-D general, 3-D box geometry	[40]
PELLPACK	single elliptic equation	Differential	2-D and 3-D general geometry	[21], [22], [23], [29], [57]
VECFEM	non-linear, elliptic, parabolic systems, eigenvalue problems	Variational e.g. $\int_{\Omega} (u_x v_x + u_y v_y) d\omega = \int_{\Omega} f v d\omega$	1-D, 2-D, 3-D general geometry	[17]
FIDSOL	nonlinear, elliptic, parabolic systems	Differential	2-D and 3-D box geometry	[43]
CADSOL	nonlinear, elliptic, parabolic systems	Differential	2-D general geometry	[42]
PDECOL	nonlinear, parabolic systems	Differential	1-D interval	[31]
ITGFS	2-D Navier-Stokes	Template, structured meshes e.g. <i>transonic turbulence flow parameter values</i>	2-D general geometry	[57]
NSC2KE	2-D Navier-Stokes	Template, structured meshes	2-D general geometry	[3]
NPARC3-D	3-D Navier-Stokes	Template, multi-block structured meshes	3-D general geometry	[10]

.....

TABLE 1. PDE systems integrated in PELLPACK, their applicability, and major characteristics

PDEONE	nonlinear, parabolic systems	Differential	1-D interval	[19]
--------	------------------------------	--------------	--------------	------

## Frameworks for //ELLPACK PDE Solvers

The design of the PELLPACK programming environment (i.e., a very high level PDE language and interactive editing tools) has been influenced by the requirements of its current solving capabilities and the structure of the solution skeletons (i.e., drivers) that the user is allowed to specify and run. Other solution frameworks, can be easily created in the PELLPACK system by utilizing the pre-defined *fixed interfaces* among the PDE solution phases, existing or new PDE software parts, and Fortran code. For example, the parallel time-stepping methodology described in [53] has been implemented in PELLPACK utilizing a variety of PELLPACK iterative solvers and its performance was measured on a variety of platforms [48]. In this section we describe the various pre-defined solution frameworks that PELLPACK currently supports.

### Elliptic and Parabolic PDE Solution Frameworks

PELLPACK allows the solution of single linear and non-linear elliptic and parabolic PDE equations defined on 2-D and 3-D domains. In this framework, the user can specify a solution method by naming (referencing) selected library modules (`discretization`, `indexing`, `solution`) corresponding to the phases of the PDE solution process [40] (see Figure 2 for an example). In the case of coupled or single-phase solvers the name of the `triple` module is specified. Framework 1 below lists the segments of this framework. The parallel elliptic framework currently supported in PELLPACK is based on geometric partitioning of the grid or mesh data. Thus, the user is required to specify the decomposition data in the form of a file with appropriate format and parameters. This segment can be generated by an



ments of this framework.

