Chong Zhen Jie A0201613Y
MA5232: Rarefied Gas Dynamics

Project 2

This pdf document answers all three questions to the project. The full code for question 3 is appended at the end of question 3 itself.

Question 1.

The original collision operator is as follows:

$$\begin{aligned} Q[f,g](\boldsymbol{v}) =& \frac{1}{2}\int_{\mathbb{R}^3}\int_0^{\pi}\int_{\boldsymbol{n}\perp\boldsymbol{g}} |\boldsymbol{g}|B(|\boldsymbol{g}|,\chi) \\ & \times [f(\boldsymbol{v'})g(\boldsymbol{v_1'}) + g(\boldsymbol{v'})f(\boldsymbol{v_1'}) \\ & - f(\boldsymbol{v})g(\boldsymbol{v_1}) - g(\boldsymbol{v})f(\boldsymbol{v_1})]\,\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1} \end{aligned}$$

For orthogonal matrix $\mathbf{R}$ and arbitrary vector $\boldsymbol{w}$, if $\boldsymbol{v}$ becomes $\mathbf{R}\boldsymbol{v}+\boldsymbol{w}$,

$$\begin{aligned} \text{New relative velocity} &: (\mathbf{R}\boldsymbol{v_1}+\boldsymbol{w}) - (\mathbf{R}\boldsymbol{v}+\boldsymbol{w}) = \mathbf{R}(\boldsymbol{v_1}-\boldsymbol{v}) = \mathbf{R}\boldsymbol{g} \\ \text{Jacobian determinant} &: |\det[\mathbf{J}(\mathbf{R}\boldsymbol{v}+\boldsymbol{w})]| = |\det[\mathbf{J}(\mathbf{R}\boldsymbol{v})]| = |\det(\mathbf{R})| = 1 \end{aligned}$$

Since the initial relative velocity has changed, the collision plane also changes. The rotated collision plane is defined by $\mathbf{R}\boldsymbol{n} \perp \mathbf{R}\boldsymbol{g}$. We define $\boldsymbol{n}_{\mathbf{R}} \perp \boldsymbol{g}_{\mathbf{R}} := \mathbf{R}\boldsymbol{n} \perp \mathbf{R}\boldsymbol{g}$. The Jacobian is similarly computed to be 1.

Due to orthogonality of matrix $\mathbf{R}$, the magnitude of the new relative velocity is

$$|\mathbf{R}\boldsymbol{g}| = \sqrt{(\mathbf{R}\boldsymbol{g})^T(\mathbf{R}\boldsymbol{g})} = |\boldsymbol{g}|$$

From this result,

$$\begin{aligned} \chi &= \chi(|\mathbf{R}\boldsymbol{g}|, b) = \chi(|\boldsymbol{g}|, b) \\ b &= b(|\mathbf{R}\boldsymbol{g}|, \chi) = b(|\boldsymbol{g}|, \chi) \\ \text{Collision kernel} &: B(|\mathbf{R}\boldsymbol{g}|, \chi) = B(|\boldsymbol{g}|, \chi) \end{aligned}$$

The angle χ and the collision kernel B remains unchanged.

Hence we have that,

$$
\begin{aligned}
Q[\tilde{f},\tilde{g}](\boldsymbol{v}) = \frac{1}{2}\int_{\mathbb{R}^3}\int_0^{\pi}\int_{\boldsymbol{n_R}\perp\boldsymbol{g_R}} & |\mathbf{R}\boldsymbol{g}|B(|\mathbf{R}\boldsymbol{g}|,\chi) \\
& \times [\tilde{f}(\boldsymbol{v'})\tilde{g}(\boldsymbol{v_1'}) + \tilde{g}(\boldsymbol{v'})\tilde{f}(\boldsymbol{v_1'}) \\
& - \tilde{f}(\boldsymbol{v})\tilde{g}(\boldsymbol{v_1}) - \tilde{g}(\boldsymbol{v})\tilde{f}(\boldsymbol{v_1})]\ \mathrm{d}\boldsymbol{n_R}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1} \\
= \frac{1}{2}\int_{\mathbb{R}^3}\int_0^{\pi}\int_{\boldsymbol{n_R}\perp\boldsymbol{g_R}} & |\boldsymbol{g}|B(|\boldsymbol{g}|,\chi) \\
& \times [\tilde{f}(\boldsymbol{v'})\tilde{g}(\boldsymbol{v_1'}) + \tilde{g}(\boldsymbol{v'})\tilde{f}(\boldsymbol{v_1'}) \\
& - \tilde{f}(\boldsymbol{v})\tilde{g}(\boldsymbol{v_1}) - \tilde{g}(\boldsymbol{v})\tilde{f}(\boldsymbol{v_1})]\ \mathrm{d}\boldsymbol{n_R}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1} \\
= \frac{1}{2}\int_{\mathbb{R}^3}\int_0^{\pi}\int_{\boldsymbol{n}\perp\boldsymbol{g}} & |\boldsymbol{g}|B(|\boldsymbol{g}|,\chi) \qquad\qquad \text{[Change of variables]} \\
& \times [f(\mathbf{R}\boldsymbol{v'}+\boldsymbol{w})g(\mathbf{R}\boldsymbol{v_1'}+\boldsymbol{w}) + g(\mathbf{R}\boldsymbol{v'}+\boldsymbol{w})f(\mathbf{R}\boldsymbol{v_1'}+\boldsymbol{w}) \\
& - f(\mathbf{R}\boldsymbol{v}+\boldsymbol{w})g(\mathbf{R}\boldsymbol{v_1}+\boldsymbol{w}) - g(\mathbf{R}\boldsymbol{v}+\boldsymbol{w})f(\mathbf{R}\boldsymbol{v_1}+\boldsymbol{w})]\ \mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1} \\
= & Q[f,g](\mathbf{R}\boldsymbol{v}+\boldsymbol{w})
\end{aligned}
$$

□

Question 2.1.

For any function $\psi(\boldsymbol{v})$, it is shown in the lecture notes that

$$\begin{aligned}\int_{\mathbb{R}^3}\psi(\boldsymbol{v})\mathcal{L}[f]\,\mathrm{d}\boldsymbol{v} &= \int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^\pi\int_{\boldsymbol{n}\perp\boldsymbol{g}}\psi(\boldsymbol{v})|\boldsymbol{g}|B(|\boldsymbol{g}|,\chi)\\ &\quad\times K[f](\boldsymbol{v},\boldsymbol{v_1},\boldsymbol{v'},\boldsymbol{v_1'})f_{eq}(\boldsymbol{v})f_{eq}(\boldsymbol{v_1})\,\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1}\,\mathrm{d}\boldsymbol{v}\\ &= \int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^\pi\int_{\boldsymbol{n}\perp\boldsymbol{g}}\psi(\boldsymbol{v})|\boldsymbol{g}|B(|\boldsymbol{g}|,\chi)\\ &\quad\times\left[\frac{f(\boldsymbol{v_1'})}{f_{eq}(\boldsymbol{v_1'})}+\frac{f(\boldsymbol{v'})}{f_{eq}(\boldsymbol{v'})}-\frac{f(\boldsymbol{v_1})}{f_{eq}(\boldsymbol{v_1})}-\frac{f(\boldsymbol{v})}{f_{eq}(\boldsymbol{v})}\right]f_{eq}(\boldsymbol{v})f_{eq}(\boldsymbol{v_1})\,\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1}\,\mathrm{d}\boldsymbol{v}\end{aligned}$$

We apply change of variables to just the first three terms. We also use the fact that by energy conservation,

$$f_{eq}(\boldsymbol{v_1'})f_{eq}(\boldsymbol{v'}) = f_{eq}(\boldsymbol{v_1})f_{eq}(\boldsymbol{v})$$

