UPPSALA UNIVERSITET

High Performance Computing II

Maya Neytcheva, Petia Boyanova, Xunxun Wu

Department of Information Technology

Uppsala University

NGSSC
FORSKARSKOLA I TEKNISK-VETENSKAPLIGA BERÄKNINGAR
NATIONAL GRADUATE SCHOOL IN SCIENTIFIC COMPUTING

UPPSALA UNIVERSITET

Introduction to *deal.II* and *Trilinos*

Plan of the lecture:

- What do we want to do?
- What is *deal.II*?
  - Where to find it
  - What does it do
  - How to use it
  - Examples
- What is *Trilinos*?
  - Where to find it
  - What does it do
  - How to use it
  - Examples

What do we want to do?

Solve a PDE numerically.
Test example: Find u such that

$$\begin{aligned} -\Delta u &= f, \quad in\, \Omega \subset \mathbb{R}^2 \\ u &= 0, \quad on\, \partial\Omega \end{aligned}$$

where $\Omega = [-1, 1]^2$, and $f = 1$.

What do we want to do, cont.

For setting up the spatial FE approximation, the first step is to rewrite the above equation in variational form.
Let $V = \{v : \|\nabla v\| + \|v\| < \infty,\ v|_{\partial\Omega} = 0\}$. Multiplying the equation with a test-function $v \in V$ and integrating over Ω using Green's formula with the homogeneous Dirichlet boundary conditions, we obtain

$$\begin{aligned}
\int_\Omega f\, v\, dx &= -\int_\Omega \Delta u\, v\, dx \\
&= \int_\Omega \nabla u \cdot \nabla v\, dx - \int_{\partial\Omega} \mathbf{n} \cdot \nabla u\, v\, dx \\
&= \int_\Omega \nabla u \cdot \nabla v\, dx
\end{aligned}$$

What do we want to do, cont.

The variational form is thus defined to be the following problem:
Find $u \in V$ such that

$$\int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega f \, v \, dx, \quad \forall v \in V$$

UPPSALA UNIVERSITET

What do we want to do, cont.

In order to formulate the discretization in space, we decompose the infinite-dimensional computational domain Ω into finite-dimensional subsets (elements) with a characteristic size h. Let $\mathcal{K}$ be a triangulation of Ω, and let $V_h \subset V$ be the subspace of continuous piecewise linears on $\mathcal{K}$

$$V_h = \{v \in V,\, v|_{\partial\Omega} = 0\}$$

With this choice of approximation space, the discrete space counterpart of the equation reads:
Find $U \in V_h$ such that

$$\int_\Omega \nabla U \cdot \nabla v \, dx = \int_\Omega f \, v \, dx, \quad \forall v \in V_h$$

What do we want to do, cont.

Next, to compute the finite element approximation U we let $\{\varphi_i\}_{i=1}^N$ be the basis for the subspace V_h. Since U belong to V_h it can be written as:

$$U = \sum_{j=1}^{N} \mathbf{u}_j \varphi_j$$

with N unknowns $\mathbf{u}_j, j = 1, 2, \cdots, N$, to be found.
This equation can be rewritten as a linear system by inserting the representation $U = \sum_{j=1}^N \mathbf{u}_j \varphi_j$. Using the notation

$$A_{ij} = \int_\Omega \nabla\varphi_i \, \nabla\varphi_j \, dx, \quad b_i = \int_\Omega f \, \varphi_i \, dx, \quad i = 1, 2, \cdots, N$$

we have

$$b_i = \sum_{j=1}^{N} A_{ij} \, \mathbf{u}_j, \quad i = 1, 2, \cdots, N$$

UPPSALA UNIVERSITET

What do we want to do, cont.

The linear system for the unknowns $\mathbf{u}_j$ in matrix form:

$$A\mathbf{u} = \mathbf{b}$$

Now we know what we are going to solve, and we can look at how to compute A_{ij} and $\mathbf{b}_i$ (form the integrals).
In the finite element method, this is most commonly done using some quadrature.

UPPSALA UNIVERSITET

What do we want to do, cont.

