FEniCS Course

Lecture 4: Time-dependent PDEs

Contributors
Hans Petter Langtangen
Anders Logg

Course outline

The heat equation

We will solve the simplest extension of the Poisson problem into the time domain, the heat equation:

$$\begin{aligned} \frac{\partial u}{\partial t} - \Delta u &= f \quad \text{in } \Omega \text{ for } t > 0 \\ u &= g \quad \text{on } \partial\Omega \text{ for } t > 0 \\ u &= u^0 \text{ in } \Omega \text{ at } t = 0 \end{aligned}$$

The solution $u = u(x, y, t)$, the right-hand side $f = f(x, y, t)$ and the boundary value $g = g(x, y, t)$ may vary in space and time. The initial value u^0 is a function of space only.

Time-discretization of the heat equation

We discretize in time using the implicit Euler (dG(0)) method:

$$\frac{\partial u}{\partial t} \approx \frac{u^n - u^{n-1}}{\Delta t}$$

Semi-discretization of the heat equation:

$$\frac{u^n - u^{n-1}}{\Delta t} - \Delta u^n = f^n$$

$$u^n - \Delta t \Delta u^n = u^{n-1} + \Delta t f^n$$

Solve for u^1, u^2, …

Variational problem for the heat equation

Find $u^n \in V^n$ such that

$$a(u^n, v) = L^n(v)$$

for all $v \in \hat{V}$ where

$$\begin{aligned} a(u, v) &= \int_\Omega uv + \Delta t \nabla u \cdot \nabla v \, \mathrm{d}x \\ L^n(v) &= \int_\Omega u^{n-1} v + \Delta t f^n v \, \mathrm{d}x \end{aligned}$$

Note that the bilinear form $a(u, v)$ is constant while the linear form L^n depends on n

Time-stepping algorithm

```
Define the boundary condition
Compute $u^0$ as the projection of the given initial value
Define the forms $a$ and $L$
Assemble the matrix $A$ from the bilinear form $a$
$t \leftarrow \Delta t$
while $t \leqslant T$ do
    Assemble the vector $b$ from the linear form $L$
    Apply the boundary condition
    Solve the linear system $AU = b$ for $U$ and store in $u^1$
    $t \leftarrow t + \Delta t$
    $u^0 \leftarrow u^1$ (get ready for next step)
end while
```

Test problem

We construct a test problem for which we can easily check the answer. We first define the exact solution by

$$u = 1 + x^2 + \alpha y^2 + \beta t$$

We insert this into the heat equation:

$$f = \dot{u} - \Delta u = \beta - 2 - 2\alpha$$

The initial condition is

$$u^0 = 1 + x^2 + \alpha y^2$$

This technique is called the *method of manufactured solutions*

Handling time-dependent expressions

We need to define a time-dependent expression for the boundary value:

```
alpha = 3
beta = 1.2

g = Expression("1 + x[0]*x[0] + \
               alpha*x[1]*x[1] + beta*t",
               alpha=alpha, beta=beta, t=0)
```

Updating parameter values:

```
g.t = t
```

Projection and interpolation

We need to project the initial value into V_h:

```
u0 = project(g, V)
```

We can also interpolate the initial value into V_h:

```
u0 = interpolate(g, V)
```

Implementing the variational problem

```
dt = 0.3

u0 = project(g, V)
u1 = Function(V)

u = TrialFunction(V)
v = TestFunction(V)
f = Constant(beta - 2 - 2*alpha)

a = u*v*dx + dt*inner(grad(u), grad(v))*dx
L = u0*v*dx + dt*f*dx

# assemble only once, before time-stepping
A = assemble(a)
```

Implementing the time-stepping loop

```python
T = 2
t = dt

while t <= T:
    b = assemble(L)
    g.t = t
    bc.apply(A, b)
    solve(A, u1.vector(), b)

    t += dt
    u0.assign(u1)
```

Programming exercise

- Write a program to solve the heat equation
- Write your program in a file named `heat.py`
- Run your program using

```
python heat.py
```

- The complete program is available[1] as

```
transient/diffusion/d1_d2D.py
```

[1] `http://fenicsproject.org/pub/book/tutorial/`

Course outline

L1 Introduction

L2 Static linear PDEs

L3 Static nonlinear PDEs

L4 Time-dependent PDEs

L5 Advanced topics