Transform first term: $(\boldsymbol{v},\boldsymbol{v_1})\leftrightarrow(\boldsymbol{v_1},\boldsymbol{v})$, followed by $(\boldsymbol{v},\boldsymbol{v_1})\leftrightarrow(\boldsymbol{v'},\boldsymbol{v_1'})$

$$\begin{aligned}&\int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^\pi\int_{\boldsymbol{n}\perp\boldsymbol{g}}\psi(\boldsymbol{v})|\boldsymbol{g}|B(|\boldsymbol{g}|,\chi)\times\left[\frac{f(\boldsymbol{v_1'})}{f_{eq}(\boldsymbol{v_1'})}\right]f_{eq}(\boldsymbol{v})f_{eq}(\boldsymbol{v_1})\,\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1}\,\mathrm{d}\boldsymbol{v}\\ =&\int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^\pi\int_{\boldsymbol{n}\perp\boldsymbol{g}}\psi(\boldsymbol{v_1})|\boldsymbol{g}|B(|\boldsymbol{g}|,\chi)\times\left[\frac{f(\boldsymbol{v'})}{f_{eq}(\boldsymbol{v'})}\right]f_{eq}(\boldsymbol{v_1})f_{eq}(\boldsymbol{v})\,\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v}\,\mathrm{d}\boldsymbol{v_1} \qquad \text{[Transform 1]}\\ =&\int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^\pi\int_{\boldsymbol{n}\perp\boldsymbol{g}}\psi(\boldsymbol{v_1'})|\boldsymbol{g}|B(|\boldsymbol{g}|,\chi)\times\left[\frac{f(\boldsymbol{v})}{f_{eq}(\boldsymbol{v})}\right]f_{eq}(\boldsymbol{v_1'})f_{eq}(\boldsymbol{v'})\,\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v}\,\mathrm{d}\boldsymbol{v_1} \qquad \text{[Transform 2]}\\ =&\int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^\pi\int_{\boldsymbol{n}\perp\boldsymbol{g}}\psi(\boldsymbol{v_1'})|\boldsymbol{g}|B(|\boldsymbol{g}|,\chi)\times\left[\frac{f(\boldsymbol{v})}{f_{eq}(\boldsymbol{v})}\right]f_{eq}(\boldsymbol{v_1})f_{eq}(\boldsymbol{v})\,\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v}\,\mathrm{d}\boldsymbol{v_1} \qquad \text{[Energy conservation]}\end{aligned}$$

Transform second term: $(\boldsymbol{v},\boldsymbol{v_1})\leftrightarrow(\boldsymbol{v'},\boldsymbol{v_1'})$

$$\begin{aligned}&\int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^\pi\int_{\boldsymbol{n}\perp\boldsymbol{g}}\psi(\boldsymbol{v})|\boldsymbol{g}|B(|\boldsymbol{g}|,\chi)\times\left[\frac{f(\boldsymbol{v'})}{f_{eq}(\boldsymbol{v'})}\right]f_{eq}(\boldsymbol{v})f_{eq}(\boldsymbol{v_1})\,\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1}\,\mathrm{d}\boldsymbol{v}\\ =&\int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^\pi\int_{\boldsymbol{n}\perp\boldsymbol{g}}\psi(\boldsymbol{v'})|\boldsymbol{g}|B(|\boldsymbol{g}|,\chi)\times\left[\frac{f(\boldsymbol{v})}{f_{eq}(\boldsymbol{v})}\right]f_{eq}(\boldsymbol{v'})f_{eq}(\boldsymbol{v_1'})\,\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1}\,\mathrm{d}\boldsymbol{v} \qquad \text{[Transform]}\\ =&\int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^\pi\int_{\boldsymbol{n}\perp\boldsymbol{g}}\psi(\boldsymbol{v'})|\boldsymbol{g}|B(|\boldsymbol{g}|,\chi)\times\left[\frac{f(\boldsymbol{v})}{f_{eq}(\boldsymbol{v})}\right]f_{eq}(\boldsymbol{v})f_{eq}(\boldsymbol{v_1})\,\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1}\,\mathrm{d}\boldsymbol{v} \qquad \text{[Energy conservation]}\end{aligned}$$

Transform third term: $(\boldsymbol{v},\boldsymbol{v_1})\leftrightarrow(\boldsymbol{v_1},\boldsymbol{v})$

$$\begin{aligned}&\int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^\pi\int_{\boldsymbol{n}\perp\boldsymbol{g}}\psi(\boldsymbol{v})|\boldsymbol{g}|B(|\boldsymbol{g}|,\chi)\times\left[\frac{f(\boldsymbol{v_1})}{f_{eq}(\boldsymbol{v_1})}\right]f_{eq}(\boldsymbol{v})f_{eq}(\boldsymbol{v_1})\,\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1}\,\mathrm{d}\boldsymbol{v}\\ =&\int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^\pi\int_{\boldsymbol{n}\perp\boldsymbol{g}}\psi(\boldsymbol{v_1})|\boldsymbol{g}|B(|\boldsymbol{g}|,\chi)\times\left[\frac{f(\boldsymbol{v})}{f_{eq}(\boldsymbol{v})}\right]f_{eq}(\boldsymbol{v_1})f_{eq}(\boldsymbol{v})\,\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v}\,\mathrm{d}\boldsymbol{v_1} \qquad \text{[Transform]}\end{aligned}$$

We now substitute the transformed terms back to the equation.

$$\begin{aligned}
\int_{\mathbb{R}^3} \psi(\boldsymbol{v})\mathcal{L}[f]\,\mathrm{d}\boldsymbol{v} &= \int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^{\pi}\int_{\boldsymbol{n}\perp\boldsymbol{g}} \psi(\boldsymbol{v})|\boldsymbol{g}|B(|\boldsymbol{g}|,\chi) \\
&\quad\times\left[\frac{f(\boldsymbol{v_1'})}{f_{eq}(\boldsymbol{v_1'})}+\frac{f(\boldsymbol{v'})}{f_{eq}(\boldsymbol{v'})}-\frac{f(\boldsymbol{v_1})}{f_{eq}(\boldsymbol{v_1})}-\frac{f(\boldsymbol{v})}{f_{eq}(\boldsymbol{v})}\right] f_{eq}(\boldsymbol{v})f_{eq}(\boldsymbol{v_1})\;\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1}\,\mathrm{d}\boldsymbol{v} \\
&= \int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^{\pi}\int_{\boldsymbol{n}\perp\boldsymbol{g}} |\boldsymbol{g}|B(|\boldsymbol{g}|,\chi) \qquad\qquad \text{[Substitution]} \\
&\quad\times\left[\psi(\boldsymbol{v_1'})+\psi(\boldsymbol{v'})-\psi(\boldsymbol{v_1})-\psi(\boldsymbol{v})\right]\frac{f(\boldsymbol{v})}{f_{eq}(\boldsymbol{v})} f_{eq}(\boldsymbol{v})f_{eq}(\boldsymbol{v_1})\;\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1}\,\mathrm{d}\boldsymbol{v} \\
&= \int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^{\pi}\int_{\boldsymbol{n}\perp\boldsymbol{g}} |\boldsymbol{g}|B(|\boldsymbol{g}|,\chi) \\
&\quad\times\left[\psi(\boldsymbol{v_1'})+\psi(\boldsymbol{v'})-\psi(\boldsymbol{v_1})-\psi(\boldsymbol{v})\right] f(\boldsymbol{v})f_{eq}(\boldsymbol{v_1})\;\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1}\,\mathrm{d}\boldsymbol{v}
\end{aligned}$$

□

Question 2.2.

We substitute $\psi(\boldsymbol{v}) = 1, \boldsymbol{v}, |\boldsymbol{v}|^2$ separately into $\int_{\mathbb{R}^3} \psi(\boldsymbol{v}) \mathcal{L}[f] \, \mathrm{d}\boldsymbol{v}$ derived in part 1 above.

At $\psi(\boldsymbol{v}) = 1$,

$$\begin{aligned}(2.2.1) \qquad \int_{\mathbb{R}^3} \mathcal{L}[f] \, \mathrm{d}\boldsymbol{v} &= \int_{\mathbb{R}^3} \int_{\mathbb{R}^3} \int_0^\pi \int_{\boldsymbol{n} \perp \boldsymbol{g}} |\boldsymbol{g}| B(|\boldsymbol{g}|, \chi) \\ &\quad \times [1 + 1 - 1 - 1] \, f(\boldsymbol{v}) f_{eq}(\boldsymbol{v_1}) \, \mathrm{d}\boldsymbol{n} \, \mathrm{d}\chi \, \mathrm{d}\boldsymbol{v_1} \, \mathrm{d}\boldsymbol{v} \\ &= \int_{\mathbb{R}^3} \int_{\mathbb{R}^3} \int_0^\pi \int_{\boldsymbol{n} \perp \boldsymbol{g}} 0 \, \mathrm{d}\boldsymbol{n} \, \mathrm{d}\chi \, \mathrm{d}\boldsymbol{v_1} \, \mathrm{d}\boldsymbol{v} \\ &= 0\end{aligned}$$

