Nečas Center for Mathematical Modeling

Numerical solution of continuum physics problems with FEniCS or how to use FEM and not vary (too much) about programming ...

Jaroslav Hron

Mathematical Institute, Charles University in Prague

CHARLES UNIVERSITY PRAGUE

faculty of mathematics and physics

lecture 1	introduction to FEniCS/python, FEM and how to solve laplace equation
lecture 2	convection-diffusion equation, SUPG and IP stabilization
lecture 3	Stokes, incompressible Navier-Stokes equations, time discretization
lecture 4	linear and non-linear elasticity
lecture 5	ALE-method, level-set method

How to log in....

Python - https://www.python.org/

- interpreted, dynamic-typed language, object oriented, extensible
- everything is an object, variables are just object "names"

```
>>> a=42
>>> id(a); dir(a); print(a)
>>> help(a)
>>> b=a ; id(a) ; id(b)
>>> b+=1 ; id(b)
```

```
>>> import math
>>> math.sqrt(4.0)
>>> from math import cos
>>> cos(0.0)
>>> from math import *
>>> sin(pi)
```

```
>>> # Lists - collection of objects
>>> a = ['apple', 'orange', 3, 11]
>>> print(a[0:2])
['apple', 'orange']
>>> print(a[3])
11
```

```
>>> # Dictionary - indexed by any object
>>> a = {'id1': 0.2, 'id2': 0.5, 'id3': 0.05}
>>> print(a['id3'])
0.05
```

Python - https://www.python.org/

- interpreted, dynamic-typed language, object oriented, extensible
- everything is an object, variables are just object "names"
- scripts in files, run by: `python script.py`

```
for␣i␣in␣range(3):
␣␣␣␣print(i)
print('done')

for␣i␣in␣['apple',␣'orange',␣3,␣11]:
␣␣if␣i>6␣:
␣␣␣␣print(i)
print('done')
```

```
def heaviside(x):
  if x>0.0:
    y = 1.0
  elif x < 0.0:
    y = -1.0
  else: y = 0.0
  return y

print(heaviside(1.0))
```

- usefull libraries:
  + NumPy - www.numpy.org
  + SciPy - www.scipy.org
  + SymPy - www.sympy.org
  + matplotlib - matplotlib.org

FEniCS - http://fenicsproject.org

- started in 2003, collaboration between University of Chicago and Chalmers University of Technology
- 2011 - version 1.0, tutorial book published (Jan 2012) with main contribution by 5 institutions (Simula Research Laboratory, University of Cambridge, University of Chicago, Texas Tech University, KTH Royal Institute of Technology)
- current 2014 - version 1.4
- open source license (GNU LGPL v3), open source developement on https://bitbucket.org/fenics-project

info:

- A. Logg, K.-A. Mardal and G.N. Wells, editors. *FENICS: Automated Solution of Differential Equations by the Finite Element Method*, volume 84 of Lecture Notes in Computational Science and Engineering. Springer, 2012.
- free electronic version of the book available
- official documentation http://fenicsproject.org/documentation

FEniCS

- core components:

  Instant Python module that allows for instant inlining of C++ code in Python (Just in Time compilation)

  Dolfin C++/Python interface of FEniCS, providing a consistent Problem Solving Environment

  FFC FEniCS Form Compiler - compiler for multilinear forms by generating code (C++)

  FIAT FInite element Automatic Tabulator (curently Lagrange, mixed FE)

  UFC Unified Form-assembly Code is a unified framework for finite element assembly

  UFL Unified Form Language is specific language for declaration of finite element discretizations of variational forms

- additional libraries: FErari, UFLACS, Viper, ...
- external libraries: PETSc, UMFPACK, Trilinos, CGAL, VTK, ...

Poisson's equation

- classical form: find $u \in \mathcal{C}^2(\bar{\Omega})$ such that

$$-\operatorname{div}(\nabla u) = f \qquad \text{in } \Omega$$
$$u = 0 \qquad \text{on } \partial\Omega$$

- weak form: find $u \in W_0^{1,2}(\Omega)$ such that

$$\int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}x = \int_\Omega fv \,\mathrm{d}x \qquad \text{for all } v \in W_0^{1,2}(\Omega)$$

- general weak form: $u \in V(\Omega)$ such that

$$a(u, v) = L(v) \text{ for all } v \in V'(\Omega)$$

where $a : V \times V' \to R$ is a bilinear form and $L : V' \to R$ is a linear form

Discretization - FEM

- discretization - way to transform a PDE into a discrete finite dimensional system $\Rightarrow \mathbf{Ax} = \mathbf{b}$

  Finite Difference Method - based on the classical strong form, approximates the equation

  Finite Volume Method - based on integral form of conservation laws, uses piece-wise constant approximations

  Finite Element Method - based on the weak form, solves the equation as is, dicretize the space of solutions, uses piece-wise polynomial approximations, very flexible, well analyzed, in generalizations like Discontinuous Galerkin can include FVM

