MA 322: Lab Assignment #8

Due on Sunday, November 2, 2015

Jiten Chandra Kalita

Silvi Pandey - 130123045

Contents

PROBLEM 1
To derive explicit and implicit finite difference approximations to the problem

$$Du_{xx} = u_t + bu$$

where $u(0,t) = u(1,t) = 0$ and $u(x,0) = g(x)$. Also, D and b are positive constants.

SOLUTION
(a) Explicit Method

$$D(\frac{U_{i+1}^n - 2U_i^n + U_{i-1}^n}{\Delta x^2}) = (\frac{U_i^{n+1} - U_i^n}{\Delta t}) + bU_i^n$$

$$\frac{D\Delta t}{\Delta x^2}(U_{i+1}^n - 2U_i^n + U_{i-1}^n) = U_i^{n+1} - U_i^n + bU_i^n \Delta t$$

$$U_i^{n+1} = U_{i+1}^n \frac{D\Delta t}{\Delta x^2} + U_{i-1}^n \frac{D\Delta t}{\Delta x^2} + (1 - b\Delta t - 2\frac{D\Delta t}{\Delta x^2})U_i^n$$

(b) Implicit Method
The implicit finite difference approximation can be obtained by replacing every grid value at the n^{th} time stamp with the average of the n^{th} and $(n+1)^{th}$ time stamp.

$$(\frac{U_i^{n+1} - U_i^n}{\Delta t}) = D(\frac{U_{i+1}^n - 2U_i^n + U_{i-1}^n}{\Delta x^2}) - bU_i^n$$

$$(\frac{U_i^{n+1} - U_i^n}{\Delta t}) = D(\frac{(\frac{U_{i+1}^{n+1} + U_{i+1}^n}{2}) - 2(\frac{U_i^{n+1} + U_i^n}{2}) + (\frac{U_{i-1}^{n+1} + U_{i-1}^n}{2})}{\Delta x^2}) - b(\frac{U_i^{n+1} + U_i^n}{2})$$

$$(1 + \alpha + \frac{\beta}{2})U_i^{n+1} - \frac{\alpha}{2}U_{i+1}^{n+1} - \frac{\alpha}{2}U_{i-1}^{n+1} = (1 - \alpha - \frac{\beta}{2})U_i^n + \frac{\alpha}{2}U_{i+1}^n + \frac{\alpha}{2}U_{i-1}^n$$

where $\alpha = \frac{D\Delta t}{\Delta x^2}$ and $\beta = b\Delta t$

PROBLEM 2
$g(x) = sin(\pi x)$, $D = 0.1$, $b = 1$. Compute the solution at $t = 1$ using Δt =0.25 , 0.125 , 0.0625. On the same axes plot the exact solution at t = 1 and the three numerical solutions, one for the explicit and the other for the implicit method.

SOLUTION

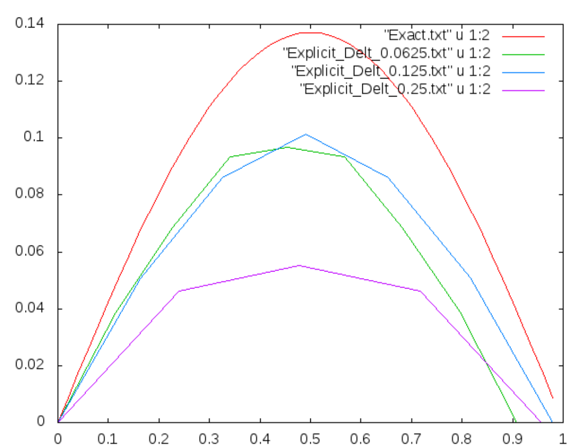
(a) EXPLICIT METHOD
Code

```
#include<bits/stdc++.h>

using namespace std;
double D = 0.1;
double b = 1;
double boundFunction(double x)
{
	return sin((double)atan(1)*(double)4*x);
}
double Getdelx(double delt)
{
	return sqrt(((double)4*D*delt)/((double)2-b*delt));
}
```

```
int main()
{
     double delt,delx;
     FILE *fp = fopen("Explicit_Delt_0.0625.txt","w");
     int countx,countt;
     double alpha,beta;
     cout<<"Enter delt"<<endl;
     cin>>delt;
     delx = Getdelx(delt);
     alpha = (D*delt)/(delx*delx);
     beta = b*delt;
     countx = floor((double)1/delx);
     countt = floor((double)1/delt);
     double arr[countx+1],arr1[countx+1];
     arr[0] = 0;
     for(int i=1;i<countx+1;i++)
     {
          arr[i] = boundFunction(i*delx);
     }
     arr[countx] = 0;
     arr1[0] = arr1[countx] = 0;
     while(countt--)
     {
          for(int i=1;i<countx;i++)
          {
               arr1[i] = arr[i+1]*alpha + arr[i-1]*alpha + (1-beta-2*alpha)*arr[i];
          }
          for(int i=1;i<countx+1;i++)
               arr[i] = arr1[i];
     }
     for(int i= 0;i<countx+1;i++)
     {
          fprintf(fp,"%lf %lf\n",i*delx,arr[i]);
          cout<<i*delx<<" "<<arr[i]<<endl;
     }
}
```

