Firedrake

(in one hour)

"Firedrake is an automated system for the solution of partial differential equations using the finite element method (FEM)."

Firedrake abstracts away FEM implementation – your program starts to read a lot more like the math.

Example:

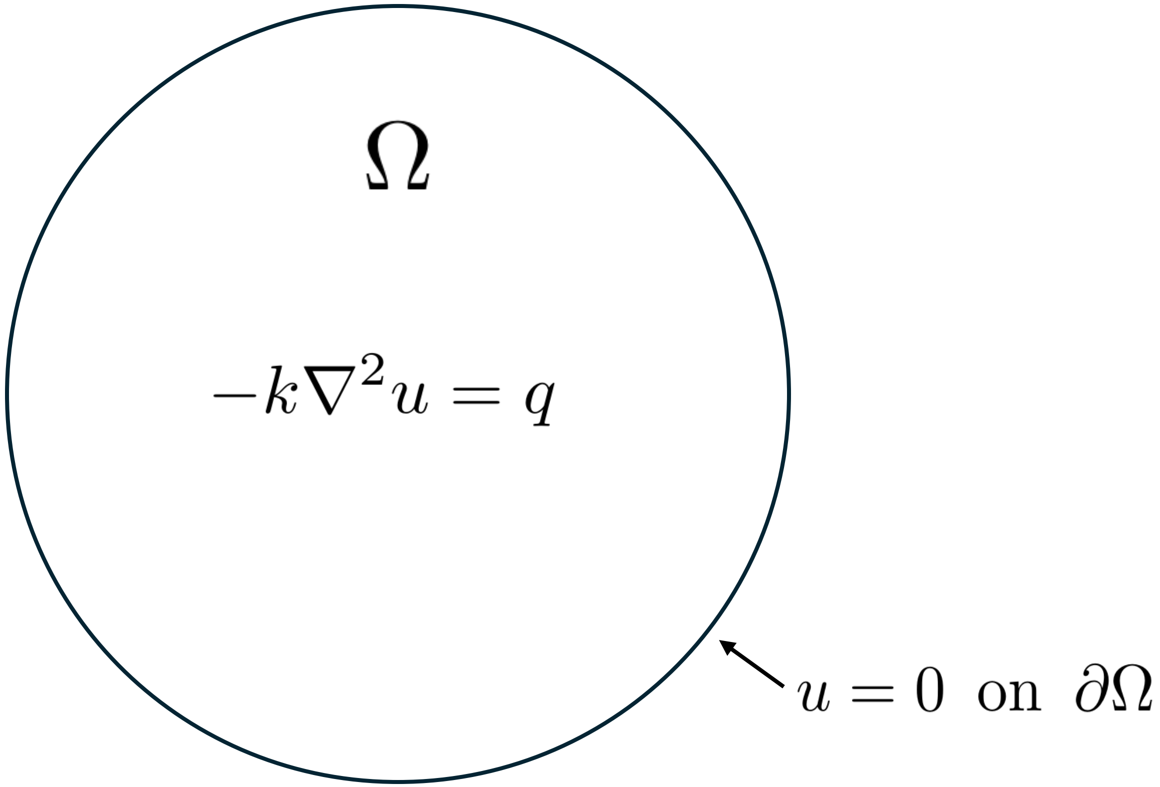
$$\int_\Omega (k(\nabla u \cdot \nabla v) - qv)\,\mathrm{d}x$$

```
(k * dot(grad(u), grad(v)) - q * v) * dx
```

Usual workflow

- Generate a weak form of the PDE(s) you want to solve
- Generate a mesh to solve on
- Pick basis function type(s)
- Pick solver options
- Write all of this up with Firedrake
- Solve?
- Debug
- Solve

Example 1 – Steady state heat equation (Poisson's equation)



k - Thermal diffusivity
u - Temperature
q - Rate of heat generation

First, need a weak form

$$-k\nabla^2 u = q \quad \leftarrow \text{strong form}$$

First, need a weak form

$$-k\nabla^2 u = q \quad \leftarrow \text{strong form}$$

$$(-k\nabla^2 u)v = qv \quad \text{multiply by test function } v$$

First, need a weak form

$$-k\nabla^2 u = q \quad \leftarrow \text{strong form}$$

$$(-k\nabla^2 u)v = qv \quad \text{multiply by test function } v$$

$$\int_\Omega (-k\nabla^2 u)v \, \mathrm{d}x = \int_\Omega qv \, \mathrm{d}x \quad \text{integrate over } \Omega$$