Poisson's equation - FEM discretization

- weak form: $u \in V(\Omega)$ such that

$$a(u,v) = L(v) \text{ for all } v \in V'(\Omega)$$

where $a : V \times V' \to R$ is a bilinear form and $L : V' \to R$ is a linear form

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}x \qquad\qquad L(v) = \int_\Omega fv \, \mathrm{d}x$$

- Let $V_h \subset V(\Omega)$ and $V'_h \subset V'(\Omega)$ then we look for $u_h \ inV_h$ such that

$$a(u_h, v) = L(v) \text{ for all } v \in V'_h$$

- assume that $\dim(V_h) = \dim(V'_h) = N$ and we have a basis $V_h = \mathrm{span}\{\phi_i\}_{i=1}^N$,$V'_h = \mathrm{span}\{\phi'_i\}_{i=1}^N$, then $u_h = \sum_{i=1}^N U_i \phi_i$

$$a(\sum_{i=1}^N U_i \phi_i, \phi'_j) = L(\phi'_j) \text{ for } j = 1..N \qquad\qquad \Rightarrow \mathbf{A}\mathbf{x} = \mathbf{b}$$

such that $\mathbf{A}_{i,j} = a(\phi_i, \phi'_j)$, $\mathbf{x}_i = U_i$ and $\mathbf{b}_i = L(\phi'_j)$

Finite elements

P. Ciarlet, The Finite Element Method for Elliptic Problems, 1978

- Definition of the finite element: The tripple (T, P, Ψ) where:

  T is bounded domain in R^d with piece-wise smooth boundary (simplex)
  $P(T)$ is finite-dimensional function space on T of dimension n (polynomials)
  Ψ is set of functionals, basis of V', $\{\psi_i\}_{i=1}^n$ (point evaluation)

- Special basis of $P(T) = \mathrm{span}\{\psi_i\}_{i=1}^n$ such that $\psi_i(\phi_j) = \delta_{i,j}$
- Computational domain Ω covered by elements domains T_k - usually defined as parametric mapping of a "reference" element.
- Continuity enforced by selection of the set Ψ.
- Result is the space

$$V_h(\Omega) = \{v \in C^k(\Omega), v/T \in P(T)\}$$

  such that $V_h \to V$ as $h \to 0$.

- Variants of elements
  - isoparametric vs. parametric vs. nonparametric
  - conforming i.e. $V_h \subset V$ v.s. non-conforming $V_h \not\subset V$
  - Lagrange (C^0), Hermite (C^1)

Periodic Table of the Finite Elements: `http://femtable.org/`

Poisson's problem in FEniCS

$$-\operatorname{div}(\nabla u) = f \quad \text{in } \Omega = [0,1]^2, \qquad u = 0 \quad \text{on } \partial\Omega$$

- $\Omega = [0,1]^2$
- Lagrange FEM:

  $$V_h = \{v \in C^0(\Omega), v/T \in P_1(T)\}$$

- Dirichlet boundary condition
  $v \in V_h \rightarrow v = 0$ on $\partial\Omega$
- $f(x,y) = 1.0$
- $a(u,v) = \int_\Omega \nabla u \cdot \nabla v \, dx$
- $L(v) = \int_\Omega f v \, dx$
- Find $u \in V_h$ such that
  $a(u,v) = L(v)$ for all $v \in V_h$.

```
from dolfin import *

mesh = UnitSquare(10, 10)
V = FunctionSpace(mesh, "Lagrange", 1)

u0 = Constant(0.0)
bc = DirichletBC(V, u0, DomainBoundary())

u = TrialFunction(V)
v = TestFunction(V)
f = Expression("1.0")

a = inner(grad(u), grad(v))*dx
L = f*v*dx

u = Function(V)
solve(a == L, u, bc)
```

Poisson's problem in FEniCS

- plot the mesh, solution

```
plot(mesh, interactive=True)
plot(u, interactive=True)
file=File("poisson.pvd")
file << u
```

- fenecs parameters:

```
info(parameters,True)

prm = parameters['krylov_solver'] # short form
prm['absolute_tolerance'] = 1E-10
prm['relative_tolerance'] = 1E-6
prm['maximum_iterations'] = 1000
```

- compute error with respect to exact solution

```
uex=Expression("....",degree=2)
e1=sqrt(assemble(pow(u-uex,2)*dx))
e=errornorm(uex, u, norm_type='l2', degree_rise=1, mesh=mesh)
print "error:", e , e1
```

Task no.1 - ex1.py

$$
\begin{aligned}
-\operatorname{div}(\nabla u) &= 2\pi^2 \sin(\pi x)\sin(\pi y) && \text{in } \Omega = [0,1]^2 \\
u &= 0 && \text{on } \partial\Omega
\end{aligned}
$$

- $u_{\text{exact}} = \sin(\pi x)\sin(\pi y)$
- Compute solution to precision $||u - u_{\text{exact}}||_2 < 10^{-6}$ in shortest time.
- Plot error vs. h, estimate convergence rate in different norms.