INF5620

Autumn 2015

Nonlinear diffusion equation

John-Anders Stende

Date: November 17, 2015

Abstract

The goal of this project is to discuss numerical aspects of a nonlinear diffusjon equation

$$\varrho u_t = \nabla \cdot (\alpha \nabla u) + f(\vec{x}, t) \tag{1}$$

with initial condition $u(\vec{x}, 0) = I(\vec{x})$ and boundary condition $\frac{\partial u}{\partial n} = 0$. ϱ is a constant, $\alpha(u)$ is a known function of u.

a)

Introduce some finite difference approximation in time that leads to an implicit scheme (say a Backward Euler, Crank-Nicolson, or 2-step backward scheme). Derive a variational formulation of the initial condition and the spatial problem to be solved at each time step (and at the first time step in case of a 2-step backward scheme).

A Backward Euler scheme for (1) reads (in operator notation)

$$[\varrho D_t^- u = \nabla \cdot (\alpha(u)\nabla u) + f(\vec{x}, t)]^n \tag{2}$$

Written out,

$$\varrho u^n - \Delta t\big(\nabla \cdot (\alpha(u^n)\nabla u^n) + f(\vec{x}, t_n)\big) = \varrho u^{n-1} \tag{3}$$

This is the time-discrete version of the PDE, but we also need to discretize in space. Variational formulation of the spatial part using Galerkin method (multiplying with a test function v and integrating over the domain) yields

$$\varrho \int_\Omega u^n v dx - \int_\Omega \Delta t(\nabla \cdot (\alpha \nabla u^n))v dx - \int_\Omega \Delta t f^n v dx = \varrho \int_\Omega u^{n-1} v dx \tag{4}$$

We do integration by parts on the integral involving derivatives of u,

$$\int_\Omega \nabla \cdot (\alpha \nabla u^n) v dx = -\int_\Omega \alpha \nabla u^n \cdot v dx + \int_{\partial\Omega} \alpha \frac{\partial u^n}{\partial n} v dx \tag{5}$$

The last term vanishes because we have the Neumann condition $\partial u^n / \partial n = 0$ for all n. Our discrete problem in space and time then reads

$$\int_\Omega (\varrho u^n v + \Delta t \alpha(u^n)\nabla u^n \cdot \nabla v - \Delta t f^n v - \varrho u^{n-1} v) dx = 0 \tag{6}$$

u is now approximated as a linear combination of basis functions ψ_i, $u = \sum_i c_i \psi_i$. The nonlinear algebraic equation F_i for u^n follow from setting $v = \psi_i$,

$$F_i = \int_\Omega (\varrho u^n \psi_i + \Delta t \alpha(u^n)\nabla u^n \cdot \nabla \psi_i - \Delta t f^n \psi_i - \varrho u^{n-1} \psi_i) dx = 0 \tag{7}$$

This is the equation that needs to be solved for each time step. The variational problem for the initial condition is simply

$$\int_\Omega (u^0 \psi_i - I\psi_i) dx = 0 \tag{8}$$

b)

Formulate a Picard iteration method at the PDE level, using the most recently computed u function in the $\alpha(u)$ coefficient. Derive general formulas for the entries in the linear system to be solved in each Picard iteration. Use the solution at the previous time step as initial guess for the Picard iteration.

Picard iteration linearize the equation by using the most recent approximation u^- to u in the coefficient $\alpha(u)$ (switching to the notation $u^n = u$ and $u^{n-1} = u^{(1)}$),

$$F_i \approx \hat{F}_i = \int_\Omega (\varrho u \psi_i + \Delta t \alpha(u^-)\nabla u \cdot \nabla \psi_i - \Delta t f \psi_i - \varrho u^{(1)} \psi_i) dx \tag{9}$$