First, need a weak form

$$-k\nabla^2 u = q \quad \leftarrow \text{strong form}$$

$$(-k\nabla^2 u)v = qv \quad \text{multiply by test function } v$$

$$\int_\Omega (-k\nabla^2 u)v \,\mathrm{d}x = \int_\Omega qv \,\mathrm{d}x \quad \text{integrate over } \Omega$$

$$\int_\Omega \nabla \cdot (-k\nabla u)v \,\mathrm{d}x = \int_\Omega qv \,\mathrm{d}x \quad \text{definition of Laplacian}$$

First, need a weak form

$$-k\nabla^2 u = q \quad \leftarrow \text{strong form}$$

$$(-k\nabla^2 u)v = qv \quad \text{multiply by test function } v$$

$$\int_\Omega (-k\nabla^2 u)v \, \mathrm{d}x = \int_\Omega qv \, \mathrm{d}x \quad \text{integrate over } \Omega$$

$$\int_\Omega \nabla \cdot (-k\nabla u)v \, \mathrm{d}x = \int_\Omega qv \, \mathrm{d}x \quad \text{definition of Laplacian}$$

$$\int_\Omega \nabla \cdot (-kv\nabla u) \, \mathrm{d}x + \int_\Omega \nabla v \cdot k\nabla u \, \mathrm{d}x = \int_\Omega qv \, \mathrm{d}x \quad \text{apply product rule}$$

First, need a weak form

$$-k\nabla^2 u = q \quad \leftarrow \text{strong form}$$

$$(-k\nabla^2 u)v = qv \quad \text{multiply by test function } v$$

$$\int_\Omega (-k\nabla^2 u)v \, \mathrm{d}x = \int_\Omega qv \, \mathrm{d}x \quad \text{integrate over } \Omega$$

$$\int_\Omega \nabla \cdot (-k\nabla u)v \, \mathrm{d}x = \int_\Omega qv \, \mathrm{d}x \quad \text{definition of Laplacian}$$

$$\int_\Omega \nabla \cdot (-kv\nabla u) \, \mathrm{d}x + \int_\Omega \nabla v \cdot k\nabla u \, \mathrm{d}x = \int_\Omega qv \, \mathrm{d}x \quad \text{apply product rule}$$

$$\int_{\partial\Omega} (-kv\nabla u) \cdot \hat{n} \, \mathrm{d}s + \int_\Omega \nabla v \cdot k\nabla u \, \mathrm{d}x = \int_\Omega qv \, \mathrm{d}x \quad \text{apply divergence theorem}$$

First, need a weak form

$$-k\nabla^2 u = q \quad \leftarrow \text{strong form}$$

$$(-k\nabla^2 u)v = qv \quad \text{multiply by test function } v$$

$$\int_\Omega (-k\nabla^2 u)v \,\mathrm{d}x = \int_\Omega qv \,\mathrm{d}x \quad \text{integrate over } \Omega$$

$$\int_\Omega \nabla\cdot(-k\nabla u)v \,\mathrm{d}x = \int_\Omega qv \,\mathrm{d}x \quad \text{definition of Laplacian}$$

$$\int_\Omega \nabla\cdot(-kv\nabla u) \,\mathrm{d}x + \int_\Omega \nabla v\cdot k\nabla u \,\mathrm{d}x = \int_\Omega qv \,\mathrm{d}x \quad \text{apply product rule}$$

$$\int_{\partial\Omega} (-kv\nabla u)\cdot\hat{n} \,\mathrm{d}s + \int_\Omega \nabla v\cdot k\nabla u \,\mathrm{d}x = \int_\Omega qv \,\mathrm{d}x \quad \text{apply divergence theorem}$$

$$\int_\Omega k\nabla u\cdot\nabla v \,\mathrm{d}x = \int_\Omega qv \,\mathrm{d}x \quad u \text{ is 0 on } \partial\Omega \text{ (and commutative dot product)}$$

