Summary of SDE for K2

Shah Saad Alam

February 7, 2024

1 Introduction and setup

The general idea is to have a method such as sde() which can then be called from the main K2 code as illustrated in main(). We use Dr. Ma's diffusion coefficients. Pseudocode is given in Section 7.

1.1 Assumptions

1. MLT is the same as the azimuthal angle ϕ (actually, not really, need to discuss this).
2. Given the position vector, $L = |\vec{r}|$.
3. We are kicking the particles at minimum magnetic field, and therefore pitch angle is the equatorial value.

2 Diffusion coefficients

We are using chorus diffusion coefficients from Dr. Ma, with separate files for upper band and lower band data provided. The QianLi Ma files are ordered by MLT bins for 1hr time bins for each L value $3.0, 3.5, ..., 7.0$. Each file for a given set of these parameters contains $(D_{\alpha_0\alpha_0}, D_{\alpha_0 p}, D_{pp}, D_{\alpha_0 E}, D_{EE})$ for given (α_0, E) coordinate pairs.

- Time: 17th March-18th March 2013, in one hour bins.
- L: $3.0 - 7.0$ in 0.5 bins.
- MLT: $0 - 4, 4 - 8, 8 - 12, 20 - 24$hr. Coefficients are assumed to be zero in the 12-20 MLT interval.
- Pitch angle: $1° - 89°$ in $2°$ intervals, with an additional point at $89.5°$, contained in file "PitchAngle_46.txt", provided by Dr. Ma.
- Energy: 0.1 keV to 10 MeV, 71 values with equal logarithmic spacing, contained in file "Energy_71.txt", provided by Dr. Ma..
- Lower band and Upper Band Chorus given separately in different files.
- The LBC files and the UBC files are each around 750 MB.

2.1 Suggestions for preprocessing

1. Add LBC and UBC to get total diffusion coefficients.
2. Strip extraneous headers and metadata.
3. Convert files from α_0, E to α_0, p.
4. Combine all the files into a single large one? This would reduce folder and file open operations at runtime.
5. Read the entire pre-processed version in at run-time from main().

Expected RAM usage: 1 GB max. **Alternatively, instead of reading in a file of diffusion coefficients, we can also do further pre-processing as illustrated in Section 4.1**.

3 Random Number Generator

We need a random number generator based on a gaussian distribution. One implementation I was able to find online employing the Mersenne Twister algorithm is: `http://www.courses.physics.helsinki.fi/fys/tilaII/files/mtfort90.f90` .

4 The kick equations

The relevant equations for the kick are detailed in Xin Tao's thesis, Eq.3.21-3.24.

$$\alpha_f = \alpha_i + b_1 \Delta s + \sigma_{11}\sqrt{\Delta s}N_1 + \sigma_{12}\sqrt{\Delta s}N_2 y \tag{1}$$

$$p_f = p_i + b_2 \Delta s + \sigma_{21}\sqrt{\Delta s}N_1 + \sigma_{22}\sqrt{\Delta s}N_2 \tag{2}$$

where Xin Tao makes the explicit choice of $\sigma_{12} = 0$. The other coefficients are:

$$\sigma_{11} = \sqrt{2D_{11}}/p_i, \qquad \sigma_{21} = \sqrt{2}D_{12}/\sqrt{D_{11}}, \qquad \sigma_{22} = \sqrt{2D_{22} - \sigma_{21}^2}, \tag{3}$$

and the b_1, b_2 are given by (subscript 1 for α_0, 2 for p)

$$b_1(t, \alpha_0, p) = \frac{1}{Gp}\frac{\partial}{\partial \alpha_0}(GD_{11}/p) + \frac{1}{G}\frac{\partial}{\partial p}(GD_{12}/p) \tag{4}$$

$$b_2(t, \alpha_0, p) = \frac{1}{Gp}\frac{\partial}{\partial \alpha_0}(GD_{12}) + \frac{1}{G}\frac{\partial}{\partial p}(GD_{22}) \tag{5}$$

We have to assume that the Jacobian is given by the **dipole** formula:

$$G = p^2 T(\alpha_0)\sin(\alpha_0)\cos(\alpha_0) \tag{6}$$

The normalised dipole bounce period is $T(y) \approx 1.3801730 - 0.639693y^{4/3}$ where $y = \sin\alpha_0$ (source: Liheng's thesis).