We first split the integral over the whole domain into integrals over all cells,

$$
\begin{aligned}
A_{ij}^K &= \sum_{K \in \mathcal{K}} \int_K \nabla \varphi_i \, \nabla \varphi_j \\
b_i^K &= \sum_{K \in \mathcal{K}} \int_K f \, \varphi_i
\end{aligned}
$$

and then approximate the integrals in each cell K by quadrature

$$
\begin{aligned}
A_{ij}^K &\approx \sum_q \int_K \nabla \varphi_i(x_q^K) \, \nabla \varphi_j(x_q^K) \, \omega_j^K \\
b_i^K &\approx \sum_q \int_K f(x_q^K) \, \varphi_i(x_q^K) \, \omega_j^K
\end{aligned}
$$

UPPSALA UNIVERSITET

What do we want to do, cont.

After A and $\mathbf{b}$ are made available, we have to choose a suitable numerical solution to solve the system.

- fast
- accurate
- robust

UPPSALA UNIVERSITET

How to choose a package?

Why *deal.II* and *Trilinos*?

UPPSALA UNIVERSITET

What is *deal.II*

deal.II: A Finite Element Differential Equations Analysis Library

- a C++ program library targeted at the computational solution of partial differential equations using adaptive finite elements
- aims: to enable rapid development of modern finite element codes, using among other aspects adaptive meshes and a wide array of tools classes often used in finite element program
- seemles using 1D, 2D or 3D programs
- locally refined grids, adaptive refinement strategies and error indicators and error estimators.

What is *deal.II*

deal.II: A Finite Element Differential Equations Analysis Library

- h, p, hp refinement
- continuous and discontinuous elements
- support for a variety of finite elements
- complete stand-alone linear algebra library
- interface to other packages such as Trilinos, PETSc and METIS
- smooth transition from serial to parallel
- online documentation

UPPSALA UNIVERSITET

What is *deal.II*

- Modern software techniques that make access to the complex data structures and algorithms as transparent as possible
- Support for several output formats
- Portable support for a variety of computer platforms and compilers
- Free source code under an Open Source license
- open to contributors

For its creation, its principal authors have received the 2007 J. H. Wilkinson Prize for Numerical Software.

Links to *deal.II*:

http://www.dealii.org/

http://dealii.sourceforge.net/index.php/Main_Pag

http://www.dealii.org/developer/index.html

UPPSALA UNIVERSITET

deal.II Homepage - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Regionbussarna/Upptåget - ... | Rayleigh quotient - Wikipedia,... | openSUSE 12.1 Screenshot ... | deal.II Homepage

www.dealii.org

Most Visited | openSUSE | Getting Started | Latest Headlines | Mozilla Firefox

deal.II

Home
README
Download

Documentation

7.1.0
7.0.0
6.3.1 ~~6.3.0~~
6.2.1 ~~6.2.0~~
6.1.0
6.0.0
(subversion)

Information

FAQ
News
Mailing lists
Wiki
Bug tracker

Resources

Authors
Development
Publications
License
Legal notice
Download statistics

Testsuite

Build tests
Regression tests
Testing info

October 9th, 2011: deal.II 7.1 released (download, changes).

deal.II: A Finite Element Differential Equations Analysis Library

What is deal.II?

deal.II is a C++ program library targeted at the computational solution of partial differential equations using adaptive finite elements. It uses state-of-the-art programming techniques to offer you a modern interface to the complex data structures and algorithms required.

The main aim of deal.II is to enable rapid development of modern finite element codes, using among other aspects adaptive meshes and a wide array of tools classes often used in finite element program. Writing such programs is a non-trivial task, and successful programs tend to become very large and complex. We believe that this is best done using a program library that takes care of the details of grid handling and refinement, handling of degrees of freedom, input of meshes and output of results in graphics formats, and the like. Likewise, support for several space dimensions at once is included in a way such that programs can be written independent of the space dimension without unreasonable penalties on run-time and memory consumption.

deal.II is widely used in many academic and commercial projects. For its creation, its principal authors have received the 2007 J. H. Wilkinson Prize for Numerical Software. It is also part of the industry standard SPEC CPU 2006 benchmark suite used to determine the speed of computers and compilers, and comes pre-installed on the machines offered by the commercial Sun Grid program.

