Proceedings of Equadiff-11
2005, pp. 1–9
ISBN 978-80-227-2624-5

DIFFPACK TECHNICAL SUMMARY

PETER BÖHM[*]

Diffpack, developed and marketed by inuTech, is an object-oriented problem-solving environment for the numerical solution of partial differential equations (PDEs).

Diffpack is used worldwide for research and development in the industry and the academic sectors. Among Diffpack customers are industrial and academic leaders such as Bosch, Cambridge, Canon, CEA, Cornell, DaimlerChrysler, Furukawa, Intel, Mitsubishi, Natexis Banque, NASA, Nestlé, Shell, Siemens, Stanford, Statoil, Petrobras, Veritas, and XEROX, just to mention a few.

The customer activities span from simple prototype applications to projects involving several man-years of simulator development.

Wide Application Scope. By its design, Diffpack provides a very high degree of modeling flexibility, while still maintaining the computational efficiency needed for the most demanding simulation problems in science and engineering. Due to its modular and object-oriented design, Diffpack also grants high efficiency with respect to human resources. These are key features of all members of the Diffpack product line.

Unlike most products for PDEs, Diffpack is not a ready-to-use application, it is rather a development framework, designed to allow easy modification and combination of all numerical building blocks making up a potentially new application. It has strong focus on the flexible construction, modeling power and computational efficiency of the numerical kernel, rather than the specific nomenclature and graphical user interfaces targeting a particular engineering area.

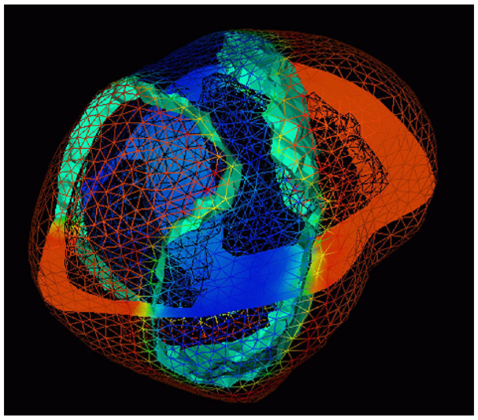
As a result, Diffpack has no inherent restrictions on the types of PDEs that can be solved. Diffpack may very well be used to solve traditional engineering PDE problems; however, it is typical for current customers that they use Diffpack to solve problems that can be characterized as "non-standard" and outside the mainstream engineering areas. Such problems often occur in pure research contexts or in the context of problems involving a variety of mutually dependent physical effects.

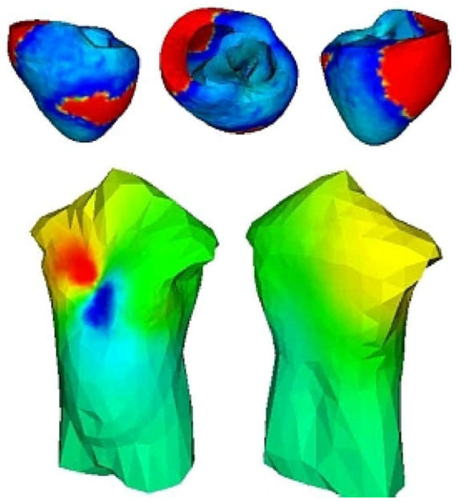
This modeling power can be efficiently employed to complement the functionality of advanced industry-standard FEM software systems like ANSYS, ABAQUS, NASTRAN, etc. ...

Diffpack's ability to construct new solvers may also be employed to create highly customized trimmed-down solvers for standard problems, closely adapted to application and user specific needs and with a minimum overhead of unused functionality and unnecessary complexity.

Electrocardial Simulations with Diffpack. As an example of a very complicated problem solved by Diffpack we consider a model of the electrical activity in the human heart (courtesy of Simula Research Laboratory AS). The mathematical model consists of 3 coupled partial differential equations (PDE) - One is modelling the propagation of the electrical signal in the heart chambers, the second one in the heart tissue, and the third models the transport from the heart surface to throughout the body. In addition to the

[*]inuTech GmbH, Fürther Strasse 212, 90429 Nrnberg, Germany (`peter.boehm@inutech.de`)


Electrical Signal in the human heart



PDEs there is a set of 12 coupled ODEs modelling the chemical reactions defined locally for each node.

