Inf2320 Chapter 7

4th November 2004

Exercises

7.1

In this model we think of the drop of ink as being $2\Delta x$ long when we inject it, and the time the injection itself takes is $2\Delta t$. This results in the source function as given. Assuming the tube is sealed at the ends, the PDE becomes

$$\begin{aligned}
\frac{\partial u}{\partial t} &= k\frac{\partial^2 u}{\partial x^2} + f(x,t), \\
u(x,0) &= 0, \\
\frac{\partial u}{\partial x} &= 0 \quad \text{for } x = 0, L.
\end{aligned}$$

The concentration of ink will increase with time. It will be strongest in the middle of the tube ($L/2$) and weaker towards the edges.

7.2

This temperature problem could e.g. represent a pipe buried vertically in the ground. In this case the lower end would be at constant temperature, while the top end would be exposed to a day-night cycle.

7.3

This could e.g. be a description of the flow of lubrication between two surfaces in a machine.

7.4

Replace the physical variables u, t, and x with the dimensionless variables $\bar{u}$, $\bar{t}$ and $\bar{x}$.

$$\bar{u} = \frac{u}{c_0}, \;\; t_c = \frac{L^2}{k}, \;\; \bar{t} = \frac{t}{t_c}, \;\; \bar{x} = \frac{x}{L}$$

Note that

$$\frac{\partial u}{\partial t} = \frac{\partial u}{\partial \bar{t}}\frac{\partial \bar{t}}{\partial t} = \frac{\partial \bar{t}}{\partial t}\frac{\partial u}{\partial \bar{t}} = \frac{1}{t_c}\frac{\partial}{\partial \bar{t}}\bar{u}c_0 = \frac{c_0}{t_c}\frac{\partial \bar{u}}{\partial \bar{t}},$$

and;

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial \bar{x}}{\partial x}\frac{\partial u}{\partial \bar{x}}\right) = \frac{\partial}{\partial \bar{x}}\frac{1}{L}\left(\frac{1}{L}\frac{\partial}{\partial x}\bar{u}c_0\right) = \frac{c_0}{L^2}\frac{\partial^2 \bar{u}}{\partial \bar{x}^2}.$$

The source term becomes

$$A\sin^2\left(\omega\bar{t}\frac{l^2}{k}\right)\exp(-L(\bar{x}-\bar{x_0})).$$

Dropping the bars, we can now write the PDE as

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{AL^2}{kc_0}\sin^2\left(\omega\bar{t}\frac{l^2}{k}\right)\exp(-L(\bar{x}-\bar{x_0})).$$

7.5

The discrete version of the Robin boundary conditions can be written as

$$\begin{aligned} \text{for } x=0: &\quad -\frac{u_2^l - u_0^l}{2\Delta x} = \alpha(u_1^l - u_s), \\ \text{for } x=1: &\quad \frac{u_{n+1}^l - u_{n-1}^l}{2\Delta x} = \alpha(u_n^l - u_s). \end{aligned}$$

These equations contain the values u_0^l and u_{n+1}^l, both of which are outside the grid. However, we can combine the equations above with the scheme (7.81) itself, which for ($x = 0$) reads:

$$u_1^{l+1} = u_1^l + \frac{\Delta t}{\Delta x^2}(u_0^1 - 2u_1^l + u_2^l) + \Delta t f_1^l$$

We now have two equations involving u_0^l and can eliminate it, resulting in the equation

$$u_1^{l+1} = u_1^l + 2\frac{\Delta t}{\Delta x^2}\left((\Delta x\alpha - 1)u_1^l - \Delta x\alpha u_s + u_2^l\right) + \Delta t f_1^l.$$

Using the same method for ($x = 1$) results in

$$u_n^{l+1} = u_n^l + 2\frac{\Delta t}{\Delta x^2}\left((\Delta x\alpha - 1)u_n^l - \Delta x\alpha u_s + u_{n-1}^l\right) + f_n^l.$$

7.6

See the program **ex76.py**.

7.7

$$\begin{aligned} u(x,t) &= e^{-\pi^2 t}\cos(\pi x) \\ \frac{\partial u}{\partial t} &= -\pi^2 e^{-\pi^2 t}\cos(\pi x) \\ \frac{\partial u}{\partial x} &= e^{-\pi^2 t}\pi(-\sin(\pi x)) \\ \frac{\partial^2 u}{\partial x^2} &= e^{-\pi^2 t}\pi\pi(-\cos(\pi x)) \end{aligned}$$

(7.91) satisfies $u_t = u_{xx}$.

$$\begin{aligned}\frac{\partial}{\partial x}u(0,t) &= e^{-\pi^2 t}\pi(-\sin(0)) = 0\\ \frac{\partial}{\partial x}u(1,t) &= e^{-\pi^2 t}\pi(-\sin(\pi)) = 0\end{aligned}$$

(7.91) satisfies Neumann boundary conditions. This solution is used in the program **ex76.py**.

