Workshop Kaskade 7 & Applications
Getting Started

M. Weiser, L. Lubkoll

Zuse Institute
Berlin

MATHEON

Ankara, 2013-10-02

Contents

Kaskade 7 in a nutshell

- what's it good for
- toolkit structure

PDE problem classes

- mathematical structure of variational problems
- examples: Poisson & Stokes
- notation

Tutorial

- describing variables, variational functionals, ansatz and test spaces
- obtaining grids
- assembly, solution, output
- examples: Poisson & Stokes

Kaskade 7 in a Nutshell

What's it good for?

MATHEON

Target problems

second order PDEs:	elliptic	$-\mathrm{div}(\sigma\nabla u) = f$	Poisson
		$-2\mu\Delta u - \lambda\nabla\mathrm{div} u = f$	Lamé-Navier
	parabolic	$\dot{T} = \Delta T + f$	heat equation
systems of PDEs:	optimization	$\begin{bmatrix} I & & \Delta \\ & \alpha & B^* \\ \Delta & B & \end{bmatrix} \begin{bmatrix} y \\ u \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ f \end{bmatrix}$	KKT
	flow	$\begin{bmatrix} -\Delta & \nabla \\ \mathrm{div} & \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}$	Stokes

... and much more

What it is... and what not

MATHEON

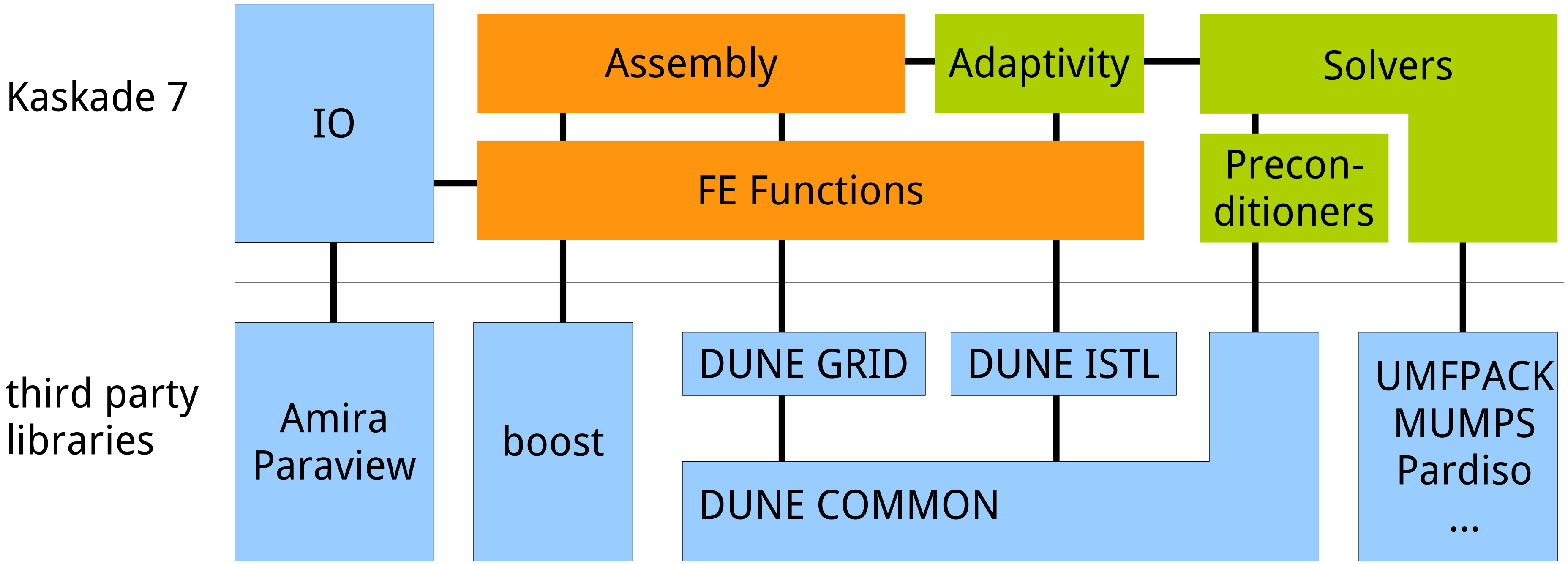
Kaskade 7 is

- ✔ a finite element library (toolbox)
  - you are in control
  - mix and match what you need
  - easily extended as required
- ✔ a research code
  - flexible
  - living, evolving
- ✔ well documented
  - for a research code
  - targeted towards developers
  - doxygen, tutorial, course ware
- ✔ a tool for real programmers
  - powerful
  - steep learning curve

and it is not
(and not intended to be)