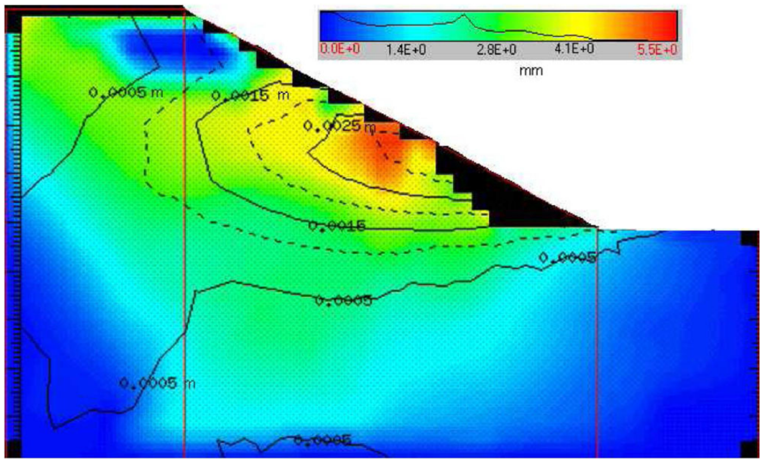
The problem was solved by finite elements using Diffpack standard FEM tools, multigrid methods and adaptive gridding (wave front). The ODEs were solved in parallel. Subproblem simulators were built and tested separately and joint by administration class. For an accurate 3D solution a grid of approx. 40.000.000 nodes was used, resulting in a discretized system of more than 900 million unknowns. On a Linux cluster of 64 processors, the solution took around 15 days (1000 sec. per time step).

Modeling of Multiphase Flow in Porous Media with Diffpack. As a second example of a very complicated problem solved by Diffpack we consider a multiphase flow in porous media (courtesy of Colenco Power Engineering, Switzerland). Coupled hydro-mechanical problems appear in many engineering problems such as subsidence, landslides, and propagation of gas in underground facilities. Moreover, also materials like, e.g. filters, sintered metals, or metal foams can be considered as multiphase porous media. Hence, coupled hydro-mechanical phenomena play an important role in a wide field of applications ranging from soil mechanics and oil extraction to automobile and aircraft industry.

The continuum mechanical model takes into account the mass and momentum conservation laws as well as the relevant constitutive equations for porous media. Based on the finite element functionality of Diffpack a program was developed which allows for the solution of such complex, highly non-linear and numerically challenging multi-phase flow problems. The numerical tools provided by Diffpack permit the rapid and fast development of a program with a wide range of applications. Examples where it was successfully applied are, e.g., the drainage of a sand column, the movements at a mountainside due to the increase of the water table or the gas pressure in an underground silo.

Functionality. Technically speaking, Diffpack is organized as a collection of C++ libraries with classes, functions and utility programs. The numerical functionality is embedded in an environment of software engineering tools supporting the management of Diffpack development projects and affiliates the needs of solving partial differential equations by the help of the finite element method (FEM), the finite difference method (FDM) and the finite volume method (FVM).

The Diffpack libraries, including the kernel and add-on toolboxes, contain a substan-



Movements at a mountainside due to the increase of the water table (courtesy of Colenco Power Engineering)

tial collection of data structures and numerical algorithms. The functionality include vectors, matrices, general multi-index arrays, strings, enhanced I/O, menu system for input data handling, simulation result database system, systems for automatic report generation, execution statistics, GUIs, preprocessors, interfaces to third party pre- and postprocessing tools, representation of linear systems (in particular large sparse systems), a large number of state-of-the-art iterative methods for sparse linear systems, solvers for non-linear systems, finite element and finite difference meshes, scalar and vector fields over meshes, a collection of finite elements, various finite element algorithms and associated data structures, numerical integration schemes, high level finite difference support, stencils (difference molecules), probability distributions, random number generators, real and complex arithmetic, solution methods for stochastic ordinary differential equations, support for random fields, adaptive meshes, error estimation, multigrid methods, generalized (mixed) finite element methods, parallel computing tools and a large collection of source code examples for problems in a variety of application areas.

