

# FEniCS Course

## Lecture 1: Introduction to FEniCS

---

### *Contributors*

Anders Logg

André Massing

Simon Funke



What is FEniCS?

# FEniCS is an automated programming environment for differential equations

- C++/Python library
- Initiated 2003 in Chicago
- 1000–2000 monthly downloads
- Part of Debian and Ubuntu
- Licensed under the GNU LGPL



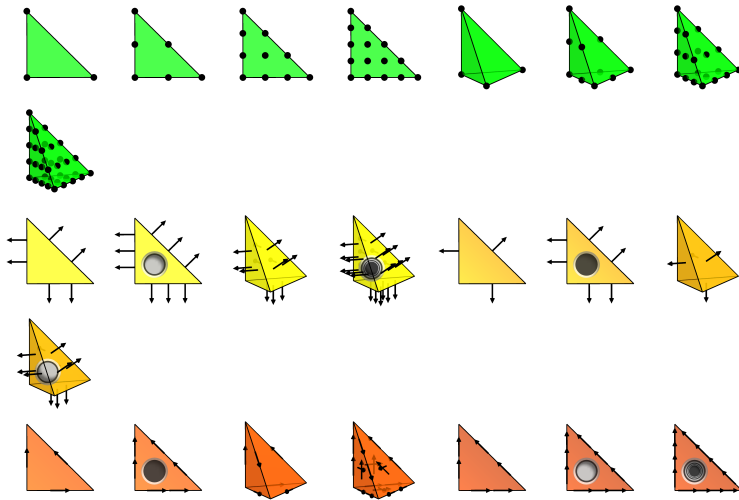
<http://fenicsproject.org/>

## Collaborators

*Simula Research Laboratory, University of Cambridge,  
University of Chicago, Texas Tech University, KTH Royal  
Institute of Technology, Chalmers University of Technology,  
Imperial College London, University of Oxford, Charles  
University in Prague, ...*

# FEniCS is automated FEM

- Automated generation of basis functions
- Automated evaluation of variational forms
- Automated finite element assembly
- Automated adaptive error control



# FEniCS code can be readable, scale with mathematical complexity, and provide high-performance

## Stokes with nonlinear viscosity

Given temperature  $T$ , find velocity  $u$  and pressure  $p$  such that

$$\begin{aligned}-\operatorname{div}(2\nu(u, T)\varepsilon(u) + pI) &= \operatorname{Ra} T g \\ \operatorname{div} u &= 0\end{aligned}$$

in  $\Omega$  with (for instance)

$$\nu(u, T) = e^{-\alpha T} (u \cdot u).$$

## Finite element formulation

Given temperature  $T$ , find

$(u, p) \in W = V \times Q$  such that

$$\begin{aligned}\int_{\Omega} 2\nu(u, T)\varepsilon(u) \cdot \varepsilon(v) + \operatorname{div}(v) p \\ + \operatorname{div}(u) q - \operatorname{Ra} T g \cdot v \, dx = 0\end{aligned}$$

for all  $(v, q) \in W$ .

```
from fenics import *

# Define viscosity
def nu(u, T):
    return exp(-10.0*T)*dot(u, u)

# Define element spaces
mesh = Mesh(...)
V = VectorFunctionSpace(mesh, "CG", 2)
Q = FunctionSpace(mesh, "CG", 1)
W = V * Q

# Define functions
T = Expression("...")
w = Function(W)
(u, p) = split(w)
(v, q) = TestFunctions(W)

# Define equation
F = (2*nu(u, T)*inner(eps(u), eps(v))
     + div(v)*p + div(u)*q
     + Ra*T*v[1])*dx
bcs = ...

# Solve F = 0 w.r.t w
solve(F == 0, w, bcs)
```

# FEniCS provides a wide range of (mixed) finite element spaces

```
from fenics import *

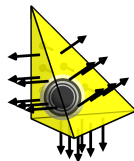
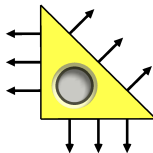
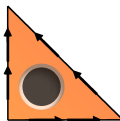
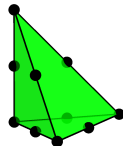
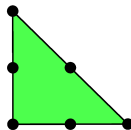
# Define viscosity
def nu(u, T):
    return exp(-10.0*T)*dot(u, u)

# Define element spaces
mesh = Mesh(...)
V = VectorFunctionSpace(mesh, "CG", 2)
Q = FunctionSpace(mesh, "CG", 1)
W = V * Q

# Define functions
T = Expression("...")
w = Function(W)
(u, p) = split(w)
(v, q) = TestFunctions(W)

# Define equation
F = (2*nu(u, T)*inner(eps(u), eps(v))
     + div(v)*p + div(u)*q
     + Ra*T*v[1])*dx
bcs = ...

# Solve F = 0 w.r.t w
solve(F == 0, w, bcs)
```