- ✗ a finite element code (framework)
  - can do only what it's designed for
  - follows pre-fabricated scenarios
- ✗ a production code
  - stable releases
  - for standard problem types
  - full fledged with pre- and postprocessing
- ✗ an unorganized pile of code
- ✗ a toy for script kiddies
  - with point-and-click GUI

Toolkit Structure



Short History of Kaskade 7

MATHEON ZIB

1989 *Concepts of an adaptive finite element code* (Deuflhard/Leinen/Yserentant)
Kaskade 1 (C, by P. Leinen)

1991 Kaskade 2 (C, by B. Erdmann & R. Roitzsch)

1992 Kaskade 3 (C++, by R. Beck)

1994 Kaskade 4 (C++, extended implementation by R. Beck)

1996 Kaskade 5 (C++, new implementation by R. Beck)

2000 Kaskade 6 (C++, new implementation by L. Zschiedrich et al.)
($\rightarrow$ commercial code JCMwave)

2007 Kaskade 7 (C++, new implementation based on Dune by M. Weiser)

Kaskade:
a sequence of independent codes sharing the same mathematical spirit

Restaurant „La Grande Cascade“, Paris

Short History of Kaskade 7

MATHEON

1989	*Concepts of an adaptive finite element code* (Deuflhard/Leinen/Yserentant) Kaskade 1 (C, by P. Leinen)
1991	Kaskade 2 (C, by B. Erdmann & R. Roitzsch)
1992	Kaskade 3 (C++, by R. Beck)
1994	Kaskade 4 (C++, extended implementation by R. Beck)
1996	Kaskade 5 (C++, new implementation by R. Beck)
2000	Kaskade 6 (C++, new implementation by L. Zschiedrich et al.) ($\rightarrow$ commercial code JCMwave)
2007	Kaskade 7 (C++, new implementation based on Dune by M. Weiser)

Kaskade:
a sequence of independent codes
sharing the same mathematical spirit

ZIB, Heilbronner Str., Berlin

Short History of Kaskade 7

MATHEON

1989	*Concepts of an adaptive finite element code* (Deuflhard/Leinen/Yserentant)
	Kaskade 1 (C, by P. Leinen)
1991	Kaskade 2 (C, by B. Erdmann & R. Roitzsch)
1992	Kaskade 3 (C++, by R. Beck)
1994	Kaskade 4 (C++, extended implementation by R. Beck)
1996	Kaskade 5 (C++, new implementation by R. Beck)
2000	Kaskade 6 (C++, new implementation by L. Zschiedrich et al.)
	($\rightarrow$ commercial code JCMwave)
2007	Kaskade 7 (C++, new implementation based on Dune, by M. Weiser)

Kaskade:
a sequence of independent codes
sharing the same mathematical spirit

ZIB, Takustr. 7, Berlin

MATHEON

PDE Problem Classes

Second Order Variational Problems

Prototype Poisson problem

$$
\begin{aligned}
-\Delta u &= f \quad x \in \Omega \\
\nabla u^T n + \gamma(u - u_\Gamma) &= 0 \quad x \text{ on } \partial\Omega
\end{aligned}
$$

strong formulation

integration by parts (Theorem of Gauß)

$$
\int_\Omega \nabla u^T \nabla v - f v \, dx + \int_{\partial\Omega} \gamma(u - u_\Gamma) v \, ds = 0 \qquad \forall v \in H^1(\Omega)
$$

weak formulation

$$
\min_{u \in H^1(\Omega)} \int_\Omega \frac{1}{2} |\nabla u|^2 - f u \, dx + \int_{\partial\Omega} \frac{\gamma}{2} (u - u_\Gamma)^2 \, ds
$$

variational functional

Second Order Variational Problems

Prototype Stokes problem

$$
\begin{aligned}
-\Delta u + \nabla p &= s & x &\in \Omega \\
-\operatorname{div} u &= 0 & x &\in \Omega \\
u &= 0 & x &\text{ on } \partial\Omega
\end{aligned}
$$