7.8

In order to prove that $u(x,t)$ is a solution to the diffusion equation we simply observe that

$$\begin{aligned}\frac{\partial}{\partial t}u(x,t) &= -ae^{-at}\sin(b(x-c))\\ \frac{\partial^2}{\partial x^2}u(x,t) &= -b^2e^{-at}\sin(b(x-c))\end{aligned}$$

Note that $a = b^2$. The function u can be made to fit certain Dirichlet boundary conditions on the form $g_0(t) = \alpha e^{-\beta t}$ by choosing $a = \beta$, and c so that $\sin(-bc) = \alpha$. For $g_1(t) = \alpha e^{-\beta t}$ this instead becomes $\sin(b(1-c)) = \alpha$. Similarly, for Neumann conditions of the same form, c must be choosen so that $b\cos(-bc) = \alpha$, or (for $x = 1$) $b\cos(b(1-c)) = \alpha$. In order to verify Algorithm 7.4 we simply choose a set of values for a, b, and c, and run the algorithm with $u(x,0) = \sin(b(x-c))$, $g_0(t) = exp(-at)\sin(-bc)$, and $g_1(t) = exp(-at)\sin(b(1-)c)$. The results of the numerical calculation should now be the same as the equation $u(x,t)$.

7.9

An exact solution of the problem is

$$u(x,t) = e^{-\frac{\pi^2}{4}t}\sin(\frac{\pi}{2}(x-1)),$$

when the initial values are set to be $I(x) = \sin\frac{\pi}{2}(x-1))$.

7.10

Look at the derivatives of $\hat{u}$:

$$\begin{aligned}\hat{u}(x,t) &= av(x,t) + bw(x,t)\\ \frac{\partial\hat{u}}{\partial t} &= a\frac{\partial v}{\partial t} + b\frac{\partial w}{\partial t}\\ \frac{\partial^2\hat{u}}{\partial x^2} &= a\frac{\partial^2 v}{\partial x^2} + b\frac{\partial w^2}{\partial x^2}\end{aligned}$$

As both v and w solves the diffusion PDE we know that $v_t = v_{xx}$ and $w_t = w_{xx}$, and so

$$\hat{u}_t = av_t + bw_t = av_{xx} + bw_{xx} = \hat{u}_{xx}.$$

The boundary and initial conditions for $\hat{u}$ are simply

$$\begin{aligned}
\hat{u}(0,t) &= av(0,t) + bw(0,t),\\
\hat{u}(1,t) &= av(1,t) + bw(1,t),\\
\hat{u}(x,0) &= av(x,0) + bw(x,0).
\end{aligned}$$

Now let us look at the function $u(x,t;k)$

$$\begin{aligned}
u(x,t;k) &= e^{-\pi^2k^2t}\sin(\pi kx),\\
\frac{\partial u}{\partial t} &= -\pi^2k^2e^{-\pi^2k^2t}\sin(\pi kx),\\
\frac{\partial^2 u}{\partial x^2} &= e^{-\pi^2k^2t}\pi k\pi k(-\sin(\pi kx)).
\end{aligned}$$

We see that $u_t = u_{xx}$ and so u is also a solution of the diffusion PDE without source terms. The boundary and initial conditions for $\hat{u}(x,t) = u(x,t;1) + u(x,t;100)$ are

$$\begin{aligned}
\hat{u}(0,t) &= 0,\\
\hat{u}(1,t) &= 0,\\
\hat{u}(x,0) &= \sin(\pi x) + 0.1\sin(100\pi x).
\end{aligned}$$

The time $t = T$ when there is no visible track of $w(x,t)$ will depend on the program used for plotting.

