PROJECT

IMPLICIT AND EXPLICIT METHODS

Numerical Solutions

Explicit and implicit methods are approaches used in numerical analysis for obtaining numerical solutions of time-dependent ordinary and partial differential equations, as is required in computer simulations of physical processes

CEP 452

Computational Aspects in Water Resources

Professor
N. K. Garg

Civil Engineering Department

Indian Institute of Technology Delhi

SUMEET KUMAR SINHA
2010CE10405

Group 1

27th March, 2014

CEP 452
IMPLICIT AND EXPLICIT METHODS

March 27, 2014

PROBLEM DESCRIPTION

[*The following problem illustrates the use of the explicit and implicit methods and their advantages and disadvantages of their use.*]

Distance between two parallel plates is 1 cm and a liquid with a kinematic viscosity ɳ is filled between the parallel plates. The lower plate is fixed and the upper plate is linearly accelerated from 0 to 10 cm/sec in 0.1 sec and then it remains constant to 10 cm/sec. Assuming fully developed and 1-Dimensional flow, determine the velocity distribution profile of the flute in-between the plates. Draw the velocity vs time at y=0.9 cm till 1 sec

Explicit Method

[Explicit methods calculate the state of a system at a later time from the state of the system at the current time, while implicit methods find a solution by solving an equation involving both the current state of the system and the later one.]

The transient (time-dependent) heat equation in 1D

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2} \tag{1}$$

In explicit finite difference schemes, the temperature at time n+1 depends explicitly on the temperature at time n. The explicit finite difference discretization of equation 1 is

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \kappa \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2} \tag{2}$$

This can rearranged in the following manner (with all quantities at time n+1 on the left-hand-side and quantities at time n on the right-hand-side)

$$T_i^{n+1} = T_i^n + \kappa \Delta t \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2} \tag{3}$$

Since we know T_{i+1}^n, T_i^n and T_{i-1}^n, we can compute T_i^{n+1}. This is schematically shown on figure 1. The major advantage of explicit finite difference methods is that they are relatively simple and computationally fast. However, the main drawback is that stable solutions are obtained only when

$$0 < \frac{\kappa \Delta t}{\Delta x^2} < 0.5 \tag{4}$$

If this condition is not satisfied, the solution becomes unstable and starts to wildly oscillate.

Program Developed in FORTRAN

[Fortran (previously FORTRAN, derived from Formula Translating System) is a general-purpose, imperative programming language that is especially suited to numeric computation and scientific computing.]

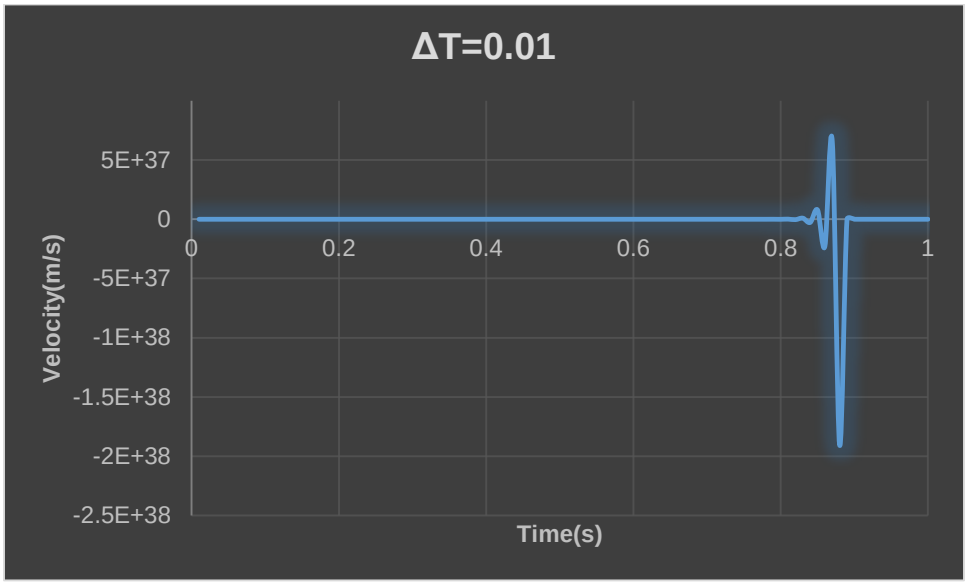
```
implicit none

!! Declaring Variables for the explicit scheme

real,dimension(10) :: fun,funw
real,dimension(100) :: vel
integer :: i,j
real :: lmda,dt,dy,nu

!! Takin Input Parameters from the user

write(*,*) 'Input the kinematic viscosity (Neu) sec/m2'
read(*,*) nu
write(*,*) 'Input the time difference (dt) sec'
read(*,*) dt
write(*,*) 'Input the distance difference (dy) m'
read(*,*) dy

!! Assigning the values to variables

lmda=nu*dt/dy/dy
fun=0.0                 !! Initializing matrix fun=0.0

!! Calculating the values of velocity of particles with time and distance

DO i=1,100,1
        IF(i<=10) then
                fun(10)=i
        ELSE
                fun(10)=10
        ENDIF
        DO j=1,9
                funw(j)=lmda*(fun(j+1)+fun(j-1)-2.0*fun(j))+fun(j)
        END DO
        vel(i)=funw(9)
        fun=funw
END DO

!! Printing the values of the velocity in file

open(20,file='velocity.res')
write(20,*) 'velocity profile at y=.9'
write(20,*) (vel(i),i=1,100)
STOP
END
```