deal.II emerged from work at the Numerical Methods Group at Universität Heidelberg, Germany, which is at the forefront of adaptive finite element methods and error estimators. Today, it is maintained by two dozen original authors at Texas A&M University, and dozens of contributors and several hundred users are scattered around the world (see our credits page for

deal.II

Selected images

UPPSALA UNIVERSITET

Main Page - WikiDeal - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Regionbussarna/Upptåget - ... | Rayleigh quotient - Wikipedia,... | openSUSE 12.1 Screenshot ... | Main Page - WikiDeal

dealii.sourceforge.net/index.php/Main_Page

Most Visited | openSUSE | Getting Started | Latest Headlines | Mozilla Firefox

Log in

page | discussion | view source | history

Main Page

This is the Wiki page for the http://www.dealii.org library. We provide it as a space for deal.II users to present projects and results obtained with the help of deal.II. Please feel free to create pages in which you explain your project, show pictures, share experiences, or show code. You may also link to descriptions of your projects elsewhere on the web.

If you find inaccuracies, the proper thing in a wiki is to edit pages as you wish. A description of how a wiki works and what you can do can be found at User's Guide. **Unfortunately, due to spambots attacking wikis, you have to be given write privileges to the wiki by hand — which we are more than happy to do, just send an email to developer@dealii.org!**

This wiki is not meant to replace the deal.II mailing list or homepage, but to supplement it. Therefore, continue submitting questions and answers to the mailing list. If the answers involve pieces of code or longer explanations, then this here might be a better place. For more information and documentation on deal.II, please also consult the deal.II homepage at http://www.dealii.org.

This wiki is set up and run by Luca Heltai. The deal.II authors are grateful for his help!

deal.II FAQ

Here we collect a number of questions which have been asked a lot of times in the deal.ii mailing list. If you feel you have a question that deserves to be here, feel free to add it somewhere in this page. Hopefully somebody will answer!

Installing deal.II

Deal.II is compatible with almost all hardware out there. Here we point out some tips and tricks for the installation on different kinds of architectures.

Gallery

This page has a number of images that were the products of simulations and which are posted mostly for the pleasure of viewing, but also with a brief explanation of what they depict.

People

A list of the people involved in deal.II and options of getting involved yourself.

Requested Features

Clicking on this link takes you to a page where we list features that are commonly requested for deal.II. Feel free to add to this list if you miss some functionality.

Events

navigation
- Main page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search

Go | Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

UPPSALA UNIVERSITET

Step-by-Step Examples - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Regionbussarna/Upptåget - ... | Rayleigh quotient - Wikipedia,... | openSUSE 12.1 Screenshot ... | Step-by-Step Examples

www.dealii.org/developer/doxygen/tutorial/index.html

Most Visited | openSUSE | Getting Started | Latest Headlines | Mozilla Firefox

Overview

Connections between programs

Programs by number

Programs by topic

Programs by number:

1 2 3
4 5 6
7 8 9
10 11 12
13 14 15
16 17 18
19 20 21
22 23 24
25 27 28
29 30 31
32 33 34
35 36 38
39 40 45
46

Back to the deal.II

The deal.II tutorial contains a collection of programs, each more or less built atop of previous ones, which demonstrate various aspects of the library. Each such example has the following structure:

1. **Introduction:** What the program does, including the mathematical model, and what programming techniques are new.
2. **The commented program:** An extensively documented listing of the source code.
3. **Results:** The output of the program, with comments and interpretation.
4. **The plain program:** The source code stripped of all comments.

You can browse the available tutorial programs

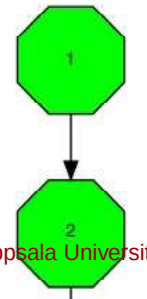
1. as **a graph (see below)** that shows how tutorial programs build upon each other.
2. as **a list** that provides a short synopsis of each program.
3. or **grouped by topic**.

Some of the programs also jointly form the geodynamics demonstration suite.

The programs are in the `examples/` directory of your local deal.II installation. After compiling the library itself, if you go into one of the tutorial directories, you can compile the program by typing **make**, and run it using **make run**. The latter command also compiles the program if that has not already been done. The Makefiles in the different directories are based on the small program Makefile template described in this section.