Next, by momentum conservation,

$$\boldsymbol{v'} + \boldsymbol{v_1'} = \boldsymbol{v} + \boldsymbol{v_1}$$

At $\psi(\boldsymbol{v}) = \boldsymbol{v}$,

$$\begin{aligned}(2.2.2) \quad \int_{\mathbb{R}^3} \boldsymbol{v} \, \mathcal{L}[f] \, \mathrm{d}\boldsymbol{v} &= \int_{\mathbb{R}^3} \int_{\mathbb{R}^3} \int_0^\pi \int_{\boldsymbol{n} \perp \boldsymbol{g}} |\boldsymbol{g}| B(|\boldsymbol{g}|, \chi) \\ &\quad \times \left[\boldsymbol{v_1'} + \boldsymbol{v'} - \boldsymbol{v_1} - \boldsymbol{v}\right] f(\boldsymbol{v}) f_{eq}(\boldsymbol{v_1}) \, \mathrm{d}\boldsymbol{n} \, \mathrm{d}\chi \, \mathrm{d}\boldsymbol{v_1} \, \mathrm{d}\boldsymbol{v} \\ &= \int_{\mathbb{R}^3} \int_{\mathbb{R}^3} \int_0^\pi \int_{\boldsymbol{n} \perp \boldsymbol{g}} 0 \, \mathrm{d}\boldsymbol{n} \, \mathrm{d}\chi \, \mathrm{d}\boldsymbol{v_1} \, \mathrm{d}\boldsymbol{v} \qquad \text{[Momentum conservation]} \\ &= 0\end{aligned}$$

Lastly, by energy conservation,

$$|\boldsymbol{v'}|^2 + |\boldsymbol{v_1'}|^2 = |\boldsymbol{v}|^2 + |\boldsymbol{v_1}|^2$$

At $\psi(\boldsymbol{v}) = |\boldsymbol{v}|^2$,

$$\begin{aligned}(2.2.3) \quad \int_{\mathbb{R}^3} |\boldsymbol{v}|^2 \, \mathcal{L}[f] \, \mathrm{d}\boldsymbol{v} &= \int_{\mathbb{R}^3} \int_{\mathbb{R}^3} \int_0^\pi \int_{\boldsymbol{n} \perp \boldsymbol{g}} |\boldsymbol{g}| B(|\boldsymbol{g}|, \chi) \\ &\quad \times \left[|\boldsymbol{v_1'}|^2 + |\boldsymbol{v'}|^2 - |\boldsymbol{v_1}|^2 - |\boldsymbol{v}|^2\right] f(\boldsymbol{v}) f_{eq}(\boldsymbol{v_1}) \, \mathrm{d}\boldsymbol{n} \, \mathrm{d}\chi \, \mathrm{d}\boldsymbol{v_1} \, \mathrm{d}\boldsymbol{v} \\ &= \int_{\mathbb{R}^3} \int_{\mathbb{R}^3} \int_0^\pi \int_{\boldsymbol{n} \perp \boldsymbol{g}} 0 \, \mathrm{d}\boldsymbol{n} \, \mathrm{d}\chi \, \mathrm{d}\boldsymbol{v_1} \, \mathrm{d}\boldsymbol{v} \qquad \text{[Energy Conservation]} \\ &= 0\end{aligned}$$

From equations (2.2.1), (2.2.2), (2.2.3), we have

$$\int_{\mathbb{R}^3} \begin{pmatrix} 1 \\ \boldsymbol{v} \\ |\boldsymbol{v}|^2 \end{pmatrix} \mathcal{L}[f] \, \mathrm{d}\boldsymbol{v} = 0$$

□

Question 2.3.

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathbb{R}^3} f\,\log\,f\,\mathrm{d}\boldsymbol{v} &= \int_{\mathbb{R}^3}\left(\frac{\partial f}{\partial t}\,\log\,f + f\cdot\frac{1}{f}\frac{\partial f}{\partial t}\right)\,\mathrm{d}\boldsymbol{v}\\ &= \int_{\mathbb{R}^3}\left(\frac{\partial f}{\partial t}\,\log\,f + \frac{\partial f}{\partial t}\right)\,\mathrm{d}\boldsymbol{v}\\ &= \int_{\mathbb{R}^3}(\mathcal{L}[f]\,\log\,f + \mathcal{L}[f])\,\mathrm{d}\boldsymbol{v}\end{aligned}$$

By mass conservation from part 2 above in equation 2.2.1,

$$\int_{\mathbb{R}^3}\mathcal{L}[f]\,\mathrm{d}\boldsymbol{v} = 0$$

Also, using $\int_{\mathbb{R}^3}\psi(\boldsymbol{v})\mathcal{L}[f]\,\mathrm{d}\boldsymbol{v}$ derived in part 1 above with $\psi(\boldsymbol{v}) = \log\,f(\boldsymbol{v})$,

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathbb{R}^3} f\,\log\,f\,\mathrm{d}\boldsymbol{v} &= \int_{\mathbb{R}^3}(\mathcal{L}[f]\,\log\,f + \mathcal{L}[f])\,\mathrm{d}\boldsymbol{v}\\ &= \int_{\mathbb{R}^3}(\mathcal{L}[f]\,\log\,f)\,\mathrm{d}\boldsymbol{v} + 0 \qquad \text{[Mass Conservation]}\\ &= \int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^{\pi}\int_{\boldsymbol{n}\perp\boldsymbol{g}} |\boldsymbol{g}|B(|\boldsymbol{g}|,\chi)\\ &\quad\times[\log\,f(\boldsymbol{v_1'}) + \log\,f(\boldsymbol{v'})\\ &\quad - \log\,f(\boldsymbol{v_1}) - \log\,f(\boldsymbol{v})]\frac{f(\boldsymbol{v})}{f_{eq}(\boldsymbol{v})}f_{eq}(\boldsymbol{v})f_{eq}(\boldsymbol{v_1})\,\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1}\,\mathrm{d}\boldsymbol{v}\end{aligned}$$

We now apply change of variables here. We also use the fact that by energy conservation, the last two terms $f_{eq}(\boldsymbol{v_1})f_{eq}(\boldsymbol{v})$ remain unchanged when transforming the variables due to:

$$f_{eq}(\boldsymbol{v_1'})f_{eq}(\boldsymbol{v'}) = f_{eq}(\boldsymbol{v_1})f_{eq}(\boldsymbol{v})$$

We now apply each transformation separately on the original expression, all indicated by:

$$\begin{aligned}&\text{Transform 1} : (\boldsymbol{v},\boldsymbol{v_1}) \leftrightarrow (\boldsymbol{v_1},\boldsymbol{v})\\ &\text{Transform 2} : (\boldsymbol{v},\boldsymbol{v_1}) \leftrightarrow (\boldsymbol{v'},\boldsymbol{v_1'})\\ &\text{Transform 3} : (\boldsymbol{v},\boldsymbol{v_1}) \leftrightarrow (\boldsymbol{v_1},\boldsymbol{v}),\ \text{followed by } (\boldsymbol{v},\boldsymbol{v_1}) \leftrightarrow (\boldsymbol{v'},\boldsymbol{v_1'})\end{aligned}$$