strong formulation

integration by parts (Theorem of Gauß)

$$
\begin{aligned}
\int_\Omega u_x : v_x - p \operatorname{div} v - s^T v \, dx &= 0 \\
\int_\Omega -q \operatorname{div} u \, dx &= 0
\end{aligned}
\qquad \forall v \in H_0^1(\Omega)^3, q \in L^2(\Omega)
$$

weak formulation

$$
\nabla \left[\int_\Omega \frac{1}{2} |u_x|_F^2 - p \operatorname{div} u - s^T u \, dx \right] = 0
$$

variational functional

Second Order Variational Problems

Example incompressible Navier-Stokes

$$
\begin{aligned}
-\nu \Delta u + u_x u + \nabla p &= s \\
\operatorname{div} u &= 0 \\
u|_{\partial\Omega} &= 0
\end{aligned}
$$

strong formulation

integration by parts (Theorem of Gauß)

$$\int_\Omega \nu u_x : v_x + u^T u_x^T v - p \operatorname{div} v - s^T v \, dx = 0$$

$$\int_\Omega q \operatorname{div} u \, dx = 0$$

$$\forall v \in H_0^1(\Omega)^3, q \in L^2(\Omega)$$

weak formulation

Second Order Variational Problems

„General“ minimization problems

$$U = U_1 \times \cdots \times U_n$$

$$\min_{u \in U} \int_\Omega f(x, u_1, \nabla u_1, \ldots, u_n, \nabla u_n)\, dx + \int_{\partial\Omega} g(x, u_1, \ldots, u_n)\, ds$$

„General“ weak formulations

$U = U_1 \times \cdots \times U_n$ ansatz space

$V = V_1 \times \cdots \times V_m$ test space

$$\int_\Omega (F_i(x, u, \nabla u) v_i + \hat{F}_i(x, u, \nabla u) \nabla v_i)\, dx + \int_{\partial\Omega} G(x, u) v_i\, ds = 0, \quad i = 1, \ldots, m$$

Second Order Variational Problems

„General“ minimization problems

$$U = U_1 \times \cdots \times U_n$$

$$\min_{u \in U} \int_\Omega f(x, u_1, \nabla u_1, \ldots, u_n, \nabla u_n)\, dx + \int_{\partial\Omega} g(x, u_1, \ldots, u_n)\, ds$$

Poisson prototype

$$U = H^1(\Omega)$$

$$f(x, u, \nabla u) = \frac{1}{2}|\nabla u|^2 - su$$
$$g(x, u) = \frac{\gamma}{2}(u - u_\Gamma)^2$$

Second Order Variational Problems

„General“ stationary problems

$$U = U_1 \times \cdots \times U_n$$

$$\frac{\partial}{\partial u}\left[\int_\Omega f(x, u_1, \nabla u_1, \ldots, u_n, \nabla u_n)\, dx + \int_{\partial\Omega} g(x, u_1, \ldots, u_n)\, ds\right] = 0$$

Stokes prototype

$$U = H_0^1(\Omega)^3 \times L^2(\Omega)$$

$$f(x, u, \nabla u, p, \nabla p) = \frac{1}{2}|\nabla u|^2 - p\,\mathrm{div}\, u - s^T u$$

$$g(x, u, p) = \frac{\gamma}{2}|u|^2, \quad \gamma \to \infty$$

Second Order Variational Problems

MATHEON

„General“ weak formulations

$$U = U_1 \times \cdots \times U_n \qquad \text{ansatz space}$$

$$V = V_1 \times \cdots \times V_m \qquad \text{test space}$$

$$\int_\Omega (F_i(x,u,\nabla u)v_i + \hat{F}_i(x,u,\nabla u)\nabla v_i)\,dx + \int_{\partial\Omega} G(x,u)v_i\,ds = 0, \quad i = 1,\ldots,m$$

