Diffpack- 
A Flexible Development Framework 
for the Numerical Modeling and Solution of Partial 
Differential Equations

Peter Böhm, Frank Vogel 
Bratislava, July 25th 2005
Outline

- inuTech - Innovative Numerical Technologies
- Diffpack - an overview
- Diffpack - the main philosophy
- Diffpack - how to realize
- Examples
  - Heart
  - Tsunami
  - CFD
inuTech - Our History

- FSV Consulting R&D, founded 1995
- inuTech GmbH was established in June 2000
  - Share Holders: Prof. Schittkowski, CADFEM International AG, F. Vogel
  - Currently 6 employees
inuTech - Our Objectives

- R&D of Numerical Methods
- Sales and Support of Software
- Closing the Gap between Engineering and Mathematics
- Seminars and Training
- Consulting

to Solve Engineering and Mathematical Problems
inuTech - Our R&D Experience

- **Structural Optimization**
  - Topology Optimization
  - Shape Optimization
  - Sizing- and Parameter Optimization
  - Large Scale Optimization
  - Multi-Disciplinary Optimization
  - Design of Experiments
  - Biological Growth
  - Free Material Optimization (Composite Design)
Diffpack - in general
inuTech - Our R&D Experience

- **Differential Equations**
  - inuTech develops and markets the **Diffpack Product Line** for the Numerical Modeling and Solution of Differential Equations
  - inuTech offers Consulting Services around Diffpack; we can deliver customized turn-key solutions for specialized simulation problems
  - **Attention:** Diffpack is a development environment, not a program!
    It allows you to generate a simulator!
The Diffpack® Team

SINTEF

Univ. of Oslo

Univ. of Erlangen

Univ. of Bayreuth

WisSOFT

CADFEM

inuTech

SpringerLink

Simula Research Laboratory

inuTech

Innovative Numerical Technologies
The **Diffpack®** Distribution

**Diffpack Development Framework**

- **Diffpack Utility Programs**
  - Diffpack Kernel (computing/numerics)
  - Diffpack Toolboxes
  - OpenSource Pre-/Postprocessors
- **Third-Party Pre-/Postprocessors**
- **Books, Manpages, Reports**
- **Support**
  - phone/e-mail

**innuTech**

Innovative Numerical Technologies
The Diffpack® Environment

Applications

- Adaptivity
- Multigrid
- Data Filters
- Parallel LA
- Parallel DD

Diffpack Kernel
List of Applications

- electrical activity in the heart (and body)
- continuous casting
- simulation of a solid-oxide fuel cell
- chemical reactor flow
- parasitics in high power electronics
- option pricing in finance
- water waves - structure impact
- hyperspherical black hole simulation
- tsunami simulation
- virtual rock physics laboratory
- heat treatment of cancer
- ...

innuTech
Diffpack - main philosophy
The Challenge

- Speed-Up 1,000,000 over last 50 years
- Hardware Power
- Numerical Methods

Scientific Computing

Software Development
The **Diffpack®** Vision

\[
\frac{\partial s}{\partial t} = F(v, s, t)
\]

“arbitrary” PDE, ODE, or other numerical problem

\[-\nabla \cdot [\kappa \nabla u] = f \quad \text{in } \Omega\]

\[u = g \quad \text{on } \partial \Omega\]

**Diffpack Kernel**

Diffpack Toolboxes (further extensions)

\[=\]

own solver
The **Diffpack®** Vision (cont’d)

- Structural mechanics
- Porous media flow
- Water waves
- Aero-dynamics
- Electro magnetics
- Incompressible flow
- Heat transfer
- Other PDE applications

Observation: Methodology basis independent of applications
The **Diffpack**® Intention

**satisfy the criteria of numerical software by using Diffpack**

- efficiency
- reliability
- robustness
- extensibility
- portability
- user-friendliness

→ focus on your own problem-specific parts
The **Diffpack®** Philosophy

- do not reinvent the wheel
- trust (to a certain extend) former developments
- make use of OOP in numerics -> OON (inheritance, virtual func.)
- support high abstraction level
- offer suitable building blocks (modular design)
  - stay as close as possible to the math
  - avoid translation errors
- simplify as much as possible (try to build your software “like a child”)
- do not patronize the developer
  - arbitrary equations, work on weak formulation
  - allow for own user defined extensions
- avoid bottle-necks of OOP
The **Diffpack®** Philosophy - common language