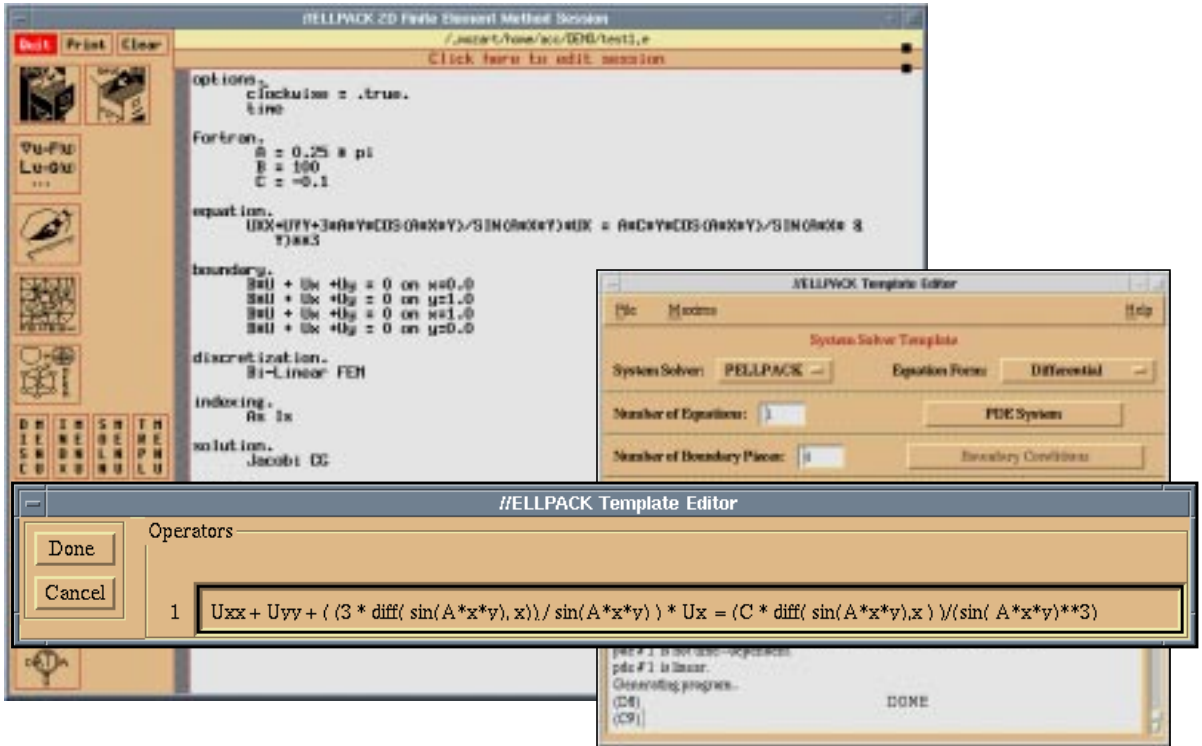


FIGURE 2. An instance of PELLPACK user interface for an elliptic framework

**FRAMEWORK 2. Nonlinear sequential elliptic PDE solution**

Segment	Description
Declarations, Options	Space for saving solution(s)
Equation, BCs	PDE problem definition
Grid/Mesh	Domain discretization
Triple	Initial guess
Fortran	Newton loop start
Linearized Elliptic Solver	Elliptic problem discretization, indexing, solution

. . . . .

**FRAMEWORK 2. Nonlinear sequential elliptic PDE solution**

Output	Format for solution output
Fortran	Convergence test
Fortran	Newton loop end
Subprograms	Initial guess, Jacobian and other support functions

Similarly, there is a framework for implementing semi-discrete parabolic PDE solvers which utilizes the available PELLPACK elliptic PDE solvers. In this case users can select pre-defined time discretization schemes or specify their own and reduce the parabolic PDE problem to a set of elliptic PDEs defined at each time-step. The framework for these solvers is described in Framework 3 and [51].

**FRAMEWORK 3. Parabolic sequential PDE solution**

<b>Segment</b>	<b>Description</b>
Declarations, Options	Space for saving solution(s)
Equation, BCs	PDE problem definition
Grid/Mesh	Domain discretization
Triple	Initial condition
Fortran	Time stepping loop start
Elliptic PDE solver	Elliptic problem discretization, indexing, solution
Output	Format for solution output
Fortran	Convergence test
Fortran	Time stepping loop end
Subprograms	Initial condition and other support functions

. . . . .

## MPlus (Matrix Partitioning) Steady-State Solution Framework

This framework is applicable to any non-time dependent PDE computation and is designed to re-use existing sequential PDE discretization software in a parallel solution scheme. It assumes that the discrete equations are generated *sequentially* with any of the existing libraries. It uses mesh/grid decomposition data or user defined partitions for the algebraic data structures associated with the selected PDE solver. The partitioned discrete PDE (i.e., algebraic) equations are loaded into the targeted multicomputer platform and solved in parallel by the available parallel solvers. Framework 4 displays the skeleton of this framework. The methodology and its performance evaluation described in [33].

**FRAMEWORK 4. Parallel matrix solution**

Segment	Description
Sequential solution framework	The PDE problem, its discretization, and sequential solver
Partition	Discrete geometric or user defined algebraic data partitioning strategy
Load	Loads partitioned algebraic system
Display	Display the structure of partitioning system
Solve	Apply a parallel solver
Output	Format for solution output

## DPlus (Domain Partitioning) Steady-State Solution Framework

This framework is currently applicable to steady-state PDE models and their derivatives (i.e., implicit parabolic solvers) defined on 2-D and 3-D domains. It is also based on a methodology to reuse sequential PDE discretization software in a parallel computation [33]. It involves

. . . . .

a decomposition of the model based on a balanced partitioning of the PDE domain with appropriate artificial interface conditions that allow the uncoupled generation of the discrete equations in each subdomain. The decomposition of the domain is obtained via the partitioning of a relative course grid or mesh [7]. Unlike MPlus, DPlus runs the sequential discretization code in parallel (i.e., each processor runs sequential code on its assigned subdomain). Framework 5 lists the segments of this framework.