Results

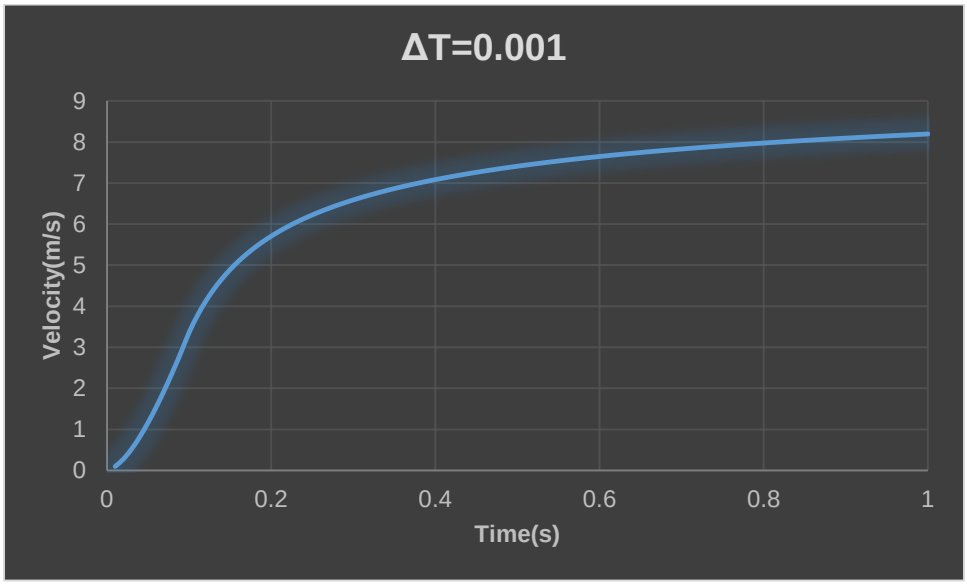
[The results were obtained for Y=0.9 cm and were plotted in Microsoft Excel]

The results are plotted at Y=0.9m for two values of ΔT= 0.01 and 0.001 for ΔZ=0.1 and μ=1.

- ΔT= 0.01 thus Lambda = 1 > 0.5 thus the graph has a lot of noise



- ΔT= 0.001 thus Lambda = .1 < 0.5 thus the graph has no noise and is stable



Implicit Method

[Explicit methods calculate the state of a system at a later time from the state of the system at the current time, while implicit methods find a solution by solving an equation involving both the current state of the system and the later one.]

In implicit finite difference schemes, the spatial derivatives $\frac{\partial^2 T}{\partial x^2}$ are evaluated (at least partially) at the new time step. The simplest implicit discretization of equation 1 is

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \kappa \frac{T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}}{\Delta x^2} \tag{5}$$

This can be rearranged so that unknown terms are on the left and known terms are on the right

$$-sT_{i+1}^{n+1} + (1+2s)T_i^{n+1} - sT_{i-1}^{n+1} = T_i^n \tag{6}$$

Where s = κΔt/Δx^2. Note that in this case we no longer have an explicit relationship for T_{i-1}^{n+1}, T_i^{n+1} and T_{i+1}^{n+1}. Instead, we have to solve a linear system of equations (now you know why we repeated linear algebra in the first lesson...). The main advantage of implicit finite difference methods is that there are no restrictions on the time step, which is good news if we want to simulate geological processes at high spatial resolution. Taking large time steps, however, may result in an inaccurate solution. Therefore it is always wise to check the results by decreasing the time step until the solution doesn't change anymore (this is called converge check).

