Gridap: introduction for FluidDynamics

About Gridap

Project started in 2020, offical Gridap repository

What:

- Written in Julia
- Provides a set of tools for solving PDEs
- FEM framework
- Almost 1:1 mathematical notation

Support for:

- MPI, GridapDistributed
- PETSc, GridapPETSc
- Pardiso, GridapPardiso
- Gmsh, GridapGmsh

Features:

- MultiField problems
- Time depended
- Discontinuous Galerkin
- Raviart Thomas elements

WorkFlow:

1. Mesh creation with Gmsh
2. Julia-MPI-Gridap
3. Open the results in Paraview

Solve the Poisson equation in 3D

Problem

$$\begin{aligned} -\Delta u = f \text{ in } \Omega, \\ u = g \text{ on } \Gamma_{\rm D}, \\ \nabla u \cdot n = h \text{ on } \Gamma_{\rm N}, \end{aligned}$$

We choose $f(x) = 1$, $g(x) = 2$, $h(x) = 3$

```
f(x) = 1.0
```

```
g(x) = 2.0;
```

```
h(x) = 3.0;
```

Find $u \in U_h$ such that $a(u,v) = b(v)$ for all $v \in V_h$

where $a(u,v) \doteq \int_{\Omega} \nabla v \cdot \nabla u \ {\rm d}\Omega, \quad b(v) \doteq \int_{\Omega} v\ f \ {\rm d}\Omega + \int_{\Gamma_{\rm N}} v\ h \ {\rm d}\Gamma_{\rm N}$

```
using Gridap
```

```
using GridapGmsh
```

```
msh_file = joinpath(@__DIR__, "models", "toy_model.msh")
```

```
model = GmshDiscreteModel(msh_file)
```

```
order = 1;
```

```
reffe = ReferenceFE(lagrangian,Float64,order);
```

```
V = TestFESpace(model,reffe,dirichlet_tags=["sides"]);
```

```
U = TrialFESpace(V,g);
```

```
neumanntags = ["circle", "triangle", "square"];
```

```
Γ = BoundaryTriangulation(model,tags=neumanntags);
```

Set up to perform the integrals in the weak form numerically. Define integration mesh, plus a Gauss-like quadrature in each of the cells in the triangulation

```
degree = 2;
```

```
dΓ = Measure(Γ,degree);
```

```
Ω = Triangulation(model);
```

```
dΩ = Measure(Ω,degree);
```

Weak form

$$a(u,v) \doteq \int_{\Omega} \nabla v \cdot \nabla u \, \mathrm{d}\Omega$$

```
a(u,v) = ∫(∇(u)⋅∇(v))dΩ ;
```

$$b(v) \doteq \int_{\Omega} v\, f \, \mathrm{d}\Omega + \int_{\Gamma_{\mathrm{N}}} v\, h \, \mathrm{d}\Gamma_{\mathrm{N}}$$

```
b(v) = ∫( v*f )*dΩ + ∫( v*h )*dΓ ;
```

```
op = AffineFEOperator(a,b,U,V) ;
```

Linear Solver

```
ls = LUSolver();
```

There are other solvers available:

- BackSlash(), like `\` in `MATLAB`
- NLSolver()
- PETSc

```
solver = LinearFESolver(ls) ;
```

```
uh = solve(solver,op);
```

```
writevtk(Ω, "Poisson", cellfields = ["uh" => uh]);
```