**FRAMEWORK 5. Parallel stationary PDE solution**

Segment	Description
Declarations, Options	Space for saving solution, parallel machine configuration and model
Equation, BCs	PDE problem definition
Mesh generation and decomposition	Parallel multiphase mesh generation and decomposition
Interior interface conditions	Interior interface BCs definition so that the generation of global discrete equations among sub-domains is decoupled
PDE discretization	Local PDE problem discretization in parallel
Solve	Parallel solution of distributed discrete PDE equations
Output	Format for solution output

## Frameworks for “Foreign” PDE Systems

Most general PDE solving systems require users to define PDE problems by writing Fortran functions with fixed argument lists and data structures for the PDE equation, boundary, and initial conditions. Users write driver programs to allocate space, initialize variables and call the solver routines with appropriate parameters and control variables. Often, Jacobians or other symbolic computations are also required,





.....

Subprograms	Fortran functions describing the PDE equations, boundary conditions, and initial conditions. These functions are interfaces to the functions used by VECFEM to describe the equations.
-------------	--

Framework 6 lists the segments of the VECFEM framework in the // ELLPACK system. Some of the PDE problem input data for VECFEM are generated by the PDE framework specification editor. For VECFEM elliptic problems, this editor supports a variational template for specifying the coefficients of bi-linear and linear forms and a functional template for entering the PDE in differential form. For the stress analysis of isotropic materials, a stress template is available for entering only the elasticity modulus and Poisson's number of the material. The differential form of the PDE equations is symbolically transformed to a variational form. Figure 3 displays an instance of the //ELLPACK graphical interface for VECFEM.

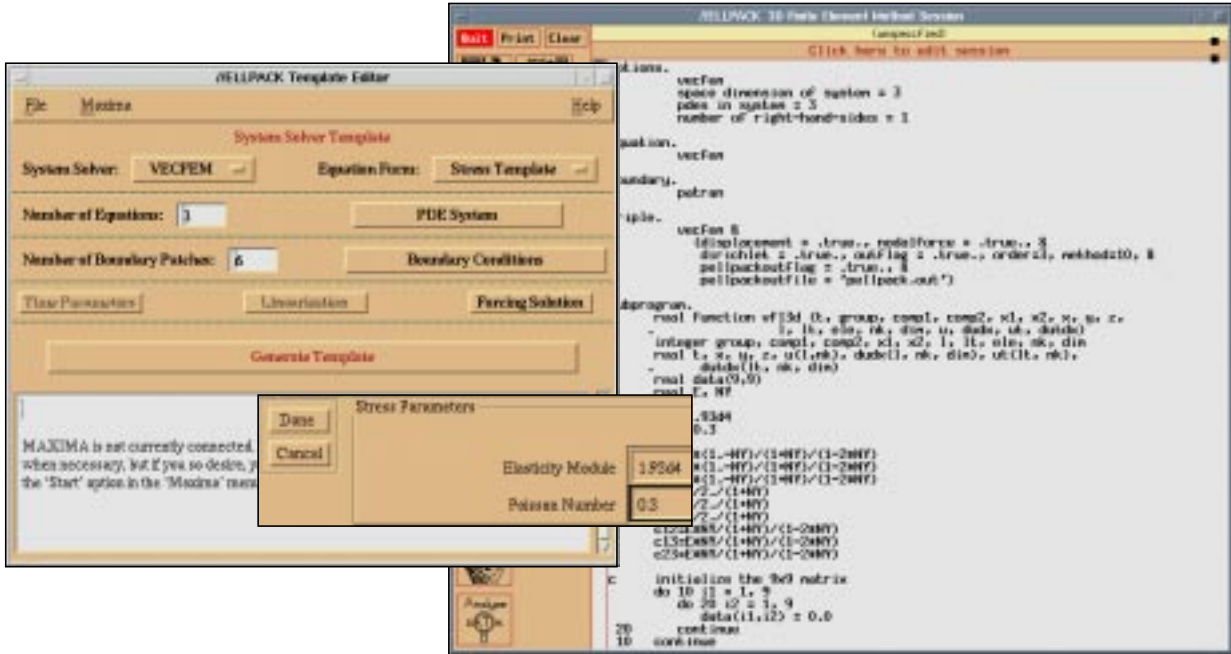


FIGURE 3. An instance of the PELLPACK interface for the VECFEM structural analysis framework

## FIDISOL Framework

FIDISOL [43] solves non-linear, time-dependent 2-D and 3-D PDE systems on rectangular domains using finite difference methods. Framework 7 describes the framework for this library. Jacobians are required for the nonlinear equations and boundary conditions; these are computed symbolically by the PDE framework specification editor. Figure 4 displays an instance of the PELLPACK interface for FIDISOL.