The implicit method described in equation 6 is second order accurate in space but only first order accurate in time (i.e., O(Δt,Δx^2)). It is also possible to create a scheme which is second order accurate both in time and in space (i.e., O (Δt^2, Δx^2)).

Program Developed in FORTRAN

[Fortran (previously FORTRAN, derived from Formula Translating System) is a general-purpose, imperative programming language that is especially suited to numeric computation and scientific computing.]

```
implicit none

!! Declaring the variables

real, dimension (100,100):: MatA
real, dimension (100):: UO, UN,R
real :: lambda,nu,dt,dz,tim
integer :: i0,j0,k0,num

!! Takin Input Parameters from the user

write(*,*) 'Input the kinematic viscosity (Neu) sec/m2'
read(*,*) nu
write(*,*) 'Input the time difference (dt) sec'
read(*,*) dt
write(*,*) 'Input the distance difference (dy) m'
read(*,*) dy

!! Assigning the values to the variables
```

```
lambda=nu*dt/(2*dz*dz)
MatA=0.0
MatA(1,1)=1
MatA(11,11)=1
UO=0
UN=0
R=0
num=1/dt

!! Assigning values to the Matrix MatA

MatA(2,1)=0
MatA(10,9)=(-1)*lambda
MatA(10,10)=1+2*lambda
DO i0=2,9
        MatA(i0,i0-1)=(-1)*lambda
        MatA(i0,i0)=1+2*lambda
        MatA(i0,i0+1)=(-1)*lambda
END DO

!! Calculation values in Loops

DO j0=1,num
        tim=j0*dt
        IF (tim.le.0.1) then
                UN(11)=  100*tim
        ELSE
                UN(11)=10
        END IF
        R(11)= UN(11)
        DO i0=2,9
                R(i0)=lambda*(UO(i0+1)+UO(i0-1)-2*UO(i0))+UO(i0)
                R(10)=lambda*(UO(11)+UO(9)-2*UO(10))+UO(10)+lambda*(R(11))
        END DO
                open(5, file='Result.txt')    !! Printing Values in Result File
                write(5,*) UN(10),',',tim
        DO i0=1,11
                UO(i0)=UN(i0)
        END DO
END DO
STOP
END

!! Subroutine function of Gaussian  Elimination Methods

SUBROUTINE Gauss(N,MatA,B,X)

implicit none
real,dimension(100,100) :: MatA
real,dimension(100,1) :: B,X
integer :: N,i0,j0,k0
real:: temp=0

!I Forward Elimination Program
```

```
 DO k0=1,N-1,1
        DO i0=k0+1,N,1
                IF( ABS(MatA(k0,k0)) < 1.0E-08) then
                        write(*,*) 'Input too Small'
                        STOP
                END If
                DO j0=k0+1,N,1
                        MatA(i0,j0)=MatA(i0,j0)-MatA(k0,j0)*MatA(i0,k0)/MatA(k0,k0)
                END DO
                B(i0,1)=B(i0,1)-B(k0,1)*MatA(i0,k0)/MatA(k0,k0)
                MatA(i0,k0)=MatA(i0,k0)-MatA(k0,k0)*MatA(i0,k0)/MatA(k0,k0);
        END DO
END DO
```

