1 Expanding the conservation laws

The conservation laws are given below in terms of $\mu = 1$. For numerical convenience, place a dimensionless μ in front of each $\vec{B}$ appearance and let $\mu = 0$ be plain compressible hydrodynamics.

$$\begin{aligned}\frac{\partial\rho}{\partial t} &= -\vec{\nabla}\cdot(\rho\vec{v})\\ \frac{\partial(\rho\vec{v})}{\partial t} &= -\vec{\nabla}\cdot(\rho\vec{v}\vec{v} - \vec{B}\vec{B}) - \vec{\nabla}P^*\\ \frac{\partial\vec{B}}{\partial t} &= \vec{\nabla}\times(\vec{v}\times\vec{B})\\ \frac{\partial E}{\partial t} &= -\vec{\nabla}\cdot\Big((E+P^*)\vec{v} - \vec{B}(\vec{B}\cdot\vec{v})\Big)\end{aligned} \tag{1}$$

where $P^* = (\gamma-1)\left(E - \rho\frac{v^2}{2} - \frac{B^2}{2}\right) + \frac{B^2}{2}$. Note that $\vec{B}\vec{B}$ is a dyadic tensor $\mathbf{B} = \vec{B}\vec{B}$ such that $\mathbf{B}_{ij} = B_iB_j$. Then the divergence $\vec{\nabla}\cdot\mathbf{B} = \partial^iB_iB_j\hat{e}_j = B_i\partial_iB_j\hat{e}_j = \left(\vec{B}\cdot\vec{\nabla}\right)\vec{B}$. On the other hand, $\vec{\nabla}\cdot\rho\vec{v}\vec{v} = \left(\vec{\nabla}\cdot\rho\vec{v}\right)\vec{v} + (\vec{v}\cdot\vec{\nabla})\rho\vec{v}$ where in general we allow $\vec{\nabla}\cdot\rho\vec{v}\neq 0$.

Some intermediate steps: $\vec{\nabla}\cdot(\rho\vec{v}\vec{v}) = \frac{\rho\vec{v}}{\rho}\cdot\vec{\nabla}(\rho\vec{v})$. Additionally, $\vec{\nabla}\times(\vec{v}\times\vec{B}) = (\vec{B}\cdot\vec{\nabla})\vec{v} - (\vec{v}\cdot\vec{\nabla})\vec{B} - \vec{B}(\vec{\nabla}\cdot\vec{v})$ since only $\vec{\nabla}\cdot\vec{B} = 0$.

Construct the vector field $\vec{u}(x,y,z)$ where $\vec{u} = (\rho, \rho v_x, \rho v_y, \rho v_z, B_x, B_y, B_z, E)$ and assume we have point-by-point cached values of $-(E+P^*)\vec{v} + \vec{B}(\vec{B}\cdot\vec{v}) = EPmDV_i$ (along with P^*, four values), then we can represent these evolution equations as (two lines b/c too damn long)