- **Engineer** has a problem
- **Physicist** has a physical description
- **Mathematician/Numeric Specialist** has a set of equations/solution methods
- **Software** available

**Diffpack**: gives common language

- Engineer gets data which tell him how to solve the initial problem
- System Admin/Hardware Specialist has computer to run on the program
- Software Specialist writes a program to solve the problem
Diffpack - how realized
**Diffpack® Implementation - high abstraction level**

Example: Solve $A^*x=b$ by a CG method:

**Procedural language approach (ITPACK, Netlib):**

```c
CALL NSPCP (PREC, ACCEL, NDMIN, MAXNZ, COEF, JCOEF, P, IP, U,
            UBAR, RHS, WKSP, IWKSP, NW, INW, IPARM, RPARM, IER)
```

**OON (C++, Diffpack):**

```cpp
//Given Matrix A, Vector x, Vector b
LinEqSystem system (A, x, b);
LinEqSolver solver (method);

solver.solve(system);
```
Diffpack® Implementation - inheritance

C++ Classes

SimCase
Inheritance
FEM
Inheritance
OwnSimulator

Fundamental library class for any simulation
Fundamental library class for finite element methods
General implementation of the problem. Focus on numerics
Diffpack® Implementation - inheritance (cont’d)
**Diffpack® Implementation - use weak formulation**

\[-\nabla \cdot [\kappa \nabla u] = f \quad \text{in } \Omega\]
\[u = g \quad \text{on } \partial \Omega\]

Use weak formulation

\[
\sum_{j=1}^{m} \left( \int_{\Omega} \kappa \nabla N_j \nabla N_i d\Omega + \right) u_j = \int_{\Omega} f \ N_i d\Omega, \quad i = 1, \ldots, m \quad \text{in } \Omega
\]
\[u = g \quad \text{on } \partial \Omega\]
Diffpack® Implementation - important member

global_menu.multipleLoop (S)

S::adm
S::define
S::scan
S::solveProblem
S::integrands
S::resultReport

FEM::makeSystem
FEM::calcElmMatVec
FEM::numItgOverElm
ConvDiff:: integrands (numerical kernel)

```cpp
void ConvDiff::integrands (ElmMatVec& elmat, const FiniteElement& fe)
{
    int i, j, q;
    real f_value = f(fe);
    real k_value = k(fe);
    const Int nbf = fe.getNBF();
    const real detJxW = fe.detJxW;
    const int nsd = fe.getNSD();

    real gradNi_gradNj;
    real Ni_gradNj;
    for (i = 1; i <= nbf; i++) {
        for (j = 1; j <= nbf; j++) {
            gradNi_gradNj = 0;
            Ni_gradNj = 0;
            for (q = 1; q <= nsd; q++) {
                gradNi_gradNj += fe.dN(i, q) * fe.dN(j, q);
                Ni_gradNj += U(q) * fe.N(i) * fe.dN(j, q);
            }
            elmat.A(i, j) += (k_value * gradNi_gradNj + Ni_gradNj) * detJxW;
        }
        elmat.b(i) += fe.N(i) * f_value * detJxW;
    }
}
```

Weak Formulation (\(\alpha=0\)):

\[
\sum_{j=1}^{m} \left( \int_{\Omega} k \nabla N_j \nabla N_i d\Omega + \int_{\Omega} \mathbf{v} \cdot \nabla N_j N_i d\Omega \right) u_j = \int_{\Omega} f N_i d\Omega, \quad i=1,...,m
\]

\[
Au = b
\]
ConvDiff :: fillEssBC
(apply essential boundary Conditions)

```c
void ConvDiff::fillEssBC ()
{
    dof->initEssBC (); // init for assignment below
    const int nno = grid->getNoNodes (); // no of nodes
    Ptv(real) x;
    for (int i = 1; i <= nno; i++) {
        // is node i subjected to any boundary indicator?
        if (grid->boNode (i,2) || grid->boNode (i,3) || grid->boNode (i,4))
            dof->fillEssBC (i, 0.0); // u=0 at boundary nodes
    }
}
```

Apply Boundary Conditions:

\[ u(x, y) = 0, \quad x, y \in \Gamma_1 \]

“essential conditions”
Diffpack® Implementation - simplification

• Whenever you add data/function:
  – do you really need the data/function at this place
  – is the current place the optimal place
  – should one use inheritance
  – avoid “if”
• try to build your software “like a child”
• keep your software as simple as possible
Edge elements
Open Source Spirit
Examples
**Diffpack® - Electrical Activity in the Heart**