The list of functionality is comprehensive. However, the major advantage for the user is the freedom with which different parts of this functionality can be combined. In the design of Diffpack, great care has been taken to make all objects and functions highly interoperable. For the user this means increased ability to construct and maintain applications and numerical algorithms.

The Diffpack product line is organized as a kernel product, referred to as the Diffpack Kernel, and a set of toolboxes providing specialized functionality. The kernel and the toolboxes are purchased as separate products.

Available toolboxes are the following:

- Diffpack Adaptivity Toolbox – Dynamic refinement of structured and unstructured meshes in 1, 2 and 3 dimensions.
- Diffpack Multilevel Toolbox – A complete and flexible lab for constructing fast multigrid linear solvers.
- Diffpack Datafilter Toolbox – A tool for import of industry standard FEM meshes into Diffpack.
- Diffpack Parallel Toolbox I – Automated parallelization of any PDE based sequential Diffpack application.

These four add-on toolboxes offer access to advanced functionality for any application with minimal development efforts. Typically, only a handful of additional code lines are necessary to bring them into operation.

Target Users. Although usage of Diffpack requires programming in C++, only minor C++ knowledge is required to get started. However, experience with programming in general is necessary.

The Diffpack learning process is supported by a comprehensive volume published by Springer-Verlag ([1]). This book introduces Diffpack programming via the style of typical FORTRAN or C codes, and then gradually introduces the object-oriented techniques characterizing more advanced Diffpack applications. The book contains over 50 application examples, which are all part of the product as delivered to customers. These examples form also a valuable resource as application templates for the user's own development.

The user is required to have a minimum of background knowledge in the mathematics of PDEs. The target user will typically have insight into the physics and mathematics of his problem area, and a need to have his mathematical and numerical ideas carefully reflected in his application program.

For these users, Diffpack provides the necessary level of control and flexibility at a minimum of development efforts. Typically, the user will concentrate almost entirely on the essential and critical numerics of his problem. Tasks related to general numerics and program management will automatically be handled by built-in Diffpack constructions. As a result, the application code will be compact and focused on the mathematical model to be solved. Still, even the simplest Diffpack application can easily offer a wide range of analysis and reporting mechanisms, which can support the understanding of the mathematical model and tailor the entities of the numerical algorithms for optimal application performance.

From Basic Models to Advanced Coupled Problems. The Diffpack software architecture follows closely the mathematical concepts involved in the PDE solution process. For instance, when solving a PDE by a finite element method (FEM), it is from a mathematical point of view natural to rephrase the PDE in terms of its weak formulation. Such weak formulations involve the evaluation of (possibly complicated) integrals over the computational domain. It turns out that the general FEM algorithm is essentially a loop over all elements in the grid, where for each element the local contribution to the discretized problem is computed by numerical integration. The integration method itself usually takes the form of an inner loop over a collection of integration points, where the integrand needs to be evaluated for each point picked.

Observing that the PDE enters the calculation only in terms of the point samples of the integrand, this mathematical/numerical view of the solution process is mirrored in the Diffpack software. As an application programmer, the user supplies the necessary expressions to evaluate the integrand of the particular PDE for a given integration point. The rest of the solution process is automatically handled by the Diffpack library components, all the way from element definitions via numerical integration methods and matrix assembly, to the solution of the resulting algebraic problems. In addition, the user might have to supply the necessary input data concerning boundary conditions and initial conditions. As a result, by coding 2-3 functions, together small enough to fit in one or two sheets of paper, the user will have a full-scale simulator providing run-time access to a wide selection of numerical solution methods.