Hence we have,

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathbb{R}^3} f\,\log\,f\,\mathrm{d}\boldsymbol{v} =& \int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^\pi\int_{\boldsymbol{n}\perp\boldsymbol{g}} |\boldsymbol{g}|B(|\boldsymbol{g}|,\chi) \qquad \text{[Original]}\\
&\times[\log\,f(\boldsymbol{v_1'})+\log\,f(\boldsymbol{v'})\\
&-\log\,f(\boldsymbol{v_1})-\log\,f(\boldsymbol{v})]\frac{f(\boldsymbol{v})}{f_{eq}(\boldsymbol{v})}f_{eq}(\boldsymbol{v})f_{eq}(\boldsymbol{v_1})\,\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1}\,\mathrm{d}\boldsymbol{v}\\
=& \int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^\pi\int_{\boldsymbol{n}\perp\boldsymbol{g}} |\boldsymbol{g}|B(|\boldsymbol{g}|,\chi) \qquad \text{[Transform 1]}\\
&\times[\log\,f(\boldsymbol{v'})+\log\,f(\boldsymbol{v_1'})\\
&-\log\,f(\boldsymbol{v})-\log\,f(\boldsymbol{v_1})]\frac{f(\boldsymbol{v_1})}{f_{eq}(\boldsymbol{v_1})}f_{eq}(\boldsymbol{v})f_{eq}(\boldsymbol{v_1})\,\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1}\,\mathrm{d}\boldsymbol{v}\\
=& \int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^\pi\int_{\boldsymbol{n}\perp\boldsymbol{g}} |\boldsymbol{g}|B(|\boldsymbol{g}|,\chi) \qquad \text{[Transform 2]}\\
&\times[\log\,f(\boldsymbol{v_1})+\log\,f(\boldsymbol{v})\\
&-\log\,f(\boldsymbol{v_1'})-\log\,f(\boldsymbol{v'})]\frac{f(\boldsymbol{v'})}{f_{eq}(\boldsymbol{v'})}f_{eq}(\boldsymbol{v})f_{eq}(\boldsymbol{v_1})\,\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1}\,\mathrm{d}\boldsymbol{v}\\
=& \int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^\pi\int_{\boldsymbol{n}\perp\boldsymbol{g}} |\boldsymbol{g}|B(|\boldsymbol{g}|,\chi) \qquad \text{[Transform 3]}\\
&\times[\log\,f(\boldsymbol{v})+\log\,f(\boldsymbol{v_1})\\
&-\log\,f(\boldsymbol{v'})-\log\,f(\boldsymbol{v_1'})]\frac{f(\boldsymbol{v_1'})}{f_{eq}(\boldsymbol{v_1'})}f_{eq}(\boldsymbol{v})f_{eq}(\boldsymbol{v_1})\,\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1}\,\mathrm{d}\boldsymbol{v}\\
=& -\frac{1}{4}\int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_0^\pi\int_{\boldsymbol{n}\perp\boldsymbol{g}} |\boldsymbol{g}|B(|\boldsymbol{g}|,\chi) \qquad \text{[Average of previous four expressions]}\\
&\times[\log\,f(\boldsymbol{v_1'})+\log\,f(\boldsymbol{v'})-\log\,f(\boldsymbol{v_1})-\log\,f(\boldsymbol{v})]\\
&\times\left[\frac{f(\boldsymbol{v_1'})}{f_{eq}(\boldsymbol{v_1'})}+\frac{f(\boldsymbol{v'})}{f_{eq}(\boldsymbol{v'})}-\frac{f(\boldsymbol{v_1})}{f_{eq}(\boldsymbol{v_1})}-\frac{f(\boldsymbol{v})}{f_{eq}(\boldsymbol{v})}\right]\\
&\times f_{eq}(\boldsymbol{v})f_{eq}(\boldsymbol{v_1})\,\mathrm{d}\boldsymbol{n}\,\mathrm{d}\chi\,\mathrm{d}\boldsymbol{v_1}\,\mathrm{d}\boldsymbol{v}
\end{aligned}$$

The last expression obtained is the average of the previous four expressions. Since the terms $|\boldsymbol{g}|, B(|\boldsymbol{g}|,\chi), f_{eq}$ are defined to be non-negative, we need to show that the following term is non-negative as well:

$$\begin{aligned}
&[\log\,f(\boldsymbol{v_1'})+\log\,f(\boldsymbol{v'})-\log\,f(\boldsymbol{v_1})-\log\,f(\boldsymbol{v})]\\
&\times\left[\frac{f(\boldsymbol{v_1'})}{f_{eq}(\boldsymbol{v_1'})}+\frac{f(\boldsymbol{v'})}{f_{eq}(\boldsymbol{v'})}-\frac{f(\boldsymbol{v_1})}{f_{eq}(\boldsymbol{v_1})}-\frac{f(\boldsymbol{v})}{f_{eq}(\boldsymbol{v})}\right]\geq 0
\end{aligned}$$

By energy conservation,

$$\begin{aligned}
f_{eq}(\boldsymbol{v_1'})f_{eq}(\boldsymbol{v'}) &= f_{eq}(\boldsymbol{v_1})f_{eq}(\boldsymbol{v})\\
\log\,f_{eq}(\boldsymbol{v_1'})+\log\,f_{eq}(\boldsymbol{v'}) &= \log\,f_{eq}(\boldsymbol{v_1})+\log\,f_{eq}(\boldsymbol{v})\\
0 &= -\log\,f_{eq}(\boldsymbol{v_1'})-\log\,f_{eq}(\boldsymbol{v'})+\log\,f_{eq}(\boldsymbol{v_1})+\log\,f_{eq}(\boldsymbol{v})
\end{aligned}$$

From this result, we have:

$$
\begin{aligned}
&\log f(\boldsymbol{v_1'}) + \log f(\boldsymbol{v'}) - \log f(\boldsymbol{v_1}) - \log f(\boldsymbol{v}) + 0 \\
&= \log f(\boldsymbol{v_1'}) + \log f(\boldsymbol{v'}) - \log f(\boldsymbol{v_1}) - \log f(\boldsymbol{v}) \\
&\quad - \log f_{eq}(\boldsymbol{v_1'}) - \log f_{eq}(\boldsymbol{v'}) + \log f_{eq}(\boldsymbol{v_1}) + \log f_{eq}(\boldsymbol{v}) \\
&= \log \frac{f(\boldsymbol{v_1'})}{f_{eq}(\boldsymbol{v_1'})} + \log \frac{f(\boldsymbol{v'})}{f_{eq}(\boldsymbol{v'})} - \log \frac{f(\boldsymbol{v_1})}{f_{eq}(\boldsymbol{v_1})} - \log \frac{f(\boldsymbol{v})}{f_{eq}(\boldsymbol{v})}
\end{aligned} \tag{2.3.1}
$$

We have the following hypothesis used in the linearization of the collision operator:

$$
\begin{aligned}
f &= (1 + \varepsilon\phi) f_{eq} \\
\frac{f}{f_{eq}} &= 1 + \varepsilon\phi
\end{aligned} \tag{2.3.2}
$$

Also, by taking the natural logarithm on both sides,

$$
\begin{aligned}
\log \frac{f}{f_{eq}} &= \log (1 + \varepsilon\phi) \approx \varepsilon\phi \\
\log \frac{f}{f_{eq}} &\approx \varepsilon\phi
\end{aligned} \tag{2.3.3}
$$

Since ε is defined to be a very small constant, equation (2.3.3) is approximated by the fact that $\log (1 + x) \approx x$ for small values of x.

Applying equations (2.3.1), (2.3.2), (2.3.3), we get the following:

$$
\begin{aligned}
&[\log f(\boldsymbol{v_1'}) + \log f(\boldsymbol{v'}) - \log f(\boldsymbol{v_1}) - \log f(\boldsymbol{v})] \\
&\quad \times \left[\frac{f(\boldsymbol{v_1'})}{f_{eq}(\boldsymbol{v_1'})} + \frac{f(\boldsymbol{v'})}{f_{eq}(\boldsymbol{v'})} - \frac{f(\boldsymbol{v_1})}{f_{eq}(\boldsymbol{v_1})} - \frac{f(\boldsymbol{v})}{f_{eq}(\boldsymbol{v})} \right] \\
&= \left[\log \frac{f(\boldsymbol{v_1'})}{f_{eq}(\boldsymbol{v_1'})} + \log \frac{f(\boldsymbol{v'})}{f_{eq}(\boldsymbol{v'})} - \log \frac{f(\boldsymbol{v_1})}{f_{eq}(\boldsymbol{v_1})} - \log \frac{f(\boldsymbol{v})}{f_{eq}(\boldsymbol{v})} \right] \\
&\quad \times \left[\frac{f(\boldsymbol{v_1'})}{f_{eq}(\boldsymbol{v_1'})} + \frac{f(\boldsymbol{v'})}{f_{eq}(\boldsymbol{v'})} - \frac{f(\boldsymbol{v_1})}{f_{eq}(\boldsymbol{v_1})} - \frac{f(\boldsymbol{v})}{f_{eq}(\boldsymbol{v})} \right] \\
&= [\varepsilon\phi(\boldsymbol{v_1'}) + \varepsilon\phi(\boldsymbol{v'}) - \varepsilon\phi(\boldsymbol{v_1}) - \varepsilon\phi(\boldsymbol{v})] \\
&\quad \times [1 + \varepsilon\phi(\boldsymbol{v_1'}) + 1 + \varepsilon\phi(\boldsymbol{v'}) - 1 - \varepsilon\phi(\boldsymbol{v_1}) - 1 - \varepsilon\phi(\boldsymbol{v})] \\
&= [\varepsilon\phi(\boldsymbol{v_1'}) + \varepsilon\phi(\boldsymbol{v'}) - \varepsilon\phi(\boldsymbol{v_1}) - \varepsilon\phi(\boldsymbol{v})]^2 \geq 0
\end{aligned}
$$

Putting everything together,

$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t} \int_{\mathbb{R}^3} f \log f \, \mathrm{d}\boldsymbol{v} = &- \frac{1}{4} \int_{\mathbb{R}^3} \int_{\mathbb{R}^3} \int_0^{\pi} \int_{\boldsymbol{n} \perp \boldsymbol{g}} |\boldsymbol{g}| B(|\boldsymbol{g}|, \chi) \\
&\times [\varepsilon\phi(\boldsymbol{v_1'}) + \varepsilon\phi(\boldsymbol{v'}) - \varepsilon\phi(\boldsymbol{v_1}) - \varepsilon\phi(\boldsymbol{v})]^2 \\
&\times f_{eq}(\boldsymbol{v}) f_{eq}(\boldsymbol{v_1}) \, \mathrm{d}\boldsymbol{n} \, \mathrm{d}\chi \, \mathrm{d}\boldsymbol{v_1} \, \mathrm{d}\boldsymbol{v} \\
\leq &\, 0
\end{aligned}
$$

Asides from the one negative term of $-\frac{1}{4}$, we have shown that the remaining terms are all non-negative. This gives us:

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\mathbb{R}^3} f \, \log \, f \, \mathrm{d}\boldsymbol{v} \leq 0$$

□

Question 3.

