AMATH 481/581 - Autumn 2022 Homework #4

Presentation skill: 2D plot.

Wietse Vaes 222448816

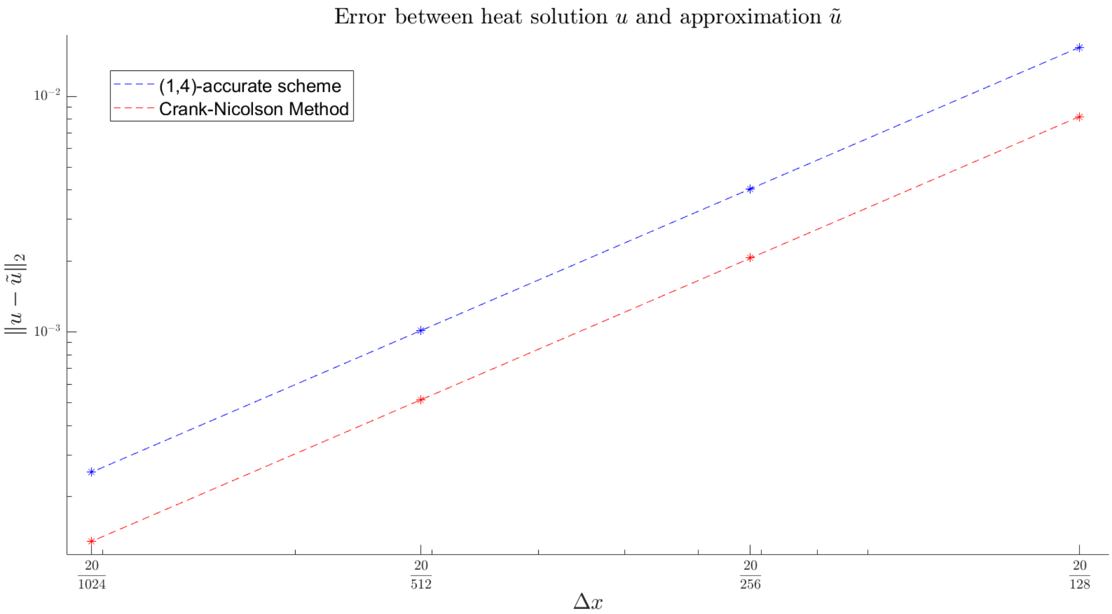

Fig. 1: The 2-norm of the error between the approximation and exact solution to the heat equation, with periodic boundary conditions, in function of mesh spacing Δx, represented by stars, using a (1,4)-accurate scheme and the Crank-Nicolson Method (using the LU decomposition). The dashed line represents the trend line of the error of their respective methods. Both have a slope of approximately 1.5, thus indicating the methods have an order of accuracy of 1.5.

AMATH 481/581 - Autumn 2022 Homework #4

Presentation skill: 3D plot.

