FEniCS.jl, solving PDE’s using Julia

Yiannis Simillides

University College London

19 June 2018
Collaborators

- Dr Chris Rackauckas, University of California, Irvine
- Dr Timo Betcke, University College London

Funding Provided via GSoC, through NumFocus - The Julia Language
Introduction

1. Collaborators
2. Julia
   - Advantages
   - Disadvantages
   - Benchmarks
3. Finite Element Method
   - Introduction to FEM
4. FEniCS
   - Introduction
   - Variational Form
5. FEniCS.jl
   - Our work so far
   - Meshing Capabilities
   - UFL
   - Hybrid code
   - Future Work
Advantages

- Numerical computing programming language
<table>
<thead>
<tr>
<th>Collaborators</th>
<th>Julia</th>
<th>Finite Element Method</th>
<th>FEniCS</th>
<th>FEniCS.jl</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>★★★</td>
<td></td>
<td>★★★★★</td>
<td>★★★★★★</td>
</tr>
</tbody>
</table>

## Advantages

- Numerical computing programming language
- Open-Source
Advantages

- Numerical computing programming language
- Open-Source
- Interoperability with C / Python / Others
Advantages

- Numerical computing programming language
- Open-Source
- Interoperability with C / Python / Others
- JIT compiled
Disadvantages

- Numerical computing programming language
Disadvantages

- Numerical computing programming language
- Not fully stable (v1.0 released "soon")
Disadvantages

- Numerical computing programming language
- Not fully stable (v1.0 released "soon")
- Packages not fully developed compared to Python
Benchmarks

General Benchmarks

https://julialang.org/benchmarks/

Figure 1: Timing Comparison Programming Languages
Introduction to FEM

- Method of solving PDE's and optimization problems
Method of solving PDE’s and optimization problems
Can handle arbitrary boundary conditions compared to Finite Difference
Introduction to FEM

- Method of solving PDE’s and optimization problems
- Can handle arbitrary boundary conditions compared to Finite Difference
- Different variations, hp-fem, CUT-fem etc
Introduction to FEM

- Method of solving PDE’s and optimization problems
- Can handle arbitrary boundary conditions compared to Finite Difference
- Different variations, hp-fem, CUT-fem etc
- FEM software available, commercial and open source
Outline

How to Solve?

- Derive variational formulation (weak form)
Outline

How to Solve?

- Derive variational formulation (weak form)
- Discretize the solution space (mesh) using shape functions
Outline

How to Solve?

- Derive variational formulation (weak form)
- Discretize the solution space (mesh) using shape functions
- Apply Boundary Conditions
Outline

How to Solve?

- Derive variational formulation (weak form)
- Discretize the solution space (mesh) using shape functions
- Apply Boundary Conditions
- Solve system of equations (linear/non-linear)
FEniCS
Open-Source Computing Platform

- High-level, near one-to-one correspondence with model
Introduction

FEniCS
Open-Source Computing Platform

- High-level, near one-to-one correspondence with model
- Many "models" implemented
FEniCS

Open-Source Computing Platform

- High-level, near one-to-one correspondence with model
- Many "models" implemented
- Python and C++ interfaces
Introduction

FEniCS
Open-Source Computing Platform

- High-level, near one-to-one correspondence with model
- Many "models" implemented
- Python and C++ interfaces
- and now Julia interfaces
Poisson Equation

\[-\nabla^2 u = f, \text{ in } \Omega\]

\[u = 1 + x^2 + y^2, \text{ on } \partial \Omega\]

\[f = -6.\]  

(1)

Figure 2: Discretized Domain $\Omega$
Variational Form

\[-\nabla^2 u = f, \text{ in } \Omega\]  \hspace{1cm} (2)

\[-\int_\Omega (\nabla^2 u)v \, dx = \int_\Omega fv \, dx.\]  \hspace{1cm} (3)

\[
\int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega fv \, dx.\]  \hspace{1cm} (4)

Code in FEniCS

```python
u_D = Expression("1+x[0]*x[0]+2*x[1]*x[1]", degree=2)
bcl = DirichletBC(V, u_D, "on_boundary")
f = Constant(-6.0)
a = dot(grad(u), grad(v))*dx
L = f*v*dx
```
We can then proceed to solve the problem, either directly or iteratively.

**Figure 3:** Poisson Solution
Our work so far

Interfaced FEniCS with Julia

- Meshing Capabilities
Our work so far

Interfaced FEniCS with Julia

- Meshing Capabilities
- Semi-native plotting
Interfaced FEniCS with Julia

- Meshing Capabilities
- Semi-native plotting
- UFL Library (Variational form)
Our work so far

Interfaced FEniCS with Julia

- Meshing Capabilities
- Semi-native plotting
- UFL Library (Variational form)
- Solving and helper functions
## Our work so far

### Interfaced FEniCS with Julia

- Meshing Capabilities
- Semi-native plotting
- UFL Library (Variational form)
- Solving and helper functions
- Extend library to solve FEM problems in Julia
Implemented a wide variety of geometrical objects and meshes.

- Access values / discretizations
Meshing/Plotting Capabilities

Implemented a wide variety of geometrical objects and meshes.

- Access values / discretizations
- Find and label Boundary Values/Nodes directly
Implemented a wide variety of geometrical objects and meshes.

- Access values / discretizations
- Find and label Boundary Values/Nodes directly
- Custom Meshes available, import/export from vtk format
Implemented a wide variety of geometrical objects and meshes.

- Access values / discretizations
- Find and label Boundary Values/Nodes directly
- Custom Meshes available, import/export from vtk format
- Plotting available via PyPlot
UFL Library

Used to efficiently describe variational forms

- Describe problem, along with boundary conditions
UFL Library

Used to efficiently describe variational forms

- Describe problem, along with boundary conditions
- Perform necessarily mathematical calculations
UFL Library

Used to efficiently describe variational forms

- Describe problem, along with boundary conditions
- Perform necessarily mathematical calculations
- ”Helper” functions, allowing retrieval of stiffness matrices etc
UFL Library

Used to efficiently describe variational forms

- Describe problem, along with boundary conditions
- Perform necessarily mathematical calculations
- "Helper" functions, allowing retrieval of stiffness matrices etc
- Variety of solvers available, linear, non-linear etc
Use parts of the FEniCS library to solve FEM problems more efficiently

**Julia Code**

```julian
dbc(x,y) = 1+x^2+y^2
f_ext(x,y) = -6

function poisson_stiffness(N, dN, i, j)
    dN[i,1]*dN[j,1] + dN[i,2]*dN[j,2]
end

mesh = UnitSquareMesh(10,10)
u = feMesh(mesh,dbc)
l= solver(u, poisson_stiffness, f_ext)
```
Comparing FEniCS to our Hybrid code gives the following results

Figure 4: Timing Comparison
Future Work

- Fully interface library
Future Work

- Fully interface library
- Extend hybrid code, more shape functions
Future Work

- Fully interface library
- Extend hybrid code, more shape functions
- Benchmark non-linear and time-dependent problems