Diffusion Equation

Vegard Falmår and Sigurd Sørlie Rustad

University of Oslo
Norway

December 4, 2020

CONTENTS

Abstract

I. INTRODUCTION

For our studies we have used c++ for heavy computation, python for visualization and automation. All the code along with instructions on how to run it, can be cloned from our GitHub repository[1].

II. THEORY

The diffusion equation

The full diffusion equation reads

$$\frac{\partial u(\mathbf{r},t)}{\partial t} = \nabla \cdot \left[D(u,\mathbf{r})\nabla u(\mathbf{r},t)\right],$$

with $\mathbf{r}$ is a positional vector. If $D(u,\mathbf{r}) = 1$ the equation simplifies to

$$\frac{\partial u}{\partial t} = \nabla^2 u(\mathbf{r},t),$$

or

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right) u(x,y,z,t) = \frac{\partial u(x,y,z,t)}{\partial t} \qquad (1)$$

in cartesian coordinates. In this report we are mainly going to work with the diffusion equation in one and two dimensions, i.e.

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t} \;\wedge\; \frac{\partial^2 u(x,t)}{\partial x^2} + \frac{\partial^2 u(x,t)}{\partial y^2} = \frac{\partial u(x,t)}{\partial t}.$$

Discretization

Equation (1) in one dimension reads

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t} \quad \text{or} \quad u_{xx} = u_t. \qquad (2)$$

With $x \in [0, L]$ and boundary conditions

$$u(0,t) = a(t), \quad t \geq 0 \quad \wedge \quad u(L,t) = b(t), \quad t \geq 0, \qquad (3)$$

we can approximate the solution by discretization. First introducing Δx and Δt as small steps in x-direction and time. Then we can define the value domain of t and x,

$$t_j = j\Delta t, \quad j \in \mathbb{N}_0 \quad \wedge \quad x_i = i\Delta x, \quad \{i \in \mathbb{N}_0 | i \leq n+1\}.$$

Where n is the number of points we evaluate $u(x,t)$ at, minus one.

[1] github.com/sigurdru/FYS3150/tree/master/Project5

Explicit and implicit schemes

Something on this here maybe?

Explicit forward Euler

The algorithm for explicit forward Euler in one dimension (from [1] chapter 10.2.1) reads

$$\begin{aligned} u(x_i, t_j + \Delta t) =& \alpha u(x_i - \Delta x, t_j) + (1-2\alpha)u(x_i, t_j) + \\ & \alpha u(x_i + \Delta x, t_j) \end{aligned}$$

where

$$\alpha = \frac{\Delta t}{\Delta x^2},$$

and the discretization is explained in the appropriate section.

Implicit Backward Euler

$$u_{xx} = \frac{u(x_i + \Delta x, t_j)2u(x_i, t_j + u(x_i - \Delta x, t_j))}{\Delta x^2} \qquad (4)$$

Implicit Crank-Nicolson

$$\begin{aligned} u_{xx} \approx& \frac{1}{2}\Bigg(\frac{u(x_i+\Delta x, t_j) - 2u(x_i, t_j) + u(x_i - \Delta x, t_j)}{\Delta x^2} + \\ & \frac{u(x_i + \Delta x, t_j + \Delta t) - 2u(x_i, t_j + \Delta t)}{\Delta x^2} + \\ & \frac{u(x_i - \Delta x, t_j + \Delta t)}{\Delta x^2}\Bigg) \end{aligned}$$

III. METHODS

Method

IV. RESULTS

Results

V. DISCUSSION

Discussion

VI. CONCLUSION

Conclusion

VII. APPENDIX

Appendix

[1] Morten Hjorth-Jensen, Computational Physics, Lecture Notes Fall 2015, August 2015, https://github.com/CompPhysics/ComputationalPhysics/blob/master/doc/Lectures/lectures2015.pdf.