Non linear diffusion equation

Yapi Donatien Achou

November 18, 2014

0.1 Introduction

In this project we solve the time dependent non linear diffusion equation, with Neumann boundary condition. To formulate a finite element method for the non linear diffusion equation, the time derivative is approximated by the backward scheme. The nonlinear diffusion equation is thus sampled at some time level say k and expressed as a spatial problem. The nonlinear term in the diffusion equation is linearised by using a Picard iteration. The discretised equation is implemented in FEniCs. Fenics is a user friendly software for solving finite element method. The first verification of the FEniCs implementation reproduce a constant solution. The second verification of the FEniCs implementation reproduce a simple analytical solution and shows that the ratio of the error and the time step is constant. The nonlinear FEniCs implementation is tested by recovering the convergence rate of the backward Euler scheme. The project is conclude by simulating a nonlinear Gaussian function for different nonlinear terms.

0.2 Mathematical problem

The goal is the solve the nonlinear diffusion equation

$$\rho\frac{\partial u}{\partial t} = \nabla\cdot(\alpha(u)\nabla u) + f(\mathbf{x},t) \quad u\in\Omega \tag{1}$$

with Initial condition

$$u(\mathbf{x},t) = I(\mathbf{x}) \tag{2}$$

and Neumann boundary condition

$$\frac{\partial u}{\partial n} = 0 \tag{3}$$

Where ρ is the density and $\alpha(u)$ is a function of u and f is a source term. Equation (1) is nonlinear because of $\alpha(u)$. If u is for example a scalar quantity such as temperature, then (1,2,3) model the temperature distribution in the domain Ω. In this case $\alpha(u)$ is the thermal diffusion and f is the heat source.

0.3 Numerical solution by finite element method

0.3.1 Variational formulation

To derive the variational formulation of (1,2,3) we first discretise the time derivative by using a backward scheme:

$$\frac{\partial u}{\partial t} \approx \frac{u^{k+1}-u^k}{\Delta t} \tag{4}$$

And at time level k we reformulate (1) as a spatial problem:

$$\rho\frac{u^k-u^{k-1}}{\Delta t} = \nabla\cdot(\alpha(u^k)\nabla u^k) + f^k(\mathbf{x},t) \tag{5}$$

Let V be the finite element space. Using the Galerkin finite element method, the trial space is the same as the test space. Let $v\in V$ be a test function. We multiply (5) by v then integrate by part. We respectively get:

$$\int_\Omega u^k v\mathrm{d}x + \frac{\Delta}{\rho}\int_\Omega \alpha(u^k)\nabla u^k\cdot\nabla v\mathrm{d}x = \int_\Omega u^{k-1}v\mathrm{d}x + \frac{\Delta}{\rho}\int_\Omega f^k v\mathrm{d}x \tag{6}$$

$$a(u,v) = \int_\Omega u^k v\mathrm{d}x + \frac{\Delta}{\rho}\int_\Omega \alpha(u^k)\nabla u^k\cdot\nabla v\mathrm{d}x$$

$$L(v) = \int_\Omega u^{k-1}v\mathrm{d}x + \frac{\Delta}{\rho}\int_\Omega f^k v\mathrm{d}x$$

The variational formulation of our problem is: find $u\in V$ such that

$$a(u,v) = L(v) \quad \forall v\in V$$

0.3.2 Picard iteration at the pde level

In the Picard iteration we use the known previewn solution in the nonlinear term $\alpha(u)$. By doing so, the nonlinear diffusion equation is linearised. More precisely given u^{k-1} find u^k such the

$$\rho\frac{u^k-u^{k-1}}{\Delta t} = \nabla\cdot(\alpha(u^{k-1})\nabla u^k) + f^k(\mathbf{x},t) \tag{7}$$

0.4 FEniCs implementation

```
def diffusion(dimension,d,alpha,I,f,rho,size):
    """
    solve the time dependent diffusion equation
    with homogeneous Neumann boundary condition.
    in the program u_1 correspond to u^k-1 and u correspond
    to u^k.
    usage of the function duffusion:

    diffusion('dimension','d',I,alpha,rho):
    dimension = 1D, 2D, 3D for a one dimension,
    two dimension, and three dimension. problem respectivelly

    d = 1 for P1 Lagrange element and d = 2 for P2 Lagrange
    element.
    """

    # specify the dimension of the problem
    if dimension == 1:
        mesh = UnitIntervalMesh(size)

    elif  dimension == 2:
        mesh = UnitSquareMesh(size,size)

    elif  dimension == 3:
        mesh = UnitCubeMesh(size,size,size)


    # finite element function space
    V = FunctionSpace(mesh, "Lagrange", 1)
    R = FunctionSpace(mesh, "R", 0)
    W = V * R # mixt function space


    u_1 = interpolate(I,V)
    u = TrialFunction(V)
    v = TestFunction(V)

    dt = 0.02
    # mixt finite element varitional form
    a = ((dt/rho)*inner(alpha(u)*grad(u), grad(v)) + (dt/rho)*c*v + u*d)*dx +u*v*dx
    L = f*v*dx + g*v*ds + u_1*v*dx


    #Compute solution
    u= Function(V)
    T = 0.05
    t = dt
    w = Function(W)
    t = dt
    while t<=T:
        I.t = t
        solve(a == L, w)
        (u, c) = w.split(True)
        u_1.assign(u)
        t+=dt
    return u
```

