1 Mathematical Framework: Two-Moment CR Transport

1.1 Governing Equations

The two-moment approach evolves the cosmic-ray (CR) energy density E_c and CR flux $\boldsymbol{F}_c$ as independent variables. In conservative form, the governing equations are

$$\frac{\partial E_c}{\partial t} + \nabla \cdot \boldsymbol{F}_c = -(\boldsymbol{v} + \boldsymbol{v}_s) \cdot \boldsymbol{\sigma}_c [\boldsymbol{F}_c - (E_c + P_c)\, \boldsymbol{v}] + Q, \tag{1}$$

$$\frac{1}{v_{\max}^2} \frac{\partial \boldsymbol{F}_c}{\partial t} + \nabla \cdot \boldsymbol{P}_c = -\boldsymbol{\sigma}_c \cdot [\boldsymbol{F}_c - \boldsymbol{v}(E_c + P_c)]\,, \tag{2}$$

where:

- $P_c = E_c/3$ is the CR pressure (isotropic closure for ultra-relativistic particles)
- $\boldsymbol{P}_c = P_c \mathsf{I}$ is the CR pressure tensor
- $\boldsymbol{v}$ is the gas velocity
- $\boldsymbol{v}_s$ is the CR streaming velocity
- $\boldsymbol{\sigma}_c$ is the CR-gas interaction coefficient tensor
- $v_{\max}$ is the maximum CR propagation speed (reduced speed of light)
- Q represents explicit CR injection/sources

The source term on the right-hand side of (1) should be read with some care when compared to Jiang & Oh (2018; arXiv:1712.07117). In the paper, the E_c equation is written with the source $(\boldsymbol{v} + \boldsymbol{v}_s) \cdot \nabla P_c + Q$, whereas the `Athena++` implementation is organized around the relaxation form written in (1), which is the form that enters the implicit source solve. In practice, the code advances the source terms through this relaxation form rather than through a separate explicit $(\boldsymbol{v} + \boldsymbol{v}_s) \cdot \nabla P_c$ update. For this reason, when reading the paper alongside the code, it is best to follow the implemented source-step equations and the associated matrix solve.

1.2 Physical Interpretation of Terms

The streaming velocity represents CR advection along magnetic field lines at the Alfvén speed:

$$\boldsymbol{v}_s = -\operatorname{sgn}(\hat{\boldsymbol{b}} \cdot \nabla P_c)\, v_A \hat{\boldsymbol{b}}, \quad \text{where } v_A = \frac{B}{\sqrt{4\pi\rho}} \tag{3}$$

The sign function ensures CRs stream *down* their pressure gradient. Near extrema where $\nabla P_c \to 0$, the streaming direction becomes delicate to evaluate numerically. One of the main motivations for the two-moment formulation is to evolve $\boldsymbol{F}_c$ directly rather than relying on an instantaneous local flux prescription there.

The interaction coefficient combines diffusion and streaming effects. In the parallel direction:

$$\frac{1}{\sigma_{c,\parallel}} = \frac{1}{\sigma_{c,\parallel}^0} + \frac{v_A(E_c + P_c)}{|\hat{\boldsymbol{b}} \cdot \nabla P_c|} \tag{4}$$

The first term represents classical diffusion with coefficient $\sigma^0_{c,\parallel}$. The second term encodes the streaming contribution to the effective coupling. As $\nabla P_c \to 0$, the parallel coupling weakens and the transport update moves toward the optically thin limit of the two-moment system. This behavior is part of the closure adopted by Jiang & Oh.

In the perpendicular direction,

$$\sigma_{c,\perp} = \sigma^0_{c,\perp}, \tag{5}$$

since streaming only occurs along field lines.

1.3 Why the Two-Moment Approach?

Traditional CR transport evolves only the energy density E_c, assuming an instantaneous steady-state closure for the flux:

$$\boldsymbol{F}_c^{\text{traditional}} = (E_c + P_c)(\boldsymbol{v} + \boldsymbol{v}_s) - \frac{\hat{\boldsymbol{b}}(\hat{\boldsymbol{b}} \cdot \nabla P_c)}{\sigma^0_c}. \tag{6}$$

This one-moment closure has two difficulties:

1. At extrema ($\nabla P_c \to 0$), the streaming direction $\boldsymbol{v}_s$ changes sign discontinuously, so the closure becomes ill-defined exactly where the gradient is least well resolved. This is the dominant source of the classical streaming instability; Sharma et al. (2010) tame it with an ad hoc smoothing, which Jiang & Oh (2018) aim to avoid.
2. Substituting the algebraic flux into the E_c equation produces a diffusive (parabolic) contribution along the field[1]. That piece has no finite signal speed and, when combined with the sign-flip behavior above, makes the discretized problem stiff and poorly conditioned near extrema.

The two-moment approach addresses both issues by evolving E_c and $\boldsymbol{F}_c$ together. A small inertial term $(1/v^2_{\text{max}})\,\partial_t \boldsymbol{F}_c$ is retained in the flux equation, exactly as in two-moment radiation hydrodynamics. This makes the transport part of the system hyperbolic, with finite characteristic speeds.