Connections between tutorial programs

The following graph shows the connections between tutorial programs and how they build on each other. Click on any of the boxes to go to one of the programs. If you hover your mouse pointer over a box, a brief description of the program should appear.



UPPSALA UNIVERSITET

Step-by-Step Examples - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Regionbussarna/Upptåget - ... | Rayleigh quotient - Wikipedia,... | openSUSE 12.1 Screenshot ... | Step-by-Step Examples

www.dealii.org/developer/doxygen/tutorial/index.html

Most Visited | openSUSE | Getting Started | Latest Headlines | Mozilla Firefox

Overview

Connections between programs

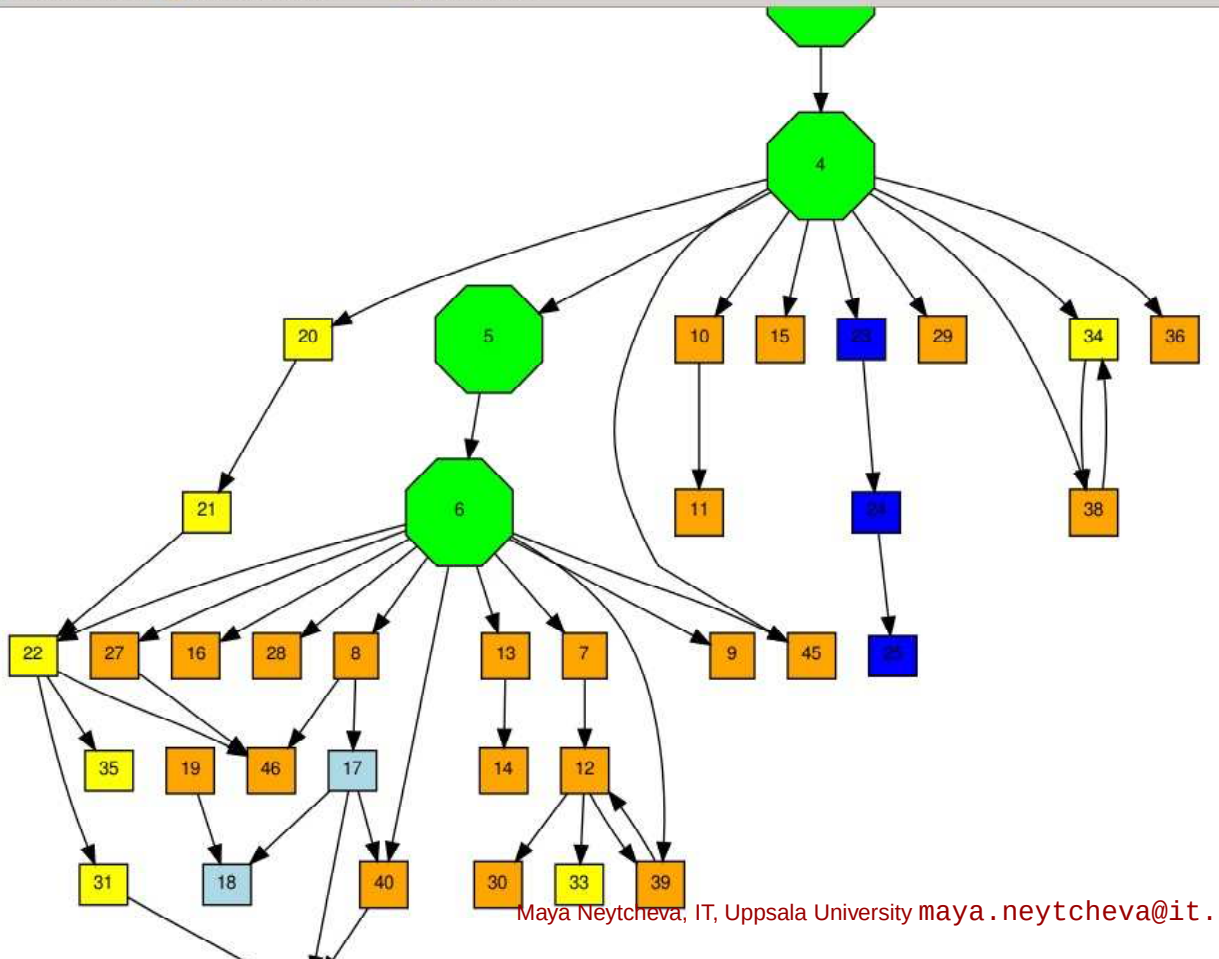
Programs by number

Programs by topic

Programs by number:

1 2 3
4 5 6
7 8 9
10 11 12
13 14 15
16 17 18
19 20 21
22 23 24
25 27 28
29 30 31
32 33 34
35 36 38
39 40 45
46

Back to the deal.II



Trillinos

What is *Trilinos*?

The Trilinos Project - Home - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Regionbussarna/Upptåget - ... | Rayleigh quotient - Wikipedia,... | openSUSE 12.1 Screenshot ... | The Trilinos Project - Home

trilinos.sandia.gov

Most Visited | openSUSE | Getting Started | Latest Headlines | Mozilla Firefox

The Trilinos Project

Sandia National Laboratories

Home | About | Capability Areas | Resources | Packages | Download

- Getting Started
- News
- Contact
- Citing Trilinos

Welcome to the Trilinos Project Home Page

The Trilinos Project is an effort to develop algorithms and enabling technologies within an object-oriented software framework for the solution of large-scale, complex multi-physics engineering and scientific problems. A unique design feature of Trilinos is its focus on packages.

Trilinos Release 10.8 Now Available

Trilinos 10.8 is now available for download.

See the 10.8 release notes for more information.

Trilinos 10.8 Release

Trilinos 10.8 is now available for download. The 10.8 release uses a CMake build system, which requires CMake version 2.8 or greater. Instructions for building Trilinos 10.0 and later are available here.

Trilinos User Group 2011: Register Now!

The 2011 Trilinos User Group Meeting (TUG) is scheduled for October 31st- November 3rd in Albuquerque. The meeting will feature a one half day tutorial, two days of talks directed at users, and one day of Trilinos developer presentations and discussions. In limited cases, interested users may be able to attend the developer day on November 3rd. Please inquire if you are interested when registering. The format will closely follow the format used last year. To learn more about TUG 2011, or to register, please visit the TUG 2011 Homepage.

Trilinos Featured in RCE HPC Podcast

Trilinos was featured in a recent Research, Computing, & Engineering (RCE) High Performance Computing (HPC) podcast hosted by Brock Palen and Jeff Squyres. The podcast includes a very high-level summary of Trilinos and touches on other interesting topics like the origin of the name "Trilinos". Visit the RCE-cast website for more details and to download the podcast.

UPPSALA UNIVERSITET

The Trilinos Project - Packages - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Regionbussarna/Upptåget - ... | Rayleigh quotient - Wikipedia,... | openSUSE 12.1 Screenshot ... | The Trilinos Project - Packages

trilinos.sandia.gov/packages/

Most Visited | openSUSE | Getting Started | Latest Headlines | Mozilla Firefox

The Trilinos Project

Sandia National Laboratories

Home | About | Capability Areas | Resources | Packages | Download

Interoperability

Publicly-Released:

- Amesos
- Anasazi
- AztecOO
- Belos
- CTrilinos
- Didasko
- Epetra
- EpetraExt
- FEI
- ForTrilinos
- Galeri
- GlobiPack
- IFPACK
- Ifpack2
- Intrepid
- Isorropia
- Kokkos
- Komplex
- LOCA
- Meros
- Mesquite
- ML
- Moertel
- MOOCHO
- NewPackage

Each Trilinos package is a self-contained, independent piece of software with its own set of requirements, its own development team and group of users. Because of this, Trilinos itself is designed to respect the autonomy of packages. Trilinos offers a variety of ways for a particular package to interact with other Trilinos packages. It also offers a set of tools that can assist package developers with builds across multiple platforms, generating documentation and regression testing across a set of target platforms. At the same time, what a package must do to be called a Trilinos package is minimal, and varies with each package.