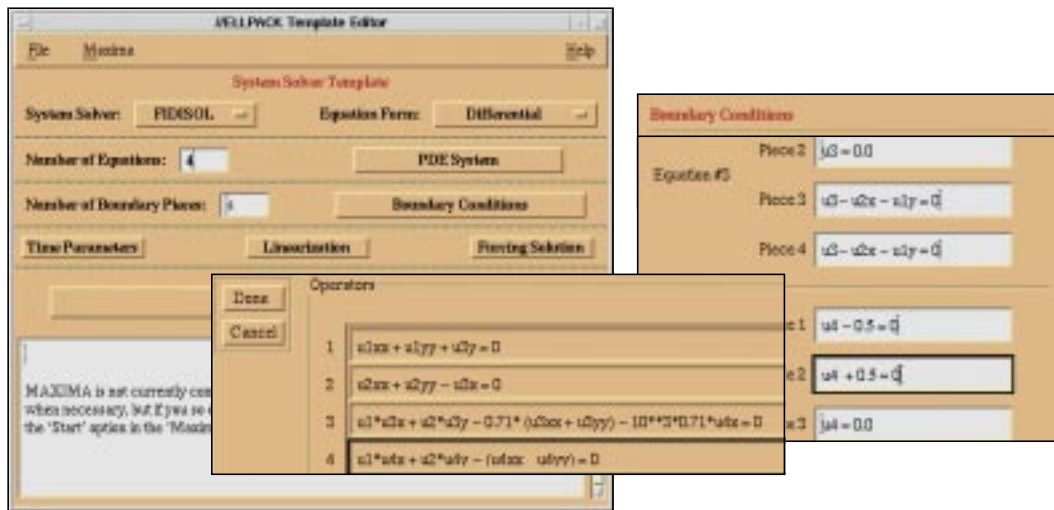


FIGURE 4. An instance of the PELLPACK interface for the FIDISOL framework

### FRAMEWORK 7. FIDISOL

Segment	Description of language interface
Options	FIDISOL id, tags indicating the type of PDE (i.e., non-linear, parabolic), number of equations in the system
Equation, BCs, IC	FIDISOL tag for all equations indicating that the equations are defined by Fortran subroutines in the subprograms segment
Boundary	2-D, 3-D box geometry
Grid	Domain discretization (uniform, non-uniform grid)

.....

Triple	FIDISOL solver and associated parameter, output specification parameters
Subprograms	Fortran functions describing the PDE equations, boundary conditions, initial conditions. These functions are identical to the functions used by FIDISOL to describe the equations. Functions describing the Jacobians for the PDE equations and boundary conditions are also placed here.

### CADSOL Framework

CADSOL [42] solves non-linear, time-dependent 2-D systems of equations on general domains using finite difference methods. Framework 8 describes the framework for CADSOL. The required Jacobians are computed by the PDE framework specification editor. Figure 5 displays an instance of the PELLPACK interface for CADSOL.