To see this, consider the homogeneous one-dimensional system (without sources):

$$\frac{\partial E_c}{\partial t} + \frac{\partial F_c}{\partial x} = 0, \tag{7}$$

$$\frac{1}{v^2_{\text{max}}}\frac{\partial F_c}{\partial t} + \frac{\partial P_c}{\partial x} = 0. \tag{8}$$

To close the system, write

$$P_c = \chi E_c, \tag{9}$$

where $\chi \equiv P_c/E_c$ is the *Eddington factor*. For isotropic particle distributions, $\chi = 1/3$.

Substituting plane-wave solutions $E_c, F_c \propto e^{i(kx-\omega t)}$ yields

$$-i\omega E_c + ikF_c = 0, \tag{10}$$

$$-\frac{i\omega}{v^2_{\text{max}}}F_c + ik\chi E_c = 0. \tag{11}$$

[1] In PDE classification, a *parabolic* equation (e.g. the diffusion equation) does not support finite wave speeds: information spreads instantaneously across the domain. In contrast, a *hyperbolic* equation has well-defined characteristics along which information propagates at finite speed, making the initial value problem well-posed.

Requiring nontrivial solutions gives the dispersion relation

$$\omega^2 = v_{\max}^2 \chi k^2. \tag{12}$$

Hence the characteristic wave speeds are

$$\lambda = \pm v_{\max}\sqrt{\chi} = \pm v_{\max}\sqrt{\frac{P_c}{E_c}}. \tag{13}$$

For the isotropic case ($\chi = 1/3$),

$$\lambda = \pm \frac{v_{\max}}{\sqrt{3}}. \tag{14}$$

The Eddington Closure: In radiation transport, the hierarchy of moments couples energy density E, flux $\boldsymbol{F}$, and pressure tensor P. To truncate the system one introduces the *Eddington tensor*, defined by $\mathsf{P} = \boldsymbol{f}\,E$. The tensor $\boldsymbol{f}$ encodes angular information: for isotropy, $\boldsymbol{f} = \frac{1}{3}\mathsf{I}$ so that $P/E = 1/3$, while for a perfectly beamed field, $\boldsymbol{f} = \hat{n}\hat{n}$. Here $\hat{n}\hat{n}$ denotes the outer product of the propagation direction with itself, so that $(\hat{n}\hat{n})_{ij} = \hat{n}_i\hat{n}_j$. Averaging over all directions on the sphere yields $\langle \hat{n}_i\hat{n}_j\rangle = \frac{1}{3}\delta_{ij}$, which explains the isotropic case. This is the familiar *Eddington approximation*. Jiang & Oh adopt the same strategy for cosmic rays: closing with $\chi = P_c/E_c$ provides finite, causal wave speeds and makes the two-moment system well-posed.

2 Fluxes and Sources in the Two-Moment CR Equations

2.1 Classification of Terms

Equations (1)–(2) contain two qualitatively different contributions:

Flux (transport) terms.

$$\nabla \cdot \boldsymbol{F}_c, \qquad \nabla P_c$$

These terms transport E_c and $\boldsymbol{F}_c$ between neighboring cells. In a finite-volume method, they are advanced *explicitly* by computing *interface fluxes* via a Riemann solver and differencing those fluxes across cell faces.

Source (local) terms.

$$-(\boldsymbol{v} + \boldsymbol{v}_s)\cdot\sigma_c[\boldsymbol{F}_c - (E_c + P_c)\boldsymbol{v}] + Q, \qquad -\sigma_c[\boldsymbol{F}_c - (E_c + P_c)\boldsymbol{v}]$$

These act within each cell. They relax $\boldsymbol{F}_c$ toward the local equilibrium form used by the closure and account for injection Q. When σ_c is large, the associated timescale is short (stiff), so these terms are advanced *implicitly* after the explicit transport update. In the MHD implementation, the parallel relaxation is handled by the implicit matrix solve, while the perpendicular work term is added separately after that solve.

2.2 Time Integration: Explicit, Implicit, and Operator Splitting

The CR two-moment system mixes two classes of terms: (i) *hyperbolic transport* (flux divergence), and (ii) *stiff relaxation sources* (scattering and energy exchange). Because their numerical behavior is very different, these are advanced with a combination of explicit and implicit steps, using operator splitting.

2.2.1 Explicit and Implicit Updates

For a generic variable U with $\partial_t U = \mathcal{R}(U)$, the two time-integration strategies are:

Explicit update.

$$U^{n+1} = U^n + \Delta t\, \mathcal{R}(U^n).$$

The right-hand side is evaluated using known data. It is simple but constrained by the CFL condition. For CR transport the timestep must satisfy $\Delta t \lesssim \Delta x / v_{\max}$. Stiff scattering makes this prohibitively restrictive if $\sigma_c \gg 1/\Delta t$.