First, need a weak form

$-k\nabla^2 u = q$ $\leftarrow$strong form

$(-k\nabla^2 u)v = qv$ multiply by test function v

$\int_\Omega (-k\nabla^2 u)v \, \mathrm{d}x = \int_\Omega qv \, \mathrm{d}x$ integrate over Ω

$\int_\Omega \nabla \cdot (-k\nabla u)v \, \mathrm{d}x = \int_\Omega qv \, \mathrm{d}x$ definition of Laplacian

$\int_\Omega \nabla \cdot (-kv\nabla u) \, \mathrm{d}x + \int_\Omega \nabla v \cdot k\nabla u \, \mathrm{d}x = \int_\Omega qv \, \mathrm{d}x$ apply product rule

$\int_{\partial\Omega} (-kv\nabla u) \cdot \hat{n} \, \mathrm{d}s + \int_\Omega \nabla v \cdot k\nabla u \, \mathrm{d}x = \int_\Omega qv \, \mathrm{d}x$ apply divergence theorem

$\boxed{\int_\Omega k\nabla u \cdot \nabla v \, \mathrm{d}x = \int_\Omega qv \, \mathrm{d}x}$ u is 0 on $\partial\Omega$ (and commutative dot product)

Firedrake utility function to generate circular mesh

(so don't need to generate it in advance)

Will use degree 1 Lagrange elements (linear basis functions)

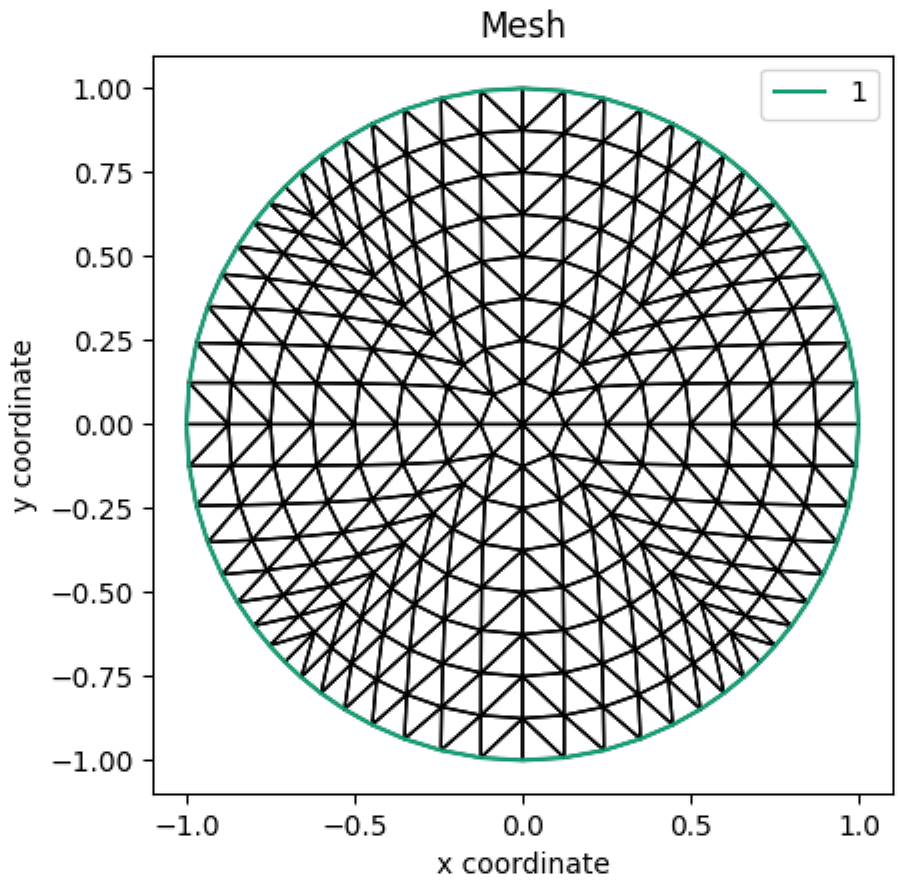
Time to write to the program.

```python
from firedrake import *
```

Bring in Firedrake goodies (from X import * makes my heart sad)

```python
mesh = UnitDiskMesh(3)
```

Use Firedrake convenience function to generate a mesh



```python
H = FunctionSpace(mesh, "CG", 1)
```