7.11

The finite difference scheme is

$$\frac{u_i^{l+1} - u_i^l}{\Delta t} = \frac{u_{i-1}^l - 2u_i^l + u_{i+1}^l}{\Delta x^2}.$$

Inserting (7.93) results in

$$\frac{A\left[\left(1-\frac{4\Delta t}{\Delta x^2}\sin^2\left(\frac{\pi k\Delta x}{2}\right)\right)^{l+1} - \left(1-\frac{4\Delta t}{\Delta x^2}\sin^2\left(\frac{\pi k\Delta x}{2}\right)\right)^{l}\right]\sin(\pi k(i-1)\Delta x)}{\Delta t} =$$
$$\frac{A\left(1-\frac{4\Delta t}{\Delta x^2}\sin^2\left(\frac{\pi k\Delta x}{2}\right)\right)^{l}\left[\sin(\pi k(i-2)\Delta x) - 2\sin(\pi k(i-1)\Delta x) + \sin(\pi ki\Delta x)\right]}{\Delta x^2},$$

which can be reduced to

$$\frac{\left[\left(1-\frac{4\Delta t}{\delta x^2}\sin^2\left(\frac{\pi k\Delta x}{2}\right)\right) - 1\right]\sin(\pi k(i-1)\Delta x)}{\Delta t} =$$
$$\frac{\left[\sin(\pi k(i-2)\Delta x) - 2\sin(\pi k(i-1)\Delta x) + \sin(\pi ki\Delta x)\right]}{\Delta x^2}.$$

This, using a few trigonometric formulas, becomes

$$\begin{aligned}&2\cos^2(\pi k\Delta x)\left[\sin(\pi k i\Delta x)\cos(\pi k\Delta x)-\cos(\pi k i\Delta x)\sin(\pi k\Delta x)\right]=\\&\quad\sin(\pi k i\Delta x)\cos(\pi k 2\Delta x)-\cos(\pi k i\Delta x)\sin(\pi k 2\Delta x)+\sin(\pi k i\Delta x),\end{aligned}$$

which in turn becomes

$$\cos^2(\pi k\Delta x)=1-\sin^2(\pi k\Delta x).$$

The boundary conditions are

$$u_1^l=A\left(1-\frac{4\Delta t}{\delta x^2}\sin^2\left(\frac{\pi k\Delta x}{2}\right)\right)^l\sin(\pi k(1-1)\Delta x)=0,$$

and

$$u_n^l=A\left(1-\frac{4\Delta t}{\delta x^2}\sin^2\left(\frac{\pi k\Delta x}{2}\right)\right)^l\sin(\pi k)=0.$$

The initial condition is

$$u_i^0=A\sin(\pi k(i-1)\Delta x).$$

7.12

Taylor expansion of e^y and $\sin^2 y$:

$$\begin{aligned}e^y&=1+y+\frac{y^2}{2}+\frac{y^3}{6}+O(y^4)\\\sin^2 y&=y^2+\frac{y^4}{3}+O(y^6)\end{aligned}$$

Now we replace the relevant parts of the function for $e(k,\Delta x,\Delta t)$ with the two first terms in their power series expansion:

$$\begin{aligned}e(k,\Delta x,\Delta t)&=e^{-\pi^2k^2\Delta t}-1+\frac{4\Delta t}{\Delta x^2}\sin^2\frac{\pi k\Delta x}{2}\\&\approx 1-\pi^2k^2\Delta t-1+\frac{4\Delta t}{\Delta x^2}\left(\frac{\pi^2k^2\Delta x^2}{4}+\frac{\pi^4k^4\Delta x^4}{48}\right)\\&=\frac{\Delta t\pi^4k^4\Delta x^2}{12}\end{aligned}$$

The leading term in the expansion of e is proportional to Δt and Δx^2.

7.13

In equation (7.93) u_i^l may increase with time if

$$\left|1-\frac{4\Delta t}{\Delta x^2}\sin^2\left(\frac{\pi k\Delta x}{2}\right)\right|$$

is greater than one. The only way for this to happen is to have

$$\frac{4\Delta t}{\Delta x^2}\sin^2\left(\frac{\pi k \Delta x}{2}\right) > 2,$$

which only happens when

$$\Delta t > \frac{1}{2}\Delta x^2.$$

In this case the numerical solution is unstable.

7.14

```
*Algorithm for solving (7.72)*

SET INITIAL CONDITIONS:
$u_i^0 = I(x_i)$, for $i = 1, \ldots, n$
for $l = 0, 1, \ldots, m$
    UPDATE ALL INNER POINTS
    $u_i^{l+1} = u_i^l + \frac{1}{\gamma_i}\frac{\Delta t}{\Delta x}\left(k_{i+\frac{1}{2}}\frac{u_{i+1}^l - u_i^l}{\Delta x} - k_{i-\frac{1}{2}}\frac{u_i^l - u_{i-1}^l}{\Delta x}\right) + \frac{\Delta t}{\gamma_i} f_i^l$
      for $i = 2, \ldots, n-1$
    INSERT BOUNDARY CONDITIONS
    $u_1^{l+1} = u_1^l + \frac{1}{\gamma_i}\frac{\Delta t(u_2^l - u_1^l)}{\Delta x^2}(k_{3/2} + k_{1/2}) + \frac{\Delta t}{\gamma_i} f_1^l$
    $u_n^{l+1} = u_n^l + \frac{1}{\gamma_i}\frac{\Delta t(u_{n-1}^l - u_n^l)}{\Delta x^2}(k_{n+1/2} + k_{n-1/2}) + \frac{\Delta t}{\gamma_i} f_n^l$
```