0.5 Verifications

The first verification of FEniCs implementation reproduce a constant solution with machine precision see file **verification.py** In the second verification of the FEniCs implementation we proved that using an analytical solution

$$u(x,y,t) = \exp(-\pi^2 t)\cos(\pi x) \tag{8}$$

The ratio

$$\frac{E}{h}$$

is approximately constant. See file **analyticalVerfication.py**. Here E is the L^2 while $h = \Delta t$. The file **verification.py** contains verification for the implementation.

0.5.1 Manufacture solution verification

Given the manufacture solution

$$u(x,t) = tx\left(\frac{1}{2} - \frac{x}{3}\right)$$

and the nonlinear term

$$\alpha(u) = 1 + u^2$$

the following figures gives the FEniCs solution and the the one given in (9).

Figure 1: FEnics implemetation at $t = 2$

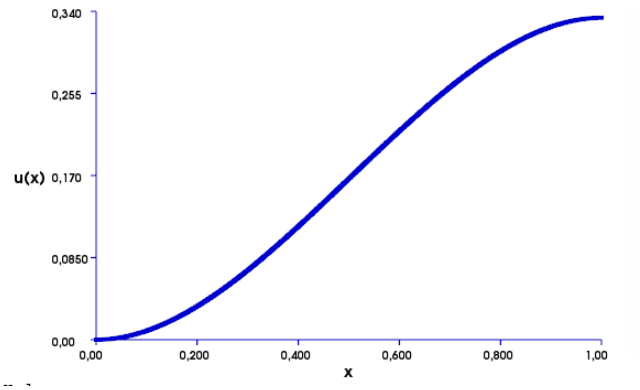
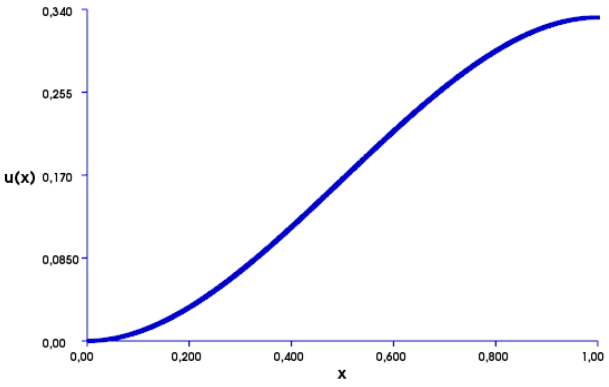


Figure 2: manufacture solution $t = 2$



0.5.2 FENiCs Errors

0.5.3 Nonlinear verification

The nonlinear FEniCs implementation is verified and the convergence rate of 1 is computed. See file name **nonLinearVerification.py**

0.6 Simulation of the nonlinear Gaussian function

The nonlinear Gaussian function

$$I(x,y) = \exp\left(-\frac{1}{2\sigma^2}(x^2+y^2)\right) \quad (x,y) \in [0,1]\times[0,1] \tag{9}$$

simulation is given by the following figures

Figure 3: Nonlinear Gaussian function at $t = 1$

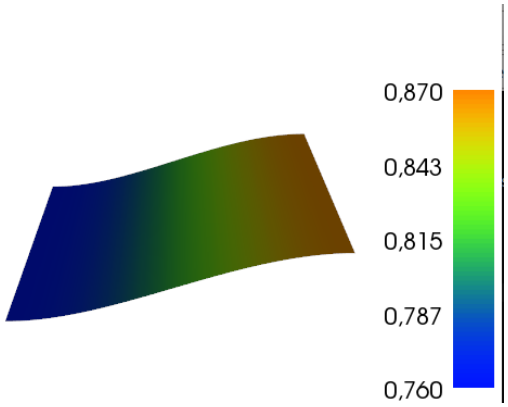


Figure 4: Nonlinear Gaussian function at $t = 2$

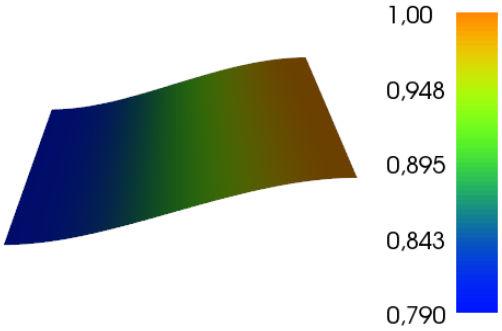


Figure 5: Nonlinear Gaussian function at $t = 3$