Implicit update. Instead one solves

$$U^{n+1} = U^n + \Delta t\, \mathcal{R}(U^{n+1}),$$

so the new state appears on both sides. Applied to the CR energy and the stored flux components, this yields a linear 4×4 system coupling E_c and $\widetilde{\boldsymbol{F}}_c$:

$$\begin{bmatrix} 1-\Delta t\,\alpha & -\Delta t\,\beta_1 & -\Delta t\,\beta_2 & -\Delta t\,\beta_3 \\ -\Delta t\,\gamma_1 & 1+\Delta t\,\delta & 0 & 0 \\ -\Delta t\,\gamma_2 & 0 & 1+\Delta t\,\delta & 0 \\ -\Delta t\,\gamma_3 & 0 & 0 & 1+\Delta t\,\delta \end{bmatrix} \begin{bmatrix} E_c^{n+1} \\ \widetilde{F}_{c1}^{n+1} \\ \widetilde{F}_{c2}^{n+1} \\ \widetilde{F}_{c3}^{n+1} \end{bmatrix} = \begin{bmatrix} \mathrm{RHS}_E \\ \mathrm{RHS}_{\widetilde{F}1} \\ \mathrm{RHS}_{\widetilde{F}2} \\ \mathrm{RHS}_{\widetilde{F}3} \end{bmatrix}.$$

The coefficients $\alpha, \beta_i, \gamma_i, \delta$ are proportional to σ_c and to components of the gas and total velocities. They originate directly from the linear source terms $F_i - v_i(E_c + P_c)/v_{\max}$. Because the system is block-triangular, Athena++ solves it analytically: E_c^{n+1} is obtained first (via Schur complement), then each F_i^{n+1} is back-substituted. Explicit expressions for the entries are collected in Appendix A.3. Here and below, the flux unknowns in the implicit solve are the stored code variables, not the unscaled physical flux. This implicit solve is unconditionally stable, removing the timestep constraint from stiff scattering.

Illustration: 1D source step. In one dimension with closure $P_c = \chi E_c$, flow velocity v, total transport velocity $v_{\mathrm{tot}} = v + v_s$, and stored flux variable

$$\widetilde{F} \equiv F_{\mathrm{phys}}/v_{\max},$$

and ignoring the separate perpendicular work term that appears in the MHD implementation, the code-facing source equations take the form

$$\frac{\partial E_c}{\partial t} = -v_{\mathrm{tot}}\,\sigma\left[\widetilde{F} - \frac{v(E_c + P_c)}{v_{\max}}\right] + Q, \tag{15}$$

$$\frac{\partial \widetilde{F}}{\partial t} = -v_{\max}\,\sigma\left[\widetilde{F} - \frac{v(E_c + P_c)}{v_{\max}}\right]. \tag{16}$$

The asymmetry is important: the scalar E_c equation carries an outer v_{tot} factor, while the flux equation carries an outer $v_{\max}$ factor. This structure is what produces the 4×4 coefficient pattern shown above, with A_{11} picking up $v_j v_j^{\mathrm{tot}}$ products and A_{1j} picking up v_j^{tot} alone. A backward Euler discretization couples E_c^{n+1} and $\widetilde{F}^{\,n+1}$ through a 2×2 block, which is solved by Schur complement as described above.

Two asymptotic checks are useful. First, with $v = v_{\rm tot} = 0$ and $Q = 0$, the stored flux damps stably as

$$\widetilde{F}^{\,n+1} = \frac{\widetilde{F}^{\,*}}{1 + \Delta t\, \sigma v_{\rm max}}.$$

Second, in the stiff limit $\Delta t\, \sigma v_{\rm max} \gg 1$, the flux equation forces

$$\widetilde{F}^{\,n+1} \to \frac{v(E_c^{n+1} + P_c^{n+1})}{v_{\rm max}},$$

equivalently $F_{\rm phys}^{n+1} \to v(E_c^{n+1} + P_c^{n+1})$, which is the steady-state closure used by the paper.

Why the input has to be $\sigma = v_{\rm max}/(3\kappa)$. A point that is easy to miss is that the code does not store the physical CR flux directly. The conserved flux variable used in the transport and source steps is the rescaled quantity

$$\widetilde{\boldsymbol{F}}_c \;=\; \frac{\boldsymbol{F}_{\rm phys}}{v_{\rm max}}.$$

This is visible in `cr_flux.cpp`, where the energy-equation flux is assembled as `vmax * w_l(fdir,i)`: the stored variable is multiplied by $v_{\rm max}$ to recover the physical energy flux that appears in $\nabla \cdot \boldsymbol{F}_c$. The rescaling keeps E_c and $\widetilde{\boldsymbol{F}}_c$ on comparable units and makes the transport eigenvalues $\pm v_{\rm max}/\sqrt{3}$ come out cleanly.