Projects

7.4.1; Diffusion of a jump

a)

$$\begin{aligned}
\varrho c_v \frac{\partial u}{\partial t} &= k\frac{\partial^2 u}{\partial x^2}, \\
-k\frac{\partial u}{\partial x} &= 0, \quad \text{for } x = 0, 2L, \\
u(0,t) &= \left\{\begin{array}{ll} U_1, & x \le L \\ U_2, & x > L \end{array}\right. .
\end{aligned}$$

(NOTE: It is assumed that the pieces are of the same length L and are made of the same metal.)

b) Scaling the problem, using

$$\bar{u} = \frac{u - U_a}{U_a - U_b},$$

and assuming $U_1 < U_2$ results in the equations

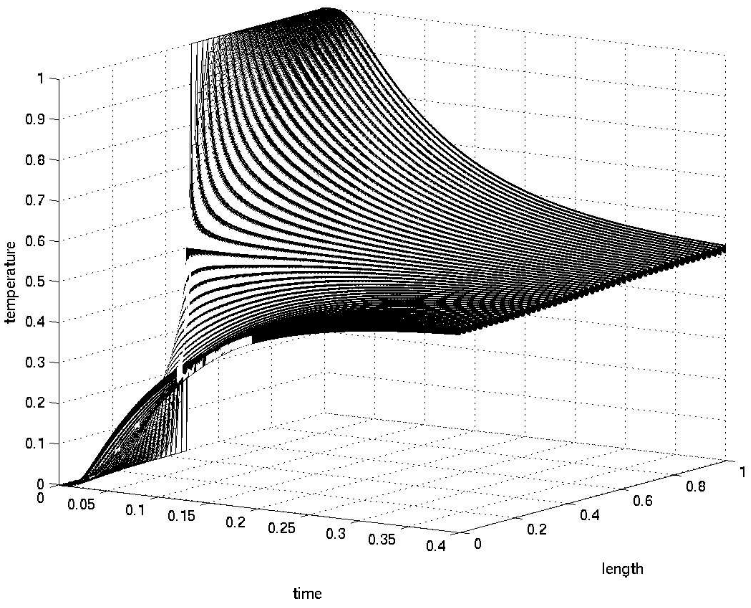

Figure 1: Project 7.1. Two pieces of metal with different temperature are brought together.

$$
\begin{aligned}
\frac{\partial u}{\partial t} &= \frac{\partial^2 u_s}{\partial x^2}, \\
\frac{\partial u}{\partial x} &= 0, \quad \text{for } x = 0, 1, \\
u(0,t) &= \begin{cases} 0, & x \leq 0.5 \\ 1, & x > 0.5 \end{cases} .
\end{aligned}
$$

c) See program **p71.m**, and Figure 1.

d) In order to show that

$$u(x,t) = e^{-\pi^2 t}\cos(\pi x)$$

is a solution, simply insert $u(x,t)$ in the equations for the scaled problem:

$$
\begin{aligned}
\frac{\partial u}{\partial t} &= -\pi^2 e^{-\pi^2 t}\cos(\pi x), \\
\frac{\partial^2 u}{\partial x^2} &= -\pi^2 e^{-\pi^2 t}\cos(\pi x), \\
\frac{\partial}{\partial x}u(0,t) &= -\pi e^{-\pi^2 t}\sin(0) = 0, \\
\frac{\partial}{\partial x}u(1,t) &= 0 - \pi e^{-\pi^2 t}\sin(\pi) = 0.
\end{aligned}
$$