# FEniCS provides an expressive form language close to mathematical syntax

```
from fenics import *

# Define viscosity
def nu(u, T):
    return exp(-10.0*T)*dot(u, u)

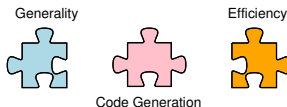
# Define element spaces
mesh = Mesh(...)
V = VectorFunctionSpace(mesh, "CG", 2)
Q = FunctionSpace(mesh, "CG", 1)
W = V * Q

# Define functions
T = Expression("...")
w = Function(W)
(u, p) = split(w)
(v, q) = TestFunctions(W)

# Define equation
F = (2*nu(u, T)*inner(eps(u), eps(v))
     + div(v)*p + div(u)*q
     + Ra*T*v[1])*dx
bcs = ...

# Solve F = 0 w.r.t w
solve(F == 0, w, bcs)
```

Language for variational forms



# FEniCS provides automated form assembly over finite element meshes and numerical linear algebra

```
from fenics import *

# Define viscosity
def nu(u, T):
    return exp(-10.0*T)*dot(u, u)

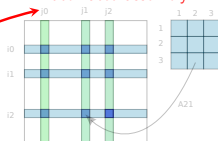
# Define element spaces
mesh = Mesh(...)
V = VectorFunctionSpace(mesh, "CG", 2)
Q = FunctionSpace(mesh, "CG", 1)
W = V * Q

# Define functions
T = Expression("...")
w = Function(W)
(u, p) = split(w)
(v, q) = TestFunctions(W)

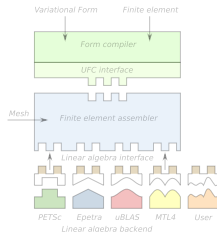
# Define equation
F = (2*nu(u, T)*inner(eps(u), eps(v))
     + div(v)*p + div(u)*q
     + Ra*T*v[1])*dx
bcs = ...

# Solve  $F = 0$  w.r.t  $w$ 
solve(F == 0, w, bcs)
```

Automated assembly



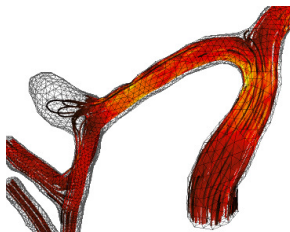
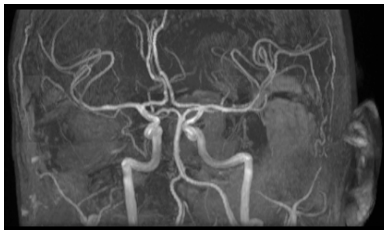
High performance linear algebra





What has FEniCS been used for?

# Computational hemodynamics



- Low wall shear stress may trigger aneurysm growth
- Solve the incompressible Navier–Stokes equations on patient-specific geometries

$$\begin{aligned}\dot{u} + u \cdot \nabla u - \nabla \cdot \sigma(u, p) &= f \\ \nabla \cdot u &= 0\end{aligned}$$

# Computational hemodynamics (contd.)



```
# Define Cauchy stress tensor
def sigma(v,w):
    return 2.0*mu*0.5*(grad(v) + grad(v).T) -
        w*Identity(v.cell().d)

# Define symmetric gradient
def epsilon(v):
    return 0.5*(grad(v) + grad(v).T)

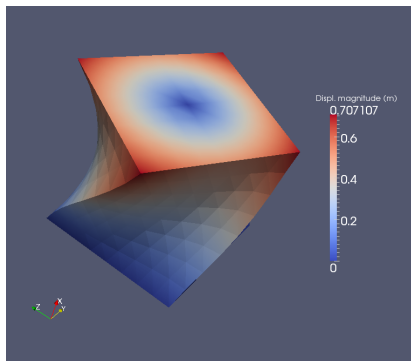
# Tentative velocity step (sigma formulation)
U = 0.5*(u0 + u)
F1 = rho*(1/k)*inner(v, u - u0)*dx + rho*inner(v,
    grad(u0)*(u0 - w))*dx \
    + inner(epsilon(v), sigma(U, p0))*dx \
    + inner(v, p0*n)*ds - mu*inner(grad(U).T*n,
        v)*ds \
    - inner(v, f)*dx
a1 = lhs(F1)
L1 = rhs(F1)

# Pressure correction
a2 = inner(grad(q), k*grad(p))*dx
L2 = inner(grad(q), k*grad(p0))*dx - q*div(u1)*dx

# Velocity correction
a3 = inner(v, u)*dx
L3 = inner(v, u1)*dx + inner(v, k*grad(p0 -
    p1))*dx
```