In terms of this stored variable, the code's source update has the form

$$\partial_t \widetilde{\boldsymbol{F}}_c = -\, v_{\rm max}\, \sigma_{\rm input} \Big[\widetilde{\boldsymbol{F}}_c - \boldsymbol{v}\, (E_c + P_c)/v_{\rm max} \Big]. \qquad \text{(coded)}$$

Substituting $\widetilde{\boldsymbol{F}}_c = \boldsymbol{F}_{\rm phys}/v_{\rm max}$ and multiplying through by $v_{\rm max}$ gives

$$\partial_t \boldsymbol{F}_{\rm phys} = -\, v_{\rm max}\, \sigma_{\rm input} \Big[\boldsymbol{F}_{\rm phys} - \boldsymbol{v}\, (E_c + P_c) \Big].$$

The continuum flux equation (2), multiplied by $v_{\rm max}^2$, reads

$$\partial_t \boldsymbol{F}_{\rm phys} = -\, v_{\rm max}^2\, \sigma_{\rm phys} \Big[\boldsymbol{F}_{\rm phys} - \boldsymbol{v}\, (E_c + P_c) \Big]. \qquad \text{(phys)}$$

Matching the two forms term by term gives a single consistent relation,

$$\sigma_{\rm input} \;=\; v_{\rm max}\, \sigma_{\rm phys}.$$

Now take the diffusion limit: set $\boldsymbol{v} = 0$, drop $\partial_t \boldsymbol{F}_{\rm phys}$, and use (2). With $P_c = E_c/3$,

$$\nabla P_c = -\, \sigma_{\rm phys}\, \boldsymbol{F}_{\rm phys} \quad \Longrightarrow \quad \boldsymbol{F}_{\rm phys} = -\, \frac{1}{3\sigma_{\rm phys}}\, \nabla E_c.$$

Matching Fick's law $\boldsymbol{F}_{\rm phys} = -\kappa\, \nabla E_c$ requires

$$\sigma_{\rm phys} \;=\; \frac{1}{3\kappa}.$$

Combined with $\sigma_{\rm input} = v_{\rm max}\sigma_{\rm phys}$, this gives

$$\sigma_{\rm input} \;=\; \frac{v_{\rm max}}{3\kappa},$$

which is the input relation used in practice (e.g. `sigma = vmax/(3.0*kappa)`). Without the $v_{\rm max}$ factor, the two-moment scheme would *not* recover Fick's law with the intended κ in the diffusion limit.

2.2.2 Operator Splitting

The CR two-moment system has two very different pieces:

- *Transport:* hyperbolic fluxes that move E_c and $\boldsymbol{F}_c$ between cells at speeds up to $v_{\rm max}$.
- *Sources:* local, stiff terms that relax $\boldsymbol{F}_c$ toward its equilibrium value and couple CRs to the gas.

Solving both at once is inefficient, because transport is best handled by explicit Godunov methods, while sources require a stable implicit update. *Operator splitting* evolves each part separately, in sequence, with the scheme suited to it.

Formally,

$$\frac{d\boldsymbol{U}}{dt} \;=\; T(\boldsymbol{U}) + S(\boldsymbol{U}), \qquad \boldsymbol{U} = \begin{bmatrix} E_c \\ \widetilde{\boldsymbol{F}}_c \end{bmatrix},$$

where T denotes the transport terms and S the source terms, and the second component uses the rescaled flux actually stored by the code (see the pedbox above). In the absence of the MHD-specific rotation and the separate perpendicular-work addition, the split update has the schematic form

$$\boldsymbol{U}^{*} = \boldsymbol{U}^{n} + \Delta t\, T(\boldsymbol{U}^{n}) \quad \text{(explicit transport)},$$
$$\boldsymbol{U}^{n+1} = \boldsymbol{U}^{*} + \Delta t\, S(\boldsymbol{U}^{\,n+1}) \quad \text{(implicit sources)}.$$

The actual implementation is more specific. The implicit step first rotates the state into the local field-aligned frame, solves the diagonal 4×4 system there, rotates back to simulation coordinates, and in the MHD branch adds the perpendicular work term to E_c afterward. Appendix A.3 gives the details.

3 The Riemann Problem, Left/Right States, and HLLE

3.1 Why Left and Right States Appear at a Face

Finite-volume methods store *cell averages* at centers. Computing the flux through a face requires values *on either side of that face.* These are obtained by *reconstruction*: from neighboring cell averages, a left state U_L (from the left cell) and a right state U_R (from the right cell) are built *at the common face.*

Reconstruction order matters:

- 1st order (donor cell): Simple but diffusive
- 2nd order (PLM): Good balance of accuracy and robustness
- 3rd order (PPM): Most accurate but requires NGHOST$\geq$3

Higher order reduces numerical diffusion, which is crucial for capturing sharp CR gradients.

In 1D with cells indexed by i, the face between cells i and $i+1$ is at $x_{i+1/2}$. Reconstruction produces

$$U_L \equiv U(x^-_{i+1/2}), \qquad U_R \equiv U(x^+_{i+1/2}).$$

3.2 The Riemann Problem at a Face

At each cell face $x_{i+1/2}$, the finite-volume scheme requires the intercell flux $F_{i+1/2}$ in order to update the cell averages. From reconstruction only the states on either side of the face are known:

$$U(x,0) = \begin{cases} U_L, & x < x_{i+1/2}, \\ U_R, & x > x_{i+1/2}. \end{cases}$$

This piecewise constant data is the initial condition of a *Riemann problem*. The Riemann problem is defined as evolving this discontinuity under the homogeneous hyperbolic system

$$\partial_t U + \partial_x F(U) = 0,$$

i.e. the transport equations without source terms.