Wietse Vaes 222448816

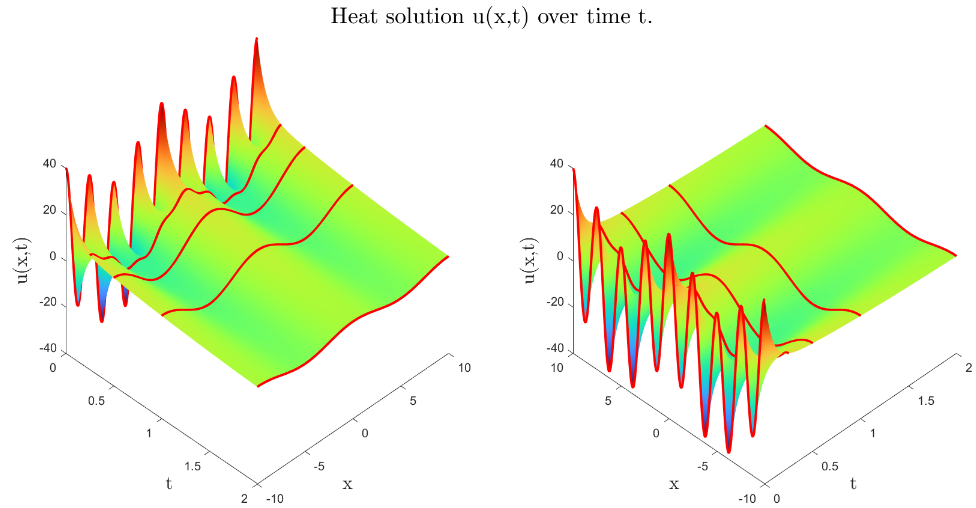

Fig. 1: The solution $u(x,t)$ to the heat equation using initial condition $u_0 = 10\cos\left(\frac{2\pi x}{10}\right) + 30\cos\left(\frac{8\pi x}{10}\right)$ and periodic boundary conditions, for $x \in [-10, 10]$ and time $t \in [0, 2]$. The approximation of the solution has been achieved by using the Crank-Nicolson Method and biconjugate gradient stabilized method. The red lines represent solutions at chosen times in order to better understand the evolution of the heat over time.

Appendix

A Code presentation skill: 2D plot

```
clear all
%% Parameters + Getting Lambda
L = 10;
T = 2;
alpha = 2;
Allnx = [128,256,512,1024];
nt = 501;

x = linspace(-L,L,Allnx(1)+1); x(end)=[];
Deltax = mean(diff(x));
t = linspace(0,T,nt); Deltat = mean(diff(t));

lambda = Deltat*alpha/(Deltax^2);


%% calculating all approximations
%Calculating aproximations using Method of Lines (1,4) and Crank-Nicolson Method
for index1 = 1:4
    nx = Allnx(index1);
    [u1{index1},u2{index1},DeltaX(index1)] = calcu(nx,lambda);
end

%% calculating erros
%Reading in exact solutions
names = {'exact_128.csv','exact_256.csv','exact_512.csv','exact_1024.csv'};
for index1 = 1:4
```

```
    u{index1} = readmatrix(names{index1});
end

%Calculating 2-norm of error
for index1 = 1:4
        err1(index1)= sqrt(trapz(-L:DeltaX(index1):L-DeltaX(index1),(u1{index1}(:,end)-u{index1}).^2));
        err2(index1)= sqrt(trapz(-L:DeltaX(index1):L-DeltaX(index1),(u2{index1}(:,end)-u{index1}).^2));
end
%% plotting
%plot
figure(1); hold on
coeff1 = polyfit(log(DeltaX),log(err1),1);
coeff2 = polyfit(log(DeltaX),log(err2),1);
loglog(DeltaX,exp(coeff1(2))*DeltaX.^coeff1(1),'--','Color',"b")
loglog(DeltaX,exp(coeff2(2))*DeltaX.^coeff2(1),'--','Color',"r")
legend('(1,4)-accurate scheme','Crank-Nicolson Method')
loglog(DeltaX,err1,'*','Color',"b");loglog(DeltaX,err2,'*','Color',"r");
%setting axis + labeling
set(groot,'defaultAxesTickLabelInterpreter','latex');
set(gca, 'YScale', 'log','XScale', 'log');xlim([min(DeltaX)-.001,max(DeltaX)+.01]);xticks(flip(DeltaX));xticklabels(["$$\frac{20}{1024}$$","$$\frac{20}{512}$$","$$\frac{20}{256}$$","$$\frac{20}{128}$$"])
xlabel('$$\Delta x$$','Interpreter','Latex','FontSize',16); ylabel('$$\|u-\tilde{u}\|_2$$','Interpreter','Latex','FontSize',16);
title('Error between heat solution $$u$$ and approximation $$\tilde{u}$$','Interpreter','Latex','FontSize',16)
%legend
```

```
lgd = legend('(1,4)-accurate scheme','Crank-Nicolson Method');
lgd.FontSize = 13;

%% Function
function [u1,u2,Deltax] = calcu(nx,lambda)
%Parameters
L = 10;
T = 2;
alpha = 2;
%x-grid
x = linspace(-L,L,nx+1); x(end)=[];
Deltax = mean(diff(x));
%t-grid
Deltat = lambda/alpha*Deltax^2;
t = 0:Deltat:T; nt = length(t);

%making D4
e1 = ones(nx,1);
D4 = spdiags([-e1,16*e1,-30*e1,16*e1,-e1],-2:2, nx, nx);
D4(1,nx-1) = -1; D4(1,nx) = 16; D4(2,nx) = -1;
D4(nx-1,1) = -1; D4(nx,1) = 16; D4(nx,2) = -1;
D4 = D4/12;

%initial condition
f = @(x) 10*cos(2*pi*x/L)+30*cos(8*pi*x/L);
u1 = zeros(nx,nt);u2 = zeros(nx,nt);
u1(:,1) = f(x)';
for index1 = 2:nt
    u1(:,index1) = u1(:,index1-1)+lambda*D4*u1(:,index1-1); %
        calculating solution using MOL
end
%Making  B and C for Crank-Nicolson Method
```

```
e1 = ones(nx,1);
B = spdiags([-lambda*e1/2,e1,-lambda*e1/2],-1:1, nx, nx); C =
    spdiags([lambda*e1/2, e1, lambda*e1/2],-1:1, nx, nx);
B = B + lambda*speye(nx,nx); C = C - lambda*speye(nx,nx);
B(1,end) = -lambda/2;B(end,1) = -lambda/2;
C(1,end) = lambda/2;C(end,1) = lambda/2;
%Calculating the LU-decomposition
[L,U,P] = lu(B);

u2(:,1) = f(x)';
for index1 = 2:nt
    u2(:,index1) = U\(L\(P*(C*u2(:,index1-1)))); %calculating
        solution using LU
end

end
```

