The second obligatory project, in INF5270

Soran Hussein Mohmmed / soranhm

November 2018

Nonlinear diffusion equation

The goal of this project is to discuss various numerical aspects of a nonlinear diffusion model:

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

$$u(\bar{x}, 0) = I(\bar{x}), \quad \frac{\partial u}{\partial n} = 0$$

ρ is constant and $\alpha(u)$ is a known function of u

Backward Euler

Backward Euler discretization (finite differencing) in the time direction gives the implicit scheme:

$$[\rho D_t u = \nabla * (\alpha(u)\nabla u) + f(\bar{x}, t)]^n$$

Written out:

$$\rho \frac{u^n - u^{n-1}}{\Delta t} = \nabla * (\alpha(u^n)\Delta u^n) + f^n$$

Sovling for u^{n-1}:

$$u^{n-1} = u^n - \frac{\Delta t}{\rho}(\nabla * (\alpha(u^n)\nabla u^n) + \frac{\Delta}{\rho} f^n \tag{2}$$

Denote u for u^n and u^{-1} for u^{n-1} and have the residual $\int_\Omega R(\bar{x})v = 0, \forall v \in V$. Rewriting, integrating and multiplying (2) by v, gives:

$$\int_\Omega [u - \frac{\Delta t}{\rho}(\nabla * (\alpha(u)\nabla u) + \frac{\Delta t}{\rho} f^n - u^{(1)}] v d\Omega$$

using integrasion by part on the second term:

$$\int_\Omega \frac{\Delta t}{\rho}(\nabla * (\alpha(u)\nabla u) = \frac{\Delta t}{\rho}[-\int_\Omega \alpha(u)\nabla u * \nabla v + \int_{\partial\Omega} \frac{\partial u}{\partial n} * v ds]$$

With the Nueman boundary condition ($\frac{\partial u}{\partial n} = 0$) second terms becomes zero. Then we have the variational formula:

$$\int_\Omega [uv + \frac{\Delta t}{\rho}\alpha(u)\nabla u * \nabla v - \frac{\Delta t}{\rho} f^n v - u^{(1)} v] d\Omega = 0$$

with the spatioal problem:

$$u = \sum_j c_j \psi_j, \quad v = \psi_j$$

Picard iteration

Rewriting the variational form as:

$$F = \int_\Omega [uv + \frac{\Delta t}{\rho}\alpha(u)\nabla u * \nabla v - \frac{\Delta t}{\rho} f^n v - u^{(1)} v] d\Omega = 0$$

Picard iteration Rewriting u as the unknown, we want to solve and u^- as the most recently computed value. Moreover u^- denotes the unknown function at the previous time level, u^{m-1}. At the beginning: $u^- = u^{(1)}$ and after each iteration, u^- is updated to u. Using the most recently computed u function in the $\alpha(u)$ coefficient:

$$F_i = \bar{F}_i = \int_\Omega [uv + \frac{\Delta t}{\rho}\alpha(u^-)\nabla u * \nabla v - \frac{\Delta t}{\rho} f^n v - u^{(1)} v] d\Omega = 0$$

This equation ($\bar{F}_i = 0$) are now linear and a system: $\sum_{j\in I_s} A_{ij} c_j = b_i, i \in I_s$. Need to solve the unknown c_j, we have $u = \sum_j c_j \psi_j$ and $v = \psi_j$:

$$\int_\Omega [uv + \frac{\Delta t}{\rho}\alpha(u^-)\nabla u * \nabla v] d\Omega = \int_\Omega [\frac{\Delta t}{\rho} f^n v - u^{(1)} v] d\Omega \tag{3}$$

This gives:

$$A_{ij} = \int_\Omega [uv + \frac{\Delta t}{\rho}\alpha(u^-)\nabla u * \nabla v] d\Omega$$
$$b_i = \int_\Omega [\frac{\Delta t}{\rho} f^n v + u^{(1)} v] d\Omega$$

After seting $u^- = u^{(1)} = u^k$ for iteration k, and reapet to seek a new improved soltuion untill u^{k+1} such that $u^{k+1} \longrightarrow u$ as $k \longrightarrow \infty$.

Restricted Picard iteration

Restricted the Picard iteration to a single iteration: $(\alpha(u) \rightarrow \alpha(u^{(1)}))$, we have:

$$\int_\Omega [uv + \frac{\Delta t}{\rho}\alpha(u^{(1)})\nabla u * \nabla v]d\Omega = \int_\Omega [\frac{\Delta t}{\rho} f^n v + u^{(1)}v]d\Omega$$

$$\int_\Omega a(u,v)dx = \int_\Omega L(v)dx$$

Writing it in python, gives:
u = TrialFunction(V)
v = TestFunction(V)
a = (u*v + (dt/rho)*inner(alpha(u_1)*nabla_grad(u),nabla_grad(v)))*dx
L = (u_1 + (dt/rho)*f)*v*dx

Constant solution

choosing ρ, α, f and I such that $u(\bar{x}, t) = C$, (C - constant). Inserting that $u(\bar{x}, t) = C$ in the original equation (1):

$$\rho\frac{\partial u}{\partial t} = \nabla * (\alpha(u)\nabla u) + f(\bar{x}, t)$$

$$\rho\frac{\partial C}{\partial t} = \nabla * (\alpha(C)\nabla C) + f(\bar{x}, t)$$

$$f(\bar{x}, t) = 0$$

The equation above gives f = 0 and that we can choos any function for $\alpha(u)$ and any constant ρ. At the initial condition we have C = $u(\bar{x}, 0) = I(\bar{x}) \rightarrow I(\bar{x}) = C$, this gives that both $u(\bar{x}, 0)$ and $I(\bar{x})$ are equal to C. Now we have $f(\bar{x}, t) = 0, I(\bar{x}))C, u(\bar{x}, t) = C$. ρ can be any constant and $\alpha(u)$ can be any function.

FEniCS implementasion

I have implementet d),e) and f) in the python file 'nonldiff.py'

Nummerical error

Some of the nummerical error is:
The discretization will give a error, since it is non-linear to linear problem the error (O(Δt) and second order error in x,y,z axis). Error will occure when $\alpha(u)$ is not constant. Since we turn it into a single Picard iteration it will occure a time error. The programming may also make a error