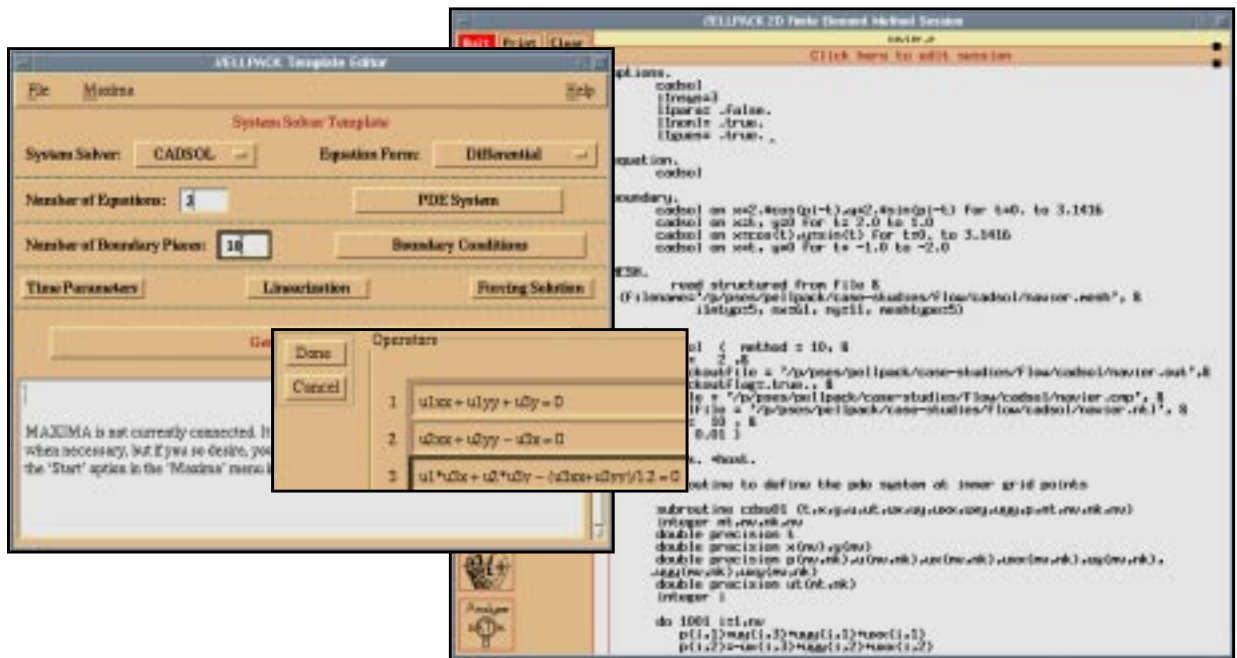


FIGURE 5. An instance of the PELLPACK interface for the CADSOL framework



Triple	PDECOL solver and parameter specification, output specification parameters
Subprograms	Fortran functions describing the PDE equations, boundary conditions, initial conditions. These functions are identical to the functions used by PDECOL to describe the equations. Functions describing the Jacobians for the PDE equations and boundary conditions are also placed here.

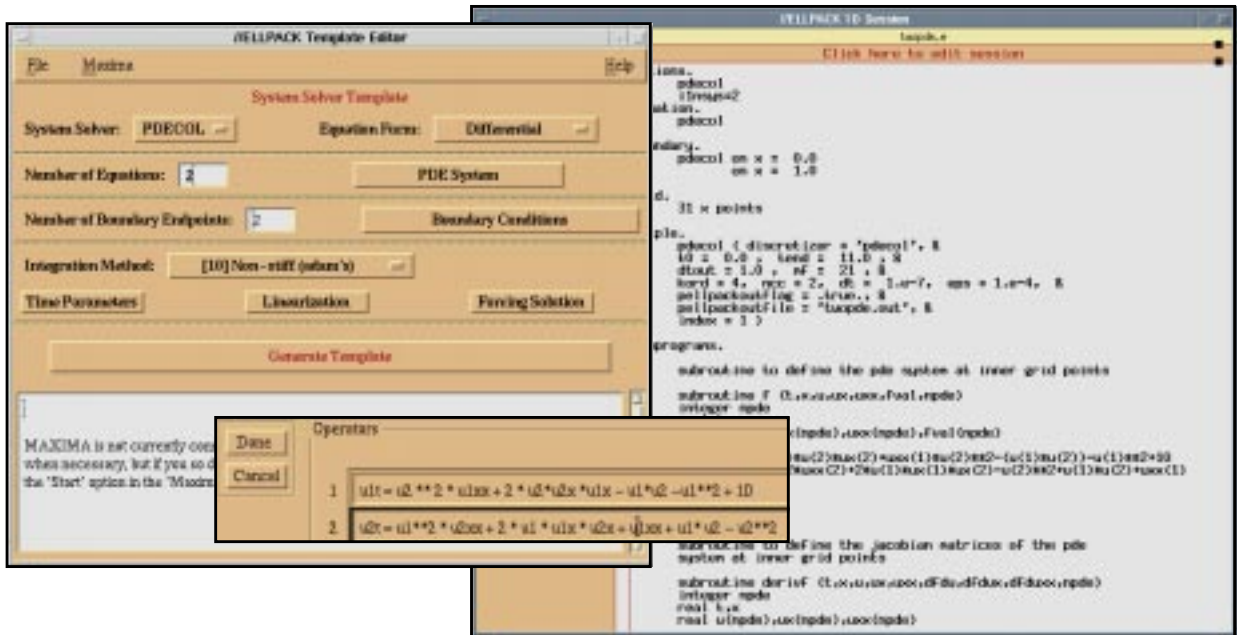


FIGURE 6. An instance of the PELLPACK interface for the PDECOL framework

## Templates for “Foreign” PDE Systems

There are PDE systems whose mathematical model and numerical solver is specified through a set of physical and numerical parameters (usually numerical data). These systems are usually associated with flow problems. In these cases the PELLPACK interface consists of a hierarchical set of templates corresponding to various models the “foreign” system supports. In general, these solvers do not require symbolic processing or Fortran code generation. Three such solvers

.....