Result



Explanation

The exact solution of the PDE can be obtained by integrating the expression given in the problem.It comes out as $e^{-b-D\pi^2} sin(\pi x)$ at $t = 1$

To get the appropriate Δx for the given values of Δt ,the conditions of stability are exploited.

The condition comes out as :

$$\Delta x^2 >= \frac{4D\Delta t}{2 - b\Delta t}$$

IMPLICIT METHOD

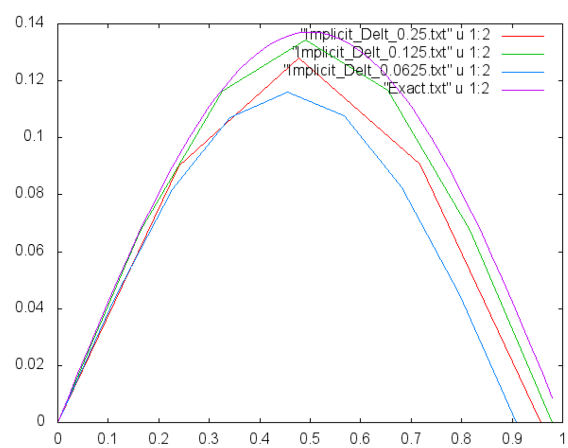
Code

```
#include<bits/stdc++.h>

using namespace std;
double D = 0.1;
double b = 1;
double boundFunction(double x)
{
     return sin((double)atan(1)*(double)4*x);
}
double Getdelx(double delt)
{
     return sqrt(((double)4*D*delt)/((double)2-b*delt));
}
void SolveGaussSeidel(double arr1[],double arr2[],double b[],double size,
double alpha,double beta);
int main()
{
     double delt,delx;
     FILE *fp = fopen("Implicit_Delt_0.0625.txt","w");
     int countx,countt;
     double alpha,beta;
     cout<<"Enter delt"<<endl;
     cin>>delt;
     delx = Getdelx(delt);
     alpha = (D*delt)/(delx*delx);
     beta = b*delt;
     countx = floor((double)1/delx);
     countt = floor((double)1/delt);
     double arr[countx+1],arr1[countx+1],b[countx+1],arr2[countx+1];
     arr[0] = arr[countx] = 0;
    b[0] = b[countx] = 0;
     for(int i=1;i<countx;i++)
     {
          arr[i] = boundFunction(i*delx);
     }
     for(int i=1;i<countx;i++)
    {
        arr1[i] = 0;
        b[i] = (1-alpha-beta/2)*arr[i] + alpha*arr[i+1]*0.5 + alpha*arr[i-1]*0.5;
    }
    arr1[0] = arr1[countx] = 0;
    arr2[0] = arr2[countx] = 0;
     while(countt--)
    {
```

```
        SolveGaussSeidel(arr1,arr2,b,countx,alpha,beta);
        for(int i=1;i<countx;i++)
            arr1[i] = 0;
        for(int i=1;i<countx;i++)
        {
            b[i] = (1-alpha-beta/2)*arr2[i] + alpha*arr2[i+1]*0.5 + alpha*arr2[i-1]*0.5;
        }
    }
    for(int i=0;i<countx+1;i++)
    {
        fprintf(fp,"%lf %lf\n",i*delx,arr2[i]);
    }
}
void SolveGaussSeidel(double arr1[],double arr2[],double b[],double size,
double alpha,double beta)
{
    int count = 0;
    while(1)
    {
        for(int i=1;i<size;i++)
        {
            arr2[i] = (b[i]+alpha*0.5*arr2[i-1]+alpha*0.5*arr1[i+1])/(1+alpha+beta*0.5);
        }
        for(int i=1;i<size;i++)
        {
            if(fabs(arr1[i]-arr2[i])/fabs(arr1[i]) <= 0.0001)
                count++;
        }
        if(count == size-1)
            break;
        count = 0;
        for(int i=1;i<size;i++)
            arr1[i] = arr2[i];
    }
}
```