[1], [2] are referenced for understanding of Question 3. Also, only the implementation of Monte Carlo integral and Newton's method are displayed within the solution. The remaining code can be found at the end of this question.

3.1. Computing macroscopic quantities using Monte Carlo integral

The spatially homogeneous BGK equations do not change ρ_i, $\boldsymbol{m}_i$ and E_i. Hence, we can compute the macroscopic quantities of the i-th species for $\psi(\boldsymbol{v}) = (1, \boldsymbol{v}, \frac{1}{2}|\boldsymbol{v}-u_i(0)|^2, \frac{1}{2}|\boldsymbol{v}|^2)^T$:

$$\begin{pmatrix} \rho \\ \boldsymbol{m} \\ E_{internal,i} \\ E_i \end{pmatrix} = m_i \int_{\mathbb{R}^3} \psi(\boldsymbol{v}) f_i(\boldsymbol{v}) \, \mathrm{d}\boldsymbol{v} = m_i \int_{\mathbb{R}^3} \psi(\boldsymbol{v}) f_i(\boldsymbol{v}, 0) \, \mathrm{d}\boldsymbol{v}$$

To do that, we use Monte Carlo integration to approximate the values as follows:

$$m_i \int_{\mathbb{R}^3} \psi(\boldsymbol{v}) f_i(\boldsymbol{v}, 0) \, \mathrm{d}\boldsymbol{v} = m_i \frac{(b-a)^3}{N} \sum_{k=1}^{N} \psi(\boldsymbol{v}) f_i(x_k)$$

We perform random sampling from a uniform distribution to generate the velocity grid and integrate over the domain of a three-dimensional cube. Below is the implementation of the Monte Carlo integral.

```
def mc_int(sp, a=a, b=b, fn=f_init, v_list=v_list, num=num, u_init=u_init):
    """
    Args:
    ===
    sp: i-th species
    a: lower limit
    b: upper limit
    fn: distribution function
    v_list: velocity grid
    num: number of velocities in v_list
    u_init: initial mean velocity to compute internal energy

    Output:
    ===
    Computes Monte Carlo integral. Returns macroscopic quantities density, momentum, internal energy, energy.
    """
    function = fn(sp)
    f_list = []
    mmt_list = []
    Ei_list = []
    E_list = []

```

```
    for v in v_list:
        # Compute f(v)
        f = function(v)
        f_list.append(f)
        # Compute v*f(v)
        mmt_list.append(v*f)
        # Compute 0.5(|v-u|**2)f(v)
        Ei_list.append(0.5 * two_norm_sq(v - u_init[sp-1]) * f)
        # Compute 0.5(|v|**2)f(v)
        E_list.append(0.5 * two_norm_sq(v) * f)

    # Compute numerical values of the respective integrals, multiplied by the mass to get the macroscopic quantities
    compute_mqty = lambda x: ((b-a)**3) * (np.sum(x, axis=0) / num) * m[sp-1]
    rho = compute_mqty(f_list)
    mmt = compute_mqty(mmt_list)
    Ei = compute_mqty(Ei_list)
    E = compute_mqty(E_list)

    print(f"density of species {sp}: {rho}")
    print(f"momentum of species {sp}: {mmt}")
    print(f"internal energy of species {sp}: {Ei}")
    print(f"energy of species {sp}: {E}")
    return rho, mmt, Ei, E
```

LISTING 1. Monte Carlo integration

We get the following results:

$$\rho_1 \approx 2.03,\ \boldsymbol{m}_1 \approx (-2.07, 0.00669, 0.00647)^T,\ E_{i,1} \approx 2.98,\ E_1 \approx 4.03,$$
$$\rho_2 \approx 1.96,\ \boldsymbol{m}_2 \approx (1.95, -0.00315, 0.0206)^T,\ E_{i,2} \approx 2.97,\ E_2 \approx 3.94,$$

$$\boldsymbol{u}_1 \approx (-1.02, 0.00330, 0.00319)^T,\ T_1 \approx 0.980,$$
$$\boldsymbol{u}_2 \approx (0.992, -0.00160, 0.0105)^T,\ T_2 \approx 2.02,$$

$$\boldsymbol{u}_{12} = \boldsymbol{u}_{21} \approx (-0.0293, 0.000889, 0.00677)^T,$$
$$T_{12} = T_{21} \approx 1.77$$

We can check that the values given in the question for ρ_i (from the product of n_i and m_i) and $\boldsymbol{u}_i$ are approximately equal to the above estimated values.

Also, $\boldsymbol{u}_{12}, \boldsymbol{u}_{21}$ are approximately the zero vector. We will see later in the simulation that the common mean velocity of both species after collision converges to around the zero vector (see Figure 2).

3.2. Newton's method to find solution

Afterwards, we can solve the optimization problem for the discretized $f_{eq,i}$ of the i-th species using Newton's algorithm. Our objective function for which we find the roots is:

$$\boldsymbol{g}(\alpha_i, \boldsymbol{\beta}_i, \gamma_i) = m_i \sum_{k=1}^{N} \frac{(b-a)^3}{N} \boldsymbol{j}_k \exp(\alpha_i + \langle \boldsymbol{\beta}_i, \boldsymbol{v}_k \rangle + \gamma_i |\boldsymbol{v}_k|^2) - \begin{pmatrix} \rho \\ \boldsymbol{m} \\ 2E \end{pmatrix}$$

$$\text{with Jacobian: } \mathbf{J}(\alpha_i, \boldsymbol{\beta}_i, \gamma_i) = m_i \sum_{k=1}^{N} \frac{(b-a)^3}{N} (\boldsymbol{j}_k \boldsymbol{j}_k^T) \exp(\alpha_i + \langle \boldsymbol{\beta}_i, \boldsymbol{v}_k \rangle + \gamma_i |\boldsymbol{v}_k|^2)$$

$$\text{where } \boldsymbol{j}_k = \begin{pmatrix} 1 \\ \boldsymbol{v}_k \\ |\boldsymbol{v}_k|^2 \end{pmatrix}$$

For the Newton direction, we directly solve the linear matrix equation $\mathbf{J}\boldsymbol{d} = \boldsymbol{g}$, where $-\boldsymbol{d}$ is the descent direction, instead of computing the inverse of the Jacobian for a slightly faster computation time. Below is the implementation of the Newton's method.

```
def run_newton(sp, inputs_init=inputs_init, tol=tol, g=g, Jacobian=Jacobian):
    """
    Args:
    ===
    sp: i-th species
    inputs_init: initial guess of solution
    tol: tolerance of error
    g: objective function
    Jacobian: Jacobian function

    Output:
    ===
    Returns the estimated roots: alpha, beta, gamma.
    """
    start = time()

    inputs = inputs_init
    while 1:
        val = g(sp, inputs)
        J = Jacobian(sp, inputs)

        change = np.linalg.solve(J, val)
        alpha, beta, gamma = change[0], np.array(change[1:-1]), change[-1]
        inputs[0] -= alpha
        inputs[1] -= beta
        inputs[2] -= gamma

        err = np.linalg.norm(change)
        if err < tol:
            print(f"For species {sp},")
            print(f"Final estimated roots: {inputs}")
```

```
            print(f"Error: {err}")
            print(f"Time elapsed: {time() - start:.2f} seconds")
            return inputs
```

LISTING 2. Newton's method

The estimated solutions are:

$$\alpha_1 = -2.56,\ \boldsymbol{\beta}_1 = (1.00, 1.37 \times 10^{-16}, 1.65 \times 10^{-16})^T,\ \gamma_1 = -0.500,$$
$$\alpha_2 = -3.26,\ \boldsymbol{\beta}_2 = (1.00, -5.95 \times 10^{-18}, -8.22 \times 10^{-17})^T,\ \gamma_2 = -0.500$$

With the estimated solutions, we can plot the initial equilibrium distribution $f_{eq,i}$ in Figure 1 below for both species. We compare this together with the final equilibrium distribution after collision, $f_{eq,ij}$, which can be computed from our estimated $\boldsymbol{u}_{ij}$ and T_{ij} and the given function, $f_{eq,ij}(\boldsymbol{v})$, in the question.

