**Finite Differences Project**

**(Parabolic PDE)**

Minxuan Song

Guancheng Qiu

Zijun Lin

1. **Introduction**

Finite difference methods (FDM) have been known for its ability to solve difference equations. As is evident, different finite different methods yield different results, which largely results from the differences in input selection. This paper focuses heavily on solving practical problems using different finite difference methods and the error analysis that arises when comparing the results. Specifically, this problem will focus on solving the famous one-dimensional heat problem, and the problem that’s considered an extension of it: the two-dimensional heat problem. Note that an analytical solution for both of the problem will be provided, in order to illustrate a better understanding to the problem.

1. **The one-dimensional heat problem**

Consider the one dimensional heat problem: find u(x,t) such that . Suppose the following condition is known: u(x,0) = sin(πx) for all x such that 0 ≤ x ≤1, and u(0, t) = 0 for all t such that 0 ≤ t ≤1, and u(1,t) = 0 for all t such that 0 ≤ t ≤ 1.

**Analytical Solution:**

By firstly doing an analytical solution, we can familiarize ourselves with the essence of the problem. The solution not only informs us of the framework of this problem but also provides us with results that we can later compare with the results we get when we’re trying to solve this problem using numerical methods.

First, assume the solution has the form:

The original equation then becomes:

where λ is an arbitrary constant. This gives us two ODE’s:

Given initial values , boundaries , we have

For a non-trivial solution, consider boundaries to be

Let us first look at the spatial problem

The solution is of the form:

which gives:

For a non-trivial solution, we have so

For the time problem

the solution is

which gives the solution to the original equation

as we let

Hence, this problem is solved analytically.

**Numerical Solutions:**

We take on this problem with three different finite difference methods: the Explicit method, the Implicit method and the Crank-Nicolson method. Note that we’re already provided with the knowledge that for , u(x,0) = sin(πx) for all x such that 0 ≤ x ≤1, and u(0,t) = 0 for all t such that 0 ≤ t ≤1, and u(1,t) = 0 for all t such that 0 ≤ t ≤ 1. This will provide us with interior nodes to construct the linear system that we’ll be using to solve this problem.

In terms of the Explicit method, we’ll use Euler’s method to approach this problem.

For LHS, we have the forward difference formula:

and for RHS, we have the centered three-point formula:

Now by ignoring error terms and equating the two, we have:

Now simplifying this equation:

Since u(x,0) = sin(, and we already know the boundaries , it will be very straightforward to solve for .

Therefore, we have a linear system as this:

Then, solving this should yields an answer to the explicit method.

Matlab implementation for this method:

M = 6;

N = 6;

linsysa = zeros(M-2,N);

tempu = zeros(M,N-2);

u = zeros(M,N);

dx = 1/(M-1);

dt = 1/(M-1);

%write bottom line

**for** i = 1:(M-1)

**for** j = 1:N

u(M,j) = sin(1/5 \* pi\*(j-1));

end

end

%write linearsystem

**for** i = 1:(M-2)

linsysa(i,i) = dt/dx^2 ;

linsysa(i,i+1) = 1-2\*dt/dx^2 ;

linsysa(i,i+2) = dt/dx^2 ;

end

%creat u without left and right colums

**for** i = 1:M

**for** j =1 :(N-2)

tempu(i,j) = u(i,j+1);

end

end

%calculate

**for** j = 1:(N-1)

tempu(M-j,:) = linsysa\*(u(M-j+1,:))';

**for** i = 1:M

**for** n =1 :(N-2)

u(i,n+1) = tempu(i,n);

end

end

end

disp(u)

In terms of Implicit method, we can also take Euler’s method to take on this problem.

For LHS, we have:

This is the backward difference formula.

For RHS, we have:

This is the centered three-point formula.

Then, by ignoring error terms and equating the two equations, we have:

By simplifying this equation, we have:

Now, given initial values and the boundaries , we can construct the linear system that we need to solve for .

Note that n is the number of partitions. Since we already know that when j – 1 = 0, u(x,0) = So by starting a loop from j = 1, and with the boundaries values , we can always have .