Simulators implemented in agreement with Diffpack's coding standard can easily be reused in different settings. The software philosophy encourages the implementation of separate, generic solvers for the types of PDEs encountered in an application field. Such solvers can thereafter be specialized or refined by use of object-oriented principles like class inheritance. This is also the key to successful construction of compound simulators solving systems of coupled PDEs, possibly subject to complicated constitutive relations. Instead of starting an elaborate development from scratch, the user may pull existing Diffpack solvers from his software toolchest and make these objects work together. This software strategy has significant pay-off in terms of reduced and structured development efforts, combined with increased reliability and robustness as each solver involved gets used within a wide range of different problem configurations.

Using Diffpack's highly optimized concepts for representation of spatial points, simulators can also be set up to run in 1D, 2D and 3D, all from the same source code. Also, Diffpack's mechanisms for complex arithmetic ensure that complex and real problems can be solved with the same implementation techniques.

Run Time Control. One important feature of most Diffpack applications is the so-called menu system. This is a hierarchical data management system used at run time to define initial values of all entities used in a Diffpack program. Such entities may be simple numerical parameters, but may as well be more abstract quantities such as matrix formats, algebraic solvers, convergence criteria, numerical integration schemes, element types, etc. Thus, the menu system gives any application the ability to select, at run time, all program entities, from simple constants to the numerical algorithms that will be used.

The configuration is usually defined in input files (*.i files), and may in addition be redefined via graphical or command line selection dialogs. When Diffpack is used as sub-libraries in context of applications with established user communication, the configuration can be loaded directly from such an input stream (without input files or selection dialogs).

The menu system allows multiple choices to each program entity. Multiple choices will result in one run per configuration, and is often used to set up consistent numerical experiments, e.g. in a batch process. As an example, the user could define a loop over different preconditioners and iterative solvers, in order to find the optimal set-up for a certain type of linear system.

From Prototype to Production Code. The development of computationally intensive applications in Diffpack is often performed in two main steps:

1. Development of a prototype with focus on correctness of model and numerical methods.
2. Converting the prototype to an optimized version suitable for running in a production context.

The first step will normally use the most general tools in Diffpack for e.g. matrix assembly, numerical integration, mesh formats, etc., without explicit utilization of any specific structural properties of the particular problem at hand. The focus is thus on the overall solution process and the verification of simulator design and numerics.

In the second step, bottlenecks in the computations are localized and structural properties of the problem are used to optimize the critical functionality. The localization may efficiently be performed by standard code profiling tools, and Diffpack functions providing CPU critical operations may be replaced by the user's own specialized implementation using knowledge of the particular problem at hand.

Extending and Interfacing Diffpack. Functionality developed by the user can be seamlessly integrated with Diffpack, in the sense that user defined entities can be given the same level of interoperability as native Diffpack entities. As an example, the user may construct his own finite element types and select these at run time via the menu system and Diffpack's GUIs together with the native Diffpack elements. Technically, such integration is facilitated by special Diffpack functions registering the user-defined entities into Diffpack's entity tables.

Most user-defined functionality can be integrated with Diffpack as library components at the level of interoperability described above. The user may thus consistently extend Diffpack into a tool tailored to his particular areas of interest.

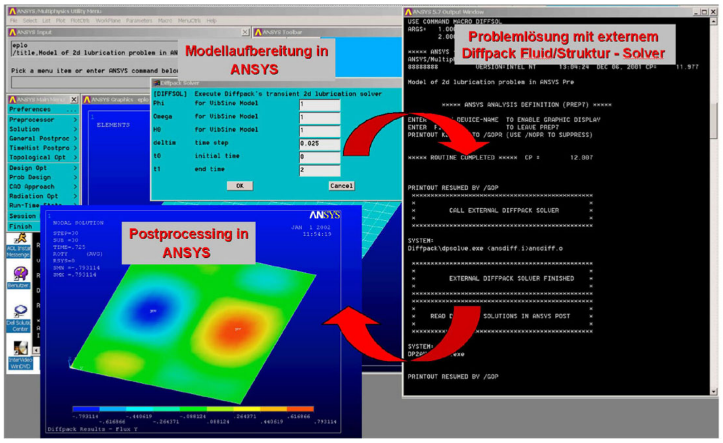
User defined library components might be programmed from scratch. Naturally, for many users it is of particular interest to extend Diffpack with well-tested legacy code, e.g. written in C or FORTRAN. Such functionality may easily be encapsulated in classes with communication interfaces following the Diffpack standard. This approach allows a seamless integration with Diffpack using the registration mechanisms explained above. With minimal efforts, the user may therefore make valuable functionality, which may have taken man-years to develop, directly available as part of Diffpack's plug'n play environment.