For illustrative purpose, we modify the velocity grid by fixing the second and third components of $\boldsymbol{v}$ to a constant 0 such that $\boldsymbol{v}$ is of the form:

$$\boldsymbol{v} = (v_x, 0, 0) \text{ for any } v_x$$

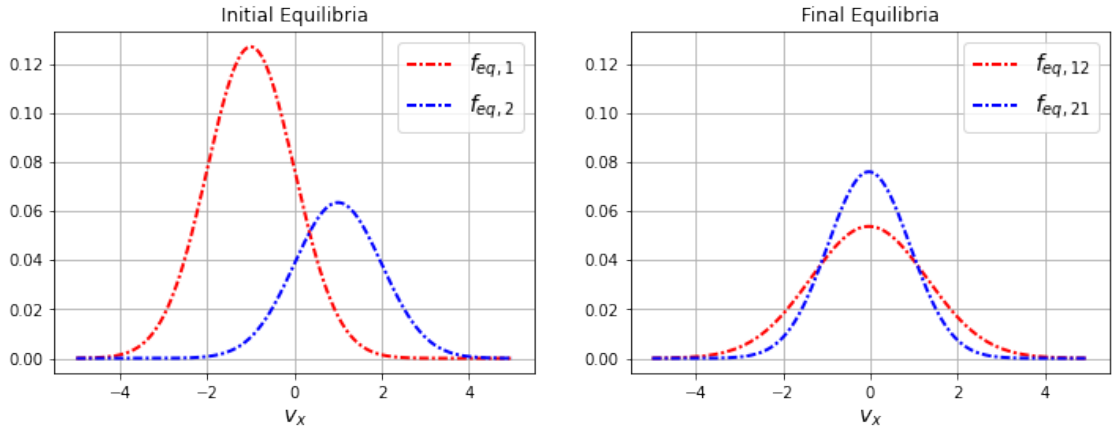


FIGURE 1. Initial and final equilibrium distribution functions

We will see later in the simulation that these two plots in Figure 1 indeed correspond to the initial and final equilibria before and after the collision respectively.

3.3. SIMULATION OF THE COLLISION

Following that, we can run the simulation to see the effects of the collision term. Again, we use the same modified velocity grid by zeroing out the second and third components. The evolution of the distribution function f_i throughout the collision process is shown in Figure 2 below. This is done by applying the below collision term to the initial distribution function iteratively.

$$\frac{\partial f_i}{\partial t} = \nu_{ii}(f_{eq,ii} - f_i) + \nu_{ij}(f_{eq,ij} - f_i)$$

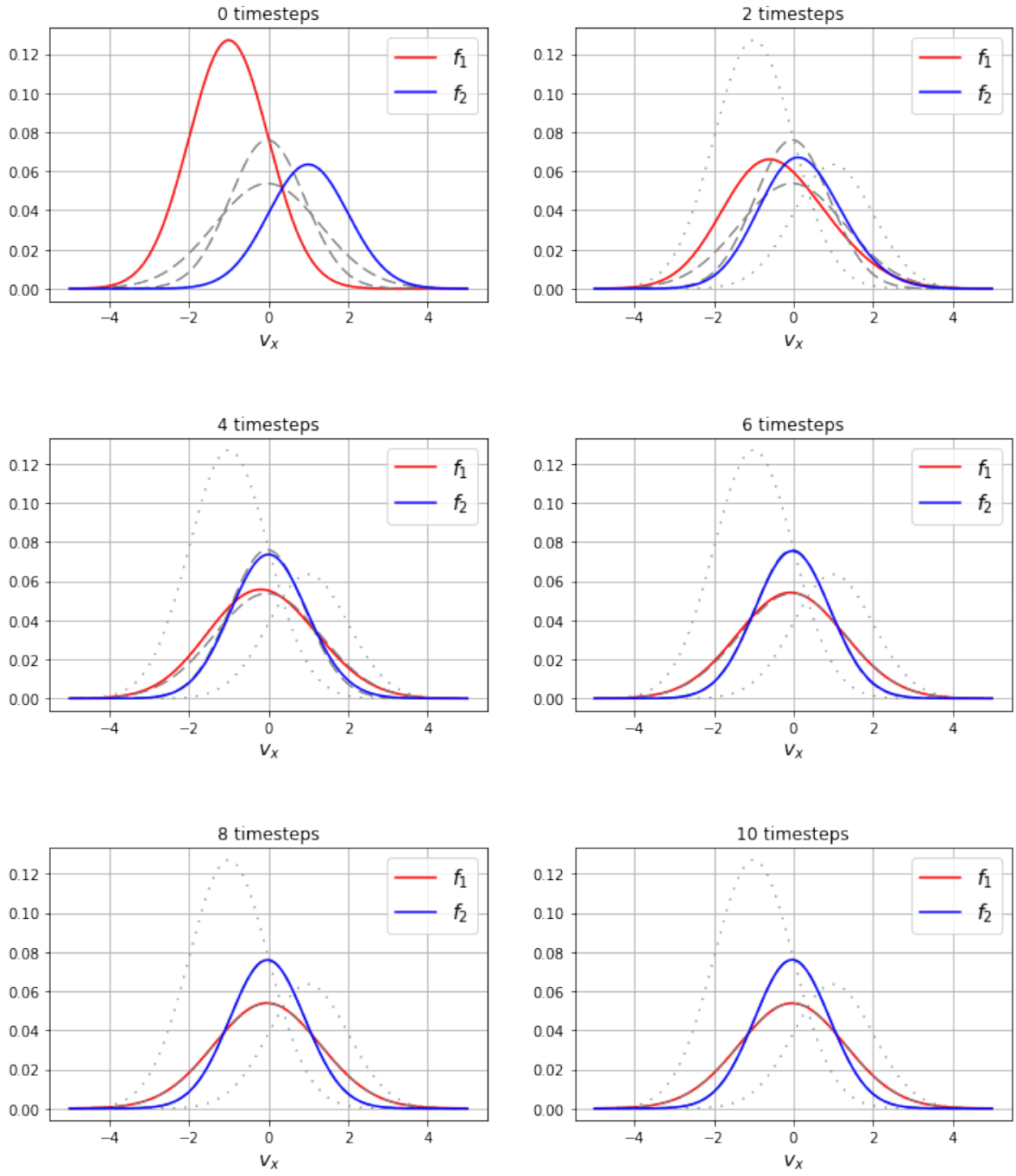


FIGURE 2. Evolution of distribution function, f_i

In Figure 2, the dotted gray lines indicate the initial equilibrium distribution functions while the dashed grey lines indicate the final equilibrium distribution functions shown previously in Figure 1. The red and blue lines indicate the evolution of the respective distribution function, f_i. One timestep is represented by a single iteration of applying the collision term once.

The evolution of the distribution functions can be described as follows:

(i) At 0 timesteps, we see that the initial distribution functions indeed correspond to their initial equilibria shown in Figure 1, being centered around their respective mean velocities.

(ii) As we apply the collision terms iteratively, the distribution functions move away from their original equilibria as shown at timesteps of 2 and 4.

(iii) At timesteps 6, 8 and 10, we see that the distribution functions converge to the final equilibria shown in Figure 1, centered around a common mean velocity, as indicated by both peaks occurring around zero.

References

[1] Jeffrey Haack, Cory Hauck, Christian Klingenberg, Marlies Pirner, and Sandra Warnecke. Numerical schemes for a multi-species BGK model with velocity-dependent collision frequency, 02 2022.

[2] Luc Mieussens. Discrete velocity model and implicit scheme for the BGK equation of Rarefied Gas Dynamics. *Mathematical Models and Methods in Applied Sciences*, 10:1121–1149, 2000.

