1. Survey of Current Finite Element Software

Anders Logg
logg@tti-c.org

Toyota Technological Institute at Chicago

Sixth Winter School in Computational Mathematics
Geilo, March 5-10 2006

Outline

- Chapter 2 in lecture notes

Classification

- Classify by design:
  - Libraries in existing languages
  - Domain-specific languages (DSL)
- Classify by level of automation:
  - Automatic assembly from variational problem
  - Finite element toolboxes
- Classify by license:
  - Free 
  - Proprietary 

Summary of projects

Project	automatic assembly	library / language	license
Analysa	yes	language	proprietary
deal.II	no	library	QPL*
Diffpack	no	library	proprietary
FEniCS	yes	both	GPL, LGPL
FreeFEM	yes	language	LGPL
GetDP	yes	language	GPL
Sundance	yes	library	LGPL

Test problem: Poisson's equation

Find $U \in V_h$ such that

$$\int_\Omega \nabla v \cdot \nabla U \, \mathrm{d}x = \int_\Omega v \, f \, \mathrm{d}x \quad \forall v \in \hat{V}_h$$

- Compare specification for different systems
- Should be close to the mathematical notation
- Obvious limitations of ASCII
- Should be general
- Should be efficient

deal.II

- Main authors:
  - Wolfgang Bangerth, ICES (formerly TICAM) in Texas
  - Ralf Hartmann, German Aerospace Center in Braunschweig
  - Guido Kanschat, Applied Mathematics in Heidelberg
- Originated in 1992 (DEAL)
- C++ finite element library
- No automatic assembly
- Licensed under the QPL (with additional restrictions)
- `http://www.dealii.org/`

Poisson's equation with deal.II

```
...
for (dof_handler.begin_active(); cell! = dof_handler.end(); ++cell)
{
  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) *
                      <value of right-hand side f> *
                      fe_values.JxW(q_point));

  ...
```

Poisson's equation with deal.II (cont'd)

```
  ...

  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);
}
```

DiffPack

- Main authors:
  - Are Magnus Bruaset, Simula in Oslo
  - Hans Petter Langtangen, Simula in Oslo
- Originated in 1991
- C++ finite element library
- No automatic assembly
- Proprietary (early versions free)
- `http://www.diffpack.com/`

Poisson's equation with Diffpack

```
for (int i = 1; i <= nbf; i++)
  for (int j = 1; j <= nbf; j++)
    elmat.A(i, j) += (fe.dN(i, 1) * fe.dN(j, 1) +
                      fe.dN(i, 2) * fe.dN(j, 2) +
                      fe.dN(i, 3) * fe.dN(j, 3)) *
                      detJxW;

for (int i = 1; i <= nbf; i++)
  elmat.b(i) += fe.N(i)*<value of f>*detJxW;
```

Sundance

- Main author:
  - Kevin Long, Sandia National Laboratories in Livermore
- C++ finite element library
  - PDE-constrained optimization
  - Powerful symbolic engine
  - Differentiation
- Automatic assembly
- Licensed under the LGPL
- `http://software.sandia.gov/sundance/`

Poisson's equation with Sundance

```
Expr v = new TestFunction(new Lagrange(1));
Expr U = new UnknownFunction(new Lagrange(1));
Expr f = new DiscreteFunction(...);

Expr dx = new Derivative(0);
Expr dy = new Derivative(1);
Expr dz = new Derivative(2);
Expr grad = List(dx, dy, dz);

Expr poisson = Integral((grad*v)*(grad*U) - v*f);
```

Analysa

- Main author:
  - L. Ridgway Scott, University of Chicago
  - Babak Bagheri, PROS Revenue Management in Texas
- A scheme-based language for finite element computation
- Automatic assembly
- Proprietary
- `http://people.cs.uchicago.edu/~ridg/al/aa.html`

Poisson's equation with Analysa

```
(integral-forms
  ((a v U) (dot (gradient v) (gradient U)))
  ((m v U) (* v U))
)

(elements
  (element (lagrange-simplex 1))
)

(spaces
  (test-space (fe element (all mesh) r:))
  (trial-space (fe element (all mesh) r:))
)
```

Poisson's equation with Analysa (cont'd)

```
(functions
  (f (interpolant test-space (...)))
)

(define A-matrix (a testspace trial-space))
(define b-vector (m testspace f))
```