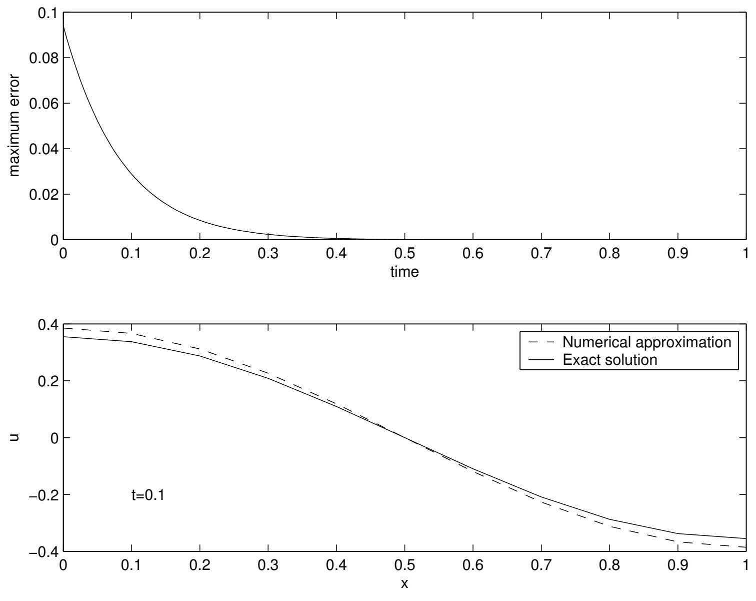

Figure 2: Project 7.1 d. Comparing the exact solution to the results of the numerical approximat, when the initial condition is $u(0,x) = \cos(\pi x)$.

Plotting both this exact solution and the approximation yielded by the numerical method (Figure 2) shows that the error in the numerical results decreases with time. This indicates that the code is free of bugs.

e) We know from experience (and thermodynamics) that if we place two items of different temperatures in contact, heat will flow from the warm item to the cold one. In the simple case of two equally big pieces of the same metal the temperature of both pieces will eventually become the average of the two starting temperatures. This is consistent with the result shown in Figure 1, where the temperature of both pieces becomes $T = 0.5$.

f) In this problem the solution will reach a stationary solution just after $T = 3$. The exact time will vary depending on Δt and Δx.

g) We have solved this problem (at least approximately) in exercise **f**, for the scaled time, $\bar{t}$. In order to find the time as a function of heat conduction and total lenght of the pieces, we simply use $t = \bar{t}t_c$ where $\bar{t} = 3$, and

$$t_c = \frac{1}{k}\varrho c_v 4L^2.$$

7.4.2; Periodical Injection of Pollution

a) The source term f can be specified as

$$f(x,t) = \begin{cases} K, \ r \leq r_0, \ i \leq t \leq i + 1/3, \ i = 1, 2, \ldots \\ 0, \ \text{otherwise} \end{cases}$$

where i is time in days, (eight hours is 1/3 day). This is implemented in the program **p72.m**.

b) Complete initial-boundary value problem:

$$\begin{aligned}
\frac{\partial}{\partial t}c(r,t) &= k\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}c(r,t)\right) + f(r,t),\\
\frac{\partial c}{\partial r} &= 0 \quad \text{for } r = 0,\\
c(t,r) &= 0 \quad \text{for } r = L,\\
c(0,r) &= 0.
\end{aligned}$$

c) Finite difference scheme for (7.97):

$$c_i^{l+1} = c_i^l + \frac{k}{r_i^2}\frac{\Delta t}{\Delta r^2}\left((r_i + \frac{\Delta r}{2})^2(c_{i+1} - c_i) - (r_i - \frac{\Delta r}{2})^2(c_i - c_{i-1})\right) + \Delta t f_i^l$$

d)

```
*Algorithm for solving (7.97)*

SET INITIAL CONDITIONS:
$c_i^0 = 0$, for $i = 1, \dots, n$
for $l = 0, 1, \dots, m$
    UPDATE ALL INNER POINTS
    $c_i^{l+1} = c_i^l + \frac{k}{r_i^2}\frac{\Delta t}{\Delta r^2}\left((r_i + \frac{\Delta r}{2})^2(c_{i+1} - c_i) - (r_i - \frac{\Delta r}{2})^2(c_i - c_{i-1})\right) + \Delta t f_i^l$
        for $i = 2, \dots, n-1$
    INSERT BOUNDARY CONDITIONS:
    $c_1^{l+1} = c_1^l + \frac{2\Delta t}{\Delta r^2}\left(c_2^l - c1^l\right) + \Delta t f_i^l$
    $c_n^{l+1} = 0$
```

e) See the program **p72.m**.

f) A very simple test of the implementation would be to set $K = 0$ and confirm that the concentration renmain zero throughout the simulation. For a more advanced test one could calculate the total pollution present at all times and compare this with the total pullution from the source term.