Basic Linear Algebra Libraries

- Epetra - Core linear algebra package. Facilitates construction and manipulation of distributed and serial graphs, sparse and dense matrices, vectors and multivectors.
- EpetraExt - Extensions to the core linear algebra package, Epetra.
- Tpetra - Next-generation, templated version of Petra, taking advantage of the newer advanced features of C++.
- Jpetra - Experimental Java version of the Petra library.
- Kokkos - Core kernel package.

Preconditioners

- AztecOO - ILU-type preconditioner. See also description under "Linear Solvers."
- IFPACK - Distributed algebraic preconditioner package. Includes incomplete factorizations and relaxation-based preconditioners in domain decomposition framework. Compatible with AztecOO.
- Ifpack2 - Contains preconditioners that operate on the templated linear-algebra objects provided by the Tpetra package. Intended as a templated replacement for Ifpack.

UPPSALA UNIVERSITET

ML - Home - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Regionbussarna/Upptåget - ... | Rayleigh quotient - Wikipedia,... | openSUSE 12.1 Screenshot ... | ML - Home

trilinos.sandia.gov/packages/ml/

Most Visited | openSUSE | Getting Started | Latest Headlines | Mozilla Firefox

ML Multi-Level Preconditioning Package

Sandia National Laboratories

- ML Home
- About
  - Overview
  - Team
- Downloads
- Mail Lists
- Documentation
  - Publications
  - Trilinos Release 10.8
  - Trilinos Release 10.6
  - Trilinos Release 10.4
  - Trilinos Release 10.2
  - Trilinos Release 10.0
  - Trilinos Release 9.0
  - Trilinos Release 8.0
  - Trilinos Release 7.0
  - Trilinos Release 6.0
  - Trilinos Release 5.0
  - Trilinos Release 4.0
  - Development
- Help
  - FAQ
  - Contact
- ML API
- User Guides
- Citations

ML: Multilevel Preconditioning Package

Welcome to the homepages for ML, Sandia's main multigrid preconditioning package. ML is designed to solve large sparse linear systems of equations arising primarily from elliptic PDE discretizations. ML is used to define and build multigrid solvers and preconditioners, and it contains black-box classes to construct highly-scalable smoothed aggregation preconditioners. ML preconditioners have been used on thousands of processors for a variety of problems, including the incompressible Navier-Stokes equations with heat and mass transfer, linear and nonlinear elasticity equations, the Maxwell equations, semiconductor equations, and more.

Please use the links on the left of this page to navigate, or use the links below:

- An **overview** of ML.
- **Documentation** generated directly from ML's source code by Doxygen.
- **User Guides**.
- **Mail lists** for users and developers.
- Relevant **papers**.
- **Who are the ML developers?**

ML documentation is created and maintained using Doxygen. Click **here** to access the latest Doxygen documentation, containing details about ML and its classes, examples of usage, how to interface with other Trilinos packages, and more.

If you use ML for your applications, please let us know by writing an e-mail to the ml developers. Please also cite ML using the following bibtex entry:

```
@TechReport{ml-guide,
  author      =  {M.W. Gee and C.M. Siefert and J.J. Hu and R.S. Tuminaro and M.G. Sala},
  title       =  {{ML} 5.0 Smoothed Aggregation User's Guide},
  institution =  {Sandia National Laboratories},
  year        =  {2006},
  number      =  {SAND2006-2649},
}
```

- Trilinos Home
- Trilinos Packages
  - Amesos
  - Anasazi
  - AztecOO
  - Belos
  - Claps
  - Didasko
  - Epetra
  - EpetraExt
  - FEI
  - ForTrilinos
  - Galeri
  - GlobiPack
  - IFPACK
  - Intrepid
  - Isorropia
  - Jpetra
  - Kokkos
  - Komplex
  - LOCA
  - Meros
  - Mesquite
  - ML
  - Moertel
  - MOOCHO
  - New_Package
  - NOX
  - Optika
  - OptiPack
  - PANGEN
  - Phalanx
  - phdMesh

UPPSALA UNIVERSITET

ML: ML: Multi Level Preconditioning Package - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Regionbussarna/Upptåget - ... | Rayleigh quotient - Wikipedia,... | openSUSE 12.1 Screenshot ... | ML: ML: Multi Level Preconditi...

