Implementing of a simple Heat-Convection Model

Florian Nikos Kitzka

May 27, 2020

Abstract

The aim of this document is to provide a simple guide how to create and implement a model for a heat-convection problem. Our approach is to start from underlying physics to pose the model-equations and then to explain difficulties arising during implementation. The full implemented result is provided by an external link, which deems to encourage the reader to extend and play with given implementation. We assume basic knowledge of Analysis and Linear Algebra as well as some understanding of Fluid Dynamics. Chapters are written quite independent of each other so that the reader can skip parts in which no interest exist.

Contents

1 Simulation Target

We consider a room of cubical shape.
We assume all walls except the bottom to allow transfer of air (open walls).
The bottom is assumed to be heated by the sun-light through radiation.

Our goal is to simulate how this heat is being transferred through the room over time. TODO picture

2 Creation of Model

There are two main mechanism of heat transfer in nature. One is diffusion the other is advection.

$$\begin{aligned} \frac{\partial T}{\partial t} &= -\mathbf{v} \cdot \nabla T + \frac{k}{\rho \alpha} \Delta T \\ \frac{\partial v_z}{\partial t} &= g \frac{T - T_A}{T_A} - v_z \frac{\partial v_z}{\partial z} \end{aligned} \tag{1}$$

3 Numerical Scheme

In order to solve the model equations (1) numerically we are going to use the so called Finite Difference Method. This requires to lay a fixed grid over our domain (the room) and replace all derivative expressions by discrete approximations.

3.1 Derivative Approximations

The main method to obtain such approximations for derivatives is using Taylor expansion. We give a detailed description for one specific derivative expression but leave the remaining for the reader since they are produced in analogy.
Also, for reasons becoming clear later, we restrict our attention to 1-dimensional functions. Consider the two Taylor expansions,

$$u(x+h) = u(x) + u'(x)h + \frac{1}{2}u''(x)h^2 + o(2)$$

$$u(x-h) = u(x) - u'(x)h + \frac{1}{2}u''(x)h^2 + o(2)$$

By adding both equations we obtain

$$u(x+h) + u(x-h) = 2u(x) + u''(x)h^2$$

By solving the later for $u''(x)$ we get a second-order approximation for the second derivative of u.

$$u''(x) = \frac{1}{h^2}(u(x+h) + u(x-h) - 2u(x)) + o(2)$$

For the numerical treatment we have to consider u discrete in space and time. We define

$$u_i^n = u(t_n, x_i)$$

or in case of 2-dimensions

$$u_{i,j}^n = u(t_n, x_i, y_j)$$

With this we can now easily list the approximations we are going to use, especially the one we just obtained for second-order derivatives.

$$\begin{aligned}
\left[\frac{\partial u}{\partial t}\right]_i^n &\approx \tfrac{1}{\Delta t}(u_i^{n+1} - u_i^n) \\
\left[\frac{\partial u}{\partial x}\right]_i^n &\approx \tfrac{1}{\Delta x}(u_i^n - u_{i-1}^n) \qquad (2) \\
\left[\frac{\partial u}{\partial x}\right]_i^n &\approx \tfrac{1}{\Delta x}(u_{i+1}^n - u_i^n) \qquad (3) \\
\left[\frac{\partial^2 u}{\partial x^2}\right]_i^n &\approx \tfrac{1}{\Delta x^2}(u_{i-1}^n - 2u_i^n + u_{i+1}^n)
\end{aligned}$$

For first derivative we have provided two approximations, (3) is called forward in space and (2) is called backward in space.

Note, we only present a small selection of possible ways to approximate derivatives. Also, since this article deems as an introduction we only focused on explicit schemes, but the reader should be aware there exists another important category of so called implicit schemes. For a more comprehensive list and treatment the reader is referred to standard literature of finite difference method.