(NPARC3-D, ITGFS, NSC2KE) have been integrated into PELL-  
 PACK. NPARC3-D is a general purpose CFD simulator for three di-  
 mensional fluid problems. ITGFS and NSC2KE are two turbulence  
 solvers for 2-D problems. ITGFS is only applicable for internal flows,  
 however it is expected to be more efficient

## NPARC3-D Template

NPARC3-D [10] is a general purpose CFD simulator, which can be  
 used for most gas flow computations, such as 2-D axisymmetric, or 3-  
 D for states of inviscid, laminar, or turbulent, and steady or transient  
 with complex geometry flow.

The original NPARC system requires the fluid problems to be defined  
 through the NPARC standard input text-file and the initial solution  
 file. This case can involve very tedious work, especially for complex  
 geometries. NPARC provides some utility tools that assist the user in  
 the pre-processing phase. In addition, the original solver must be re-  
 compiled when the mesh sizes changes. We have created PELLPACK  
 templates for the NPARC system that support a graphical user inter-  
 face to allow direct access to the NPARC utilities for redefinition of  
 global parameters, including memory allocation options. The memory  
 space for the solver is automatically allocated without recompiling the  
 NPARC library. Further work is necessary for this solver to fully uti-  
 lize the pre- and post-processing capability of the PELLPACK envi-  
 ronment. Template 1 depicts the items of the NPARC template.

**TEMPLATE 1. NPARC3-D**

Segment	Description of language interface
Options	NPARC id
Equation	NPARC tag indicate model specific equations
Domain, BC	NPARC tag indicates model specific boundary conditions

.....

Mesh	uses blocked structured meshes specified in PLOT3D or GRIDGEN format [10], and an initial NPARC solution file in binary format
Triple	NPARC solver and associated parameter, output specification parameters

### ITGFS Template

The internal turbulence gas-flow solver ITGFS [57] is designed for the simulations of transonic turbulence flow in an internal flow field. The equations governing the flow consist of two-dimensional, compressible, time-dependent, Reynolds averaged Navier-Stokes equations, supplemented by an equation of state together with the constant total temperature assumption. Template 2 describes the items of this template.

**TEMPLATE 2. ITGFS**

Segment	Description of language interface
Options	ITGFS id
Equation	ITGFS tag identifies model specific equations
Domain, BC	specified graphically by the 2-D domain editor or textually by boundary parametrization; boundary conditions are model-specific tags: inflow, outflow, wall
Mesh	generated by PELLPACK's structured mesh generator
Triple	ITGFS-turbulent solver and associated parameters, output specification parameters

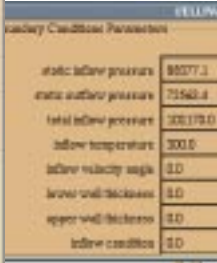
We now use the PELLPACK problem solving environment to solve a separated, transonic diffuser flow problem. We will illustrate how each PELLPACK subsystem is used in the solution process, and indicate how the components of Figure 1 are used.

The user scenario within the *PDE Problem Specification Subsystem* is depicted in Figure using snapshots from the PELLPACK system along with a brief commentary for each of the editors.

The user scenario within the *PDE Solution Specification Subsystem* is illustrated in Figure 8. Since we have already specified the PDE solver library via the framework selection, we need only to generate the appropriate domain discretization and specify the solver parameters.



selecting the CFD Template for ITGFS



entering parameters for the boundary conditions



entering parameters for governing equations



FIGURE 7. PDE Problem Specification

### The 2D Geometry Editor and the Boundary Conditions Editor

The domain can be drawn with the Geometry Editor, or the boundary can be parameterized by the user and dynamically loaded into the editor. Note that the upper and lower wall of the boundary have been divided into 3 pieces. This allows the specification of a varying grid density across the domain.