The equation $\hat{F}_i = 0$ to be solved for each Picard iteration is now linear and can be translated to a linear system $\sum_j A_{i,j} c_j = b_i$ for the unknown coefficients c_i by inserting $u = \sum_i c_i \psi_i$ and moving all unknown terms to the rhs and known terms to the lhs,

$$A_{i,j} = \int_\Omega (\varrho \psi_j \psi_i + \Delta t \alpha(u^-)\nabla \psi_j \cdot \nabla \psi_i) dx, \quad b_i = \int_\Omega (\Delta t f \psi_i + \varrho u^{(1)} \psi_i) dx \tag{10}$$

The iteration begins with setting $u^- = u^{(1)}$.

c)

Restrict the Picard iteration to a single iteration. That is, simply use a u value from the previous time step in the $\alpha(u)$ coefficient. Implement this method with the aid of the FEniCS software (in a dimension-independent way such that the code runs in 1D, 2D, and 3D).

Implemented in script *diffusion.py*

d)

The first verification of the FEniCS implementation may reproduce a constant solution. Find values of the input data ϱ, α, f and I such that $u(\vec{x},t) = C$ where C is some chosen constant. Write test functions that verify the computation of a constant solution in 1D, 2D, and 3D for P1 and P2 elements (use simple domains: interval, square, box).

Inserting $u(\vec{x},t) = C$ in the PDE, considering that the derivative of u is zero, yields

$$\varrho \cdot 0 = \nabla \cdot (\alpha(C) \cdot 0) + f(\vec{x},t) \tag{11}$$

and we see that f must be zero, while ϱ and α can have any value. Obviously, $I(\vec{x}) = C$.

Function *test_constant* tests whether u stays constant for P1 and P2 elements in 1D, 2D and 3D. Running *py.test -v diffusion.py* gives the following output:

```
============================= test session starts ==============================
platform linux2 -- Python 2.7.6 -- py-1.4.20 -- pytest-2.5.2 -- /usr/bin/python
collected 1 items

diffusion.py:81: test_constant PASSED

=========================== 1 passed in 3.71 seconds ===========================
```

e)

If we assume $\alpha(u) = 1$, $f = 0$, $\Omega = [0,1] \times [0,1]$, P1 elements and $I(x,y) = cos(\pi x)$, then the exact solution is $u_e(x,y,t) = e^{-\pi^2 t} cos(\pi x)$. The error in space is then $\mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta x^2)$ while the error in time (for Backward Euler) is $\mathcal{O}(\Delta t)$. We can then write a model for an error measure:

$$E = K_t \Delta t + K_x \Delta x^2 + K_y \Delta y^2 = Kh \tag{12}$$

where h is a common discretization measure. A suitable E can be taken as the discrete L2 norm of the solution at the nodes for a fixed time. In function *error_measure* I compute K as the mesh in space and time is simultaneously refined:

```
E = 0.135665, h = 0.500000: K = 0.271331
E = 0.045869, h = 0.250000: K = 0.183476
E = 0.020504, h = 0.125000: K = 0.164029
E = 0.008679, h = 0.062500: K = 0.138866
E = 0.003978, h = 0.031250: K = 0.127282
E = 0.001812, h = 0.015625: K = 0.115954
E = 0.000857, h = 0.007812: K = 0.109738
E = 0.000422, h = 0.003906: K = 0.108033
```

We see that K stays approximately constant as the mesh is refined, although it does get a bit smaller for each iteration. This change is reduced for each new h though, so K probably converge to a specific value.

f)

To verify NONlinear implementation with a source term, we use the method of manufactured solutions. We now restrict our problem to one space dimension, $\Omega = [0,1]$ and choose

$$u(x,t) = t \int_0^x q(1-q)dq = tx^2(\frac{1}{2} - \frac{x}{3}) \tag{13}$$

and $\alpha(u) = 1+u^2$. *sympy* can be used to compute a source term so that u is a solution to the PDE problem. Function *manufactured_solution* computes E as above for three different values of t. The results are

```
E = 0.000000207, h = 0.050000: t = 0.100000
E = 0.000013021, h = 0.050000: t = 0.500000
E = 0.005162306, h = 0.050000: t = 1.000000
```