3.2 Advection Scheme

//TODO describe which term in model equiation this is refering to

The following scheme is used for advection with positive velocity ($c > 0$)

$$u_i^{n+1} = u_i^n - c\frac{\Delta t}{\Delta x}(u_i^n - u_{i-1}^n) \qquad (4)$$

and for negative velocity we use

$$u_i^{n+1} = u_i^n - c\frac{\Delta t}{\Delta x}(u_{i+1}^n - u_{i-1}^n) \qquad (5)$$

The reason for splitting the scheme based on the direction of velocity becomes clear in chapter 3.4.

//TODO describe which term in model equiation this is refering to

3.3 Diffusion Scheme

For diffusion we use the scheme

$$u_i^{n+1} = u_i^n + \alpha\frac{\Delta t}{\Delta x^2}(u_{i-1}^n - 2u_i^n + u_{i+1}^n) \qquad (6)$$

3.4 Stability

One of the most important things with the use of approximation schemes is to ensure stability. Actually one could argue that since we have justified all our derivative approximations by Taylor-expansions we are ready and can blindly implement the discrete equations into a computer-system. This approach turns out to be too naive.
In general each computation is accompanied with rounding errors. Although these can be very small they can sum-up drastically when we have to repeat calculations iterative. Our scheme has the form

$$u_{n+1} = A(u_n) \tag{7}$$

where A is some linear function.
If we assume the u_n's to be computed without rounding errors (hypothetically), and our 'realistic' calculated values are $\tilde{u}_n$ then we have

$$\tilde{u}_n = u_n + \epsilon_n \tag{8}$$

where ϵ_n denotes the deviation of step n. Therefore instead of (7) we actually deal with

$$u_{n+1} + \epsilon_{n+1} = A(u_n + \epsilon_n) \tag{9}$$

We observe, at step n we obtained a deviated solution which we use to compute step $n+1$. Thus at step $n+1$ we are faced to the rounding error made in this step and the fact that we were actually using the deviated solution, $u_n + \epsilon_n$, as input. So the very important question arises: How does the ϵ_n's evolve over time?
Combining equations (7), (9) and using linearity of A we obtain

$$\epsilon_{n+1} = A(\epsilon_n) \tag{10}$$

This gives us a direct relation between errors at different steps. The striking idea is to formulate a scheme, that is choose A, so that for all n

$$|\epsilon_{n+1}| \leq |\epsilon_n| \tag{11}$$

Note, if above equation holds, then we can be sure that although we still have to face rounding errors, the fact that we use a deviated version, $u_n + \epsilon_n$, as input in step $n+1$ does not impact the overall outcome of the iteration.
In order to show a given scheme to fulfill (11) we use a method invented by John von Neumann.
The idea behind this is to assume the errors can be approximated (or estimated) by a Fourier-expansion

$$\epsilon(t, x) = e^{at} \sum_k e^{ikx} \tag{12}$$

In other words, some function which decreases or increases exponentially in time. Having any concrete scheme A, we can use above representation of $\epsilon(t, x)$ and use this in (10). Also since A is linear it is enough to restrict attention to a specific k of the expansion.

//TODO show advection and diffusion scheme is stable

3.5 Operator and Term Splitting

You might have wondered that our approximation schemes all were targeted at 1-dimensional problems, but our model is formulated in two dimensions. In turns out that by an elegant trick we actually only need the 1-dimensional approximations.

//TODO give exemplary proof of one split case

3.6 Final Scheme

//TODO

4 Implementation

You can find all code here
`https://github.com/applied-math-coding/basic-diffusion-transport`
and a running example here
`https://applied-math-coding.github.io/basic-diffusion-transport/`.

//TODO descirbe the usfulness of op-splitting in implementation descr use of matrix-slides and functional-progr

5 Further Reading

This article intends to give an introduction into all treated areas. There are many good books or only tutorials about numerical treatment of partial differential equations. Or if you are more specialized on fluid dynamics you will many good accounts on this field too. In case your are more interested in the implementation side, be encouraged to clone the entire project from `https://github.com/applied-math-coding/basic-diffusion-transport` and to extend or play around.