g) Scaling equation (7.97):

$$\frac{\partial}{\partial t}c = \frac{\partial}{\partial \bar{t}}\frac{\partial \bar{t}}{\partial t}\bar{c}c_0 = \frac{c_0}{t_c}\frac{\partial \bar{c}}{\partial \bar{t}}$$

$$k\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}c\right) = k\frac{1}{\bar{r}^2r_0^2}\frac{1}{r_0}\frac{\partial}{\partial \bar{r}}\left(\bar{r}^2r_0^2\frac{1}{r_0}\frac{\partial}{\partial \bar{r}}c_0\bar{c}\right) = k\frac{c_0}{\bar{r}^2r_0^2}\frac{\partial}{\partial \bar{r}}\left(\bar{r}^2\frac{\partial}{\partial \bar{r}}\bar{c}\right)$$

The complete equation now becomes

$$\frac{c_0}{t_c}\frac{\partial \bar{c}}{\partial \bar{t}} = k\frac{c_0}{\bar{r}^2r_0^2}\frac{\partial}{\partial \bar{r}}\left(\bar{r}^2\frac{\partial}{\partial \bar{r}}\bar{c}\right) + K\bar{f}.$$

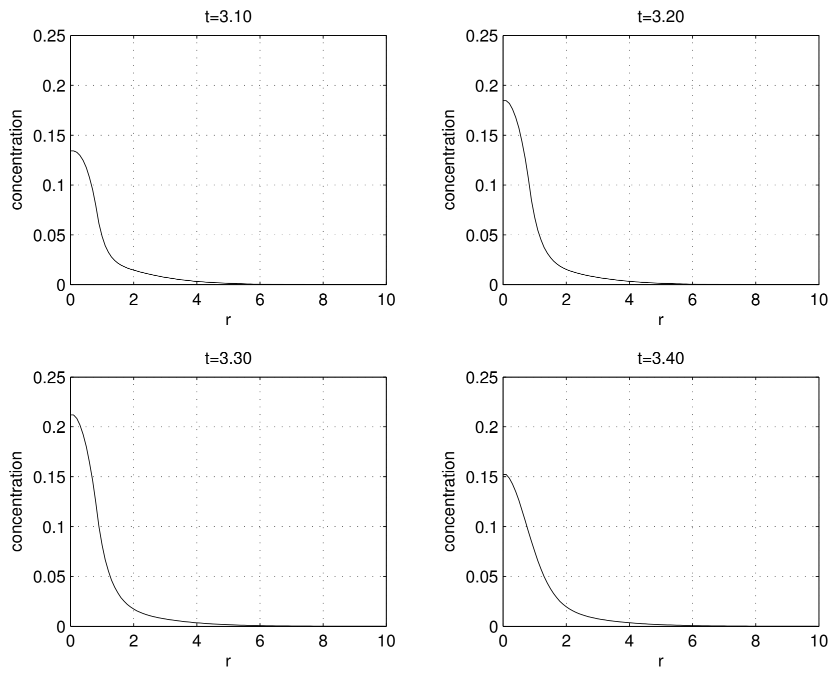

Figure 3: Project 7.2. Concentration of pollution by distance from the factory pipe. In the three first figures the factory is active, and pollution is being injected. In the last figure ($t = 3.40$) the factory is closed for the day.

Using $t_c = r_0^2/k$, and removing the bars this becomes

$$\frac{\partial c}{\partial t} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}c\right) + \alpha f,$$

where $\alpha = r_0K/kc_0$. If we choose t_cK as the scale for concentration α equals unity.

h) The ideal value for L will vary with the induvidual users accuracy requirements. We have used $L = 20r_0$.

i) The printed page is unsuited as a medium for animations, see instead figure 3 for a look at how the concentration developes in time.

7.4.3; Compare different scalings

a) See the program **p73.m**. By running this program with different scalings and values of β we see that scaling 1 fails when $\beta \gg 1$ and scale 2 fails when $\beta \ll 1$. (In this context failing means that the values for u stray far from the values between zero and one where we prefer to work.)

b) See figure 4.