Specialized boundary conditions such as inflow, outflow, wall, and slipping are recognized within this framework, and can be assigned to the boundary pieces.

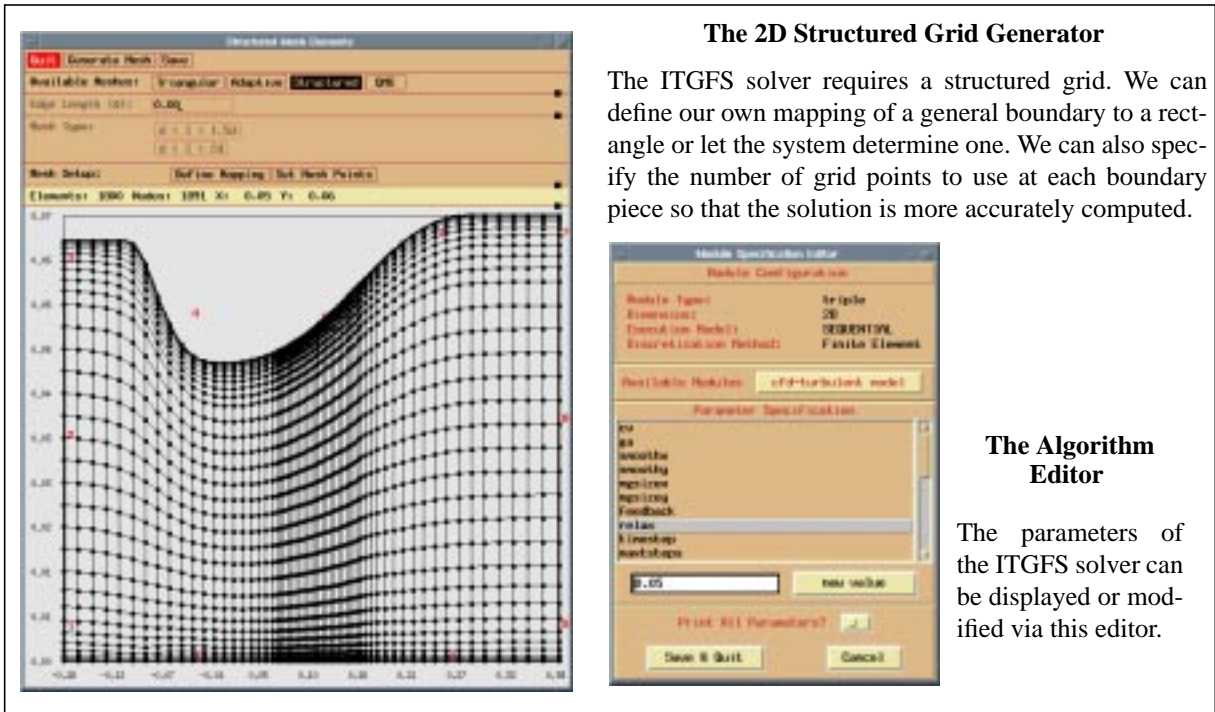
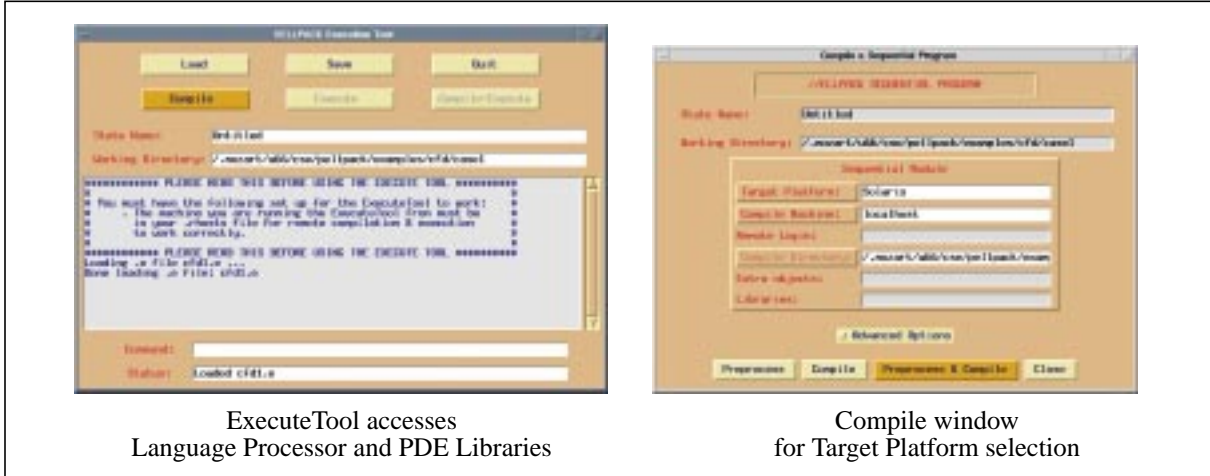