Full code for MA5232: Project 2 Question 3

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from time import time
```

Set parameters and initial conditions

[2]:
```
m = [1., 2.]
kB = 1.
colfreq = [1., 2., 3/2]
n = [2., 1.]

u_init = [np.array([-1.,0,0]), np.array([1.,0,0])]
T_init = [1., 2.]

def two_norm_sq(vector):
    """
    Args:
    ===
    v: any vector

    Output:
    ===
    Intermediate function to return the squared value of a vector two-norm.
    """
    return np.linalg.norm(vector)**2

def f_init(sp, u_init=u_init, T_init=T_init, n=n, m=m, kB=kB):
    """
    Args:
    ===
    sp: i-th species
    u_init: initial mean velocity
    T_init: initial temperature
    n: number density
    m: mass
    kB: Boltzmann's constant
    v: velocity
```

```
    Output:
    ===
    Computes initial distribution function for species: sp = 1, 2. Returns value␣
↪at given velocity v.
    """
    if sp not in [1,2]:
        print("ERROR: Invalid species index.")

    def compute_f_init(v):
        return n[sp-1] * ((m[sp-1] / (2*np.pi*kB*T_init[sp-1]))**1.5) * np.
↪exp(-(m[sp-1]*two_norm_sq(v-u_init[sp-1])) / (2*kB*T_init[sp-1]))

    return compute_f_init
```

3.1. Computing macroscopic quantities using Monte Carlo integral

[3]:
```
# Set parameters for Monte Carlo integration

a = -5
b = 5
num = 10**5
seed = 5232

np.random.seed(seed)
v_list = np.random.uniform(a,b, (num, 3))
```

[4]:
```
# Define function for Monte Carlo integration

def mc_int(sp, a=a, b=b, fn=f_init, v_list=v_list, num=num, u_init=u_init):
    """
    Args:
    ===
    sp: i-th species
    a: lower limit
    b: upper limit
    fn: distribution function
    v_list: velocity grid
    num: number of velocities in v_list
    u_init: initial mean velocity to compute internal energy

    Output:
    ===
    Computes Monte Carlo integral. Returns macroscopic quantities density,␣
↪momentum, internal energy, energy.
    """
```

```
    function = fn(sp)
    f_list = []
    mmt_list = []
    Ei_list = []
    E_list = []

    for v in v_list:
        # Compute f(v)
        f = function(v)
        f_list.append(f)
        # Compute v*f(v)
        mmt_list.append(v*f)
        # Compute 0.5(|v-u|**2)f(v)
        Ei_list.append(0.5 * two_norm_sq(v - u_init[sp-1]) * f)
        # Compute 0.5(|v|**2)f(v)
        E_list.append(0.5 * two_norm_sq(v) * f)

    # Compute numerical values of the respective integrals, multiplied by the␣
↪mass to get the macroscopic quantities
    compute_mqty = lambda x: ((b-a)**3) * (np.sum(x, axis=0) / num) * m[sp-1]
    rho = compute_mqty(f_list)
    mmt = compute_mqty(mmt_list)
    Ei = compute_mqty(Ei_list)
    E = compute_mqty(E_list)

    print(f"density of species {sp}: {rho}")
    print(f"momentum of species {sp}: {mmt}")
    print(f"internal energy of species {sp}: {Ei}")
    print(f"energy of species {sp}: {E}")
    return rho, mmt, Ei, E
```

[5]:
```
# Compute macroscopic quantities

rho1, mmt1, Ei_1, E1 = mc_int(1)
rho2, mmt2, Ei_2, E2 = mc_int(2)
```

```
density of species 1: 2.029559214033467
momentum of species 1: [-2.06501999  0.00669451  0.00647429]
internal energy of species 1: 2.984263396066316
energy of species 1: 4.03450377781187
density of species 2: 1.9631638895279249
momentum of species 2: [ 1.94806889 -0.00314541  0.02055484]
internal energy of species 2: 2.968657875293191
energy of species 2: 3.9351448204068307
```

[6]:
```
# Compute other macroscopic quantities
```

```
u1 = mmt1 / rho1
T1 = (2 * m[0] * Ei_1) / (3 * kB * rho1)

u2 = mmt2 / rho2
T2 = (2 * m[1] * Ei_2) / (3 * kB * rho2)

print(f"mean velocity of species 1: {u1}")
print(f"temperature of species 1: {T1}")

print(f"mean velocity of species 2: {u2}")
print(f"temperature of species 2: {T2}")
```

```
mean velocity of species 1: [-1.01747216  0.0032985   0.00319   ]
temperature of species 1: 0.9802665115431666
mean velocity of species 2: [ 0.99231088 -0.00160222  0.01047026]
temperature of species 2: 2.0162404787013166
```

[7]:
```
# Compute macroscopic quantities for mixed

u_mixed = ((rho1 * colfreq[-1] * u1) + (rho2 * colfreq[-1] * u2)) / ((rho1 *␣
 ↪colfreq[-1]) + (rho2 * colfreq[-1]))
T_mixed = ((n[0] * colfreq[-1] * T1) + (n[1] * colfreq[-1] * T2)) / ((n[0] *␣
 ↪colfreq[-1]) + (n[1] * colfreq[-1])) + ((rho1 * colfreq[-1] * (two_norm_sq(u1)␣
 ↪- two_norm_sq(u_mixed))) + (rho2 * colfreq[-1] * (two_norm_sq(u2) -␣
 ↪two_norm_sq(u_mixed)))) / (3 * (n[0] * colfreq[-1] + n[1] * colfreq[-1]))

print(f"mean velocity of mixed: {u_mixed}")
print(f"temperature of mixed: {T_mixed}")
```

```
mean velocity of mixed: [-0.02929106  0.00088889  0.0067696 ]
temperature of mixed: 1.773462448728367
```

3.2. Newton's method to find solution

[8]:
```
# Define functions needed to compute objective function g and Jacobian

def col_v(input1, input2, input3):
    """
    Args:
    ===
    input1: scalar
    input2: vector of dimension 3
    input3: scalar

    Output:
    Intermediate function to concatenate inputs and return as a column vector of␣
 ↪dimension 5.
```

```
    """
    condition1 = isinstance(input1, (int, float))
    condition2 = isinstance(input2, np.ndarray)
    condition3 = isinstance(input3, (int, float))

    if condition1 and condition2 and condition3:
        return np.append(input1, np.append(input2, input3))

    print("ERROR: Inputs have wrong dimensions.")
    return

def Feq(inputs):
    """
    Args:
    ===
    inputs: list of parameters: alpha, beta, gamma
    v: velocity

    Output:
    ===
    Computes distribution function of solution. Returns value at given velocity␣
 ↪v.
    """
    def compute_Feq(v):
        alpha, beta, gamma = inputs
        return np.exp(alpha + np.dot(beta, v) + gamma*two_norm_sq(v))
    return compute_Feq

w = ((b-a)**3) / num
mqty = [col_v(rho1, mmt1, 2*E1), col_v(rho2, mmt2, 2*E2)]

def g(sp, inputs, fn=Feq, v_list=v_list, m=m):
    """
    Args:
    ===
    sp: i-th species
    inputs: list of parameters: alpha, beta, gamma
    fn: distribution function
    v_list: velocity grid
    m: mass

    Output:
    ===
    Computes objective function in the optimization problem. Returns vector of␣
 ↪dimension 5.
    """
    function = fn(inputs)
```

```
    # initialize
    rho_eqn = 0
    mmt_eqn = np.zeros(3)
    E_eqn = 0

    for v in v_list:
        val = w * function(v)
        rho_eqn += val
        mmt_eqn += val * v
        E_eqn += val * two_norm_sq(v)

    return m[sp-1]*np.append(rho_eqn, np.append(mmt_eqn, E_eqn)) - mqty[sp-1]

def Jacobian(sp, inputs, fn=Feq, v_list=v_list, m=m):
    """
    Args:
    ===
    sp: i-th species
    inputs: list of parameters: alpha, beta, gamma
    fn: distribution function
    v_list: velocity grid
    m: mass

    Output:
    ===
    Returns the Jacobian matrix of dimension 5 by 5.
    """
    function = fn(inputs)

    J = np.zeros((5,5))
    for v in v_list:
        # col_v concatenates inputs into a vector of dimension 5
        col = col_v(1, v, two_norm_sq(v))
        J_k = np.outer(col, col)
        J += w * J_k * function(v)

    return m[sp-1] * J
```

[9]:
```
# Set parameters and define function for the algorithm

tol = 1e-14
inputs_init = [0., np.array([0.,0,0]), 0.]

