Assignment VI

M693B

DR. BLOMGREN

MAY 9, 2018

MATTEO POLIMENO

1 Problem 6.3.11 from Strikverda

In this problem we have to numerically solve the heat equation

$$u_t = u_{xx}.$$

Initial data is given to be

$$u(0,x) = \begin{cases} 1 - |x| & \text{if } |x| \leq \frac{1}{2} \\ \frac{1}{4} & \text{if } |x| = \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

The x-interval is given by $[-1, 1]$ while for the time t interval it holds $t \in [0, 0.5]$.

The PDE will be solved using the forward time and central space scheme and also using the Crank Nicolson scheme. For the former the value of $\mu = 0.4$ and $h = 1/10,\ 1/20,\ 1/40$. For the latter, in part b we have $\lambda = 1$, while in part c $\mu = 5$. And we have all those same values for h as well.

1.1 Part a

We first use the forward time central space scheme. Here is the code

```
clear all
tic
h = 1/20;
b = 1;
mu = .4;
tmax = .5;


k = mu*h^2;
t = 0:k:tmax;
qt = length(t);
x = -1:h:(1+h);
px = length(x);
R = 0:200;

for m = 1:px %initial conditions
    u(1,m) = u0(x(m));
end

wp = zeros(qt,px);

for ii = 1:qt %calculates exact solution
    for jj = 1:px
        S1 = ((((-1).^R)./(pi*(2*R+1))) + 2./((pi^2)*((2*R+1).^2)))...
```

```
            .*cos(pi*(2*R+1)*x(jj)).*(exp(1).^(-(pi^2)*((2*R+1).^2)*t(
                ii)));
        S2 = (cos(2*pi*(2*R+1)*x(jj))./((pi^2)*((2*R+1).^2)))...
            .*(exp(1).^(-4*(pi^2)*((2*R+1).^2)*t(ii)));
        wp(ii,jj) = (3/8) + sum(S1) + sum(S2);
    end
end


for i = 1:qt-1
    u(i,1) = wp(i,1);
    for j = 1:px-2
        u(i+1,j+1) = (1-2*b*mu)*u(i,j+1)+b*mu*(u(i,j+2)+u(i,j));
        u(i,px) = u(i,px-2);
    end

end

u(qt,1) = wp(qt,1);

time = toc


for i = 1:qt
    plot(x,u(i,:),'b-o',x,wp(i,:),'k');
    ylim([-0.1,1.5])
    xlim([-1,1])
    title(['Heat equation: \mu = .4, h = ' num2str(h) ', Time = '
        num2str(t(i))])
    xlabel('x')
    ylabel('u')
    grid on
    M(i) = getframe;
end


supnorm = max(abs(wp(qt,:)-u(qt,:)))

L2norm = norm(wp(qt,:)-u(qt,:))
```

and we plot our results

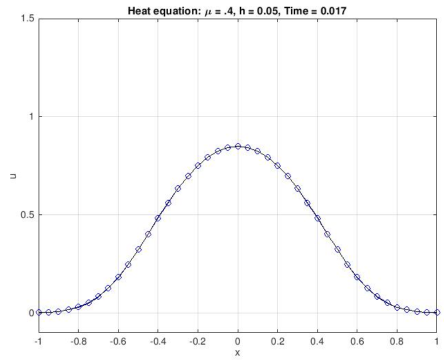

Figure 1: Analytical solution vs. scheme

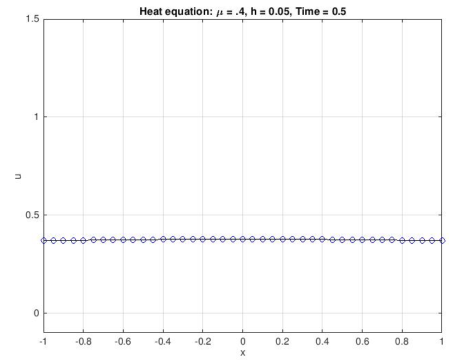

Figure 2: Analytical solution vs. scheme

Thus, we see that the scheme does an excellent job and it is very useful.