The error is small, but increases with time. It seems like the numerical and exact solution do not end up in exactly the same diffused state. This could be a natural consequence of the discretiation errors together with the fact that we use only one Picard iteration.

g)

The different sorts of numerical errors in the implementation are

- Discretization error in space: $\mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta y^2)$
- Discretization error in time: $\mathcal{O}(\Delta t)$
- Error due to single Picard iteration
- Numerical integration errors in the FEniCS software

h)

A good way to test the implementation is to compute the convergence rate. To do this, we must eliminate the error du to a single Picard iteration so it doesn't pollute the convergence rate. This can be done by finding a manufactured solution that fulfills the PDE with $\alpha(u^{(1)})$. Choosing the manufactured solution (13) and $\alpha(u) = 1 + u^2$, we can use *sympy* to compute the necessesary source term f.

For decreasing time steps $\Delta t = h$, the convergence rate can be found by comparing two consecutive experiments

$$r_{i-1} = \frac{ln(E_{i-1}/E_i)}{ln(\Delta t_{i-1}/\Delta t_i)}$$

where E is the L2 norm of the error. Δt is halved for each run. We observe that $E \sim h^r$, r should converge to 1 for this particular discretization because the error is $\mathcal{O}(\Delta t)$ for Backward Euler. Function *convergence_rate* computes r, and we see that it does indeed converge to $r = 1$:

```
r: [1.19, 2.26, 1.38, 0.87, 1.18, 0.82, 0.91, 1.07, 1.04]
```

i)

We will now simulate the nonlinear diffusion of Gaussian function. Due to symmetry of the Gaussian function (with respect to $x = 0$ and $y = 0$), it suffices to simulate one quarter of the domain:

$$I(x, y) = e^{-(x^2+y^2)/2\sigma^2} \quad (x, y) \in \Omega = [0, 1] \times [0, 1] \tag{14}$$

where σ measures the width of the Gaussian function. Choose $\alpha(u) = 1 + \beta u^2$ where β is a constant. See function *gaussian* for implementation. Setting $f = 0$ leads to the following results

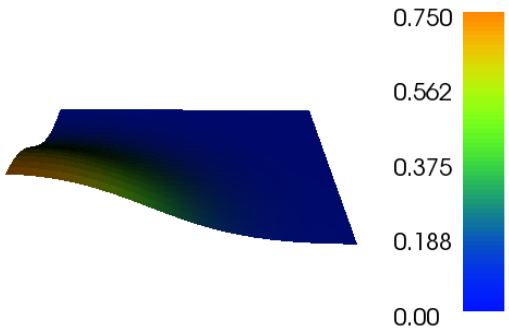

Figure 1: t = 0

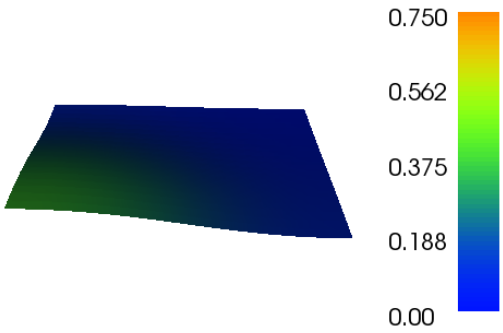

Figure 2: t = 0.05

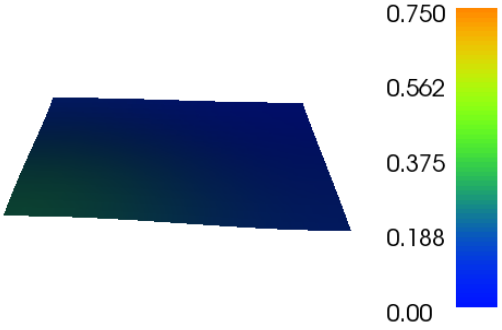

Figure 3: t = 0.1

Figure 4: t = 0.3

We see that the solution starts out as a Gaussian in the corner $(x, y) = (0, 0)$ before it spreads out evenly, i.e. diffuses until it ends up in a completely diffused state, as expected.