- 3 coupled PDEs - 1 in torso, 2 in heart
- 12 coupled ODEs sitting in each node
- Solved by finite elements using Diffpack standard FEM tools, multigrid methods, adaptive gridding (wave front)
- Parallel solution using Domain Decomposition /ODEs solved in parallel
- Sub-problem simulators build and tested separately. Joined by administration class.
- Dimension independent code
- Around 10,000 lines of code

**3D Case:**
- 40,000,000 nodes in the heart
- 1,000,000 nodes in the body
- 900,000,000 unknowns update every time step
- About 1000 sec per time step
- Optimal preconditioning, 64 processors: 15 days

**Accurate 2D solution:**
- 1,000,000 elements, 32 processors, 4 hours, 1 Gb

**Accurate 3D solution:**
- 900,000,000 unknowns, 64 processors, 1000s per time step, 312 Gb
**Diffpack® - Electrical Activity in the Heart**

\[
\frac{\partial s}{\partial t} = F(v, s)
\]

\[
\chi C_m \frac{\partial v}{\partial t} + \chi I_{ion}(v, s) = \nabla \cdot (M_i \nabla v) + \nabla \cdot (M_v \nabla u_e) + \nabla \cdot ((M_i + M_c) \nabla u_c)
\]

where

- \( u_e \) Extra cellular potential
- \( v \) Transmembrane potential, \( v = u_i - u_e \)
- \( s \) State vector (concentrations and gates)
- \( F \) Rate functions
- \( I_{ion} \) Total ionic current
- \( C_m \) Membrane capacitance
- \( \chi \) Membrane surface area to volume ratio
- \( M_i \) Intra cellular conductivity tensor
- \( M_c \) Extra cellular conductivity tensor

*Courtesy of Simula Research Lab*
Diffpack® - Tsunami Simulation

- Slides/impact
- Large destructive water waves

Tsunami Simulation - Storegga (Norway)

Courtesy of International Centre for Geohazards
Diffpack® - Computational Fluid Dynamics

Viscous 3D flow around a cylinder, Re=265, velocity isosurface

Large-eddy simulation of flow around two cylinders in a tandem arrangement, Re=22000, velocity isosurface and pressure distribution

Courtesy of SINTEF, Applied Mathematics
## Diffpack® Benchmarks

<table>
<thead>
<tr>
<th>Project</th>
<th>Author (Year)</th>
<th>Equation</th>
<th>unknowns</th>
<th>CPU</th>
<th>System</th>
<th>WT^{(3)} [s]</th>
<th>RAM [Gb]</th>
<th>Speed up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heart</td>
<td>Simula (2005)</td>
<td>Electrocardiac(^{(1)})</td>
<td>10.4 Mio + 150 Mio</td>
<td>32 / 16</td>
<td>Origin3800/ Itan.-Clu.(^{(2)})</td>
<td>1197/953 s</td>
<td>25</td>
<td>-</td>
</tr>
<tr>
<td>Heart</td>
<td>Simula (2004)</td>
<td>Electrocardiac</td>
<td>84 Mio + 1260 Mio</td>
<td>32 / 64</td>
<td>Origin3800</td>
<td>5263 / 2840 s</td>
<td>200</td>
<td>-</td>
</tr>
<tr>
<td>Tsunami</td>
<td>Simula (2005)</td>
<td>Boussinesq</td>
<td>11 Mio</td>
<td>20</td>
<td>Itan.-Clu.</td>
<td>780 s</td>
<td>8</td>
<td>19.4</td>
</tr>
</tbody>
</table>

\(^{(1)}\)2 PDEs & 30 ODEs, \(^{(2)}\)Linux Itanium2 Cluster (1.3 GHz, 64bit), \(^{(3)}\)wall clock time (per time-step)
Simplify!

Sometimes I feel like our life has gotten too complicated... that we've accumulated more than we really need... that we've accepted too many demands...

Well, Thoreau says, "Simplify, simplify." Maybe we need to do that.

But how?

I hate it when they look at me that way.
Thank you for attention!
Minisymposium on "Numerical Software for PDEs" EQUADIFF 11

Diffpack® - Computational Fluid Dynamics

Large-eddy simulation of flow around two objects in a tandem arrangement

Viscous 3D flow around a cylinder

Courtesy of SINTEF, Applied Mathematics

inuTech

Innovative Numerical Technologies