Result



PROBLEM 3

Assuming $g(x) = sin(\pi x)$, $D = 110$, $b = 1$, plot the maximum error at $t = 1$ as function of M where $t = \frac{1}{M}$ and $\Delta x = \frac{1}{10}$ for $M = 4, 8, 16, 32$. On the same axes also plot the maximum error at $t = 1$ for $M = 4, 8, 16, 32$ and $\Delta x = \frac{1}{20}$. Explain the behaviour of these two curves using the stated truncation error.

SOLUTION

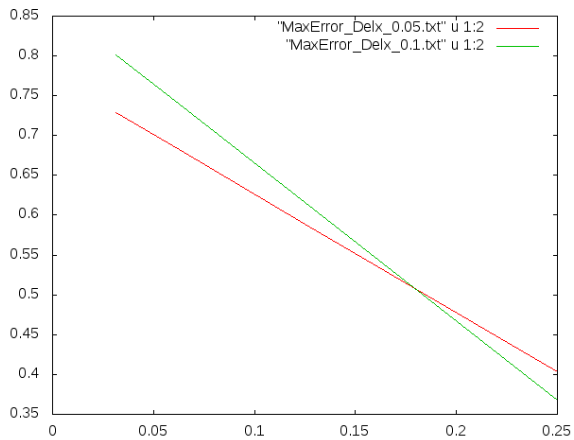
Code

```
#include<bits/stdc++.h>

using namespace std;
double D = 0.05;
double b = 1;
double delx = 0.1;
double pi = (double)4*atan(1);
double coeff = exp(-b-D*pi*pi);
double exactVal(double x)
{
     return coeff*sin(pi*x);
}
double boundFunction(double x)
{
     return sin((double)atan(1)*(double)4*x);
}
int main()
{
     FILE *fp = fopen("MaxError_Delx_0.05.txt","w");
     int countx,countt;
     double alpha,beta,maxError,error;
     double arr[1000],arr1[1000];
     double delt[] = {0.25,0.125,0.0625,0.03125};
     for(int j=0;j<4;j++)
     {
          alpha = (D*delt[j])/(delx*delx);
          beta = b*delt[j];
          maxError = 0.0;
          countx = floor((double)1/delx);
          countt = floor((double)1/delt[j]);
          arr[0] = arr[countx] = 0;
          for(int i=1;i<countx;i++)
          {
               arr[i] = boundFunction(i*delx);
          }
          arr1[0] = arr1[countx] = 0;
          for(int i=1;i<countx;i++)
          {
               arr1[i] = arr[i+1]*alpha + arr[i-1]*alpha + (1-beta-2*alpha)*arr[i];
          }
          for(int i=1;i<countx+1;i++)
               arr[i] = arr1[i];
          for(int i= 1;i<countx+1;i++)
          {
               error = fabs(arr[i] - exactVal(i*delx));
```

```
            if(error > maxError)
                maxError = error;
        }
        fprintf(fp,"%lf %lf\n",delt[j],maxError);
    }
}
```

Result



Explanation

The nature of the graph is linear.

The reason is the order of Δt and Δx in the truncation error expression.

Truncation error in the FDE is $O(\Delta t) + O(\Delta x^2)$.

In either of the cases, Δx is fixed which makes the maximum error a linear function of Δt