Implementing of a simple Heat-Convection Model

Florian Nikos Kitzka

May 28, 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 it. 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. This all deems as an introduction into the subject. Numerical treatment of Fluid Dynamic belongs to one of the most challenging tasks in numerical mathematics. For the given problem, which is quite easy in each nature our chosen method might work quite satisfactory, but be aware that for other problems much more sophisticated methods are to be taken into consideration.

Contents

1 Simulation Target **2**

2 Creation of Model **2**

3 Numerical Scheme **2**

3.1 Derivative Approximations 2

3.2 Advection Scheme 3

3.3 Diffusion Scheme 4

3.4 Stability 4

3.4.1 Diffusion Scheme 5

3.4.2 Advection Scheme 6

3.5 Operator and Term Splitting 6

3.6 Final Scheme 6

4 Implementation **6**

5 Further Reading **7**

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.
Moreover, as we have seen in section 2, physical properties like heat and momentum are transported by diffusion and advection processes.
Our goal in this section is to provide a generic scheme for both mechanisms. Hereby we are going to consider both processes independent of each other and restrict attention to the 1-dimensional case. It will turn out later that this treatment is already sufficient for the final implementation.

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

In general advaction processes have the form

$$\frac{\partial u}{\partial t} = -c\frac{\partial u}{\partial x} + S$$

Here S denotes a possible source term for the property u.
By using approximations for derivatives as given in section 3.1 we can formulate the following scheme:

For positive velocity ($c \geq 0$):

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

For negative velocity:

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

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

3.3 Diffusion Scheme

Diffusion processes are of the form

$$\frac{\partial u}{\partial t} = \alpha\frac{\partial^2 u}{\partial x^2}$$

and again by using derivative approximations from section 3.1 we formulate the following scheme:

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

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

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

In other words, some function which decreases or increases exponentially in time. Our aim is to find a scheme A which enforces $a < 0$ in above representation of $\epsilon(t, x)$.

3.4.1 Diffusion Scheme

We replace $\epsilon(t, x)$ in equation (11) by using for A the scheme given in (6). Since A is linear we can restrict attention to one specific index k in (12) to obtain

$$e^{a(t+\Delta t)} e^{jkx_i} = e^{at} e^{jkx_i} + \alpha \frac{\Delta t}{\Delta x^2} \left(e^{at} e^{jk(x_i - \Delta x)} - 2e^{at} e^{jkx_i} + e^{at} e^{jk(x_i + \Delta x)} \right)$$

Note the we have used $x_{i-1} = x_i - \Delta x$, $x_{i+1} = x_i + \Delta x$ and $\epsilon(t_{n+1}, x_i) = \epsilon(t_n + \Delta t, x_i)$.
Further by canceling out factors on both sides we get

$$e^{a\Delta t} = 1 + \alpha \frac{\Delta t}{\Delta x^2} \left(e^{-jk\Delta x} - 2 + e^{jk\Delta x} \right)$$

Further by use of Euler's formula the r.h.s can be combined to get

$$e^{a\Delta t} = 1 + \alpha \frac{\Delta t}{\Delta x^2} \left(2cos(k\Delta x) - 2 \right)$$

Since the *cos* always has absolute values below 1 we infer a sufficient condition the l.h.s to be lower 1 is $2\alpha \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2}$ or equivalently,

$$\Delta t \leq \frac{1}{4} \frac{\Delta x^2}{\alpha} \tag{13}$$

In other words, in order to ensure our diffusion-scheme to be stable, we have to impose (13) onto our implementation - we cannot choose arbitrary large time steps Δt for given a grid-width Δx.

3.4.2 Advection Scheme

The same steps as above but by using as A the scheme (4) we obtain

$$e^{at} = 1 - c\frac{\Delta t}{\Delta x}\left(1 - e^{-jk\Delta x}\right)$$

We can use the triangle inequality to estimate

$$e^{at} \leq |1 - c\frac{\Delta t}{\Delta x}| + c\frac{\Delta t}{\Delta x}$$

Note, we have used the known identity $|e^{jy}| = 1$.
From the we can conclude the r.h.s to be lower 1 if $0 \leq c\Delta t/\Delta x \leq 1$ or equivalent

$$\Delta t \leq \frac{\Delta x}{c} \tag{14}$$

For the case of $c < 0$ above consideration would show scheme (4) to never be stable. But if the same steps as above are applied on scheme (5) instead we find the same stability criterion (14).

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. Moreover we have treated diffusion and advection independent of each other whereas in our model they interact with each other.
The secrete behind this is an elegant trick referred as Operator and Term Splitting which allows us in the final implementation to separate dimensions and transport mechanisms. In other words we can implement for each dimension and each transport mechanism a scheme and then combine these scheme which each other.

//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.