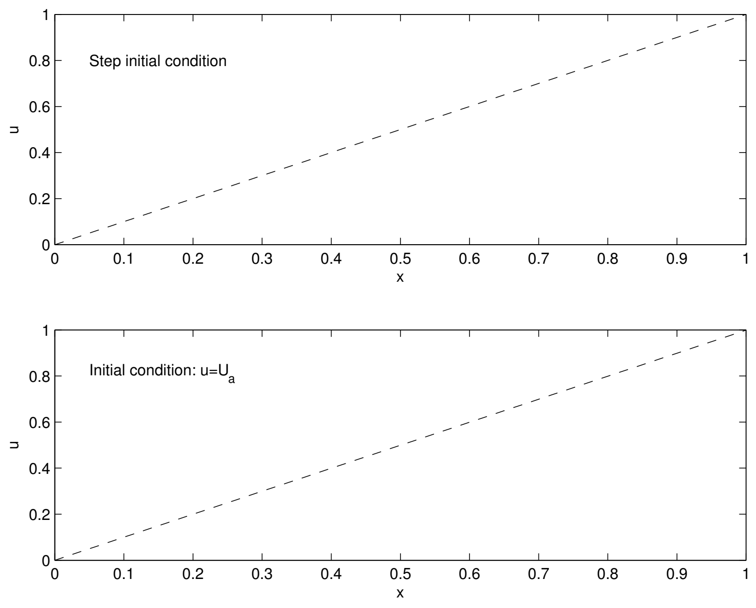

Figure 4: Project 7.4.3: The solutions after a certain time are the same, irrespective of initial conditions. ($\beta = 1$)

c) The *stationary solution*, u_s, obviously does not change with time, and we have determined in exercise **c** that the initial condition have no influence. The equations determining u_s are:

$$\begin{aligned} k\frac{\partial^2 u_s}{\partial x^2} &= 0 \\ u_s(a) &= U_a \\ k\frac{\partial u_s}{\partial x} &= Q_b \quad \text{for } x = 2L \end{aligned}$$

Scaled versions:
Scale 1, $\bar{u} = \frac{u-U_a}{U_b-U_a}$:

$$\begin{aligned} \frac{\partial^2 u_s}{\partial x^2} &= 0 \\ u_s(a) &= 0 \\ \frac{\partial u_s}{\partial x} &= \beta \quad \text{for } x = 1 \end{aligned}$$

where $\beta = \frac{Q_b L}{k(U_b-U_a)}$.
Scale 2, $\bar{u} =$:

$$\frac{\partial^2 u_s}{\partial x^2} = 0$$

$$\begin{aligned} u_s(a) &= U_a \\ \frac{\partial u_s}{\partial x} &= 1 \quad \text{for } x = 1 \end{aligned}$$

d) The time derivative of u_s is 0, and as u_s, which means that $\frac{\partial^2}{\partial x^2} u_s$ is zero. (i.e. u_s is a straight line.) We can find the stationary solution by integrating twice:

$$u_s'(x) = \int u_s''(x)\, dx = \int 0\, dx = C_1$$

In the case of scale 1 we know that $u''(x) = \beta$ at $x = 1$, which implies that $C_1 = \beta$.

$$u_s(x) = \int u_s'(x)\, dx = \int \beta\, dx = \beta x + C_2$$

We know that $u_s(x) = 0$ at $x = 0$, i.e. $C_2 = 0$. The stationary solution is simply $u_s(x) = \beta x$. If β is much greater than unity then so is u_s. For scaling 2 the stationary solution is simply $u_s(x) = x$.

Programs (`Python`, matlab)

ex76.py

```
#!/usr/bin/env python
# The diffusion equation with Neumann boundary conditions.
# Compares the results with the exact solution from exercise 7.7.

from Numeric import *

def diffeq(I, f, g0, g1, dx, dt, m, action=None):
   n = int(1/dx + 1)
   h = dt/(dx*dx)
   x = arrayrange(0, 1+dx/2, dx, Float)
   user_data= []

   um = I(x)
   u = zeros(n, Float)
   for l in range(m+1):
     t = l*dt
     # Update all inner points
     for i in range(1,n-1,1):
       u[i] = um[i] + h*(um[i-1] - 2*um[i] + um[i+1]) + dt*f(x[i], t)
     # Neumann boundary conditions:
     u[0] =  um[0] + (2*dt/(dx*dx))*(um[1] -um[0])
     u[n-1] = um[n-1] + (2*dt/(dx*dx))*(um[n-2] -um[n-1] )

     for i in range(len(u)): um[i] = u[i]
     if action is not None:
       r= action(u, x, t)
       if r is not None:
         user_data.append(r)

   return user_data

C = 1.2
def f0(x,t):  return 0.0
def IC_0(x):  return zeros(len(x), Float) +C
def IC_1(x):  return cos(pi*x)
def g0_0(t):  return 0.0
def g1_0(t):  return 0.0

def const_u(u, x, t):
   e=sum(u- exp(-pi*pi*t)*cos(pi*x))
   return e

dx = 0.1; dt = dx*dx/2.0; m = int(0.5/dt)
e = diffeq(IC_1, f0, g0_0, g1_0, dx,dt, m, action=const_u)
print "errors at time levels: ", e
```