trilinos.sandia.gov/packages/docs/dev/packages/ml/doc/html/index.html

Most Visited | openSUSE | Getting Started | Latest Headlines | Mozilla Firefox

ML Version of the Day

Main Page | Related Pages | Namespaces | Classes | Files | Directories | Search

ML: Multi Level Preconditioning Package

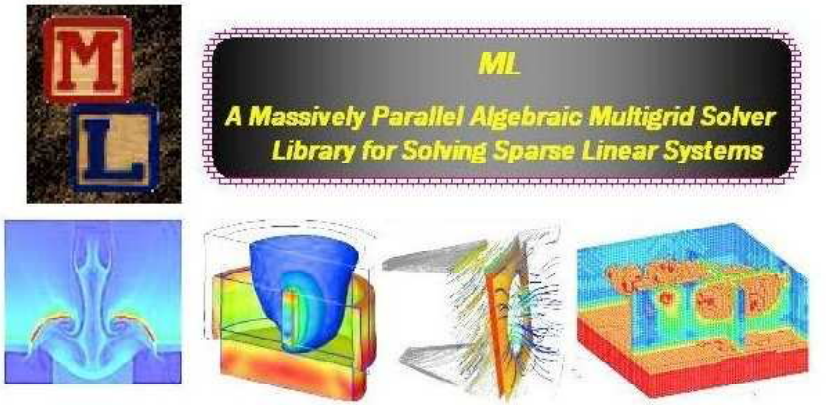


Table of Contents

How do we program with *deal.II*

Note: we need

- mesh (choice of finite elements)
- assembly of matrices (choice of basis functions)
- solution methods (nonlinear and linear)
- paraelization tools
- visualization

Mesh- and finite-element related:

```
#include <deal.II/grid/tria.h>
#include <deal.II/dofs/dof_handler.h>
#include <deal.II/grid/grid_generator.h>
#include <deal.II/grid/tria_accessor.h>
#include <deal.II/grid/tria_iterator.h>
#include <deal.II/dofs/dof_accessor.h>

#include <deal.II/fe/fe_values.h>
#include <deal.II/base/quadrature_lib.h>

#include <deal.II/base/function.h>
```

Matrix/vector data structure and

```
#include <deal.II/lac/vector.h>
#include <deal.II/lac/full_matrix.h>
#include <deal.II/lac/sparse_matrix.h>
#include <deal.II/lac/compressed_sparsity_patter

#include <deal.II/lac/solver_cg.h>
#include <deal.II/lac/precondition.h>
#include <deal.II/lac/sparse_direct.h>

#include <deal.II/lac/trilinos_precondition.h>
```

Making the grid:

```
void laplace_problem::make_grid ()
{
  GridGenerator::hyper_cube (triangulation, -1,

  triangulation.refine_global (n_refinement_step
  std::cout << "Total number of cells: "
    << triangulation.n_cells()
    << std::endl;
}
```

```
void laplace_problem::setup_system ()
{
  dof_handler.distribute_dofs (fe);
  std::cout << "Number of degrees of freedom: "
    << dof_handler.n_dofs()
    << std::endl;
  CompressedSparsityPattern c_sparsity(dof_handler.n_dofs());
  DoFTools::make_sparsity_pattern (dof_handler, c_sparsity);
  sparsity_pattern.copy_from(c_sparsity);

  system_matrix.reinit (sparsity_pattern);

  solution.reinit (dof_handler.n_dofs());
  system_rhs.reinit (dof_handler.n_dofs());
}
```

```
void laplace_problem::assemble_system ()
{
  QGauss<2>  quadrature_formula(2);
  FEValues<2> fe_values (fe, quadrature_formula,
 update_values | update_gradients | update_JxW_v
  const unsigned int   dofs_per_cell = fe.dofs_p
  const unsigned int   n_q_points    = quadratur

  FullMatrix<double>   cell_matrix (dofs_per_cel
  Vector<double>       cell_rhs (dofs_per_cell);

  std::vector<unsigned int> local_dof_indices (d
```

```
  DoFHandler<2>::active_cell_iterator
    cell = dof_handler.begin_active(),
    endc = dof_handler.end();
  for (; cell!=endc; ++cell)
    {
      fe_values.reinit (cell);
      cell_matrix = 0;
      cell_rhs = 0;
      for (unsigned int i=0; i<dofs_per_cell; ++i)
for (unsigned int j=0; j<dofs_per_cell; ++j)
  for (unsigned int q_point=0; q_point<n_q_points; ++q_point)
    cell_matrix(i,j) += (fe_values.shape_grad (i, q_point) *
 fe_values.shape_grad (j, q_point) *
 fe_values.JxW (q_point));
```

```
      for (unsigned int i=0; i<dofs_per_cell; ++i)
for (unsigned int q_point=0; q_point<n_q_points; ++q_point)
  cell_rhs(i) += (fe_values.shape_value (i, q_point) *
  1 *
  fe_values.JxW (q_point));

      cell->get_dof_indices (local_dof_indices);

      for (unsigned int i=0; i<dofs_per_cell; ++i)
for (unsigned int j=0; j<dofs_per_cell; ++j)
  system_matrix.add (local_dof_indices[i],
     local_dof_indices[j],
     cell_matrix(i,j));

      for (unsigned int i=0; i<dofs_per_cell; ++i)
system_rhs(local_dof_indices[i]) += cell_rhs(i);
    }
```

```
  std::map<unsigned int,double> boundary_values;
  VectorTools::interpolate_boundary_values (dof_
  MatrixTools::apply_boundary_values (boundary_v
      system_matrix,
      solution,
      system_rhs);
}
```