1.2 Part b and c

Here is the function that represents the analytical solution

```
function J = ustar(t,x,p,q) %elle emme
    J = 3/8;
    for jj = 0:p
        for ii = 0:q
            J = J + ((-1)^jj/(pi*(2*jj+1))+2/(pi^2*(2*jj+1)^2))*cos(pi*(2*jj+1)*x)*exp(-pi^2*(2*jj+1)^2)*t ...
            + (cos(2*pi*(2*ii+1)*x)/(pi^2*(2*ii+1)^2))*exp(-4*pi^2*(2*ii+1)^2)*t;
        end
    end

end
```

and here is the numerical scheme

```
clc
clear all

u0 = @(x) (abs(x) < 0.5)/(1-abs(x))+(abs(x) == 0.5)/4;
tmax = 1/2;
p = 100;
q = 100;
hvals = [1/10 1/20 1/40 1/80];

for lambda_switch = 1:2
    for h = hvals
```

```
    if lambda_switch == 1
        mu = 1/h; %lambda=mu*h and for lambda=1 we have mu=1/h
    else
        mu = 5;
    end

    k = mu*h^2;
    x = (-1:h:1)';
    m = length(x);
    b = ones(m,1);
    v = zeros(m,1);
    time = 0;

    coeff_matrx_n1 = [(-mu/2)*b (1+mu)*b (-mu/2)*b];
    coeff_matrx_n  = [(mu/2)*b (1-mu)*b (mu/2)*b];

    A = spdiags(coeff_matrx_n1,[-1 0 1], m,m); %make triadiagonal matrix %diagonals are at the -1 (meaning 1 down), main and 1 (meaning 1 up)
    B = spdiags(coeff_matrx_n, [-1 0 1], m,m);
    A(1,2) = 0;
    A(m,m-1) = 0;

    v(1) = ustar(0,1,p,q);
    for M = 2:m-1
        v(M) = u0(x(M)); %initial data
    end
    v(m) = ustar(0,m,p,q);

    while time < tmax
        time = time+k; %compute time

        A(1,1) = ((1-mu)*v(1)+mu/2*v(2))/ustar(time,1,p,q);
        A(m,m) = (mu/2*v(m-1)+(1-mu)*v(m))/ustar(time,m,p,q);
        v = A\(B*v);
    end

    uxsol = ustar(time,x,p,q);
    figure;
    hold on
    if lambda_switch == 1
        type = '\lambda=1';
    else
        type = '\mu = 5';
    end
    title(['h = ' num2str(h) ',' type]);
```

```
        plot(x,uxsol,'*',x,v,'x-');
        xlabel('x')
        ylabel('u')
        axis([-1 1 0 1])
        hold off;
        if lambda_switch == 1
            disp(['h:  ' num2str(h)]);
            disp(['Supremum Norm:  ' num2str(max(abs(uxsol-v)))]);
            disp(['L2 Norm:  ' num2str(norm(uxsol-v))]);
        end
    end
end
```