- The Navier–Stokes solver is implemented in Python/FEniCS
- FEniCS allows solvers to be implemented in a minimal amount of code

# Hyperelasticity



```

from fenics import *

mesh = UnitCubeMesh(24, 16, 16)
V = VectorFunctionSpace(mesh, "Lagrange", 1)

left = CompiledSubDomain("(std::abs(x[0])
< DOLFIN_EPS) && on_boundary")
right = CompiledSubDomain("(std::abs(x[0] - 1.0)
< DOLFIN_EPS) && on_boundary")

c = Expression(("0.0", "0.0", "0.0"), degree=0)
r = Expression(("0.0",
"0.5*(y0+(x[1]-y0)*cos(t)-(x[2]-z0)*sin(t)-x[1])",
"0.5*(z0+(x[1]-y0)*sin(t)+(x[2]-z0)*cos(t)-x[2])"),
y0=0.5, z0=0.5, t=pi/3, degree=3)
bcl = DirichletBC(V, c, left)
bcr = DirichletBC(V, r, right)
bcs = [bcl, bcr]
v = TestFunction(V)
u = Function(V)
B = Constant((0.0, -0.5, 0.0))
T = Constant((0.1, 0.0, 0.0))
I = Identity(V.cell().d)
F = I + grad(u)
Ic = tr(F.T*F)
J = det(F)
E, nu = 10.0, 0.3
mu, lambda = Constant(E/(2*(1 + nu))),
Constant(E*nu/((1 + nu)*(1 - 2*nu)))
psi = (mu/2)*(Ic - 3) - mu*ln(J) +
(lambda/2)*(ln(J))**2
Pi = psi*dx - dot(B, u)*dx - dot(T, u)*ds
F = derivative(Pi, u, v)

solve(F == 0, u, bcs)
plot(u, interactive=True, mode="displacement")

```

How to use FEniCS?

# Hello World in FEniCS: problem formulation

## Poisson's equation

$$\begin{aligned} -\Delta u &= f && \text{in } \Omega \\ u &= 0 && \text{on } \partial\Omega \end{aligned}$$

## Finite element formulation

Find  $u \in V$  such that

$$\underbrace{\int_{\Omega} \nabla u \cdot \nabla v \, dx}_{a(u,v)} = \underbrace{\int_{\Omega} f v \, dx}_{L(v)} \quad \forall v \in V$$

# Hello World in FEniCS: implementation

```
from fenics import *

mesh = UnitSquareMesh(32, 32)

V = FunctionSpace(mesh, "Lagrange", 1)
u = TrialFunction(V)
v = TestFunction(V)
f = Expression("x[0]*x[1]", degree=2)

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

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

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

# Basic API

- Mesh, Vertex, Edge, Face, Facet, Cell
  - FiniteElement, FunctionSpace
  - TrialFunction, TestFunction, Function
  - grad(), curl(), div(), ...
  - Matrix, Vector, KrylovSolver, LUSolver
  - assemble(), solve(), plot()
- 
- Python interface generated semi-automatically by SWIG
  - C++ and Python interfaces almost identical



Sounds great, but how do I find my way through the jungle?



# Three survival advices



Use the right Python  
tools



Explore the  
documentation



Ask, report and  
request

Use the right Python tools!

# Python tools

## Doc tools

- Standard terminal:  
    `> pydoc dolfin`  
    `> pydoc dolfin.Mesh`
- Python console  
    `>>> help(dolfin)`  
    `>>> help(dolfin.Mesh)`

## Sophisticated Python environments

**IDLE** the official (but rather limited) Python IDE

**IPython** <http://ipython.org/>  
provides a Python shell and notebook including syntax highlighting, tab-completion, object inspection, debug assisting, history ...

**Eclipse** plugin <http://pydev.org/>  
includes syntax highlighting, code completion, unit-testing, refactoring, debugger ...

# IPython/Jupyter Notebook

IP[y]: Notebook

Actions

HowOpenDownloadipybPrint

Cell

ActionsDelete

FormatCodeMarkdown

OutputToggleClearAll

InsertAboveBelow

MoveUpDown

RunSelectedAll

Autosave: ☒

Kernel

ActionsInterruptRestart

Kill kernel upon exit: ☐

Help

LinksPythonIPythonNumPySciPyMPLSymPy

Shift-Enter: run selected cell  
Ctrl-Enter: run selected cell in-place  
Ctrl-m h: show keyboard shortcuts

Configuration

Tooltip on tab: ☒  
Smart completer: ☒  
Time before tooltip: 1200 milliseconds

solving-poissonSaveQuickHelp

Let's solve numerically the following variational problem: Find  $u \in H_0^1(\Omega)$  such that  $a(u, v) = L(v) \forall v \in H_0^1(\Omega)$  where  $a(u, v) = \int_{\Omega} \nabla u \nabla v \, dx$  and  $L(v) := \int_{\Omega} f v \, dx$ . To do that in FEniCS we start by defining a mesh:

In [26]: 

