Nonlinear diffusion equation with finite elements

Fredrik E Pettersen
fredriep@student.matnat.uio.no

November 27, 2012

About the problem

In this project we will solve a nonlinear diffusion equation using the finite element method and the FEniCS software package. More precicely we will be looking at the equation

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

where ρ is a constant, $\alpha(u)$ is a known function of u, and where we have the initial condition $u(\mathbf{x}, 0) = I(\mathbf{x})$, and Neuman boundary conditions $\frac{\partial u}{\partial n} = 0$.

Discretization

In order to solve equation (1) we will need to discretize it. The standard approach is then to pick a nummerical approximation to the derivatives. We will use a finite difference discertization in time, more specifically the Backward Euler (BE) discretization, and a finite element approximation in space. The BE discertization gives us

$$\rho u^n = \Delta t \nabla \cdot (\alpha(u)\nabla u) + \Delta t f(\mathbf{x}, t) + \rho u^{n-1}$$
$$\rho u^n - \Delta t \nabla \cdot (\alpha(u)\nabla u) - \Delta t f(\mathbf{x}, t) - \rho u^{n-1} = R$$

we now approximate R on a functionspace $V = span\{\phi_i\}$, where ϕ_i denotes the P1 elements, and minimize the error by demanding

$$\big(R, v\big) = 0 \quad \forall\, v \in V$$

which gives us

$$\rho(u^n, v) - \Delta t(f(\mathbf{x}, t_n), v) - \rho(u^{n-1}, v) - \Delta t(\nabla \cdot (\alpha(u)\nabla u^n), v) = 0$$

The first three terms are ok for now, but the last term has a double derivative in u, which is something we dont want seeing as we are trying to approximate u by P1 elements as $u \simeq \sum_k u_k \phi_k$, and the double derivative of a linear function is simply zero. To get around this problem we try to integrate by parts

$$\Delta t(\nabla \cdot (\alpha(u)\nabla u^n), v) = \Delta t \left(\int_\Omega \nabla \cdot (\alpha(u)\nabla u^n) v dx \right)$$
$$= \Delta t \left([\alpha(u)\nabla u v]_{\partial\Omega} - \int_\Omega \alpha(u)\nabla u^n \nabla v dx \right) = -\Delta t \int_\Omega \alpha(u)\nabla u^n \nabla v dx$$

where we have inserted for the boundary conditions $\frac{\partial u}{\partial n} = \mathbf{n} \cdot \nabla u = 0$ on $\partial\Omega$. We have now arrived on a linear variational problem to be solved at each timestep

$$\rho(u^n, v) - \Delta t(f(\mathbf{x}, t_n), v) - \rho(u^{n-1}, v) + \Delta t(\alpha(u)\nabla u^n, \nabla v) = 0$$
$$\underbrace{\rho(u^n, v) + \Delta t(\alpha(u)\nabla u^n, \nabla v)}_{a(u,v)} = \underbrace{\Delta t(f(\mathbf{x}, t_n), v) + \rho(u^{n-1}, v)}_{L(u,v)}$$

Notice that we have a nonlinearity in $\alpha(u)$. There are several way to get around this nonlinearity, we will use Picard iteration meaning that we start out with $\alpha(u^{n-1})$ and solve the linear system to get a solution $\tilde{u}^n$. Then we update $\alpha(\tilde{u}^n)$ and solve the system again. We keep dionig this untill $\|\tilde{u}^n - \tilde{u}^{n-1}\| \leq \epsilon$ where ϵ is a predefined

tolerance. Hopefully this converges to the correct solution, and we can set our $\tilde{u}^n = u^n$. Picard iteration leads us to the following linear system for each iteration

$$\rho \int_\Omega u^n v dx + \Delta t \int_\Omega \alpha(u^{n-1}) \nabla u^n \nabla v dx = \rho \int_\Omega u^{n-1} v dx + \Delta t \int_\Omega f(x, t_n) v dx$$

we will now insert for $u \simeq \sum_k u_k \phi_k$ and because we are working with an undefined number of cells of potentially variable size we will transform all the integrals to a reference cell where $-1 \leq X \leq 1$. We will also do the calculations in 1 dimension, but they easily generalize to multiple dimensions by simply setting each element in say the vector $\mathbf{u}^n$ equal to a vector. The same approach gives us a block matrix for M and K.

$$\rho \sum_{i=1}^{N_x} \int_{-1}^{1} (\phi_i \phi_j) u_i^n dx + \Delta t \sum_{i=1}^{N_x} \int_{-1}^{1} \alpha(u_i^{n-1})(\phi_i' \phi_j') u^n dx = \rho \sum_{i=1}^{N_x} \int_{-1}^{1} (\phi_i \phi_j) u_i^{n-1} dx + \Delta t \sum_{i=1}^{N_x} \int_{-1}^{1} f_i(x, t_n) \phi_j dx$$

If we now set $M_{ij} = \rho \int_0^L (\phi_j \phi_i) dx$ and $K_{ij} = \Delta t \int_0^L \alpha(u^{n-1})(\phi_j' \phi_i') dx$ we get

$$M_{ij} = \int_0^L (\phi_j \phi_i) dx \implies M_{00} = \int_{-1}^{1} (\phi_0 \phi_0) dx$$

We are usin g P1 elements, and so the only nonzero integrals we will get are when $i = j$ and $i \pm 1 = j$, meaning M and K will become tridiagonal.
VIS HVA ELEMENTENE BLIR
we are left with

$$\begin{aligned} M\mathbf{u}^n + K(u^{n-1})\mathbf{u}^n &= M\mathbf{u}^{n-1} + \mathbf{b} \\ \implies \mathbf{u}^n &= (M + K(u^{n-1}))^{-1}(M\mathbf{u}^{n-1} - \mathbf{b}) \end{aligned}$$

to be solved for each Picard iteration. The crudest Picard iteration will be just one iteration, which is the same as solving the system using $\alpha(u^{n-1}) \simeq \alpha(u^n)$ and this is our chosen approach.

Implementation

To make the program as general as possible, we will make it dimension independent by providing the number of spatial points on the commandline as n_x, n_y, n_z and choose a unit interval, square or cube accordingly. Also, even though we will only use one Picard iteration, we will implement this as a function where the maximum nuber of iterations is given as an argument. The resulting code can be found in the appendix.

Verification

For the first verification of the program we will assume $\rho = \alpha(u) = 1$, $f(\mathbf{x}, t) = 0$ and $I(\mathbf{x}) = \cos(\pi x)$. We will wrok on the domain $\Omega = [0, 1] \times [0, 1]$. This should give an analytic solution of $u(x, y, t) = e^{-\pi^2 t} \cos(\pi x)$

Errors

As in any (nummerical) approximation there will be an error in our solution. We can distinguish two types of errors, there is the error we do in the approximation of the derivatives, and there is the nummerical error.
Our approximation using the Backward Euler scheme has an error which goes like $\mathcal{O}(\Delta t)$ for other reasons the spatial error goes like $\mathcal{O}(\Delta x^2)$. Other potential errors which will arise in the FEniCS program are errors due to loss of prescision in the nummerical integration we do upon assembly of the matrices M and K. Depending on what kind of nummerical integration we use, the error from integration could be $\mathcal{O}(\Delta x)$ for the rectangle (midpoint) rule or it could even be zero if the solution is a polynomial of degree $2n-1$ and we integrate by some form of Gaussian quadrature. n is here the order of elements used. There is allways a possibillity of loosing nummerical precision because of adding a small and a large number since we only have 16 valid decimals. In our case we also have an obvious error in the approximation of the nonlinear term.

Analytic solution

Results

Final comments