The scheme was run for the listed values of h and we plotted the results

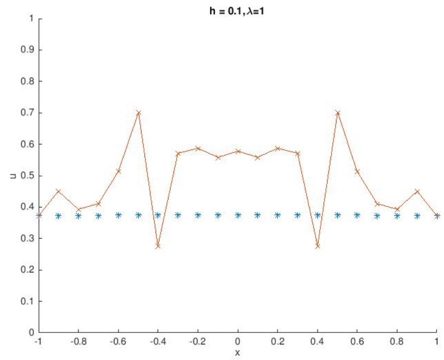

Figure 3: Analytical solution vs. scheme

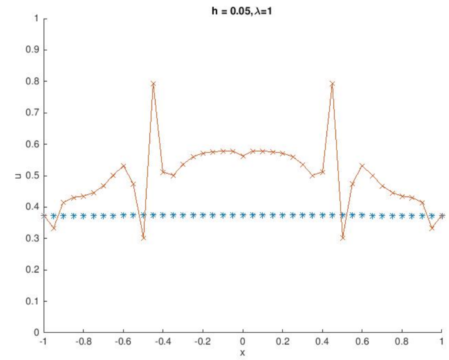

Figure 4: Analytical solution vs. scheme

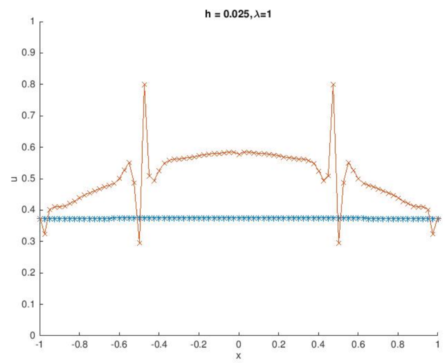

Figure 5: Analytical solution vs. scheme

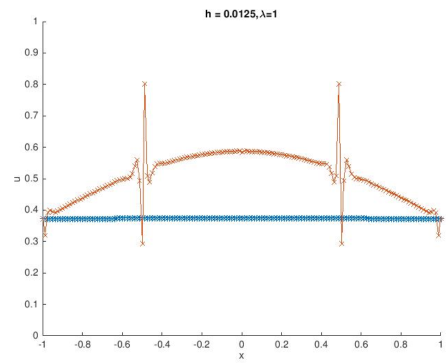

Figure 6: Analytical solution vs. scheme

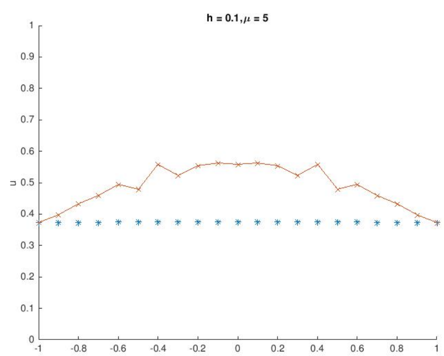

Figure 7: Analytical solution vs. scheme

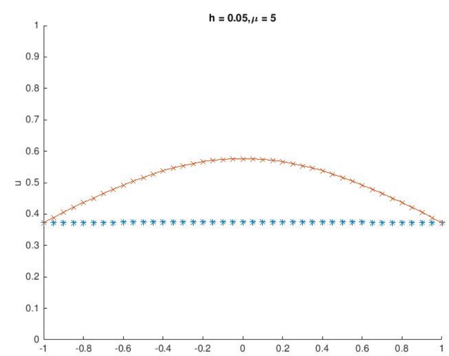

Figure 8: Analytical solution vs. scheme

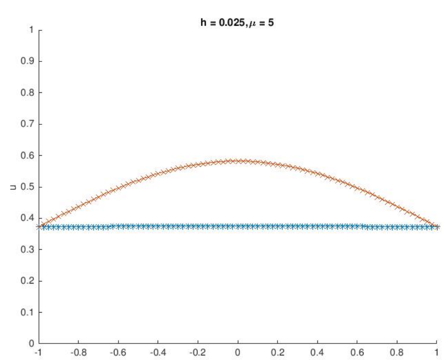

Figure 9: Analytical solution vs. scheme

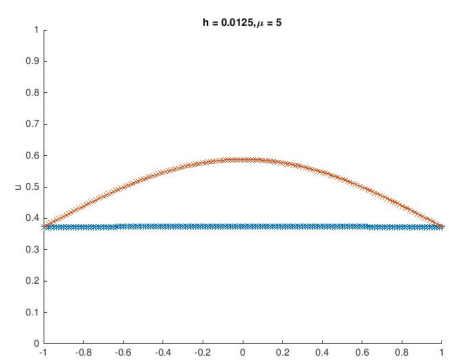

Figure 10: Analytical solution vs. scheme

Thus, we can see that it seems that the scheme does not do a good job in reproducing the dynamics of the solution. However, there might still probably be some error in the way the scheme was implemented in the code, therefore it is unclear if any definitive conclusion can be made about the scheme itself. Anyway, we compute the L2 norm and the Supremum Norm.