p71.m

```
function  p71(Dx,Dt,ts)
% Solves the scaled problem from exercise 7.4.1c.
% Will stop either when there is no more change
% in the temperature or at ts.
% example: p71(0.02,0.0002,0.2)

     x=0:Dx:1;
     t=0:Dt:ts;

alpha=Dt/(Dx*Dx)
u=zeros(length(x),length(t));

% initial condition:
i=fix(length(x)/2)+1
u(i:length(x),1)=1;
% For exercise d, instead use:
% u(:,1) =cos(pi.*x)';

   l= 1; k=1;
   while (any(k) ~= 0 & l< length(t) )
     for i=2:(length(x)-1)
       u(i,l+1)=u(i,l) + alpha*( u(i-1,l)-2*u(i,l) +u(i+1,l));
     end
% boundary conditions:
   u(1,l+1)=u(1,l) + 2*alpha*( u(2,l)- u(1,l) );
   u(length(x),l+1)=u(length(x),l) + 2*alpha*( u(length(x)-1,l) -u(length(x),l)
   l=l+1;
   k=u(:,l) - u(:,l-1)
end
```

```
% Plotting
mesh(t(1:l),x,u(:,[1:l]));
xlabel('time');
ylabel('length');
zlabel('temperature');
```

p72.m

```
function  p72(Dr,ts,L,K)
%
% Solves the scaled problem from project 7.2.
% Plots concentration by distance from factory pipe
% at time ts.
% example: p72(0.1,3,20,1)

Dt= Dr*Dr/4;
     r=0:Dr:L;
     t=0:Dt:ts;

% initial condition:
c=zeros(length(r),1);

% calculation:
     for j=1:lenght(t)
     for i=2:length(r)-1
     c(i)=c(i)+Dt*...
        ( (r(i)+Dr/2)^2*(c(i+1)-c(i))-(r(i)-Dr/2)^2*(c(i)-c(i-1)) )/(Dr*r(i))^2
     Dt*source(j*Dt,i*Dr,K);
     % boundary conditions:
     c(1)=c(2);
     c(length(r))=0;
     end
end

% Plotting
plot(r,c);
xlabel('r');
ylabel('concentration');

function s=source(t,r,K)
 if (rem(t,1) < 1/3 & r<1)
    s=K;
else
    s=0;
end
```

p73.m

```
function  p73(Dx,Dt,ts,s,beta)
```

```
%
% The heat conduction problem from project 7.4.3.
% Scaled in two different ways.
% Will stop either when there is no more change
% in the temperature or at ts.
% example: p73(0.02,0.0002,0.2,1,1)

     x=0:Dx:1;
     t=0:Dt:ts;

alpha=Dt/(Dx*Dx);
u=zeros(length(x),length(t));

if s==1
% Scaling 1
% initial condition:

i=fix(length(x)/2)+1;
u(i:length(x),1)=1;

% u(:,1) =cos(pi.*x)';
length(t)
   l= 1; k=1;
   while (k > 1e-6  & l< length(t) )
     for i=2:(length(x)-1)
       u(i,l+1)=u(i,l) + alpha*( u(i-1,l)-2*u(i,l) +u(i+1,l));
     end
% boundary conditions:
   u(1,l+1)=0;
   u(length(x),l+1)=u(length(x),l) + 2*alpha*( u(length(x)-1,l) -u(length(x),l)
   l=l+1;
   k = max(abs(u(:,l) - u(:,l-1)) );
end

plot(x,u(:,l),'r--');
xlabel('x'); ylabel('u');

elseif s==2
% Scaling 2
% initial condition:

i=fix(length(x)/2)+1;
u(i:length(x),1)=1/beta;

  l= 1; k=1;
   while (k > 1e-6 & l< length(t) )
     for i=2:(length(x)-1)
       u(i,l+1)=u(i,l) + alpha*( u(i-1,l)-2*u(i,l) +u(i+1,l));
     end
% boundary conditions:
   u(1,l+1)=0;
   u(length(x),l+1)=u(length(x),l) + 2*alpha*( u(length(x)-1,l) -u(length(x),l)
   l=l+1;
   k = max(abs(u(:,l) - u(:,l-1)) );
end

plot(x,u(:,l),'b:');
xlabel('x'); ylabel('u');

end
```