Project 4, FYS 3150 / 4150, fall 2013

Odd Petter Sand and Nathalie Bonatout

November 1, 2013

All our source code can be found at our GitHub repository for this project: https://github.com/NathalieB/Project4/

1 Introduction

x

2 Theory and Technicalities

2.1 Closed form solution

We subtitute

$$v(x,t) = u(x,t) - u_s(x) = u(x,t) + x - 1$$

where $u_s(x) = 1 - x$ is the steady state solution that satisfies our boundary and initial conditions. We then set up the diffusion equation for $v(x,t)$:

$$\frac{\partial^2 v(x,t)}{\partial x^2} = \frac{\partial v(x,t)}{\partial t}$$

with known boundary conditions

$$v(0,t) = v(1,t) = 0 \qquad t \geqslant 0.$$

The initial condition $u(x,0) = 0$ then becomes

$$v(x,0) = u(x,0) - u_s(x) = 0 - (1-x) = x - 1 \qquad 0 < x < 1.$$

The solution of this equation is known from pp. 313-314 in the lecture notes, using $L = 1$:

$$v(x,t) = \sum_{n=1}^{\infty} A_n \sin(n\pi x) e^{-n^2\pi^2 t}.$$

We now find the Fourier series coefficients by partwise integration

$$A_n = 2\int_0^1 v(x,0)\sin(n\pi x)dx = 2\int_0^1 (x-1)\sin(n\pi x)dx$$

$$= 2\left(\left[-(x-1)\frac{1}{n\pi}\cos(n\pi x)\right]_0^1 - \int_0^1 -\frac{1}{n\pi}\cos(n\pi x)dx\right)$$

$$= \frac{2}{n\pi}\left([(1-x)\cos(n\pi x)]_0^1 + \left[\frac{1}{n\pi}\sin(n\pi x)\right]_0^1\right)$$

$$= \frac{2}{n\pi}(1+0) = \frac{2}{n\pi}$$

and finally, by substitution, our closed form solution is

$$u(x,t) = v(x,t) + u_s(x) = 1 - x + \sum_{n=1}^{\infty} \frac{2}{n\pi}\sin(n\pi x)e^{-n^2\pi^2 t}.$$

2.2 Algorithms

2.2.1 Explicit scheme

```
input: nSteps (# of interior points), time

deltaX = 1 / (nSteps + 1)
alpha = 0.5
deltaT = alpha * deltaX ^ 2
tSteps = 1 / deltaT

define v, vNext
```

```
for( i = 1 --> nSteps )
        x = i * deltaX
        v[i] = v(x, 0)
        vNext[i] = 0

for( t = 1 --> tSteps )
        for( i = 1 --> nSteps )
                vNext[i] = (1 - 2 * alpha) * v[i]
                if( i > 0 ) : vNext[i] += alpha * v[i-1]
                if( i < nSteps ) : vNext[i] += alpha * v[i+1]
        v = vNext

for( i = 1 --> nSteps )
        x = i * deltaX
        u[i] = 1 - x + v[i]

output: u
```

2.2.2 Implicit scheme

x

2.2.3 Crank-Nicholson scheme

x

2.3 Tridiagonal form of the implicit schemes

2.3.1 Implicit scheme

x

2.3.2 Crank-Nicholson scheme

x

2.4 Truncation errors and stability

x

3 Results and analysis

3.1 Explicit scheme

x

3.2 Implicit scheme

x

3.3 Cranck-Nicholson scheme

x

4 Conclusion

What we learned.

4.1 Critique

x