Its solution consists of a fan of waves (shocks, rarefactions, contacts) that spread away from the interface. The numerical flux is obtained by *sampling the Riemann solution at the interface itself*:

$$F_{i+1/2} = F\big(U(x_{i+1/2}, t)\big) \quad (t > 0).$$

In words: given one state to the left and one to the right, the code solves the Riemann problem and reads off the state sitting at the face. This sampled flux is then used in the finite-volume update.

For the CR two-moment system, the characteristic waves propagate at speeds $\pm v_{\max}/\sqrt{3}$.

3.3 HLLE Numerical Flux

The HLLE (Harten–Lax–van Leer–Einfeldt) solver approximates the full Riemann fan by just its slowest and fastest waves. Let a_L and a_R denote estimates of the leftmost and rightmost signal speeds (eigenvalues of the flux Jacobian). Then the HLLE flux at an interface is

$$F^{\mathrm{HLLE}}(U_L, U_R) = \frac{a_R\, F(U_L) - a_L\, F(U_R) + a_L a_R\, (U_R - U_L)}{a_R - a_L},$$

valid when the fan straddles the interface ($a_L < 0 < a_R$).

In the degenerate cases the formula reduces to simple upwinding:

$$F^{\mathrm{HLLE}} = \begin{cases} F(U_L), & a_L \geq 0 \quad \text{(all waves to the right)}, \\ F(U_R), & a_R \leq 0 \quad \text{(all waves to the left)}. \end{cases}$$

This matches the exact Riemann solution: if all signals move rightward, the interface state is simply U_L; if all move leftward, it is U_R. The general formula above continuously limits to these cases as $a_L \to 0^+$ or $a_R \to 0^-$.

3.4 Wave Speed Estimates with Diffusion

The naive choice $a_L = -v_{\max}/\sqrt{3}$, $a_R = +v_{\max}/\sqrt{3}$ works but is overly diffusive in the streaming regime. The code uses refined estimates:

$$a_L^{\text{raw}} = \min\left(\frac{v_{n,L}+v_{n,R}}{2} - \frac{v_{\text{diff},L}+v_{\text{diff},R}}{2},\ v_{n,L} - v_{\text{diff},L}\right) \tag{17}$$

$$a_R^{\text{raw}} = \max\left(\frac{v_{n,L}+v_{n,R}}{2} + \frac{v_{\text{diff},L}+v_{\text{diff},R}}{2},\ v_{n,R} + v_{\text{diff},R}\right) \tag{18}$$

These incorporate:

- v_n: gas velocity (advection with the flow)
- v_{diff}: effective diffusion speed (see Section A.2)

The final bounds ensure the characteristic estimates remain within the hyperbolic envelope of the two-moment system:

$$a_L = \max(a_L^{\text{raw}}, -v_{\max}/\sqrt{3}), \quad a_R = \min(a_R^{\text{raw}}, +v_{\max}/\sqrt{3}). \tag{19}$$

The sign structure required by HLLE is then imposed in the usual Einfeldt form by replacing $a_R \to \max(a_R, 0)$ and $a_L \to \min(a_L, 0)$ before assembling the numerical flux.

4 Numerical Road Map: From Equations to Code

CRAdvance timestep:
1. `CalculateFluxes` → calls `CRFlux` for HLLE fluxes.
2. `FluxDivergence` → accumulates finite-volume update.
3. `AddSourceTerms` → implicit scattering/streaming solve, couples to gas.

Having derived the governing equations and outlined the numerical scheme, the remaining sections turn to the `Athena++` implementation. This provides a file-by-file guide to the relevant modules, clarifying how each component of the code realizes the steps of the algorithm.

High-Level Structure

The timestep driver for cosmic rays lives in `cr.cpp`. Each timestep is split into two stages:

1. *Transport:* compute fluxes across cell faces and accumulate their divergence.
2. *Sources:* apply scattering, streaming, and diffusion, and couple CR energy and momentum to the gas.

The routines are modular, so that the update of $\boldsymbol{F}_c$ can be traced directly through these steps.

Transport Step (`CRFlux`, `CalculateFluxes`)

The transport step advances the hyperbolic part of the system, written in terms of the stored flux $\widetilde{\boldsymbol{F}}_c = \boldsymbol{F}_{\text{phys}}/V_m$:

$$\partial_t \begin{bmatrix} E_c \\ \widetilde{\boldsymbol{F}}_c \end{bmatrix} + \nabla \cdot \begin{bmatrix} V_m\, \widetilde{\boldsymbol{F}}_c \\ V_m\, \mathbf{P}_c \end{bmatrix} = 0,$$

whose eigenvalues are $\pm V_m/\sqrt{3}$. Both entries of the flux vector carry a leading V_m: the first because the conserved E_c is transported by the physical flux $V_m\,\widetilde{\boldsymbol{F}}_c$, and the second because the continuum flux equation $(1/V_m^2)\,\partial_t \boldsymbol{F}_{\rm phys} + \nabla\cdot\mathbf{P}_c = 0$, multiplied by V_m^2 and rewritten in terms of $\widetilde{\boldsymbol{F}}_c$, has the flux $V_m\,\mathbf{P}_c$. Both factors are built directly in `cr_flux.cpp`: the CR-energy flux as `vmax * w_l(fdir,i)` and the pressure contribution to the CR-momentum flux as `vmax * eddl * w_l(CRE,i)`, i.e. $V_m\,P_c$ with the isotropic Eddington factor 1/3.