The Diffpack/ANSYS Integration. Diffpack's modeling power can be used to extend the functionality of advanced industry-standard software systems. We can identify three different ways how to integrate Diffpack solvers into e.g. ANSYS:

- Loose coupling where ANSYS is mainly used as a pre- and postprocessor with a stand-alone solver based on Diffpack.
- Partial coupling where ANSYS is used as a pre- and postprocessor and for solving a subset of a multi-physics problem, whereas Diffpack solves the remaining subset of the multi-physics problem.
- Tight coupling where finite element matrices generated by ANSYS directly get incorporated into a linear equation system handled by Diffpack. Diffpack may add additional terms to the system, e.g. from control theory.

In the following we give two examples of a loose Diffpack/ANSYS integration which basically works as follows:

- Problem set-up with ANSYS preprocessor
- ANSYS-GUI used to communicate input values
- Diffpack solver started from GUI as separate process
- Postprocessing in ANSYS



```
eplo
/title,Model of 2d lubrication problem in AN
Pick a menu item or enter ANSYS command belo
```

```
[DIFFSOL]  Execute Diffpack's transient 2d lubrication solver
Phi        for VibSine Model      1
Omega      for VibSine Model      1
H0         for VibSine Model      1
deltim     time step              0.025
t0         initial time           0
t1         end time               2
```

```
NODAL SOLUTION
STEP=30
SUB =30
TIME=.725
ROTY      (AVG)
RSYS=0
SMN =-.793114
SMX =.793114
JAN  1 2002
11:54:19
```

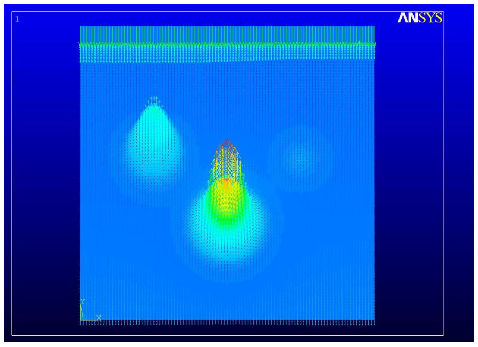
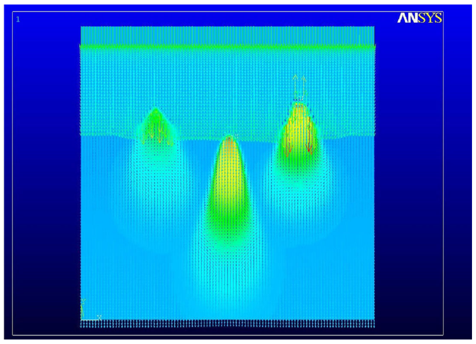
```
USE COMMAND MACRO DIFFSOL
ARGS=   1.00
        2.00

***** ANSYS
ANSYS/Multip
88888888         VERSION=INTEL NT      13:04:24  DEC 06, 2001 CP=      11.977

Model of 2d lubrication problem in ANSYS Pre

         ***** ANSYS ANALYSIS DEFINITION (PREP7) *****

ENTER      DEVICE-NAME  TO ENABLE GRAPHIC DISPLAY
ENTER  F                TO LEAVE PREP7
PRINTOUT K          TO /GOPR (USE /NOPR TO SUPPRESS)

***** ROUTINE COMPLETED *****  CP =        12.807

PRINTOUT RESUMED BY /GOP
 ***********************************************
 *                                             *
 *         CALL EXTERNAL DIFFPACK SOLVER       *
 *                                             *
 ***********************************************

SYSTEM=
Diffpack\dpsolve.exe <ansdiff.i>ansdiff.o

 ***********************************************
 *                                             *
 *         EXTERNAL DIFFPACK SOLVER FINISHED   *
 *                                             *
 ***********************************************
 ***********************************************
 *                                             *
 *    READ D        SOLUTIONS IN ANSYS POST    *
 *                                             *
 ***********************************************

SYSTEM=
DP2A      .exe

PRINTOUT RESUMED BY /GOP
```

For both examples, the additional Diffpack solver consists of a few hundred lines of C++ code, of which about 20% makes up the numerical core. The Diffpack solver fully replaces the ANSYS solver.