The action of a bilinear form

Bilinear form:

$$a : \hat{V}_h \times V_h \to \mathbb{R}$$

Action of the bilinear form:

$$a(\hat{V}_h, \cdot) : V_h \to \mathbb{R}^N$$

Notation:

$$w = a(\hat{V}_h, U) = a(\hat{V}_h, V_h)U = AU$$

where

$$\begin{aligned} w_i &= a(\hat{\phi}_i, U), & i &= 1, 2, \ldots, N \\ A_{ij} &= a(\hat{\phi}_i, \phi_j), & i, j &= 1, 2, \ldots, N \end{aligned}$$

FreeFEM

- Main authors:
  - Olivier Pironneau, Laboratoire Jacques-Louis Lions in Paris
  - Frédéric Hecht, Laboratoire Jacques-Louis Lions in Paris
  - Antoine Le Hyaric, Laboratoire Jacques-Louis Lions in Paris
- Originated in {before 1999}
- A C++-based language for finite element computation
- Automatic assembly
- Licensed under the LGPL
- `http://www.freefem.org/`

Poisson's equation with FreeFEM

```
fespace V(mesh, P1);
V v, U;
func f = ...;

varform a(v, U) = int2d(mesh)(dx(v)*dx(U) +
                              dy(v)*dy(U));

varform L(v) = int2d(mesh)(v*f);
```

GetDP

- Main authors:
  - Patrick Dular, University of Liége (in Belgium)
  - Christophe Geuzaine, CalTech in Pasadena
- Originated in 1997
- A specification-driven computational engine
- Automatic assembly
- Licensed under the GPL
- `http://www.geuz.org/getdp/`

Poisson's equation with GetDP

```
FunctionSpace {
  { Name V; Type Form0;
    BasisFunction {
      { ... }
    }
  }
}
```

Poisson's equation with GetDP (cont'd)

```
Formulation {
  { Name Poisson; Type FemEquation;
    Quantity {
      { Name v; Type Local; NameOfSpace V; }
    }
    Equation {
      Galerkin { [Dof{Grad v}, {Grad v}];
          ....
      }
    }
  }
}
```

The FEniCS project

- Main authors:
  - Johan Hoffman, KTH in Stockholm
  - Johan Jansson, Chalmers in Göteborg
  - Robert C. Kirby, University of Chicago
  - Matthew G. Knepley, ANL in Chicago
  - Anders Logg, TTI in Chicago
  - Garth N. Wells, Delft University
  - Dupont, Johnson, Larson, Scott
- Originated in 2003
- Library or language depending on interface
- Automatic assembly
- Licensed under the (L)GPL
- `http://www.fenics.org/`

FEniCS components

- DOLFIN
  - The C++/Python interface of FEniCS
  - Developed by Hoffman/Jansson/Logg/Wells
- FFC
  - The FEniCS Form Compiler
  - Developed by Anders Logg
- FIAT
  - The FInite element Automatic Tabulator
  - Developed by Robert C. Kirby
- Ko
  - A mechanical simulator
  - Developed by Johan Jansson
- Puffin
  - A light-weight educational implementation for Octave/MATLAB
  - Developed by Hoffman/Logg

FEniCS and the Automation of FEM

The *automation of the finite element method* realizes the *automation of discretization* and is thus a key step towards the Automation of CMM:

FIAT	automates the tabulation of basis functions
FFC	automates the computation of the element tensor
DOLFIN	automates the assembly of the discrete system

Poisson's equation with FEniCS

```
element = FiniteElement(...)

v = BasisFunction(element)
U = BasisFunction(element)
f = Function(element)

a = dot(grad(v), grad(U))*dx
L = v*f*dx
```

The action of Poisson with FEniCS

```
element = FiniteElement(...)

v = BasisFunction(element)
U = Function(element)

a = dot(grad(v), grad(U))*dx
```

Upcoming lectures

0. Automating the Finite Element Method
1. Survey of Current Finite Element Software
2. The Finite Element Method
3. Automating Basis Functions and Assembly
4. Automating and Optimizing the Computation of the Element Tensor
5. FEniCS and the Automation of CMM
6. FEniCS Demo Session