Implementation details:

- `CalculateFluxes` reconstructs left and right states at each cell face.
- `CRFlux` applies the HLLE Riemann solver using bounding wave speeds. In the diffusion limit, these speeds are dynamically reduced to avoid excessive numerical diffusion (see Appendix A.2).
- `FluxDivergence` accumulates the face fluxes into cell-centered updates.

Source Step (`AddSourceTerms`)

The source step relaxes the CR flux toward its steady-state value, consistent with the diffusion coefficient κ, provided the user supplies $\sigma = V_m/(3\kappa)$. In schematic form, writing $\boldsymbol{v}_{\rm tot} = \boldsymbol{v} + \boldsymbol{v}_s$ and using the stored flux variable $\widetilde{\boldsymbol{F}}_c$:

$$\partial_t \widetilde{\boldsymbol{F}}_c = -V_m\sigma\Big(\widetilde{\boldsymbol{F}}_c - \boldsymbol{v}(E_c + P_c)/V_m\Big),$$
$$\partial_t E_c = -\boldsymbol{v}_{\rm tot}\cdot\sigma\Big(\widetilde{\boldsymbol{F}}_c - \boldsymbol{v}(E_c + P_c)/V_m\Big) + (\text{perpendicular work, MHD only}).$$

The two equations share the same bracketed residual, but the E_c equation carries an outer $\boldsymbol{v}_{\rm tot}$ factor that the flux equation does not. This factor is the origin of the $v_j\,v_j^{\rm tot}$ products in A_{11} and of the $v_j^{\rm tot}$ entries in A_{1j} in the 4×4 implicit system; it is visible directly in `cr_source.cpp` as the `vtot1`, `vtot2`, `vtot3` factors multiplying `sigma_x`, `sigma_y`, `sigma_z`. The perpendicular-work term, $\boldsymbol{v}_\perp\cdot\nabla P_{c,\perp}$, is not part of that matrix: it is precomputed in `cr_transport.cpp` as `ec_source_` and added to the post-solve E_c via `new_ec += dt * ec_source_` in the MHD branch of `AddSourceTerms`.

The complication is that scattering and streaming are anisotropic. The relaxation rate differs parallel and perpendicular to $\boldsymbol{B}$, so applying the operator directly in Cartesian coordinates is not sufficient. The implementation therefore proceeds in four steps:

1. Rotate $(\boldsymbol{F}_c, \boldsymbol{v})$ into the local field-aligned frame.
2. Apply the diagonal operator: $\sigma_\parallel$ on the parallel flux, $\sigma_\perp$ on the perpendicular fluxes.
3. Rotate the updated state back into simulation coordinates.
4. Apply equal and opposite source terms to the gas.

The transport step does not rotate the conserved state itself into the field-aligned frame; the hyperbolic characteristic structure of the two-moment system is isotropic with fixed speeds $\pm V_m/\sqrt{3}$. Field geometry does still enter the transport step indirectly, however: the diffusion-reduced signal speeds $v_{\rm diff}$ are first constructed in a field-aligned sense and then rotated back to simulation coordinates before entering the Riemann solve (see Appendix A.2, Step 1, and the `InvRotateVec` call in `cr_transport.cpp`).

What is specific to the source step is rotating the conserved flux *itself* into the field frame. This is done there because the scattering/streaming operator is anisotropic and encodes field-aligned microphysics: $\sigma_\parallel$ and $\sigma_\perp$ act differently on the parallel and perpendicular components. Without that rotation, the anisotropic relaxation would spuriously mix Cartesian flux components, especially in tangled-field geometries.

Rotation Utilities (b_angle)

The routine `b_angle` computes the polar and azimuthal angles of the local magnetic field. From these, the code constructs rotation matrices that align $\boldsymbol{B}$ with the x-axis. Any vector or tensor can then be transformed into this frame, updated with diagonal operators, and rotated back.

Because the rotation is recomputed independently in every cell, the method remains valid even in simulations with highly curved or tangled fields. Cosmic rays always relax along the local tangent $\hat{\boldsymbol{b}}$, provided the mesh resolves the curvature scale. The rotation is a purely algebraic operation and adds negligible cost compared to the Riemann solver.

Where to Look in the Code

- `CRIntegrator::CRFlux` – HLLE flux calculation.
- `CRIntegrator::CalculateFluxes` – reconstruction and flux calls.
- `CRIntegrator::FluxDivergence` – divergence accumulation.
- `CRIntegrator::AddSourceTerms` – anisotropic scattering and gas coupling.
- `b_angle` – construction of field-aligned rotation matrices.

A Mathematical Details of Rotation, Diffusion, and Implicit Solve

This appendix collects the derivations underlying the algorithm described in Section 4.

