Weekly Report

Damien

February 15, 2024

1 Summary

I have a tentative implementation for the sweeping method on the Boltzmann equation. The code does not converge successfully yet and I am still in the process of debugging. However, I believe that I am close to having a working implementation. If you want to view the code it is available on my GitHub under the `boltzmann_research/boltzmann` folder.

2 Quick Review

Here I add a quick review of the problem we are solving to have everything in one place. Most of this is copied from my update in `1_12_24.pdf`. I also filled in some details that this report was missing.

2.1 Simplification of Boltzmann Equation for 2D Maxwell Molecules

Since the collision kernel is constant we can simplify the above integral. Note that both f and ρ (this will be defined later) are functions of $\mathbf{x}$ but we drop this dependence for convenience of notation.

$$\begin{aligned}
\mathcal{Q}(f,f)(\mathbf{v}) = \int_{\mathbb{R}^d}\int_{S^{d-1}} \frac{1}{2\pi}[f(\mathbf{v}'_*)f(\mathbf{v}') - f(\mathbf{v}_*)f(\mathbf{v})]\mathrm{d}\sigma\mathrm{d}\mathbf{v}_* = \\
\int_{\mathbb{R}^d}\int_{S^{d-1}} \frac{1}{2\pi}f(\mathbf{v}'_*)f(\mathbf{v}')\mathrm{d}\sigma\mathrm{d}\mathbf{v}_* - \int_{\mathbb{R}^d}\int_{S^{d-1}} \frac{1}{2\pi}f(\mathbf{v}_*)f(\mathbf{v})\mathrm{d}\sigma\mathrm{d}\mathbf{v}_* = \\
\mathcal{Q}^+(f,f)(\mathbf{v}) - \frac{1}{2\pi}f(\mathbf{v})\int_{\mathbb{R}^d}\int_{S^{d-1}} f(\mathbf{v}_*)\mathrm{d}\sigma\mathrm{d}\mathbf{v}_* = \\
\mathcal{Q}^+(f,f)(\mathbf{v}) - \frac{1}{2\pi}f(\mathbf{v})\int_{\mathbb{R}^d} 2\pi f(\mathbf{v}_*)\mathrm{d}\mathbf{v}_* = \\
\mathcal{Q}^+(f,f)(\mathbf{v}) - C\rho f(\mathbf{v})
\end{aligned} \tag{1}$$

The constant C comes from rescaling that is done in the derivation of the Boltzmann equation. This can be observed in this article. The ρ comes from the fact that we are integrating out velocity from the phase space probability function f which leaves us with only the spatial density, ρ.

2.2 Normal Shock Problem

We will tackle the normal shock problem as stated in section 5.2 of Hu et al. [2]. We take $R = 1$, $d = 2$, hence $\gamma = 2$, $M_L = u_L/\sqrt{2T_L}$. In the following, the spatial domain is chosen as $x_1 \in [-30, , 30]$ with $N_{\mathbf{x}} = 1000$; and the velocity domain is $(v_1, v_2) \in [-L_{\mathbf{v}}, L_{\mathbf{v}}]^2$.

We choose the upstream and downstream conditions as

$$(\rho_L, \rho_R) = \left(1, \frac{3M_L^2}{M_L^2+2}\right), \quad (u_L, u_R) = \left(\sqrt{2}M_L, \frac{\rho_L u_L}{\rho_R}\right), \quad (T_L, T_R) = \left(1, \frac{4M_L^2-1}{\rho_R}\right)$$

and the downstream conditions as

$$\rho_0(x_1) = \frac{\tanh(\alpha x_1)+1}{2(\rho_R-\rho_L)} + \rho_L, \quad T_0(x_1) = \frac{\tanh(\alpha x_1)+1}{2(T_R-T_L)} + T_L, \quad \mathbf{u}_0(x_1) = \left(\frac{\tanh(\alpha x_1)+1}{2(u_R-u_L)}, 0\right),$$

with $\alpha = 0.5$.

When showing the numerical results, we are mainly interested in the macroscopic quantities: density $\rho(x_1)$, bulk velocity $u(x_1)$,and temperature $T(x_1)$. Their normalized calues will be plotted, which are defined by

$$\hat{\rho}(x_1) = \frac{\rho(x_1) - \rho_L}{\rho_R - \rho_L}, \quad \hat{u}(x_1) = \frac{u(x_1) - u_L}{u_R - u_L}, \quad \hat{T}(x_1) = \frac{T(x_1) - T_L}{T_R - T_L}.$$

The initial condition for the phase space density function is

$$f_0(x_1, \mathbf{v}, t) = \frac{\rho_0}{(2\pi R T_0)^{d/2}} \exp\left(-\frac{(v_1 - u_0)^2 + v_2^2 + \ldots + v_d^2}{2RT_0}\right)$$

In Hu et al. [2] there is an example of a strong as well as a weak shock. We will model both of these.