L2 Norm and Supremum Norm		
h value	Supremum Norm	L2 Norm
1/10	0.32678	0.74834
1/20	0.41911	1.0274
1/40	0.42452	1.4152
1/80	0.4268	1.9686

2 Problem 10.4.1 from Strikwerda

In this problem, we solve

$$u_t = u_{xx}$$

in the interval $-1 \leq x \leq 1$ for $t \in [0, 1]$. The values of h will be chosen to be $1/10,\ 1/20,\ 1/40$ and $1/80$.

We will use the forward-time central-space scheme

2.1 Part a

Initial data is given to be

$$u(0, x) = \begin{cases} 1 & \text{if } |x| \leq \frac{1}{2} \\ \frac{1}{2} & \text{if } |x| = \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

Here is the numerical scheme implemented

```
clear all
tic
h = 1/40;
b = 1;
mu = .4;
tmax = 1;


k = mu*h^2;
td = 0:k:tmax;
q = length(td);
xd = -1:h:1;
```

```
p = length(xd);
L = 0:500;
u0 = @(x) (abs(x) < 0.5)+(abs(x) == 0.5)/2;
%u0 = @(x) (cos(pi*x));
for m = 1:p %initial conditions
    u(1,m) = u0(xd(m));
end

w = zeros(q,p);

for ii = 1:q %calculates exact solution
    for jj = 1:p
        S1 = exp(-td(ii).*(2.*L+1).^2.*pi^2).*(-1).^L./(2.*L+1).*cos
            ((2.*L+1).*pi.*xd(jj));
        w(ii,jj) = (1/2) + (2/pi).*sum(S1);
    end
end


for i = 1:q-1
    u(i,1) = w(i,1);
    for j = 1:p-2
        u(i+1,j+1) = (1-2*b*mu)*u(i,j+1)+b*mu*(u(i,j+2)+u(i,j));
        u(i,p) = u(i,p-2);
    end

end
u(q,1)=u(q,p-1);
u(q,1) = w(q,1);

time = toc


for i = 1:q
    plot(xd,u(i,:),'b-o',xd,w(i,:),'k');
    ylim([-1.5,1.5])
    xlim([-1,1])
    title(['Heat equation: \mu = .4, h = ' num2str(h) ', Time = '
        num2str(td(i))])
    xlabel('x')
    ylabel('u')
    grid on
    M(i) = getframe;
end
```

```
supnorm = max(abs(w(q,:)-u(q,:)))

L2norm = norm(w(q,:)-u(q,:))
```