A.1 Rotation into the $\hat{\boldsymbol{b}}$-Frame

To diagonalize the anisotropic scattering tensor, the local basis is rotated to align with the magnetic field. Given $\boldsymbol{B} = (B_x, B_y, B_z)$, define

$$\hat{\boldsymbol{b}} = \boldsymbol{B}/|\boldsymbol{B}|.$$

Then construct orthonormal vectors $\{\hat{\boldsymbol{b}}, \hat{\boldsymbol{e}}_1, \hat{\boldsymbol{e}}_2\}$ using spherical angles (θ, ϕ):

$$\sin\theta = \frac{\sqrt{B_x^2 + B_y^2}}{B}, \qquad \cos\theta = \frac{B_z}{B}, \tag{20}$$

$$\sin\phi = \frac{B_y}{\sqrt{B_x^2 + B_y^2}}, \qquad \cos\phi = \frac{B_x}{\sqrt{B_x^2 + B_y^2}}. \tag{21}$$

From these,

$$\hat{\boldsymbol{b}} = (\sin\theta\cos\phi,\ \sin\theta\sin\phi,\ \cos\theta), \tag{22}$$

$$\hat{\boldsymbol{e}}_1 = (\cos\theta\cos\phi,\ \cos\theta\sin\phi,\ -\sin\theta), \tag{23}$$

$$\hat{\boldsymbol{e}}_2 = (-\sin\phi,\ \cos\phi,\ 0). \tag{24}$$

The rotation matrix

$$\mathsf{R} = [\,\hat{\boldsymbol{b}}\;\;\hat{\boldsymbol{e}}_1\;\;\hat{\boldsymbol{e}}_2\,]$$

maps vectors as $\boldsymbol{v}' = \mathsf{R}^T\boldsymbol{v}$ and $\boldsymbol{v} = \mathsf{R}\boldsymbol{v}'$. In this frame the scattering tensor becomes

$$\boldsymbol{\sigma}_c = \mathrm{diag}(\sigma_\parallel,\ \sigma_\perp,\ \sigma_\perp).$$

Since $\boldsymbol{B}(x, y, z)$ may vary arbitrarily, the rotation is recomputed independently in every cell. This ensures that CRs always relax toward the local tangent $\hat{\boldsymbol{b}}$. If the field curvature is resolved by the mesh, the scheme captures anisotropic transport along tangled fields. If not, small perpendicular leakage appears, a limitation of spatial resolution rather than of the rotation method itself.

A.2 HLLE Flux and Diffusive Signal Speed

The HLLE flux is assembled in `CRIntegrator::CRFlux` (`cr_flux.cpp`). That function needs left/right estimates of the fastest signal speeds, which it calls `al` and `ar`. Those speeds ultimately depend on the effective diffusion velocity $v_{\rm diff}$, computed earlier in `cr_transport.cpp` from the cell optical depth.

Step 1: Optical depth. The code first computes an effective scattering coefficient σ_c:

- If only diffusion is enabled, $\sigma_c = \sigma_{\rm diff}$.
- If streaming is enabled, σ_c is combined harmonically with the "advective" opacity $\sigma_{\rm adv}$:

$$\sigma_c = \left(\frac{1}{\sigma_{\rm diff}} + \frac{1}{\sigma_{\rm adv}}\right)^{-1}.$$

  This is the effective coupling used in both the transport and source updates when both diffusion and streaming are active; the harmonic form means the smaller of the two opacities dominates.

The raw cell optical depth is

$$\tau \;=\; \tau_{\rm fact}\,\sigma_c\,\Delta x,$$

with `tau_fact` a user parameter that can inflate or deflate the effective depth. The code then constructs a rescaled argument

$$\tilde{\tau} \;\equiv\; \frac{\tau^2}{2\,f_{nn}},$$

where f_{nn} is the face-normal Eddington factor (`eddxx` in the code, equal to 1/3 for the isotropic closure). It is $\tilde{\tau}$, not τ, that enters the reduction factor below.

Step 2: Reduction factor. The reduction factor applied to the free-streaming signal speed is

$$R(\tilde{\tau}) \;=\; \sqrt{\frac{1 - e^{-\tilde{\tau}}}{\tilde{\tau}}}.$$

In the code this is implemented as

```
if (taux < tau_asymptotic_lim) {
    diffv = sqrt(1.0 - 0.5 * taux);
} else {
```

```
    diffv = sqrt((1.0 - exp(-taux)) / taux);
}
v_diff = vmax * sqrt(eddxx) * diffv;
```

The `taux` variable in the source is $\tilde{\tau}$, and the small-argument branch is the Taylor expansion of R near zero, used to avoid catastrophic cancellation.

Two limits matter:

- $\tilde{\tau} \ll 1$: $R \approx 1$, giving the free-streaming bound $v_{\rm diff} \approx V_m/\sqrt{3}$.
- $\tilde{\tau} \gg 1$: $R \approx 1/\sqrt{\tilde{\tau}}$, suppressing the signal speed so the scheme approaches the diffusive limit.