B Code presentation skill: 3D plot

```
clear all
%% Parameters + Getting Lambda
L = 10;
T = 2;
alpha = 2;
nt = 501;

x = linspace(-L,L,128+1); x(end)=[];
Deltax = mean(diff(x));
t = linspace(0,T,nt); Deltat = mean(diff(t));

lambda = Deltat*alpha/(Deltax^2);


%% calculating approximations
%using bicgstab
[u3,x,t] = calcubicgstab(512,lambda);

%Making u3 plotable.
u3(size(u3,1)+1,:) = u3(1,:);
x(length(x)+1) = 10;
nt = length(t);
%% plotting
[TT,XX] = meshgrid(t,x);
clf;
figure(1);
TTT = [1,nt,1000,2000,4000]; %Timestamp when solution will be
    explicitly shown
angle = [45,45;-45,45]; %angles of plotting
for index2 = 1:2
```

```
    subplot(1,2,index2);hold on
    surf(TT,XX,u3); shading interp;colormap turbo; %Plotting full solution
    for index1 = 1:length(TTT)
        plot3(ones(length(x),1)*t(TTT(index1)),x,u3(:,TTT(index1)),'Color','r','LineWidth',2); %plotting solution on specific timestamps
    end
    view(angle(index2,:));
    xlabel('t','Interpreter','Latex','FontSize',16);ylabel('x','Interpreter','Latex','FontSize',16);zlabel('u(x,t)','Interpreter','Latex','FontSize',16);
    hold off
end
sgtitle('Heat solution u(x,t) over time t.','Interpreter','Latex','FontSize',19)
%% Function
function [u, x, t] = calcubicgstab(nx,lambda)
%parameters
L = 10;
T = 2;
alpha = 2;

%grid definition
x = linspace(-L,L,nx+1); x(end)=[];
Deltax = mean(diff(x));
Deltat = lambda/alpha*Deltax^2;

t = 0:Deltat:T; nt = length(t);

%making matrix B and C
e1 = ones(nx,1);
```

```
B = spdiags([-lambda*e1/2,e1,-lambda*e1/2],-1:1, nx, nx); C =
    spdiags([lambda*e1/2, e1, lambda*e1/2],-1:1, nx, nx);
B = B + lambda*speye(nx,nx); C = C - lambda*speye(nx,nx);
B(1,end) = -lambda/2;B(end,1) = -lambda/2;
C(1,end) = lambda/2;C(end,1) = lambda/2;

%initial condition
f = @(x) 10*cos(2*pi*x/L)+30*cos(8*pi*x/L);
u(:,1) = f(x)';
for index1 = 2:nt
    [u(:,index1),~] = bicgstab(B,C*u(:,index1-1)); %computing
        solution
end

end
```