```
from dolfin import *  
m = UnitSquare(10, 10)  
print m  
  
<Mesh of topological dimension 2 (triangles) with 121 vertices and 200 cells, ordered>
```

Now we need some function space. Let's take P1 elements

In [27]: 

```
V = FunctionSpace(m, "CG", 1)
```

It's time to define some test and trial functions.

In [28]: 

```
u = TrialFunction(V)  
v = TestFunction(V)
```

And finally we can define the variational problem:

In [29]: 

```
a = inner(grad(u), grad(v)) * dx  
f = Constant(1)  
L = f * v * dx  
def boundary(x, on_boundary):  
    return on_boundary  
u = Function(V)  
zero = Constant(0)  
bc = DirichletBC(V, zero, boundary)  
solve(a==L, u, bc)
```

In [30]: 


```
plot(u)
```

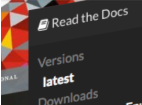


Out[30]: <viper.viper\_dolfin.Viper at 0x4b76890>

In [31]: 

```
interactive()
```

Explore the FEniCS documentation!


[FENICS'18](#)
[DOWNLOAD](#)
[DOCUMENTATION](#)
[COMMUNITY](#)
[GOVERNANCE](#)
[CITING](#)
[DONATE](#)
[ARCHIVES](#)

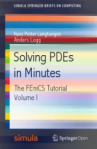
## Documentation

Our documentation includes several books, a collection of documented demo programs (notebooks), and a reference manual.

If you are new to FEniCS and want to quickly get started with solving PDEs in Python, the [FEniCS Tutorial](#) is the best starting point. For advanced users looking for details on how to build, program and develop with FEniCS, the [FEniCS Reference Manual](#) contains all you need to know about FEniCS.

## The FEniCS Tutorial

The book [Solving PDEs in Python - The FEniCS Tutorial Volume I](#) is the perfect guide for new users. The tutorial explains fundamental concepts of the finite element method, FEniCS programming, and demonstrates how to quickly solve a number of PDEs. The tutorial assumes no prior knowledge of the finite element method. The tutorial is an updated and expanded version of the popular first chapter of the [FEniCS Book](#).



<http://fenicsproject.org/documentation/>

The screenshot shows the DOLFIN 2016.2.0 documentation page for Python. The left sidebar contains a search bar and three links: 'Programmer's reference for DOLFIN (Python)', 'Collection of documented demos', and 'Quick Programmer's Reference (Python)'. The main content area has a breadcrumb 'Docs » Documentation for DOLFIN-2016.2.0 (Python)' and a 'View page source' link. The title is 'Documentation for DOLFIN-2016.2.0 (Python)'. Below it are three sections: 'The Demos' with a link to 'The demos (Collection of documented demo programs)', 'The Programmer's Reference' with a link to 'Programmer's Reference Index (Classes, functions, terms)', and 'DOLFIN modules' with a link to 'DOLFIN Module Index (Modules)'. A 'Next' button with a right arrow is at the bottom right.

DOLFIN  
2016.2.0

Search docs

Programmer's reference for DOLFIN (Python)

Collection of documented demos

Quick Programmer's Reference (Python)

Docs » Documentation for DOLFIN-2016.2.0 (Python) [View page source](#)

## Documentation for DOLFIN-2016.2.0 (Python)

### The Demos

[The demos](#) (Collection of documented demo programs)

### The Programmer's Reference

[Programmer's Reference Index](#) (Classes, functions, terms)

### DOLFIN modules

[DOLFIN Module Index](#) (Modules)

[Next](#) ➔

<https://fenicsproject.org/olddocs/dolfin/2016.2.0/python/>



Hans Petter Langtangen  
Anders Logg

# Solving PDEs in Minutes The FEniCS Tutorial I

**simula**



Springer Open

Ask questions, report bugs and request new features!

# Development community is organized via bitbucket.org

The screenshot shows the Bitbucket web interface for the repository `fenics-project / DOLFIN`. The browser's address bar shows the URL `https://bitbucket.org/fenics-project/dolfin`. The repository is a C++ project. The left sidebar contains navigation links: Overview (selected), Source, Commits, Branches, Pull requests (5), Issues (99), Wiki, Downloads (1), and Settings. The main content area is titled "Overview" and includes a table with repository statistics:

Last updated	6 minutes ago	99+ Branches	3 Tags
Language	C++	48 Forks	73 Watchers
Access level	Admin (revoke)		

Below the table is a section for "DOLFIN" with a description: "DOLFIN is the C++/Python interface of FEniCS, providing a consistent PSE (Problem Solving Environment) for ordinary and partial differential equations." It also includes an "Installation" section with a code block:

```
mkdir build
cd build
cmake ..
make install
```

For detailed instructions, see the file INSTALL. Below this is a "License" section stating that DOLFIN is free software under the GNU Lesser General Public License.


On the right side, there is a "Recent activity" section showing three commits:

- 1 commit: Pushed to fenics-project/dolfin. Merge branch 'garth/replace-boo...'. 8bc2b98. Garth Wells - 7 minutes ago.
- 1 commit: Pushed to fenics-project/dolfin. Merge branch 'garth/replace-lexi...'. 8db4a67. Garth Wells - 10 minutes ago.
- 1 commit: Pushed to fenics-project/dolfin. Replace Boost lexical\_cast with ... ab59d44. Garth Wells - 10 minutes ago.

At the bottom right, another commit is visible: 1 commit: Pushed to fenics-project/dolfin. Merge branch 'garth/replace-boo...'. be6345d. Garth Wells - 35 minutes ago.

<http://bitbucket.org/fenics-project/>

# Community help is available via QA forum

 FEniCS Project (detail)

Join Community

latest frequent trending

Show unanswered only

6 votes

article

463 views

Article: Do not post install or compile questions: email to fenics-support@googlegroups.com or use Slack

Community: FEniCS Project

updated 3 months ago by Ta S

9 votes

article

664 views

Article: Read before posting: How do I get by my question answered?

Community: FEniCS Project

updated 6 months ago by Garth Wells

0 votes

1 answer

17 views

Solving a set of 5 non linear PDEs simultaneously

Community: FEniCS Project

updated 33 minutes ago by Michal Habera

-1 votes

1 answer

23 views

Error with GenericLinearOperator and argument type

Community: FEniCS Project

updated 3 hours ago by Minas Mattheakis

0 votes

1 answer

17 views

Plasticity simulation tutorial

Community: FEniCS Project

updated 3 hours ago by Minas Mattheakis

-1 votes

1 answer

57 views

How do you restrict a DG function to only an internal boundary?

Community: FEniCS Project

updated 14 hours ago by Dan Sweeney

0 votes

1 answer

24 views

Point inside mesh

Community: FEniCS Project

updated 23 hours ago by Alexander G. Zimmerman

-1 votes

0 answers

68 views

Implementation of subdomains in the 2017.2.0 version changed (compared to 1.6.0) ?

Community: FEniCS Project

updated 1 day ago by J.H.59

f

t

+

e

Community experts

pf4d

1240 points

Adam Janecka

950 points

Jan Blechta

630 points

Hernán Mella

520 points

Chris Richardson

500 points

Community members

Recent votes

- How to sort an array according to the global degree of freedom indexing.
- Incorrect Boundary Reaction
- Error with GenericLinearOperator and argument type
- Export solution data and import geometries
- Implementation of subdomains in the 2017.2.0 version changed (compared to 1.6.0) ?

Recent replies

- C: Solving a set of 5 non linear PDEs simultaneously by Michal Habera
- I would say he meant "compiling" the

[allanswered.com/community/s/fenics-project](https://allanswered.com/community/s/fenics-project) (new)

[fenicsproject.org/qa](https://fenicsproject.org/qa) (old)

# Installation alternatives



☞ Docker images on Linux, Mac, Windows



☞ PPA with apt packages for Debian and Ubuntu



☞ Anaconda packages



☞ Build from source

<http://fenicsproject.org/download/>

# *The FEniCS challenge!*

- ❶ Install FEniCS on your laptop!

<http://fenicsproject.org/download/>

- ❷ Download and execute `demo_cahn-hilliard.py`, try to visualize the results with Paraview.
- ❸ Which elements are supported in `dolfin`  
Hint: Check documentation of *dolfin.FunctionSpace*?

See also <https://femtable.org>