MECE 5397: Scientific Computing for Mechanical Engineers

Solution of the discretized diffusion equation in two spatial dimensions with several time integration schemes

David Newman

TODO: Table of Contents page

# Abstract

Several different time integration schemes are explored in the pursuit of solving the diffusion equation in two spatial dimensions. Grid independence as well as error analyses are carried out for several relevant time steps in the domain of interest. Explorations are undertaken to determine the effective CPU run time for each discretization as the number of discrete grid points in the discretization is increased. Finally, optimization in the MATLAB code used to determine the solution is discussed.

# Problem Statement

Write a computer code to solve the two-dimensional diffusion equation

on the domain

Subject to

# Discretizations

For the purposes of this assignment, the solution to the differential equation will be denoted by *u* and the exact solution will be noted by *uexact*. For compactness, the solution at time coordinate *tn* and spatial coordinates *xj* and *yk* will be given by

The second order centered discretization of the second derivatives in space will be denoted,

With this notation in place, the Crank-Nicolson discretization of the diffusion equation is a two-level time scheme given by

Which we know is unconditionally stable for Dirichlet boundary conditions and second order accurate in time and space. Note that the diffusion coefficient, *D*, is shown in equation (7) and hereafter to allow for a general discussion of discretizations of the diffusion equation.

While the Crank-Nicolson method will certainly solve the differential equation given in (1), its implementation in two spatial dimensions will require the solution of a matrix with five diagonals. There is no simple algorithm for solving such a system, so we will instead use the Alternating Direction Implicit (ADI) scheme which is similar in form. The ADI scheme is characterized by taking two half steps in time, one implicit in x and one implicit in y. Refer to equation (8) and equation (9).

The ADI scheme results in a tri-diagonal matrix structure for each half step for which there are efficient algorithms to solve. In addition, the ADI scheme is second order accurate in the temporal and spatial domains.

Another method used to solve the equation in (1) is the explicit method, so named for the capability of isolating a single unknown at each time step. The discretization is shown in equation (10).

This method is efficient to solve, however stability analysis reveals the stability criteria given in equation (11) which is extremely restrictive for two dimensions of space.

For a fine spatial grid spacing, the time step will have to be reduced to extremely small levels to meet the stability criteria.

# Description of the Numerical Method

**A Note on Implementing Neumann Boundary Conditions**

Neumann boundary conditions are implemented for both schemes using the second order centered difference formula given by equation (12) which leads to the creation of so called “ghost nodes” which lie outside the domain of interest.

For this discussion, we effectively replace the right-hand sides of equations of equations (2) and (3) with *v* and *w* for generality. At the first row in the domain we have *k = 0*,

So that

Similarly, for the last row in the domain we have *k = N* and following a similar procedure we obtain

These expressions are then inserted into the discretization of the scheme providing for a second order accurate representation of the first derivative specified in equations (2) and (3).

**ADI Scheme**

For the ADI scheme, the following pseudocode describes the algorithm used to solve the problem:

1. Obtain all relevant parameters that specify the problem such as boundary conditions in space, an initial condition in time, and constant coefficients.
2. Compute derived parameters from step (1) that will be used frequently in the scheme.
3. Build the diagonals of each half step array to be solved. These arrays stay the same for each time step.
4. Main loop:
   1. Create the right-hand side for each tri-diagonal system in the first half step (each row of points in the spatial domain).
   2. Solve the first half step.
   3. Create the right-hand side for each tri-diagonal system in the second half step (each column of points in the spatial domain).
   4. Solve the second half step.
   5. Repeat starting at step *a)* until a specified number of iterations has been reached.

The result of this computation is a three-dimensional array with each page representing the results from a single time step, or equivalently, a single iteration of the main loop.

Some rearrangement of equations (8) and (9) reveals the following equation for the first and second half steps, respectively.

Where and

Equation (12) is iterated for each row of points in the domain and results in a tri-diagonal system. Equation (13) is iterated for each column of points in the domain and results in a similar structure. It should be noted that the lower and upper boundaries are included with the interior points as unknowns because of the Neumann conditions given by equations (2) and (3). The resulting system for each half step is shown below:

While this represents the large-scale structure of the problem, in practice each row or column may be solved separately as its own tri-diagonal system. Only the diagonal elements of each array are stored in computer memory since both systems are sparse. Each system is solved using the Thomas algorithm.

**Explicit Scheme**

The following pseudocode is used to solve equation (1) via the explicit scheme:

1. Obtain all relevant parameters that specify the problem such as boundary conditions in space, an initial condition in time, and constant coefficients.
2. Compute derived parameters from step (1) that will be used frequently in the scheme.
3. Main Loop:
   1. Loop through first and last row simultaneously and determine the value of the solution at these points. Special treatment is required here due to the upper and lower boundary conditions.
   2. Loop through interior rows of the domain and determine the value of the solution at these points.

The explicit scheme given by (10), after solving for the single unknown is given by equation (14).

There is no matrix system to solve for this scheme, making it very simple and quick to construct and execute.

# Technical Specifications of Machine Used for Simulations

Table 1. Technical specifications of computer used for simulations

|  |  |
| --- | --- |
| CPU Model | Intel Core i7-7700K |
| Number of Cores | 4 |
| Number of Threads | 8 |
| Base Frequency | 4.20 GHz |
| Cache | 8 MB SmartCache |
| TDP | 91 W |
| Main Memory Quantity | 16 GB (2 x 8 GB DDR3) |
| Storage (Boot) | 250 GB (M.2 NVMe SSD) |
| Storage (HDD) | 1 TB |
| GPU Model | NVIDIA GeForce GTX 1080 |