Declare function space

```python
u = Function(H)
v = TestFunction(H)
```

Declare functions

```python
q = Constant(1.0)  # heat production
k = Constant(0.1)  # thermal conductivity
```

Declare constants (these could also be functions...)

```python
F = (k * dot(grad(u), grad(v)) - q * v) * dx
```

Weak form (dx is a Firedrake object)

$$\int_\Omega (k(\nabla u \cdot \nabla v) - qv)\,\mathrm{d}x$$

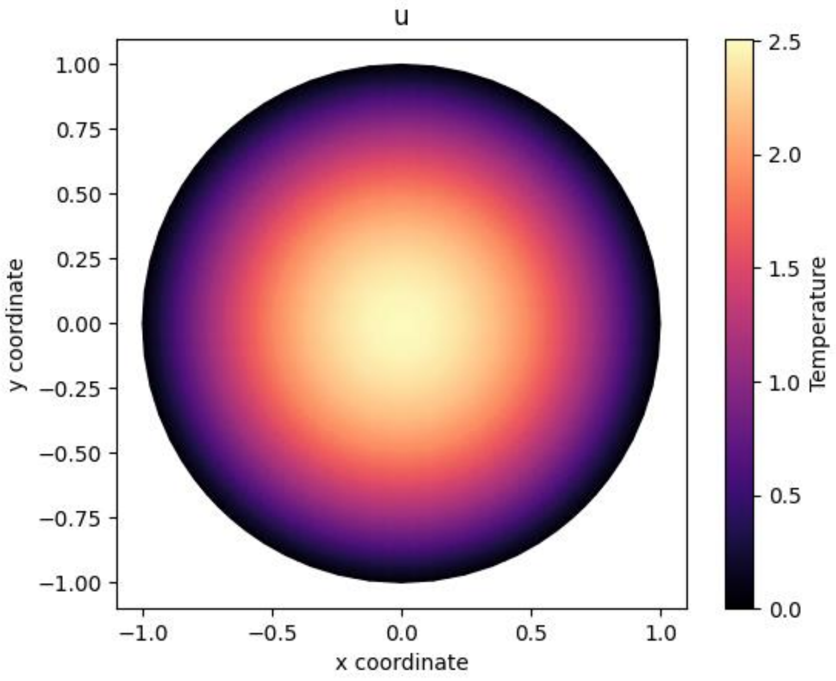
```python
bc = DirichletBC(H, Constant(0.0), 1)
```

Define boundary condition (Neumann BC would be applied as part of the weak form)

```
solve(
    F == 0,
    u,
    bcs=[bc],
)
```