Incompressible Navier Stokes

$$U = V = H_0^1(\Omega)^3 \times L^2(\Omega)$$

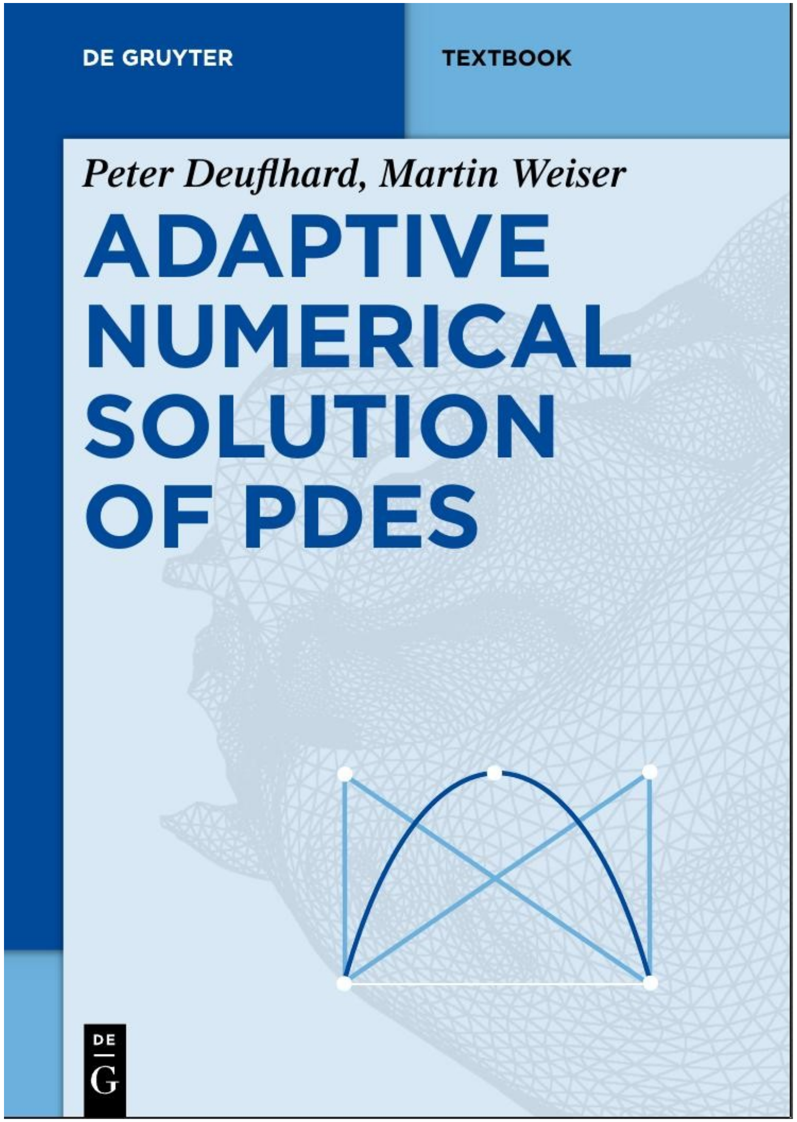
$$F_1 = u^T u_x^T - s^T \quad \hat{F}_1 = \nu u_x - p\mathbf{1}^T$$

$$F_2 = \operatorname{div} u \qquad \hat{F}_2 = 0$$

$$G_1 = \gamma u^T, \quad \gamma \to \infty$$

$$G_2 = 0$$

Reference



Contents

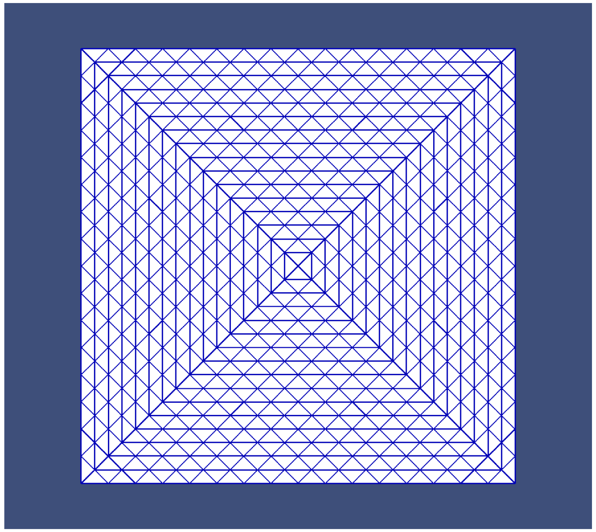
1. Elementary PDEs
2. PDEs in Science and Technology
3. Finite Difference Methods for Poisson Problems
4. Galerkin Methods
5. Numerical Solution of Linear Elliptic Grid Equations
6. Construction of Adaptive Hierarchical Meshes
7. Adaptive Multigrid Methods for Linear Elliptic Problems
8. Adaptive Solution of Nonlinear Elliptic Problems
9. Adaptive Integration of Parabolic Problems