Matlab implementation for this method:

M = 6;

N = 6;

u = zeros(M,N);

linsysa = zeros(M-2,N);

linsystemp = zeros(M,N-2);

dx = 1/(M-1);

dt = 1/(M-1);

%write bottom line

**for** i = 1:(M-1)

**for** j = 1:N

u(M,j) = sin(1/5 \* pi\*(j-1));

end

end

%write linearsystems

**for** i = 1:(M-2)

linsysa(i,i) = -dt/dx^2 ;

linsysa(i,i+1) = 2\*dt/dx^2 +1 ;

linsysa(i,i+2) = -dt/dx^2 ;

end

%creat temp system

**for** i = 1:M

**for** j =1 :(N-2)

linsystemp(i,j) = u(i,j+1);

end

end

%calculate

**for** j = 1:(N-1)

u(M-j,:) = linsysa\(linsystemp(M-j+1,:))';

**for** i = 1:M

**for** j =1 :(N-2)

linsystemp(i,j) = u(i,j+1);

end

end

end

disp(u)

Using Crank-Nicolson:

For LHS, we have:

This is the forward difference formula.

Then, for RHS, we have:

Note that different from the explicit method and the implicit method, this is the centered three-point formula applied twice.

Now, by ignoring error terms and equating two of the equations, we have:

Further simplifying this:

Therefore, we can construct a linear system accordingly, for each j:

Matlab implementation:

M = 6;

N = 6;

linsysa = zeros(M-2,N);

linsysb = zeros(M-2,N);

linsystemp = zeros(4)';

u = zeros(M,N);

dx = 1/(M-1);

dt = 1/(M-1);

%write linear systems

**for** i = 1:(M-2)

linsysa(i,i) = -dt/dx^2 ;

linsysa(i,i+1) = 2 + 2\*dt/dx^2 ;

linsysa(i,i+2) = -dt/dx^2 ;

linsysb(i,i) = dt/dx^2 ;

linsysb(i,i+1) = 2 - 2\*dt/dx^2 ;

linsysb(i,i+2) = dt/dx^2 ;

end

%write u bottom line

**for** i = 1:(M-1)

**for** j = 1:N

u(M,j) = sin(1/5 \* pi\*(j-1));

end

end

%calculate

**for** j = 1:(N-1)

linsystemp = linsysb\*(u(M-j+1,:))';

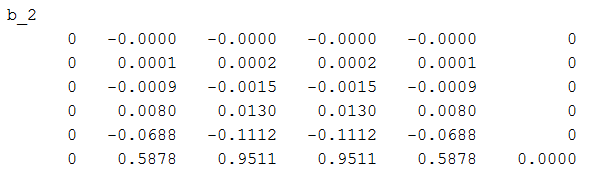
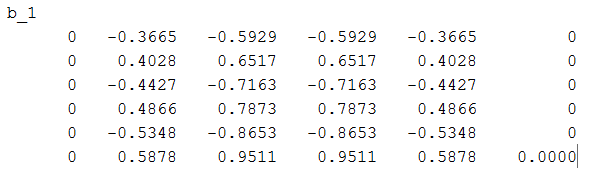
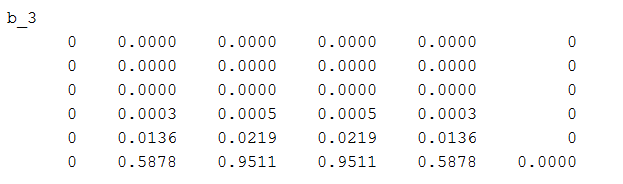
u(M-j,:) = linsysa\linsystemp;

end

disp(u)

Now, we have solved this problem using all three numerical methods. The differences, however, that exist in the results that each of the methods yields, will lead to a better understanding of the numerical methods that we have chosen to use, which we’ll elaborate further in the next section, error analysis.