Hand off to Firedrake

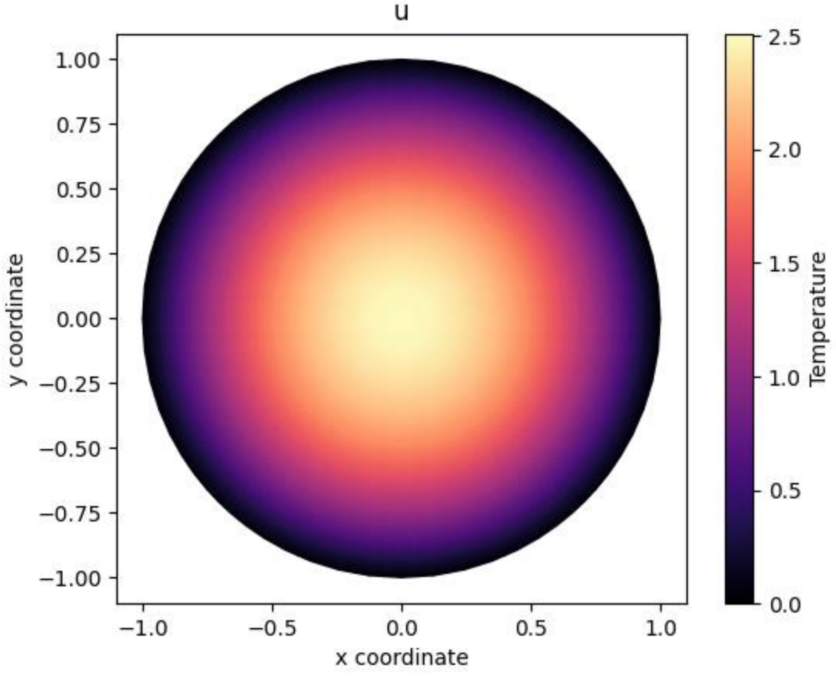
Try running heat0.py

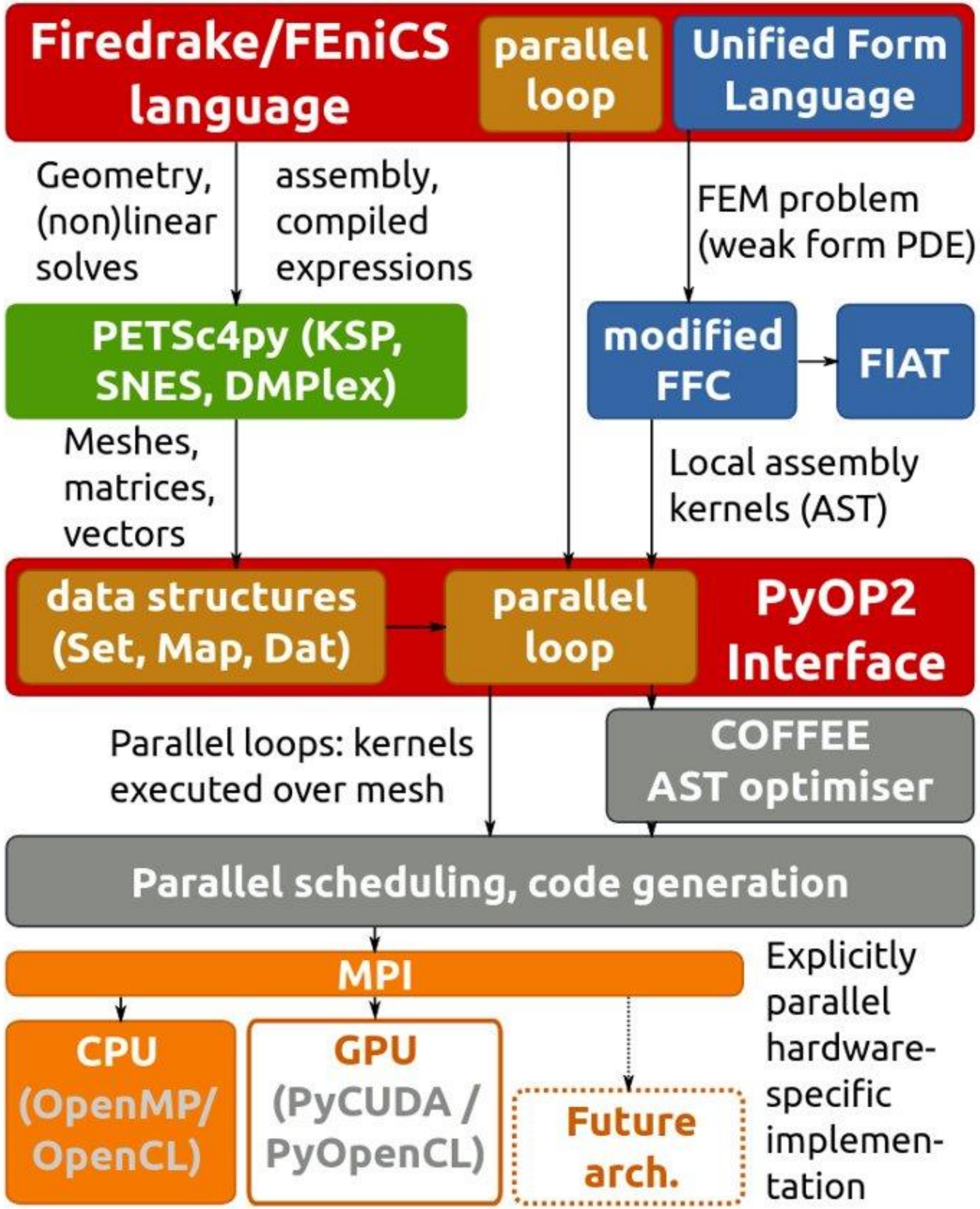


```
solve(
    F == 0,
    u,
    bcs=[bc],
)
```