Kaskade 7 Tutorial

Poisson Equation

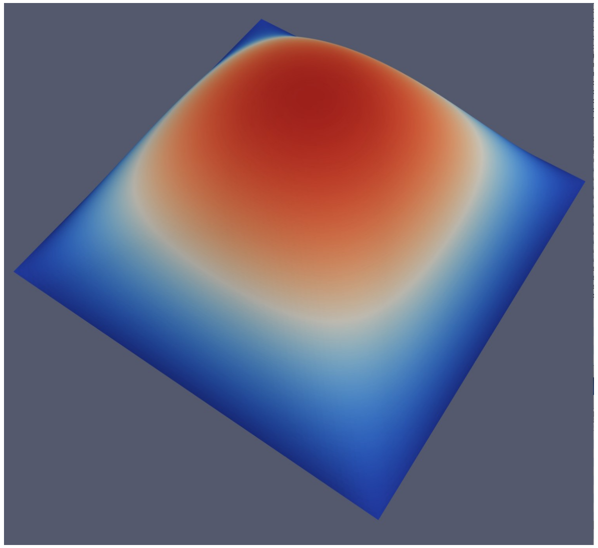
PDE and boundary conditon

$$\begin{aligned} -\Delta y &= 1 \quad \text{in } \Omega =]0,1[^2 \\ y &= 0 \quad \text{on } \partial\Omega \end{aligned}$$



Discretization

- triangulation of region (mesh)
- finite elements of order 2



see tutorial/laplace/

Poisson Equation

MATHEON ZIB

main program – definition of mesh and finite element space

```
// two-dimensional space: dim=2
typedef Dune::UGGrid<dim> Grid;

// a gridmanager is constructed
// as connector between geometric and algebraic information
Dune::FieldVector<double,dim> c0(0.0), dc(1.0);       // corner and side lengths for unit rectangle
GridManager<Grid> gridManager( createRectangle<Grid>(c0,dc,1.0,true) );

// construction of linear continuous finite element space
// for the scalar solution u
typedef FEFunctionSpace<ContinuousLagrangeMapper<double,LeafView> > H1Space;
H1Space temperatureSpace(gridManager,gridManager.grid().leafView(),1);

typedef boost::fusion::vector<H1Space const*> Spaces;     // list of used fe-spaces
Spaces spaces(&temperatureSpace);

// construct variable list.
typedef Variable< SpaceIndex<0>, Components<1>, VariableId<0> > Variable_0;
typedef boost::fusion::vector<Variable_0> VariableDescriptions;
std::string varNames[1] = { "u" };

typedef VariableSetDescription<Spaces,VariableDescriptions> VariableSetDesc;
VariableSetDesc variableSetDescription(spaces,varNames); // contains all information on variables
 ...
```

Poisson Equation

MATHEON ZIB

main program – assemble and solve

```
// construct variational functional
typedef HeatFunctional<double,VariableSetDesc> Functional;
Functional F;

//construct Galerkin representation
typedef VariationalFunctionalAssembler<LinearizationAt<Functional> > Assembler;

typedef VariableSetDesc::CoefficientVectorRepresentation<>::type CoefficientVectors;
Assembler assembler(spaces);

VariableSetDesc::Representation u(variableSetDescription);      // representation of a fe-function

assembler.assemble(linearization(F,u));

CoefficientVectors rhs(assembler.rhs());            // F'
CoefficientVectors
solution(VariableSetDesc::CoefficientVectorRepresentation<>::init(variableSet));

AssembledGalerkinOperator<Assembler> A(assembler, onlyLowerTriangle); // F''

// solve with 1 Newton step
// i.e. solution = solution - A^{-1}*rhs
directInverseOperator(A,directType,property).applyscaleadd(-1.0,rhs,solution);
u.data = solution.data;
...
```

Poisson Equation

main program – output for graphic device

```
// output of solution in VTK format for visualization,
// the data are written as ascii stream into file temperature.vtu,
// possible is also binary, order > 2 is not supported
writeVTKFile(gridManager.grid().leafView(),u,"temperature");
std::cout << "data in VTK format is written into file temperature.vtu \n";
```