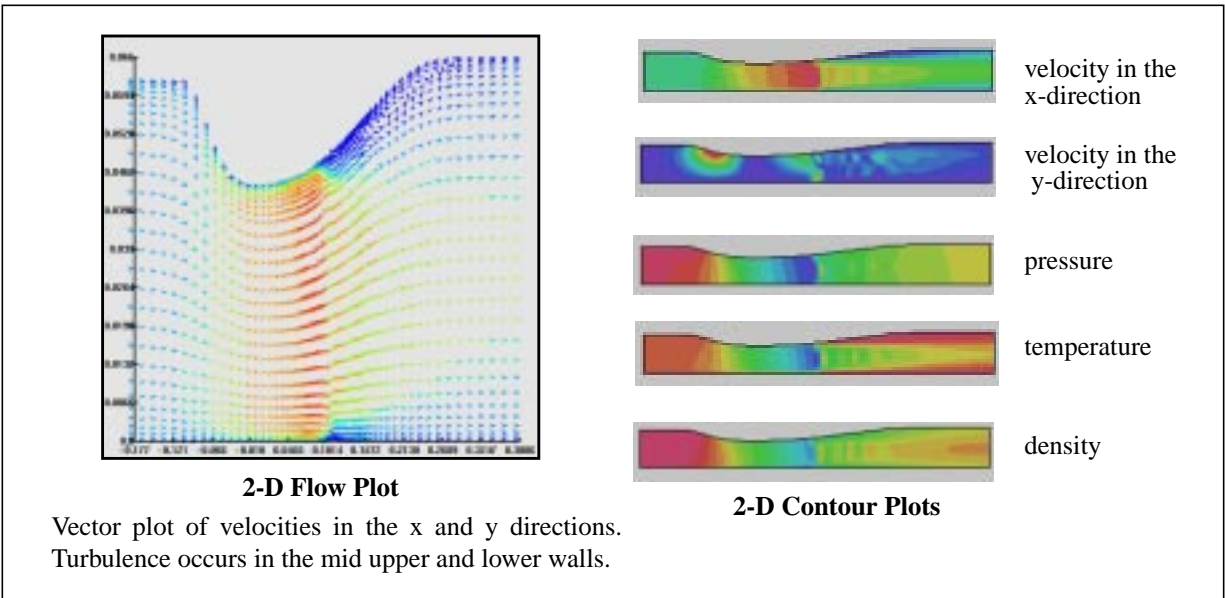
FIGURE 8. PDE Solution Specification

A PELLPACK language description of this PDE problem (.e file) is generated by the editors and written to the PELLPACK session. The language processor within the *Execution Environment Subsystem* converts the “.e file” to a Fortran driver program. The driver is linked with the PELLPACK CFD libraries, and then executed. Below are snapshots from the Execution Environment.

PELLPACK format output is generated during execution, and can be loaded into the OutputTool within the *Post-processing Subsystem* for solution visualization. Figure 10 contains snapshots from several visualizers available from the OutputTool.



**FIGURE 9. Execution Environment**



**FIGURE 10. Post-processing Environment**

. . . . .

## NSC2KE Template

NSC2KE [3] is a 2-D axisymmetric fluid flow solver applied on unstructured meshes. It solves the Euler equations using a Roe, Osher, and a Kinetic solvers and the Navier-Stokes equations using a *k-epsilon* method with two approaches of wall-laws and a two-layer model of the near wall turbulence. Template 3 describes the items of this template.

**TEMPLATE 3. NSC2KE**

<b>Segment</b>	<b>Description of language interface</b>
Options	NSC2KE id
Equation	NSC2KE tag identifies model specific equations
Domain, BC	specified graphically by the 2-D domain editor or textually by boundary parametrization; boundary conditions are model-specific tags: inflow, outflow, wall
Mesh	generated by PELLPACK's structured mesh generator
Triple	NSC2KE solver and associated parameters, output specification parameters

.....