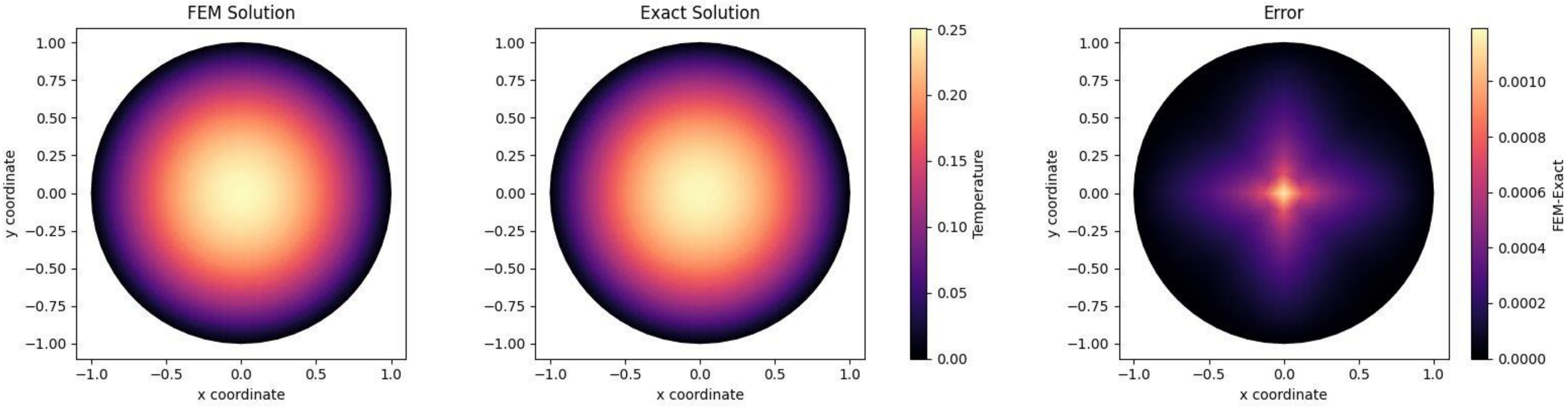
Hand off to Firedrake

Try running heat0.py





Comparison with exact solution



Things to fiddle with – how does error value/structure change when modifying:

mesh refinement (the argument supplied to UnitDiskMesh)

element degree (integer argument to FunctionSpace)

heat1.py - q(x,y) instead of constant

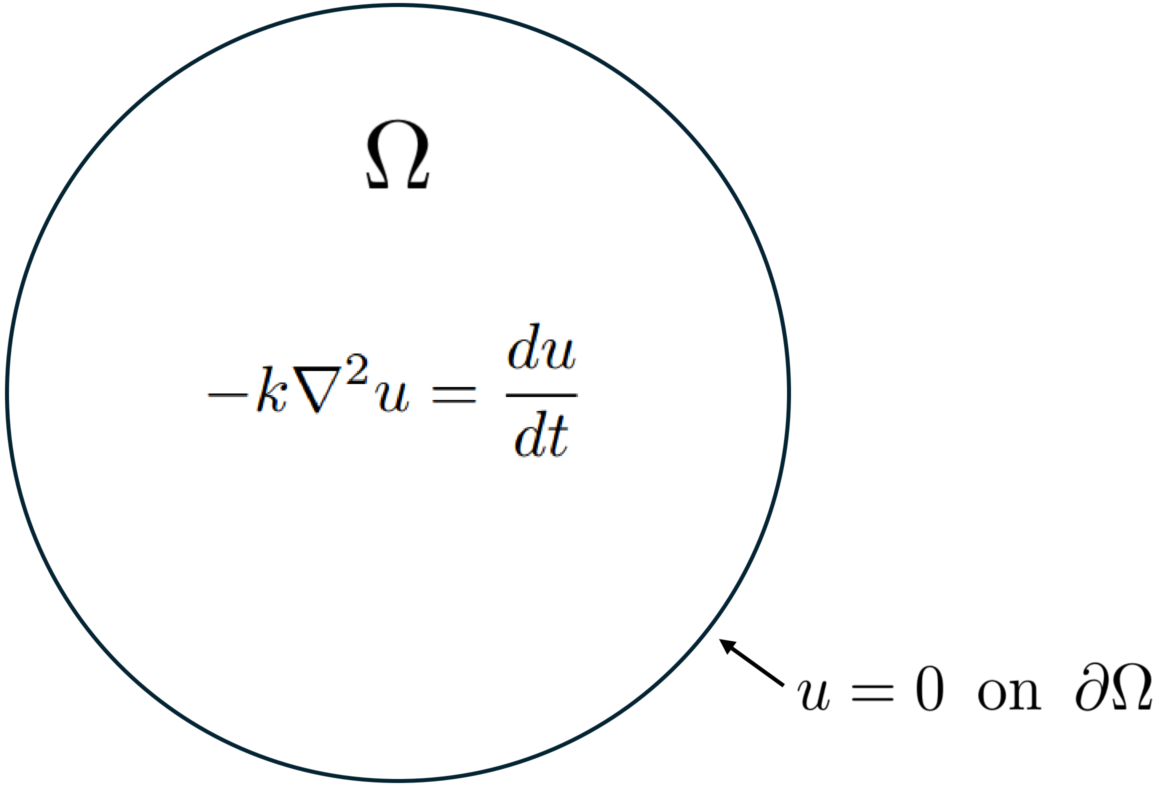
$$q(x, y) = e^{-10(x^2+(y-0.25)^2)}$$

```
x, y = SpatialCoordinate(mesh)
q = Function(H).interpolate(exp(-10 * (x**2 + (y - 0.25) ** 2)))
```

