Final project(nonlinear diffusion equation)

Wei Zhang

December 17, 2012

a)

We use Backward Euler approximation in time

$$u^l = u^{l-1} + \frac{\Delta t}{\rho}(\nabla \cdot \alpha(u^l)\nabla u^l + f^l),$$

$$u^0 = I(x)$$

$$u^l \approx \hat{u}^l = \sum_1^M u_j^l N_j(x)$$

$$\int_\Omega \left(\hat{u}^l - \hat{u}^{l-1} - \frac{\Delta t}{\rho}(\nabla \cdot \alpha(\hat{u}^l)\nabla \hat{u}^l + f^l \right) N_i d\Omega = 0,$$

$$\int_\Omega \left(\hat{u}^l N_i - \frac{\Delta t}{\rho}[N_i \alpha(\hat{u}^l)\nabla \hat{u}^l]_0^L + \frac{\Delta t}{\rho}(\nabla N_i \cdot \alpha(\hat{u}^l)\nabla \hat{u}^l) \right) d\Omega = \int (\hat{u}^{l-1} + f^l \frac{\Delta t}{\rho}) N_i d\Omega,$$

weak form:

$$\sum_{j=1}^M (\int_\Omega N_i N_j d\Omega) u_j^0 = \int_\Omega N_i I(x) d\Omega, \tag{1}$$

$$\int_\Omega \left(\hat{u}^l N_i + \frac{\Delta t}{\rho}(\nabla N_i \cdot \alpha(\hat{u}^l)\nabla \hat{u}^l) \right) d\Omega = \int_\Omega (\hat{u}^{l-1} + f^l \frac{\Delta t}{\rho}) N_i d\Omega, \tag{2}$$

b)

The Picard iteration can be used directly at the PDE level. For our problem, we get

$$\rho u_t^{k+1} = \nabla \cdot \alpha(u^k)\nabla u^{k+1} + f^l \tag{3}$$

$$u^{l,k+1} = u^{l-1} + \frac{\Delta t}{\rho}(\nabla \cdot \alpha(u^{l,k})\nabla u^{l,k+1} + f^l) \tag{4}$$

weak form of (4)

$$\int_\Omega u^{l,k+1} v d\Omega = \int_\Omega u^{l-1} v d\Omega - \frac{\Delta t}{\rho}\int_\Omega \nabla v \cdot \alpha(u^{l,k})\nabla u^{l,k+1} d\Omega + \frac{\Delta t}{\rho}\int_\Omega f^l v d\Omega \quad (5)$$

The iterations stop when $\|u^{l,k+1} - u^{l,k}\| < tol$. We use the last know value u^{l-1} as the initial guess.

c)

$$u^{l,k+1} = u^{l-1} + \frac{\Delta t}{\rho}(\nabla \cdot \alpha(u^{l-1})\nabla u^{l,k} + f^l) \quad (6)$$

$$\int_\Omega \left(u^{l,k+1} - u^{l-1})v + \frac{\Delta t}{\rho}\nabla v \cdot \alpha(u^{l-1})\nabla u^{l,k} dx - \frac{\Delta t}{\rho} f^l v \right) d\Omega = 0 \quad (7)$$

d)

We choose $\alpha(u) = 1, f = 0$, $I(x, y) = \cos(\pi x), \rho = 1$, the weak form

$$\int_\Omega \left(u^{l,k+1} - u^{l-1})v + \frac{\Delta t}{\rho}\nabla v \cdot \nabla u^{l,k} \right) d\Omega = 0 \quad (8)$$

We choose different mesh in space and time simultaneously. We choose the fixed point of time $T = 0.5$ and find out that $E/h = 0.176$. See `d.py`.

e)

We use the method of *manufactured solutions* to verify the implementation. we choose

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

$$f(x,t) = -\rho\frac{x^3}{3} + \rho\frac{x^2}{2} + 8t^3\frac{x^7}{9} + 7t^3\frac{x^5}{2} - 5t^3\frac{x^4}{4} + 2tx - t$$