2.3 Numerical Scheme for the Normal Shock Problem

For our numerical scheme we will be applying the methods developed in Chen et al. [1] and applying them to the equation described above. For the left-to-right sweep the numerical discretization is

$$\frac{v + |v|}{2} \frac{f_i^{(l+1)} - f_{i-1}^{(l+1)}}{\Delta x} + \frac{v - |v|}{2} \frac{f_{i+1}^{(l)} - f_i^{(l+1)}}{\Delta x} = Q^+(f^{(l)}, f^{(l)}) - C\rho_i^{(l)} f_i^{(l+1)}$$

and for the right-to-left sweep the numerical discretization is

$$\frac{v + |v|}{2} \frac{f_i^{(l+1)} - f_{i-1}^{(l)}}{\Delta x} + \frac{v - |v|}{2} \frac{f_{i+1}^{(l+1)} - f_i^{(l+1)}}{\Delta x} = Q^+(f^{(l)}, f^{(l)}) - C\rho_i^{(l)} f_i^{(l+1)}.$$

The update rule is acquired by isolating $f_i^{(l+1)}$ on one side.

3 Progress

I implemented the above method in Python. There is still a bug or two in the code and things are blowing up. I have parsed through my code carefully to check for any sort of typo. I could not find any. My issues are likely due to some misunderstanding that I have as opposed to a simple arithmetic error. Here I outline the details of my code to illustrate my understanding and how I chose to implement the method. All the code can be found under the `boltzmann` folder on my GitHub repository. Add GitHub link

3.1 Validation of Integration Scheme

The first and easiest thing to validate were my integration schemes for recovering the density, bulk velocity, and temperature of the solution. Recall that

$$\int_{\mathbb{R}^d} f(\mathbf{x}, \mathbf{v}, t) \begin{bmatrix} 1 \\ \mathbf{v} \\ \frac{1}{2}|\mathbf{v}|^2 \end{bmatrix} \mathrm{d}\mathbf{v} = \begin{bmatrix} \rho(\mathbf{x}, t) \\ \rho(\mathbf{x}, t)\mathbf{u}(\mathbf{x}, t) \\ \frac{1}{2}\rho(\mathbf{x}, t)|\mathbf{u}(\mathbf{x}, t)|^2 + \frac{d}{2}\rho(\mathbf{x}, t)RT(\mathbf{x}, t) \end{bmatrix}.$$

We use these formulas to recover the density, bulk velocity, and temperature of the solution. In the following plots we confirm that my methods for computing the integrals are accurate.

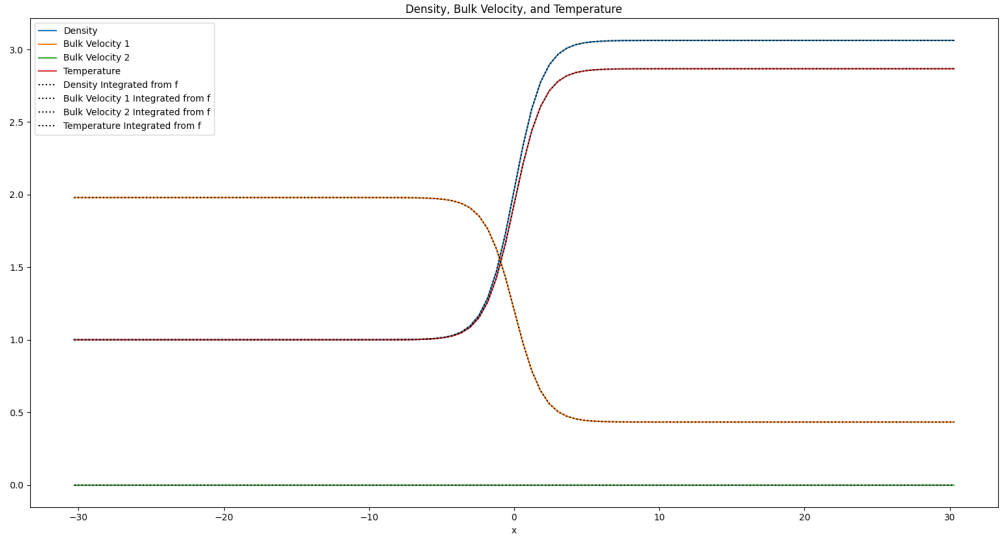

Figure 1: The solid lines are the initial distributions for the the density, bulk velocity, and temperature. The dotted lines are the curves that we recover after integrating the phase space density function, f.