Exercise – make thermal conductivity vary over the domain as well

$$\frac{1}{1 + e^{-5x}}$$

Exercise (with help) – time stepping heat equation



k - Thermal diffusivity
u - Temperature

$u(t=0) = \text{some value}$

Exercise (with help) – time stepping heat equation

Weak form for backward Euler (implicit) time stepping with no source term:

$$\int_\Omega \left(\frac{u_n - u_{n-1}}{\Delta t} v + k(\nabla u_n \cdot \nabla v) \right) \, \mathrm{d}x = 0$$

Copy and modify ex0.py

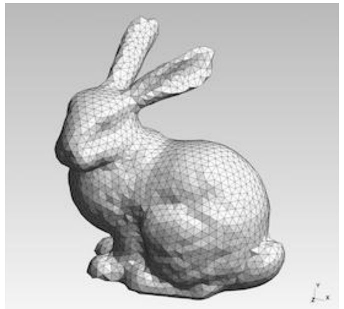
Things to change

- Get rid of exact solution stuff and mesh plotting
- Need a new function… uold?
  - Give it some initial value (uold.interpolate(10))
- Weak form loses source term and gains time step term
  - q can be deleted
  - time step size (call it dt) must be defined
- Solve repeatedly instead of just once, each time write the new temperature field to uold (uold.assign(u))
- Plot at every iteration?
- Try k=0.1, dt=0.1

$$\int_\Omega \left(\frac{u_n - u_{n-1}}{\Delta t} v + k(\nabla u_n \cdot \nabla v) \right) \mathrm{d}x = 0$$

```python
from firedrake import *

# Set up mesh
mesh = UnitDiskMesh(3)

# Define function space and functions
H = FunctionSpace(mesh, "CG", 1)

u = Function(H)
uold = Function(H)
v = TestFunction(H)

# Weak form and boundary condition
k = Constant(0.1)  # thermal diffusivity
dt = Constant(0.1)  # time step
uold.interpolate(10)

F = (((u - uold) / dt) * v + (k * dot(grad(u), grad(v)))) * dx

bc = DirichletBC(H, Constant(0.0), 1)

# Hand off to Firedrake
for j in range(10):
    solve(
        F == 0,
        u,
        bcs=[bc],
    )
    uold.assign(u)
```

Some things we didn't talk about



Meshing

- Many tools for this. Gmsh seems popular. Complex meshes are complex to generate.

PETSc options

- Lots to do here. Firedrake defaults seem to in general work fine for small problems, but for bigger problems you probably want (need) to mess with this.

Element types

- Many types of finite elements for different purposes.

Writing Paraview files

- Powerful/convenient way to visualize results (particularly in 3D)

Useful and/or interesting links

Firedrake website

PETSc

UFL

Gmsh

Glen-Stokes flow tutorial

Scalar wave equation tutorial