Although the developer of the solver will need to know Diffpack programming, the user of the application will hardly notice Diffpack. He will stay inside the ANSYS work environment all the time, and just briefly encounter Diffpack in connection to the call of the external Diffpack solver.

Example I: Two-Phase Flow with Hot Spots. A chemical reactor can be constructed as a vertical cylinder where solutes are fed in at the top and refinery products trickle out at the bottom. In oil refinery, the chemical reactions can form unwanted hot-spots, which can lead to severely high pressures and shut-down of the reactor.

In this example we are interested in studying the effects of such hot spots on the flow and pressure of two solutes in a cylindrical reactor. Mathematically, the problem is represented by a set of three coupled partial differential equations representing the saturation of one of the solutes, the pressure and the temperature inside the reactor. The Diffpack simulator consists correspondingly of three independent FEM-simulators that are coupled via an administrator class.

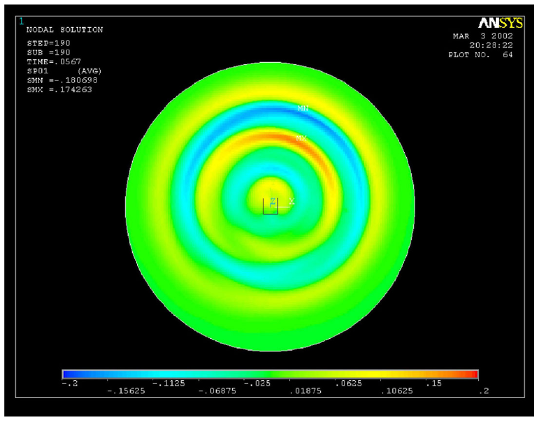
The figures show two snap shots from the ANSYS postprocessor at given time steps during a simulation. The vector field of velocities are plotted on top of the heat field.
Initially the reactor is filled with oil. A heavier solute enters at the top and displaces the oil by pushing it downwards. There is a rather sharp front between the two phases.
Three hot spots light up in the oil phase at slightly different times and stay lit for a short time period. When lit, they heat up the oil locally to form a hot oil bubble.
Due to decreased density, the oil bubble starts to propagate upwards while it diffuses and cools down. As a particle in the bubble is cooling, the upward movement slows down until the particle starts moving downwards again.
Depending on timing, heat and size of the hot spot, the oil bubble can break through the front between the phases and continue to propagate into the other phase.

In the current situation, the chemical reactions are omitted from the model, and the hot spots are represented as source terms. A more realistic model would include the chemistry in the coupled set of equations and compute the heat field as a result of the chemical reactions. Such a model would be implemented according to the software design principles mentioned above.

Example II: Acoustics with Open Boundaries. A physical interpretation of our second example is to compute the propagation of sound waves emitted from a loudspeaker placed in open space.

The Diffpack FEM-simulator developed for this problem basically solves the wave equation, and was developed independent of the spatial dimension. In 3D, the solver uses a representation of the loudspeaker as a rectangular box with the front wall removed. A single point source is located at the back wall and represents the sound source.

The image shows a snap-shot from the ANSYS postprocessor with the solution in 2D obtained on a mesh with about 100,000 elements.



The propagation of sound waves in open space is handled by embedding the box in a spherical domain with boundary far away relative to the size of the box.
At this boundary, the Sommerfeldt's radiation condition is applied, with the waves hitting the boundary in the normal direction. The latter is an approximation that becomes increasingly correct with growing size of the sphere.

The simulator was developed independent of spatial dimension and was tested primarily on 1D problems. When turning to 2D and 3D, few corrections were needed to achieve correct results. This principle for software testing is typical when developing Diffpack simulators.

