PDE Project Course

4. DOLFIN, Systems of PDEs

Johan Jansson

johanjan@math.chalmers.se

Department of Computational Mathematics

Lecture plan

- Goal of DOLFIN
- Solving a PDE with DOLFIN
- Systems of PDEs

Goal of DOLFIN

Goal of DOLFIN

- Automatic solution of PDE (variational formulation)
- Why?
  - Efficiency
    - Execution time
    - Development time
  - Safety - reduce chance of human error (bugs)

Goal of DOLFIN

- Solving PDEs:
  - Modeling - problem-specific
  - Method - standard methods (FEM, FD)
  - General solver - still open problem
- How to construct an efficient general solver for PDEs? (instead of making a special solver for every problem)

Goal of DOLFIN

- How to construct an efficient general solver for PDEs?
- DOLFIN, Puffin try to answer question with:

$$(A_h)_{ij} = a(\varphi_j, \hat{\varphi}_i).$$

- DOLFIN: algebra of basis functions to compute general form
- Likely answer: form compilation (precompute basis function expression and integrals)

Solving a PDE with DOLFIN

Solving Poisson's equation

```
int main()
{
  Mesh mesh("mesh.xml.gz");
  Problem poisson("poisson", mesh);

  poisson.set("source", f);
  poisson.set("boundary condition", mybc);

  poisson.solve();

  return 0;
}
```

Implementing a solver

```
void PoissonSolver::solve()
{
  Galerkin     fem;
  Matrix       A;
  Vector       x, b;
  Function     u(mesh, x);
  Function     f(mesh, "source");
  Poisson      poisson(f);
  KrylovSolver solver;
  File         file("poisson.m");

  fem.assemble(poisson, mesh, A, b);
  solver.solve(A, x, b);

  u.rename("u", "temperature");
  file << u;
}
```

Automatic assembling

```
class Poisson : public PDE {
  ...
  real lhs(const ShapeFunction& u, const ShapeFunction& v)
  {
    return (grad(u),grad(v)) * dK;
  }

  real rhs(const ShapeFunction& v)
  {
    return f*v * dK;
  }
  ...
};
```

Automatic assembling

```
class ConvDiff : public PDE {
  ...
  real lhs(const ShapeFunction& u, const ShapeFunction& v)
  {
    return (u*v + k*((b,grad(u))*v + a*(grad(u),grad(v))))*dK;
  }

  real rhs(const ShapeFunction& v)
  {
    return (up*v + k*f*v) * dK;
  }
  ...
};
```

Abstract assembly algorithm

$$
\begin{aligned}
(A_h)_{ij} &= a(\varphi_j, \hat{\varphi}_i) = \int_\Omega A(\varphi_j)\hat{\varphi}_i \, dx \\
&= \sum_{K \in \mathcal{T}} \int_K A(\varphi_j)\hat{\varphi}_i \, dx = \sum_{K \in \mathcal{T}} a(\varphi_j, \hat{\varphi}_i)_K .
\end{aligned}
$$

Iterate over all elements K and for each element K compute the contributions to all $(A_h)_{ij}$, for which φ_j and $\hat{\varphi}_i$ are supported within K.

Assembly in DOLFIN

```
void FEM::assembleInterior(PDE& pde, Mesh& mesh, Matrix& A,
   FiniteElement::Vector& element, Map& map,
   Quadrature& interior_quadrature,
   Quadrature& boundary_quadrature)
{
  // Iterate over all cells in the mesh
  for (CellIterator cell(mesh); !cell.end(); ++cell)
  {
    // Update map
    map.update(*cell);

    // Update element
    for (unsigned int i = 0; i < element.size(); ++i)
      element(i)->update(map);

    // Update equation
    pde.updateLHS(element, map, interior_quadrature, boundary_quadrature);

    // Iterate over test and trial functions
    for (FiniteElement::Vector::TestFunctionIterator v(element); !v.end(); ++v)
      for (FiniteElement::Vector::TrialFunctionIterator u(element); !u.end(); ++u)
        A(v.dof(*cell), u.dof(*cell)) += pde.lhs(*u, *v);
  }
}
```

Demonstration

- ...

Systems of PDEs

Systems of PDEs

A, u and f have n components:

$$\begin{aligned} A_1(u_1) &= f_1 \\ A_2(u_2) &= f_2 \\ &\dots \\ A_n(u_n) &= f_n \end{aligned}$$

Variational formulation

- V^n space of functions with n components.
- V_h^n finite dimensional space of functions with n components.
- Assume $n = 2$ for compactness of notation.
- $u = (u_1, u_2), v = (v_1, v_2), f = (f_1, f_2)$

$$\begin{aligned} A(u) &= f \Rightarrow \\ \int_\Omega A(u)v &= \int_\Omega fv \Rightarrow \end{aligned}$$

Variational formulation

$$
\begin{aligned}
\int_\Omega (A_1(u_1), A_2(u_2))(v_1, v_2) &= \int_\Omega (f_1, f_2)(v_1, v_2) \Rightarrow \\
\int_\Omega A_1(u_1)v_1 + A_2(u_2)v_2 &= \int_\Omega f_1 v_1 + f_2 v_2 \Rightarrow \\
a(u, v) &= l(v)
\end{aligned}
$$

Finite element basis

- Every component of V_h^n is a scalar V_h.
- What is a basis for V_h^n?
- Standard basis for a vector space ($n = 2$):

$$
\begin{aligned}
\hat{\phi}_1 &= (\phi_1, 0) \\
\hat{\phi}_2 &= (0, \phi_1) \\
\hat{\phi}_3 &= (\phi_2, 0) \\
\hat{\phi}_4 &= (0, \phi_2) \\
&\dots \\
\hat{\phi}_{2N-1} &= (\phi_N, 0) \\
\hat{\phi}_{2N} &= (0, \phi_N)
\end{aligned}
$$

Ansatz

- The ansatz is just as before:

$$U(x) = \sum_{j=1}^{N} \xi_j \varphi_j(x).$$

- Concretely:
  - ξ_j is the dof (value of U) for component 1 of node j
  - ξ_{j+1} is the dof for component 2 of node $j+1$

Example: linear elasticity

The equations of linear elasticity:

$$\begin{aligned} \dot{u} - v &= 0 \quad \text{in } \Omega^0, \\ \dot{v} - \nabla \cdot \sigma &= f \quad \text{in } \Omega^0, \\ \sigma &= E\epsilon(u) = E(\nabla u^\top + \nabla u) \\ v(0,\cdot) &= v^0, \quad u(0,\cdot) = u^0 \quad \text{in } \Omega^0. \end{aligned}$$

Variational formulation

- $\epsilon(u) = \nabla u^\top + \nabla u$
- $\sigma = E\epsilon(u)$, E linear transformation (material properties)
- Turns out that:

$$a(u, v) = \int_\Omega \sigma(u)\epsilon(v)$$

$$l(v) = \int_\Omega fv$$