I choose $t = 0.5, \Delta t = 0.005$ and $t = 1, \Delta t = 0.01$, 41 points which are distributed in [0,1] evenly, P1 element. The exact solution and error are

```
t= 0.5
dt= 0.005
exact solution
[ 0.          0.00015365  0.00060417  0.00133594  0.00233333  0.00358073
  0.0050625   0.00676302  0.00866667  0.01075781  0.01302083  0.0154401
  0.018       0.0206849   0.02347917  0.02636719  0.02933333  0.03236198
```

```
  0.0354375   0.03854427  0.04166667  0.04478906  0.04789583  0.05097135
  0.054       0.05696615  0.05985417  0.06264844  0.06533333  0.06789323
  0.0703125   0.07257552  0.07466667  0.07657031  0.07827083  0.0797526
  0.081       0.0819974   0.08272917  0.08317969  0.08333333]

u_e-u
[ -2.42151158e-06  -2.42311539e-06  -2.42795714e-06  -2.43612316e-06
  -2.44775318e-06  -2.46304747e-06  -2.48228008e-06  -2.50581437e-06
  -2.53411758e-06  -2.56777202e-06  -2.60748098e-06  -2.65406784e-06
  -2.70846770e-06  -2.77171086e-06  -2.84489845e-06  -2.92917023e-06
  -3.02566552e-06  -3.13547809e-06  -3.25960631e-06  -3.39889989e-06
  -3.55400492e-06  -3.72530880e-06  -3.91288690e-06  -4.11645272e-06
  -4.33531338e-06  -4.56833216e-06  -4.81389976e-06  -5.06991580e-06
  -5.33378188e-06  -5.60240748e-06  -5.87222947e-06  -6.13924600e-06
  -6.39906501e-06  -6.64696755e-06  -6.87798540e-06  -7.08699229e-06
  -7.26880767e-06  -7.41831116e-06  -7.53056579e-06  -7.60094708e-06
  -7.62527485e-06]


t= 1
dt= 0.01
exact solution
[ 0.          0.00030729  0.00120833  0.00267187  0.00466667  0.00716146
  0.010125    0.01352604  0.01733333  0.02151563  0.02604167  0.03088021
  0.036       0.04136979  0.04695833  0.05273438  0.05866667  0.06472396
  0.070875    0.07708854  0.08333333  0.08957813  0.09579167  0.10194271
  0.108       0.11393229  0.11970833  0.12529688  0.13066667  0.13578646
  0.140625    0.14515104  0.14933333  0.15314063  0.15654167  0.15950521
  0.162       0.16399479  0.16545833  0.16635938  0.16666667]

u_e-u
[ -9.82241728e-07  -9.81071906e-07  -9.77821137e-07  -9.73234463e-07
  -9.68532826e-07  -9.65481095e-07  -9.66504037e-07  -9.74819260e-07
  -9.94561581e-07  -1.03087836e-06  -1.08998005e-06  -1.17913478e-06
  -1.30659971e-06  -1.48148608e-06  -1.71355817e-06  -2.01296980e-06
  -2.38994485e-06  -2.85441088e-06  -3.41559696e-06  -4.08160861e-06
  -4.85899403e-06  -5.75231646e-06  -6.76374802e-06  -7.89270003e-06
  -9.13550451e-06  -1.04851603e-05  -1.19311560e-05  -1.34593810e-05
  -1.50521324e-05  -1.66882250e-05  -1.83432096e-05  -1.99897006e-05
  -2.15978133e-05  -2.31357090e-05  -2.45702424e-05  -2.58677036e-05
  -2.69946456e-05  -2.79187847e-05  -2.86099575e-05  -2.90411165e-05
  -2.91893422e-05]
```

See `e.py`.

f)

There are two main sources of numerical errors in the program. The first is *input error*. The second is *roundoff error*. The *input error* comes from prior computations. In other words, the value u of previous time step will affect their following iterations. The *roundoff error* floating-point operations during the algorithm.

g)

This problem is similar with the problem (e). The difference is the expression f. But how can we choose $\alpha(u)$?

h)