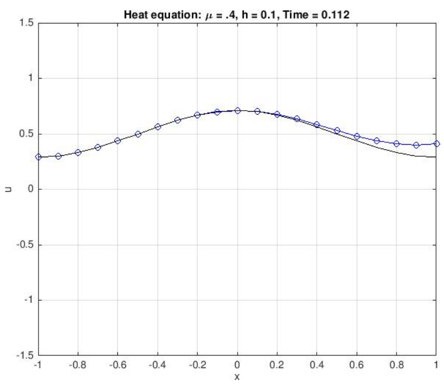

Figure 11: Analytical solution vs. scheme

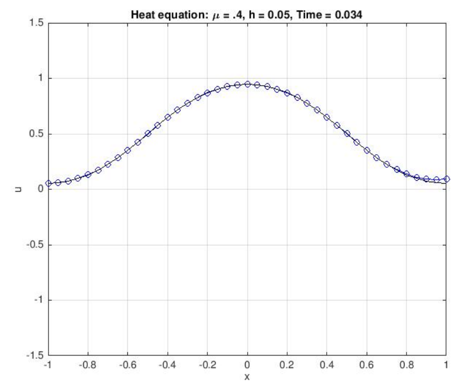

Figure 12: Analytical solution vs. scheme

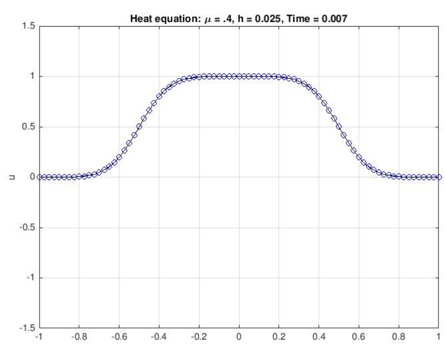

Figure 13: Analytical solution vs. scheme

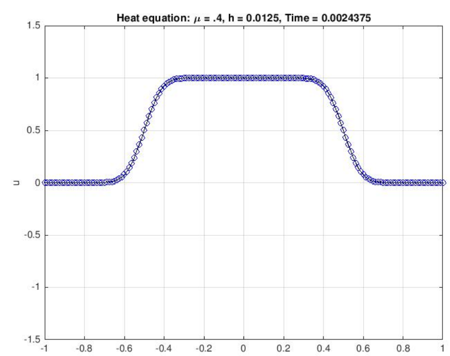

Figure 14: Analytical solution vs. scheme

Therefore we see that for $h = 1/10$ and $h = 1/20$, we lose some information at the right boundary, while for $h = 1/40$ and $h = 1/80$ the scheme does an excellent job in terms of accuracy.

2.2 Part b

We use the same numerical scheme and code, but here we use the initial data

$$u_0 = \cos(\pi x)$$

We display our results

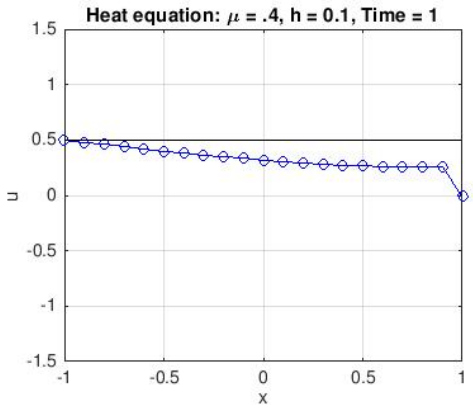

Figure 15: Analytical solution vs. scheme

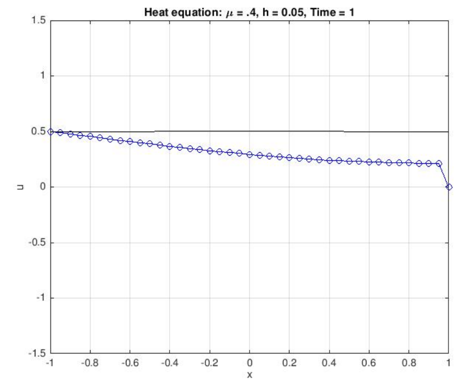

Figure 16: Analytical solution vs. scheme

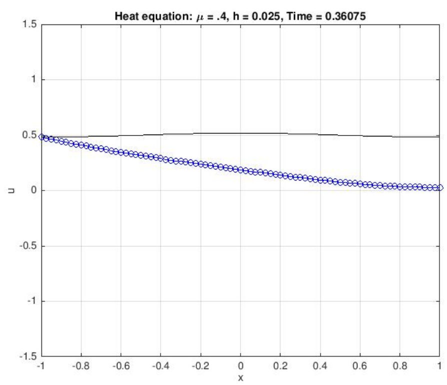

Figure 17: Analytical solution vs. scheme

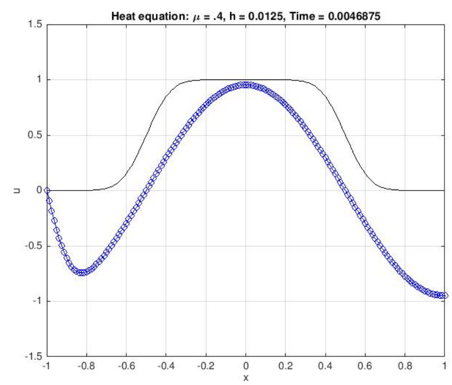

Figure 18: Analytical solution vs. scheme

There seems to be some error in the way the boundary conditions were defined in the code, but, at this stage, I was not able to fix it properly.