def run_newton(sp, inputs_init=inputs_init, tol=tol, g=g, Jacobian=Jacobian):
    """
    Args:
```

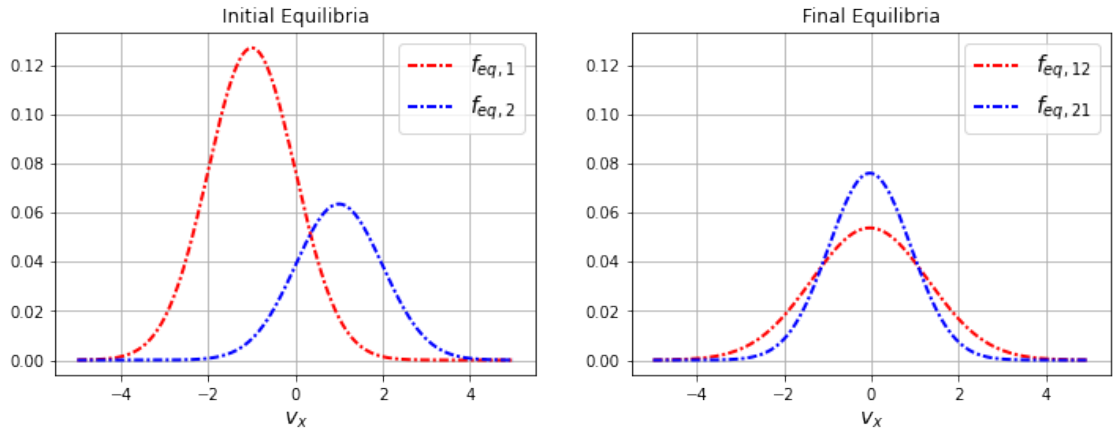
```
    ===
    sp: i-th species
    inputs_init: initial guess of solution
    tol: tolerance of error
    g: objective function
    Jacobian: Jacobian function

    Output:
    ===
    Returns the estimated roots: alpha, beta, gamma.
    """
    start = time()

    inputs = inputs_init
    while 1:
        val = g(sp, inputs)
        J = Jacobian(sp, inputs)

        change = np.linalg.solve(J, val)
        alpha, beta, gamma = change[0], np.array(change[1:-1]), change[-1]
        inputs[0] -= alpha
        inputs[1] -= beta
        inputs[2] -= gamma

        err = np.linalg.norm(change)
        if err < tol:
            print(f"For species {sp},")
            print(f"Final estimated roots: {inputs}")
            print(f"Error: {err}")
            print(f"Time elapsed: {time() - start:.2f} seconds")
            return inputs
```

[10]:
```
# Run the algorithm

alpha1, beta1, gamma1 = run_newton(1)
alpha2, beta2, gamma2 = run_newton(2)
```

```
For species 1,
Final estimated roots: [-2.56366841905406, array([-1.00000000e+00,
-1.37028533e-16, -1.64766138e-16]), -0.49999999999999545]
Error: 5.65268830620655e-15
Time elapsed: 89.87 seconds
For species 2,
Final estimated roots: [-3.2568155996139914, array([ 1.00000000e+00,
-5.94856832e-18, -8.21850847e-17]), -0.5000000000000073]
Error: 3.4235123310148745e-15
Time elapsed: 61.79 seconds
```

[11]:
```
# Plot distribution of species 1 and 2 at their initial and mixed equilibrium␣
 ↪states

def Feq_ij(sp, n=n, m=m, kB=kB, T_mixed=T_mixed, u_mixed=u_mixed):
    """
    Args:
    ===
    sp: i-th species
    n: number density
    m: mass
    kB: Boltzmann's constant
    T_mixed: temperature of mixed
    u_mixed: mean velocity of mixed
    v: velocity

    Output:
    ===
    Computes equilibrium distribution function for species sp = 1, 2. Returns␣
 ↪value at given velocity v.
    """
    if sp not in [1,2]:
        print("ERROR: Invalid species index.")

    def compute_Feq_ij(v):
        return n[sp-1] * ((m[sp-1] / (2*np.pi*kB*T_mixed))**1.5) * np.
 ↪exp(-(m[sp-1]*two_norm_sq(v-u_mixed)) / (2*kB*T_mixed))

    return compute_Feq_ij

feq1 = Feq([alpha1, -beta1, gamma1])
feq2 = Feq([alpha2, beta2, gamma2])
v_sim = v_list.copy()
v_sim[:, 1] = 0
v_sim[:, 2] = 0
v_x = v_sim[:, 0]
v_x.sort()

feq1_list = list(map(feq1, v_sim))
feq2_list = list(map(feq2, v_sim))

feq12_list = list(map(Feq_ij(1), v_sim))
feq21_list = list(map(Feq_ij(2), v_sim))

plt.figure(figsize=(12,4))

ax0 = plt.subplot(1,2,1)
plt.plot(v_x, feq1_list, ls=(0,(3,1,1,1)), color="red", lw=2, label="$f_{eq,1}$")
```

```
plt.plot(v_x, feq2_list, ls=(0,(3,1,1,1)), color="blue", lw=2,␣
 ↪label="$f_{eq,2}$")
plt.grid()
plt.legend(fontsize=14)
plt.title("Initial Equilibria")
plt.xlabel("$v_{x}$", fontsize=14)

plt.subplot(1,2,2,sharey=ax0)
plt.plot(v_x, feq12_list, ls=(0,(3,1,1,1)), color="red", lw=2,␣
 ↪label="$f_{eq,12}$")
plt.plot(v_x, feq21_list, ls=(0,(3,1,1,1)), color="blue", lw=2,␣
 ↪label="$f_{eq,21}$")
plt.grid()
plt.legend(fontsize=14)
plt.title("Final Equilibria")
plt.xlabel("$v_{x}$", fontsize=14)

plt.show()
```



3.3. Simulation of the collision

[12]:
```
# Define functions for simulation

def run_sim(colfreq=colfreq, mixed_f=Feq_ij, indv_f=f_init, v_sim=v_sim, itr=10):
    """
    Args:
    ===
    colfreq: collision frequencies
    mixed_f: distribution function for mixed
    indv_f: distribution function for individual
```

```
    v_sim: velocity grid with the second and third components of each velocity␣
↪zeroed out
    itr: number of iterations

    Output:
    ===
    Returns plots of the evolution of the distribution functions.
    """
    plt.figure(figsize=(12,14))
    plt.subplots_adjust(hspace=.5)

    for sp in [1,2]:
        color = ["red" if sp==1 else "blue"][0]
        f = indv_f(sp)
        Feq = mixed_f(sp)

        init = np.array(list(map(f, v_sim)))
        mixed = np.array(list(map(Feq, v_sim)))

        ax0 = plt.subplot(3,2,1)
        plt.plot(v_x, feq1_list, ls=(0,(1,7)), color="grey", lw=1)
        plt.plot(v_x, feq2_list, ls=(0,(1,7)), color="grey", lw=1)
        plt.plot(v_x, feq12_list, ls=(0,(10,5)), color="grey", lw=1)
        plt.plot(v_x, feq21_list, ls=(0,(10,5)), color="grey", lw=1)
        plt.plot(v_x, init, label=f"$f_{sp}$", color=color)
        plt.title("0 timesteps")
        plt.xlabel("$v_{x}$", fontsize=14)
        plt.legend(fontsize=14)
        plt.grid(visible=True)

        val = init
        count = 1
        for i in range(1, itr+1):
            val += colfreq[sp-1]*(init-val) + colfreq[-1]*(mixed-val)
            if i%2 == 0:
                count += 1
                plt.subplot(3,2,count,sharey=ax0)
                plt.plot(v_x, feq1_list, ls=(0,(1,7)), color="grey", lw=1)
                plt.plot(v_x, feq2_list, ls=(0,(1,7)), color="grey", lw=1)
                plt.plot(v_x, feq12_list, ls=(0,(10,5)), color="grey", lw=1)
                plt.plot(v_x, feq21_list, ls=(0,(10,5)), color="grey", lw=1)
                plt.plot(v_x, val, label=f"$f_{sp}$", color=color)
                plt.title(f"{i} timesteps")
                plt.xlabel("$v_{x}$", fontsize=14)
                plt.legend(fontsize=14)
                plt.grid(visible=True)
    return
```

[13]:
```
# run simulation

run_sim()
```