We simulate the nonlinear diffusion of Gaussian function, the initial function

$$I(x,y) = exp(-\frac{1}{2\sigma^2}(x^2+y^2)) \tag{9}$$

See `h.py`.

i)

In the group finite element method, a nonlinear function f is represented like

$$f(\hat{u}) \approx \sum_{j=1}^{M} f(u_j)N_j \tag{10}$$

for the heat equation, the equation number i

$$\int_\Omega \left(\hat{u}^l N_i + \frac{\Delta t}{\rho}(\nabla N_i \cdot (\sum_s \alpha(u_s^l)N_s)\nabla \hat{u}^l)\right) d\Omega = \int_\Omega (\hat{u}^{l-1} + f^l\frac{\Delta t}{\rho})N_i d\Omega,$$

j)

From the weak form, we get a system of equations at each time level:

$$F_i(u_1^l, \cdots, u_M^l) = 0, \quad i = 1, \ldots, M,$$

where

$$F_i = \int_\Omega \left((\hat{u}^l - \hat{u}^{l-1} - f^l\frac{\Delta t}{\rho})N_i + \frac{\Delta t}{\rho}(\nabla N_i \cdot \alpha(\hat{u}^l)\nabla \hat{u}^l)\right) d\Omega \tag{11}$$

The equation to be solved in each iteration

$$\sum_{j=1}^{M} J_{i,j}\delta u_j^{k+1} = -F_i \tag{12}$$

$$J_{i,j} = \frac{\partial F_i}{\partial u_j^l} = \int_\Omega \left[N_i N_j + \frac{\Delta t}{\rho}(\nabla N_i \cdot \frac{d\alpha(\hat{u}^l)}{d\hat{u}^l}N_j \nabla \hat{u}^l + \nabla N_i \cdot \alpha(\hat{u}^l)\nabla N_j)\right] d\Omega \tag{13}$$

k)

$$F_i = \sum_{j=1}^{M}(\int_\Omega (N_i N_j dx)u_j^l - \sum_{j=1}^{M}(\int_\Omega (N_i N_j dx)u_j^{l-1} - \int_\Omega f^l \Delta t N_i dx + \frac{\Delta t}{\rho}\sum_{j=1}^{M}\int_\Omega (\alpha(\hat{u}^l)N_i' N_j' dx)u_j^l d\Omega \tag{14}$$

We use p1 element and element-by-element formulation

$$\bar{M}_{i,j}^{(e)} = \int_\Omega N_i N_j dx = \frac{h}{6}\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

$$\bar{K}_{i,j}^{(e)} = \int_\Omega N_i' N_j' dx = \frac{1}{h}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

we get

$$\begin{aligned} F_i = \frac{h}{6}u_{i-1}^l + \frac{4h}{6}u_i^l + \frac{h}{6}u_{i+1}^l - (\frac{h}{6}u_{i-1}^{l-1} + \frac{4h}{6}u_i^{l-1} + \frac{h}{6}u_{i+1}^{l-1}) \\ + \Delta t f_i^l + \frac{\Delta t}{\rho h}[\alpha_{i+\frac{1}{2}}(u_{i+1} - u_i) - \alpha_{i-\frac{1}{2}}(u_i - u_{i-1})] \end{aligned} \tag{15}$$

$$\alpha_{i+\frac{1}{2}} = \frac{1}{2}(\alpha(u_{i+1}) + \alpha(u_i)), \quad \alpha_{i-\frac{1}{2}} = \frac{1}{2}(\alpha(u_{i-1}) + \alpha(u_i)) \tag{16}$$

i)

We use finite difference scheme

$$\rho\frac{u_i^l - u_i^{l-1}}{\Delta t} = \frac{1}{h^2}[(\alpha_{i+\frac{1}{2}})(u_{i+1}^l - u_i^l) - (\alpha_{i-\frac{1}{2}})(u_i^l - u_{i-1}^l)] + f_i^l \tag{17}$$

We can see that in finite element method, the time derivative leads to a tridiagonal matrix. We can use the *Lumped Mass Matrix method* to make it diagonal.