This makes the flux scheme asymptotic-preserving: it smoothly transitions between free streaming and diffusion depending on whether a cell is optically thin or thick.

Step 3: Effective diffusive speed. The final $v_{\rm diff}$ passed into `CRFlux` is

$$v_{\rm diff} \;=\; V_m \sqrt{f_{nn}}\, R(\tilde{\tau}),$$

with optional padding terms (e.g. a CR sound speed contribution) added if `vel_flx_flag` is enabled in the code.

Step 4: Candidate wave speeds at a face. Inside `CRFlux`, the left and right states provide $(v_{n,L}, v_{\rm diff,L})$ and $(v_{n,R}, v_{\rm diff,R})$. The code forms

```
meanadv   = 0.5 * (vl + vr);
meandiffv = 0.5 * (vdiff_l(i) + vdiff_r(i));

al = min(meanadv - meandiffv, vl - vdiff_l(i));
ar = max(meanadv + meandiffv, vr + vdiff_r(i));
```

and then clips:

```
ar = min(ar, vmax * sqrt(1/3));
al = max(al,-vmax * sqrt(1/3));
```

so the eigenvalues never exceed the $\pm V_m/\sqrt{3}$ bounds.

Step 5: Assemble HLLE flux. With upwind factors

```
bp = (ar > 0.0 ? ar : 0.0);
bm = (al < 0.0 ? al : 0.0);
```

the code builds left and right fluxes for each CR moment and combines them into

$$F^{\rm HLLE} \;=\; \frac{S_R F_L - S_L F_R + S_L S_R (U_R - U_L)}{S_R - S_L},$$

where $S_R \leftrightarrow bp$ and $S_L \leftrightarrow bm$. This is exactly the HLLE formula derived earlier, just written in a midpoint form in the source.

In short: `cr_transport.cpp` determines $v_{\rm diff}$ by computing $\tilde{\tau}$ from the local σ_c and reducing it with $R(\tilde{\tau})$. `cr_flux.cpp` takes those $v_{\rm diff}$ values, combines them with the advective velocities to estimate `al` and `ar`, clips to the $\pm V_m/\sqrt{3}$ bound, and finally plugs them into the HLLE flux formula.

A.3 Implicit Source Solve

After transport, the stiff source terms are updated implicitly. In the $\hat{\boldsymbol{b}}$-frame, backward Euler discretization produces a block-triangular 4×4 system for $\{E_c, \widetilde{F}_{\parallel}, \widetilde{F}_{\perp,1}, \widetilde{F}_{\perp,2}\}$:

$$\mathsf{A} \begin{bmatrix} E^{n+1} \\ \widetilde{F}_{\parallel}^{n+1} \\ \widetilde{F}_{\perp,1}^{n+1} \\ \widetilde{F}_{\perp,2}^{n+1} \end{bmatrix} = \begin{bmatrix} E^{*} + \Delta t\, Q \\ \widetilde{F}_{\parallel}^{*} \\ \widetilde{F}_{\perp,1}^{*} \\ \widetilde{F}_{\perp,2}^{*} \end{bmatrix}. \tag{25}$$

The nonzero entries, read directly from `cr_source.cpp`, are

$$\mathsf{A}_{11} = 1 - \frac{4\,\Delta t}{3\,V_m} \sum_j \sigma_j\, v_j\, v_j^{\mathrm{tot}}, \tag{26}$$

$$\mathsf{A}_{1j} = +\Delta t\, \sigma_j\, v_j^{\mathrm{tot}}, \tag{27}$$

$$\mathsf{A}_{j1} = -\tfrac{4\,\Delta t}{3}\, \sigma_j\, v_j, \tag{28}$$

$$\mathsf{A}_{jj} = 1 + \Delta t\, V_m\, \sigma_j. \tag{29}$$

Note the sign of A_{1j}: it is positive, opposite in sign to A_{j1}. This follows from the asymmetry noted earlier between the E_c source (outer $\boldsymbol{v}_{\mathrm{tot}}$ factor) and the flux source (outer V_m factor); the schematic matrix written in Section 2 uses the convention $\mathsf{A}_{1j} = -\Delta t\, \beta_j$, which simply absorbs the sign into $\beta_j < 0$.

Because flux components couple only through E_c, the system is solved efficiently by Schur complement:

$$E^{n+1} = \frac{b_1 - \sum_j \mathsf{A}_{1j}\, b_{j+1}/\mathsf{A}_{jj}}{\mathsf{A}_{11} - \sum_j \mathsf{A}_{1j}\, \mathsf{A}_{j1}/\mathsf{A}_{jj}}, \tag{30}$$

$$\widetilde{F}_j^{n+1} = \frac{b_{j+1} - \mathsf{A}_{j1}\, E^{n+1}}{\mathsf{A}_{jj}}. \tag{31}$$

The updated CR variables are rotated back to the lab frame, and equal and opposite terms are applied to the gas. In the MHD implementation, the parallel relaxation is handled by this implicit matrix solve, while the perpendicular work term $v_\perp \cdot \nabla P_{c,\perp}$ is added separately after the solve, mirroring the structure of `cr_source.cpp` and `cr_transport.cpp`.