Solving the linear system: direct method

```
void laplace_problem::solve_direct ()
{
    SparseDirectUMFPACK direct_solver;
    direct_solver.initialize(system_matrix);
    direct_solver.vmult (solution, system_rhs);
}
```

Solving the linear system: unpreconditioned CG

```
void laplace_problem::solve_cg ()
{

  SolverControl   solver_control (system_matrix.m(), 1e-12);

  SolverCG<>      solver (solver_control);

  solver.solve (system_matrix, solution, system_rhs,
PreconditionIdentity());
  std::cout<< "CG iterations without preconditioner:"...
              <<solver_control.last_step()<<std::endl;
}
```

Solving the linear system: AMG-preconditioned CG

```
void laplace_problem::solve_amg ()
{
  Amg_preconditioner.reset ();
  Amg_preconditioner = std_cxx1x::shared_ptr<TrilinosWrappers::Preco
                          (new TrilinosWrappers::PreconditionAMG());

  std::vector<std::vector<bool> > constant_modes;
  std::vector<bool>  components (3,true);
  components[2] = false;
  DoFTools::extract_constant_modes (dof_handler, components,
                                          constant_modes);
  TrilinosWrappers::PreconditionAMG::AdditionalData Amg_data;
  Amg_data.constant_modes = constant_modes;
  Amg_data.elliptic = true;
  Amg_data.higher_order_elements = true;
  Amg_data.smoother_sweeps = 2;
  Amg_data.aggregation_threshold = 0.02;
  Amg_preconditioner->initialize(system_matrix, Amg_data);
```

The actual execution part:

```
void laplace_problem::run (int n_refs)
{
  Vector<double>  init_sol;
  Vector<double>  init_rhs;
  double mesh_size;
  n_refinement_steps = n_refs;
  std::cout<< "Number of refinements: " <<n_refinement_steps<< std::

  mesh_size = 2*std::pow(0.5, double(n_refinement_steps));
  pcout << "Mesh size: " << mesh_size<< std::endl;
  make_grid ();
  setup_system();
  assemble_system ();
  init_rhs = system_rhs;
  init_sol = solution;
}
```

```
computing_timer.enter_section("Solve system directly");
 solve_direct ();
 computing_timer.exit_section("Solve system directly");

 solution = init_sol;
 system_rhs = init_rhs;
 computing_timer.enter_section("Solve system (CG)");
 solve_cg ();
 computing_timer.exit_section("Solve system (CG)");

 solution = init_sol;
 system_rhs = init_rhs;
 computing_timer.enter_section("Solve system (AMG)");
 solve_amg ();
 computing_timer.exit_section("Solve system (AMG)");

 output_results ();
}
```