3.2 Potential Problem Areas

There are three places where I may have gone wrong with my code. In each of the following three sections I outline what I did in each of these sections to help expedite the identification of the issue.

3.2.1 Scaling of Phase Distribution Function

When the velocity is integrated out of f_0, we recover ρ_0. ρ_0 is the physical density of the system and not a probability density. This implies that f_0 is not a probability distribution in the phase space. This is contrary to what I had believed previously. Should f be a physical density function or a probability density function? If f is supposed to be a probability density then how do we determine the total mass of the system? Furthermore, since $\int f(\mathbf{x}, \mathbf{v}, t)\mathrm{d}\mathbf{v}$ integrates to $\rho(\mathbf{x}, \mathbf{v}, t)$ I set $C = 1$ in Equation 1.

3.2.2 Collision Operator

I changed the code in `CBoltz2_Carl_Maxwell` slightly to return only the gain term of the collision operator as opposed to the entire collision operator. I call this adjusted function `Q_plus`.

```
def Qplus(f, N, R, L, Ntheta):
    """
    Carleman spectral method for the classical Boltzmann collision operator
    2D Maxwell molecule
    N # of Fourier modes: f(N,N), Q(N,N)
    theta: mid-point rule
    """
    temp = np.concatenate((np.arange(0,N//2),np.arange(-N//2,0,1)))
    l1 = np.array([[row]*N for row in temp])
    l2 = l1.T

    FTf = fft2(f)

    QG = np.zeros((N, N), dtype=np.complex_)
```

```
    bb = np.zeros((N, N), dtype=np.complex_)

    wtheta = np.pi / Ntheta
    theta = np.arange(wtheta / 2, np.pi, wtheta)
    sig1 = np.cos(theta)
    sig2 = np.sin(theta)

    for q in range(Ntheta):
        aa1 = alpha2(l1 * sig1[q] + l2 * sig2[q], R, L)
        aa2 = alpha2(np.sqrt(l1**2 + l2**2 - (l1 * sig1[q] + l2 * sig2[q])**2), R, L)

        QG += 2 * wtheta * ifft2(aa1 * FTf) * ifft2(aa2 * FTf)
        bb += 2 * wtheta * aa1 * aa2

    # Original code ##################
    # QL = f * ifft2(bb * FTf)
    # Q = np.real(QG - QL)
    ##################################

    # Adjusted code ##################
    Q = np.real(QG)
    ##################################

    return Q
```

We do this so that we can make the stepping method implicit as we see in the next section.

3.2.3 Sweeping Method

The second source of error is in the sweeping method. I will describe this method for the case where $v > 0$ to simplify notation. For the forward direction this gives

$$v\frac{f_i^{(l+1)} - f_{i-1}^{(l+1)}}{\Delta x} = Q^+(f^{(l)}, f^{(l)}) - C\rho_i^{(l)} f_i^{(l+1)}.$$

Re-arranging terms we end up with

$$f_i^{(l+1)} = \left(1 + \frac{C\rho_i^{(l)}\Delta x}{v}\right)^{-1} \left(\frac{\Delta x}{v} Q^+\left(f^{(l)}, f^{(l)}\right) + f_{i-1}^{(l+1)}\right). \tag{2}$$

All the tests I have run have been with $C = 1$ and $v = 1$. Let's put this in Python notation to be as clear as possible about how the computation works. In the algorithm, `f` is a $1001 \times 32 \times 32$ array. The first dimension is the spatial dimension and the other dimensions are velocity dimensions. We use the line `Qplus_grid = get_Qplus_grid(f,Nv,R,Lv,Ntheta)` to get the gain term in the collision operator at every point in phase space. The `get_Qplus_grid` function applies the `Qplus` function to f for each x-value on the grid. The code is the following.

```
def get_Qplus_grid(f,Nv,R,Lv,Ntheta): # get the gain term of the collision operator on the grid
    Q_plus_grid = np.empty(f.shape)
    for i in range(f.shape[0]):  # Iterate over the second dimension.
        Q_plus_grid[i,:,:] = Qplus(f[i,:,:],Nv,R,Lv,Ntheta)
    return Q_plus_grid
```

`Qplus` is a $1001 \times 32 \times 32$ array, same as `f`. Translating Equation 2 into code we get

```
def sweep(f,p,Qplus_grid,v,C,dx,n): # sweeping method for the Boltzmann equation
    # Forward Sweep
    for i in range(1,n):
        f[i,:,:] = ((dx/v) * Qplus_grid[i,:,:] + f[i-1,:,:])/(1 + (C * p[i] * dx)/v)
    # Backward Sweep
    for i in range(n-1,0,-1):
        f[i,:,:] = ((dx/v) * Qplus_grid[i,:,:] + f[i-1,:,:])/(1 + (C * p[i] * dx)/v)
    return f
```

3.3 Results

The code runs and does not immediately break. However, the density function quickly starts to diverge as one can observe in the following images.