!! Backward Substitution

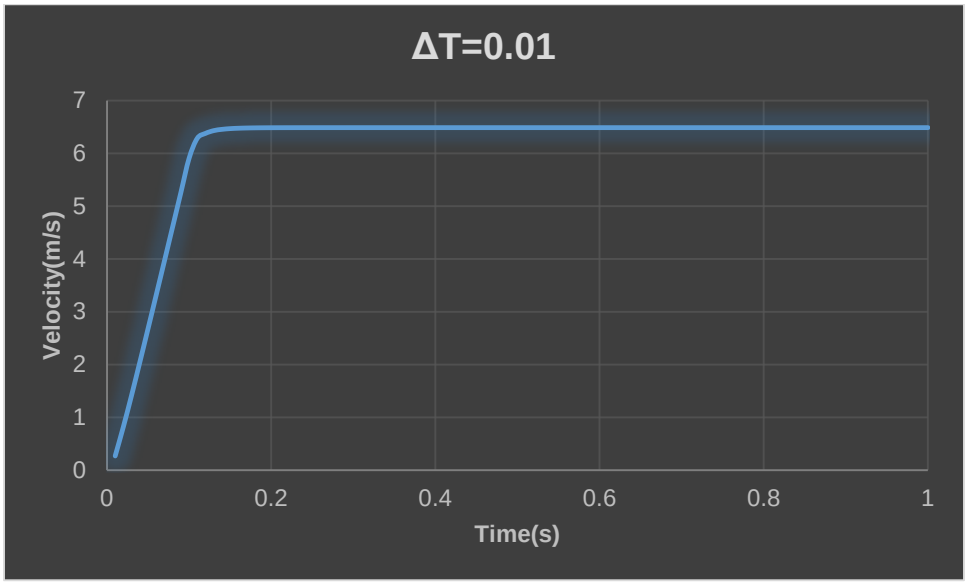
```
DO k0=N,1,-1
        DO i0=N,k0,-1
                temp=temp+X(k0,1)
        END DO
        X(k0,1)=(B(k0,1)-temp)/MatA(k0,k0)
        temp=0
END DO
RETURN
END
```

Results

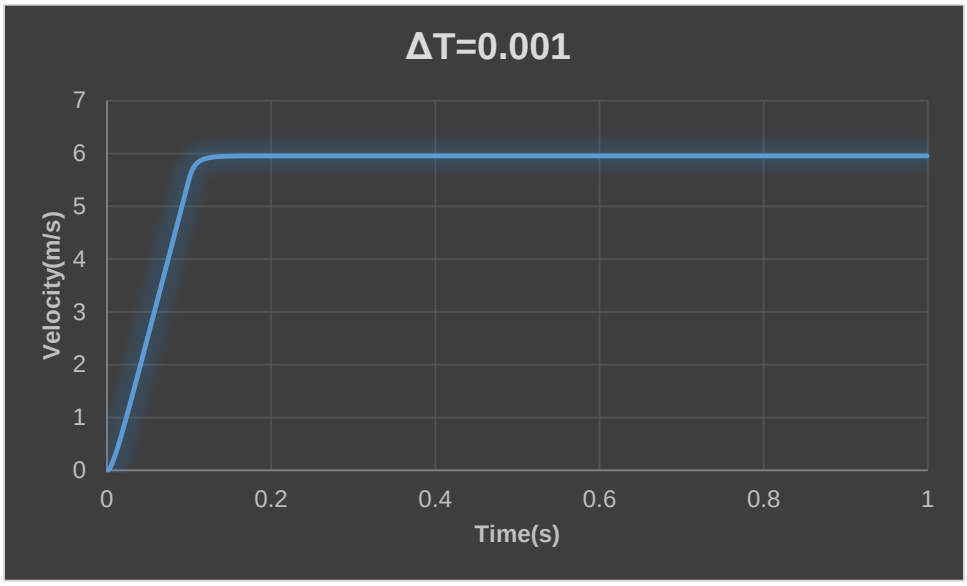
> *[The results were obtained for Y=0.9 cm and were plotted in Microsoft Excel]*

The results are plotted at Y=0.9m for two values of ΔT= 0.01 and 0.001 for ΔZ=0.1 and μ=1.

- ΔT= 0.01 thus Lambda = 1. The graph does not have a noise and is stable



- ΔT= 0.001 thus Lambda = .1 < 0.5 thus the graph has no noise and is stable



RESULT COMPARISON OF EXPLICIT AND IMPLICIT METHODS

From the above example given in the assignment the following observations and conclusions can be made.

- From the above graphs of implicit and explicit methods to the given problem it is clear that explicit function gives the result fast and is computationally fast but there is an issue of stability whereas on the other hand implicit method gives stable result with any value of ΔT, ΔZ and η combination.
- The explicit method gives stable results only when the value of lambda= $\Delta T * \eta / (\Delta Z)^2 < 0.5$.
- Explicit Function is computationally time and space efficient whereas Explicit has high complexity of time and space.
- Implicit Scheme is stable for all values of ΔT, ΔZ and η combination

Thus both the methods have their advantages and disadvantages. Thus one must use the approach depending upon the complexity and the time required to solve the problem.