$$\begin{aligned}\frac{\partial\vec{u}}{\partial t} = f(\vec{u}) = &\begin{pmatrix} -\frac{\partial u_2}{\partial x} - \frac{\partial u_3}{\partial y} - \frac{\partial u_4}{\partial z}\\ -\frac{u_2}{u_1}\frac{\partial u_2}{\partial x} - \frac{u_3}{u_1}\frac{\partial u_2}{\partial y} - \frac{u_4}{u_1}\frac{\partial u_2}{\partial z} + u_5\frac{\partial u_2}{\partial x} + u_6\frac{\partial u_2}{\partial y} + u_7\frac{\partial u_2}{\partial z} - \frac{\partial P^*}{\partial x}\\ -\frac{u_2}{u_1}\frac{\partial u_3}{\partial x} - \frac{u_3}{u_1}\frac{\partial u_3}{\partial y} - \frac{u_4}{u_1}\frac{\partial u_3}{\partial z} + u_5\frac{\partial u_3}{\partial x} + u_6\frac{\partial u_3}{\partial y} + u_7\frac{\partial u_3}{\partial z} - \frac{\partial P^*}{\partial y}\\ -\frac{u_2}{u_1}\frac{\partial u_4}{\partial x} - \frac{u_4}{u_1}\frac{\partial u_2}{\partial y} - \frac{u_4}{u_1}\frac{\partial u_4}{\partial z} + u_5\frac{\partial u_4}{\partial x} + u_6\frac{\partial u_4}{\partial y} + u_7\frac{\partial u_4}{\partial z} - \frac{\partial P^*}{\partial z}\\ \frac{u_5\frac{\partial u_2}{\partial x} + u_6\frac{\partial u_2}{\partial y} + u_7\frac{\partial u_2}{\partial z} - u_2\frac{\partial u_5}{\partial x} - u_3\frac{\partial u_5}{\partial y} - u_4\frac{\partial u_5}{\partial z} - u_5\left(\frac{\partial u_2}{\partial x} + \frac{\partial u_3}{\partial y} + \frac{\partial u_4}{\partial z}\right)}{u_1}\\ \frac{u_5\frac{\partial u_3}{\partial x} + u_6\frac{\partial u_3}{\partial y} + u_7\frac{\partial u_3}{\partial z} - u_2\frac{\partial u_6}{\partial x} - u_3\frac{\partial u_6}{\partial y} - u_4\frac{\partial u_6}{\partial z} - u_6\left(\frac{\partial u_2}{\partial x} + \frac{\partial u_3}{\partial y} + \frac{\partial u_4}{\partial z}\right)}{u_1}\\ \frac{u_5\frac{\partial u_4}{\partial x} + u_6\frac{\partial u_4}{\partial y} + u_7\frac{\partial u_4}{\partial z} - u_2\frac{\partial u_7}{\partial x} - u_3\frac{\partial u_7}{\partial y} - u_4\frac{\partial u_7}{\partial z} - u_7\left(\frac{\partial u_2}{\partial x} + \frac{\partial u_3}{\partial y} + \frac{\partial u_4}{\partial z}\right)}{u_1}\\ \frac{\partial EPmDV_1}{\partial x} + \frac{\partial EPmDV_2}{\partial y} + \frac{\partial EPmDV_3}{\partial z}\end{pmatrix}\\ &+\begin{pmatrix} 0\\ -\frac{u_2}{u_1}\left(\frac{\partial u_2}{\partial x} + \frac{\partial u_3}{\partial y} + \frac{\partial u_4}{\partial z}\right)\\ -\frac{u_3}{u_1}\left(\frac{\partial u_2}{\partial x} + \frac{\partial u_3}{\partial y} + \frac{\partial u_4}{\partial z}\right)\\ -\frac{u_4}{u_1}\left(\frac{\partial u_2}{\partial x} + \frac{\partial u_3}{\partial y} + \frac{\partial u_4}{\partial z}\right)\\ 0\\ 0\\ 0\\ 0\end{pmatrix}\end{aligned} \tag{2}$$

Note that all derivatives are given by the expression $\dfrac{\partial f_{x,y,z}}{\partial x} = \dfrac{f_{x+1,y,z} - f_{x-1,y,z}}{2\,dx}$, which is the slope of the extrapolated quadratic fit between points $(x+1, y, z), (x, y, z), (x-1, y, z)$, unless we are at a boundary, in which case we simply take $\dfrac{f_{x+1,y,z} - f_{x,y,z}}{dx}$ for instance.

Lastly, let's compute the components for $P^*, EPmDV$ and obatin

$$u_9 = P^* = (\gamma - 1)\left(u_8 - \frac{u_2^2 + u_3^2 + u_4^2}{2u_1} + \frac{u_5^2 + u_6^2 + u_7^2}{2}\right) \tag{3}$$

$$\begin{pmatrix} u_{10} \\ u_{11} \\ u_{12} \end{pmatrix} = EPmDV = \begin{pmatrix} -(u_8 + u_9)u_2 + u_5(u_2u_5 + u_3u_6 + u_4u_7) \\ -(u_8 + u_9)u_3 + u_6(u_2u_5 + u_3u_6 + u_4u_7) \\ -(u_8 + u_9)u_4 + u_7(u_2u_5 + u_3u_6 + u_4u_7) \end{pmatrix} \tag{4}$$

1.1 Divergence-free

The condition $\vec{\nabla} \cdot \vec{B} = 0$ must always be maintained for physical results. While this is theoretically enforced in the continuous limit since the divergence of a curl is zero and $\dfrac{\partial \vec{B}}{\partial t} \propto \vec{\nabla} \times (\dots)$, this is not enforced to machine precision upon discretization, and manual treatment should be effected. We do not in this present iteration of the code.

2 Numerics Considerations

- Runge-Kutta takes the entire $12L^3$ size $\vec{u}$ (including the $EPmDV_i, P^*$ variables) and updates it at once by computing a $d\vec{u}$. This requires three temporary arrays before obtaining $d\vec{u}$. Since this is evolving the entire array forward at each timestep, we require $4N$ operations to move forward a timestep, the same as if we run R-K at each point individually (fixing all other points on the grid constant) just requiring more space. However, the upshot is that we require storing updates to the whole grid to compute $d\vec{u}$ accurately using R-K, while using something like Euler we don't need temporary values and *can* ignore updates to the grid with no accuracy penalty.

  GPU optimizations must be considered. Since GPU-side storage is limited and computing all these temporary arrays seems slow, it would seem that some local approximation to the R-K at each pixel would be preferable. While this loses some accuracy and actually requires at least as many operations, it may be preferred for its decreased memory footprint.