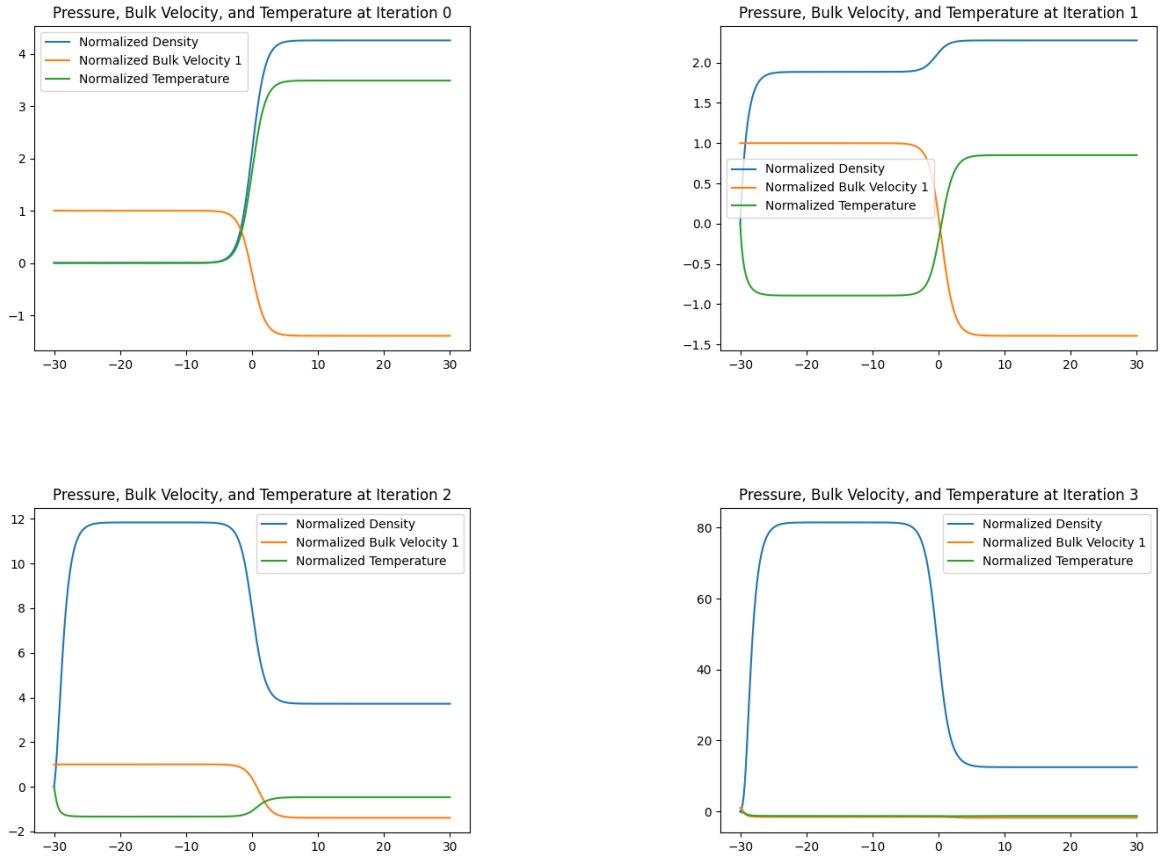

Figure 2: The above plots show the normalized density, bulk velocity, and temperature from the normal shock problem. These plots should ideally converge to look like the right plot in Figure 2 for [2]. However, the density function continually grows without bound. All experiments were done with $v = 1$

4 To Do

The main priority is to debug the code. I have some ideas for how to diagnose what is going wrong with the code.

1. I could forget the sweeping method and solve the time-dependent equation to ensure that it works correctly

2. I can explore the value of the Knudsen number. There is perhaprs some scaling that I am not accounting for

3. I can make test cases for the collision operator function in python to ensure that my adjusted function is correct

4. I can take out the collision operator to see whether the method converges to a constant steady state as it should

Once this is done I plan to implement the shock problem given to me by Jingwei two meetings ago.

References

[1] Weitao Chen, Ching-Shan Chou, and Chiu-Yen Kao. Lax–friedrichs fast sweeping methods for steady state problems for hyperbolic conservation laws. *Journal of Computational Physics*, 234:452–471, 2013.

[2] Jingwei Hu and Yubo Wang. An adaptive dynamical low rank method for the nonlinear boltzmann equation, 2021.