**Error Analysis:**

For all three of the different numerical methods that we have come up with, we can see that the results they yield differ largely. Matlab yields the following results:  

We first of all have discovered that the acurracy of all three of the methods relies heavily on the number of the interior nodes chosen. If we are to set the nodes given to be 12 instead of 6, meanwhile making the intervals between each nodes to be smaller, the accutacy for all three of the methods increase significantly. We also see clearly that for the implicit method and the Crank-Nicolson method, there are many elements inside the result table that appear to be zero. This is because both of the methods in some fashion presumes to calculate the nodes from top to bottom, and solving such linear system can easily lead to this kind of results. Now, provided that we already have the analytical solution, for each method, we can take the absolute value of the difference between the exact value and the value that the specific numerical method yields of each of the interior nodes, and by adding them up and taking the square root of the sum of such, we can see how far each FDM is from the actual answer. Then the accuracy for each of our FDM will be conspicuous.

1. **Extension: The two-dimensional heat problem**

With all the information reduced from the one-dimensional heat problem, we can take our research to the next level. Consider the following problem: a square plate of steel is 15 cm on a side. Initially, all points are at 0ºC. If two adjacent sides are suddenly brought to 100ºC and held at that temperature. The plate is insulated on its flat surfaces. so that heat flows only in the x and y directions. We are given that k = 0.13, ϕ = 7.8, c = 0.11 in c.g.s units. Now, solve

**Analytical Solution:**

We can solve this problem analytically in the same fashion that we did for the one-dimensional heat problem. Note that because the situation in terms of the number of parameters becomes significantly harder, the interior nodes we use increase in number. In fact, if we are to plot everything on a graph, it is straightforward to observe that this problem largely differs from the one-dimensional heat problem because the graph turns from 2D to 3D. Also, our initial condition changes. It has now become: u except for boundaries. For boundaries, it is: . . Therefore, using the same reduction logic, we can arrive at a satisfying expression that is similar in composition to the ones that we came up with for the one-dimensional heat problem.

**Numerical Solution:**

For numerical methods, we’ll be using the same three methods (explicit, implicit and Crank-Nicolson) that we applied to the one-dimensional heat problem.

In terms of the Explicit method, we’ll also use Euler’s method to approach this problem.

For LHS, we have the forward difference formula:

and for RHS, we have the centered three-point formula:

Now by ignoring error terms and equating the two, we have:

Now simplifying this equation:

Given except for the boundaries, and we know that for the boundaries, , it is straightforward to solve for .

Therefore, we have a linear system as this:

+ =

Then, solving this should yields an answer to the explicit method.

Matlab implementation:

k = 0.13;

phi = 7.8;

c = 0.11;

M = 9;

N = 9;

Q = 9;

linsysa = zeros(M-2,N);

linsysb = zeros(M,2\*N);

linsystemp = zeros(4)';

u = zeros(M,N,Q);

dx = 15/(M-1);

dt = 1;

dy = 15/(M-1);

**for** i = 1:(M)

linsysa(i,i) = -dt/dx^2 ;

linsysa(i,i+1) = c\*phi/k - 2\*dt/dx^2 -2\*dt/dy^2 ;

linsysa(i,i+2) = -dt/dx^2 ;

end

**for** i = 1:(M)

linsysb(i,2\*i-1) = dt/dy^2 ;

linsysb(i,2\*i) = dt/dy^2;

end

**for** i = 1:(M)

**for** j = 1:N

**for** z = 1:Q

u(1,j,z) = 100;

u(i,1,z) = 100;

end

end

end

**for** i = 1:(M)

**for** j = 1:N

**for** z = 1:Q

u(i,j,z+1) = k/c/phi\*linsysa \*(u(i,j,z))' + linsysb\*(u(i+1,j,k);

end

end

end

disp(u)

In terms of Implicit method, we can also take Euler’s method to take on this problem.

For LHS, we have:

This is the backward difference formula.

For RHS, we have:

This is the centered three-point formula.

Then, by ignoring error terms and equating the two equations, we have:

By simplifying this equation, we have:

Now, given except for the boundaries, and for the boundaries, we have , so we can construct a linear system, for each j and for

each k:

\*

Using Crank-Nicolson:

For LHS, we have:

This is the forward difference formula.

Then, for RHS, we have:

Now, by ignoring error terms and equating two of the equations, we have:

Further simplifying this:

Therefore, we can construct a linear system accordingly, for each j and each k:

\*

Solving this linear system should give us a satisfying result that is based on the Crank-Nicolson method.