An interesting extension of the current solver is to replace the point source with a vibrating plate. This will lead to a coupled acoustic-structure interaction problem. This variant can be developed by coupling the present acoustic simulator to a separately developed elasticity simulator by using the principles discussed in Example I.

Flexibility and Efficiency. The plug'n play characteristics of Diffpack application development are facilitated by a software design heavily based on object-oriented programming. Such constructions can increase human ability and efficiency significantly, but may in cases of improper use lead to a general decrease of computational efficiency.

In Diffpack, object-oriented mechanisms are used only for tasks with minimal influence on computational efficiency. In particular, low-level, computationally intensive operations are always performed in a FORTRAN or C type style, while the object-oriented principles are mainly used for higher level administrative tasks. Although object-oriented, this means that most Diffpack applications will spend almost all CPU time in functions containing very basic loops over array structures, as is the case in well written numerical FORTRAN codes. This ensures high computational performance of Diffpack applications.

Comparisons ([2]) to tailored FORTRAN benchmark codes show insignificant differences in computational efficiency between Diffpack applications and finely tuned FORTRAN solvers.

GUIs, Databases, Pre- and Postprocessing. Diffpack defines an internal format (SimRes) for storage of mesh and field structures. This format is used for storage of stand-alone entities (e.g. a simple grid file) as well as complete simulation scenarios (e.g. a SimRes database containing several mutually dependent grid and field objects).

For postprocessing, Diffpack provides a suite of filters to convert SimRes data to formats used by third party visualization products, as e.g. Matlab, Gnuplot, IRIS Explorer, AVS and VTK. For preprocessing, filters are provided for mesh formats generated by Abaqus, Ansys and Nastran via the Diffpack Datafilter Toolbox. All these filters can operate either as file-to-file converters or they may be linked to an application for direct data exchange. Some of the postprocessors may also be started directly from a Diffpack application.

In addition to filters to third party preprocessors, Diffpack provides internal preprocessors supporting lattice grids, transfinite mappings and super-element techniques as well as interfaces to the public domain grid generators GeomPack and Triangle.

On Unix, any application will operate with graphical menu selections provided it is compiled with the appropriate options. On Windows, any application can be equipped with a GUI providing application control, result browsing, curve plotting and state-of-the-art visualization (VTK).

Quality Assurance. Diffpack is available for about 10 different platforms (hardware/compiler configurations) and is maintained as one single source code across all these architectures. The broad selection of compilers ensure that only portable and safe C++ constructions are used, and makes it easy to include new platforms in the portfolio when necessary. It is therefore easy to move an application from one platform to another, normally without need for any changes in the application source code.

Correctness and stability of the Diffpack functionality is regularly verified on a suite of several hundred verification applications on all supported platforms. The verification tools are also available to the users for easy QA of their own development.

Diffpack motivates the user to apply a safe and consistent programming style, and offers a variety of tools to enhance efficiency of the development process, including automatic memory handling and powerful debugging tools.

The user's applications can easily follow the upgrade path of Diffpack, due to automatic code transformation tools upgrading his application source code from an older to a newer version of Diffpack.

Documentation Tools. Diffpack provides powerful mechanisms for documentation of numerical experiments. Full project reports containing e.g. problem description, tabulated numerical results, images, videos and, e.g., source code of the application itself, can be generated in ASCII, LaTeX or HTML formats. The reporting mechanisms interface to the menu system, and can be used to create reports of multiple runs with overview summaries and links to details for individual runs.

REFERENCES

[1] H. P. Langtangen, *Computational Partial Differential Equations, Numerical Methods and Diffpack Programming,* Texts in Computational Science and Engineering, Vol. **1**, Springer-Verlag, 2003.

[2] H. P. Langtangen and A. Tveito (Eds.), *Advanced Topics in Computational Partial Differential Equations, Numerical Methods and Diffpack Programming,* Lecture Notes in Computational Science and Engineering, Vol. **33**, Springer-Verlag, 2003, ISBN 3-540-01438-1 / XIX, 659 p.