4.1 Calculating the σ and b coefficients

We can generate pre-processed values for σ and b coefficients for each (α_0, E) pair in the diffusion coefficient files. The σ's may be trivially calculated. For the b coefficients, convert:

$$\frac{\partial}{\partial p} \to \frac{d(\log E)}{dp}\frac{\partial}{\partial(\log E)} \tag{7}$$

as the spacing in the grid of diffusion coefficient coordinates is uniform in $\log(E)$. We then use first order centred finite differencing for a given quantity F:

$$\frac{\partial F}{\partial(\log E)} = \frac{F_{n+1} - F_{n-1}}{\log(E_{n+1}) - \log(E_{n-1})}, \qquad \frac{\partial F}{\partial \alpha_0} = \frac{1}{2}\left(\frac{F_{n+1} - F_n}{\alpha_{n+1} - \alpha_n} + \frac{F_n - F_{n-1}}{\alpha_n - \alpha_{n-1}}\right) \tag{8}$$

The differencing scheme for the α_0 derivatives is a suggested method for when subsequent grid intervals are not uniform.

The pre-processed files containing these coefficients can then be read in on the grid at runtime in main(). **This process would require an $<$ 1GB of RAM at runtime**. When we require these coefficients at any given point (α_0, p), we can find the grid cube containing the point and interpolate using the known value at the cube's vertices (bi-linear interpolation such as in `https://en.wikipedia.org/wiki/Bilinear_interpolation` should be sufficient).

5 K2 units and equations

Given units in cgs, with primed units denoted dimensionalised units:

$$M' = \frac{M}{qR_E};\; p' = \frac{p}{m_0 c};\; B' = \frac{qR_E}{m_0 c^2}B \tag{9}$$

it can be shown that:

$$(E_0 + E_k)^2 = (pc)^2 + E_0^2 \Rightarrow (1 + E_k')^2 = p'^2 + 1 \tag{10}$$

and

$$M = \frac{p_\perp^2}{2m_0 B} \Rightarrow M' = \frac{p'^2_\perp}{2B'} \tag{11}$$

6 Questions

1. Converting from K2 units to Xin Tao's units in main() ?
2. How specific should the read, write and findvertices() operations should be to the Dr. Ma diffusion file?
3. What to do for diffusion coefficient values for regions outside the coordinate grid used in Dr. Ma' files?
4. Is the pre-processing idea viable?

7 Pseudocode

Suggested or tentative code is in red.

Algorithm 1 Main

```
procedure MAIN
    Unit Conversions
    M_i                                              ▷ Initial M value
    p_i||                                            ▷ Initial p_|| value
    r⃗                                                ▷ Initial position vector at B_min
    B                                                ▷ Local magnetic field strength
    t                                                ▷ Time
    M_f                                              ▷ Final M value
    p_f||                                            ▷ Final p_|| value
    Δs                                               ▷ Step size in time
    Initialise randomgauss()                         ▷ Random num gen with Gaussian dist N(0,1)
    cREAD all diffusion coefficients into memory     ▷ See question above
    (M_f, p_f||) ← sde(M_i, p_i||, r⃗, B, t, Δs)       ▷ Routine for doing the kick
end procedure
```

Algorithm 2 sde

```
procedure SDE(M_i, p_i||, r⃗, B⃗, t, Δs)
    L ← |r⃗|                                          ▷ Adiabatic invariant L
    p_i⊥ ← sqrt(2 m M_i B)                           ▷ Initial perpendicular momentum
    p_i ← sqrt(p_i⊥^2 + p_i||^2)
    φ ← arctan(y/x)                                  ▷ Azimuthal angle same as MLT? Uniqueness of result of the arctan function?
    α_i ← arccos(p_i|| / p_i)                        ▷ Initial equatorial pitch angle
    (σ_11, σ_12, σ_22) ← getσ⃗(α_i, p_i, L, φ, t)      ▷ Obtain the sde sigma coefficients
    (b_1, b_2) ← getb⃗(α_i, p_i, L, φ, t)              ▷ Get the b coefficients
    N_1 ← randomgauss()
    N_2 ← randomgauss()
    α_f ← α_i + b_1 Δs + σ_11 sqrt(Δs) N_1 + σ_12 sqrt(Δs) N_2    ▷ Eq3.24 Xin Tao's Thesis
    p_f ← p_i + b_2 Δs + σ_21 sqrt(Δs) N_1 + σ_22 sqrt(Δs) N_2    ▷ Eq3.25 Xin Tao's Thesis
    M_f ← p_f^2 / (2 m B)
    p_f|| ← p_f cos(α_f)
    return M_f, p_f||
end procedure
```

Algorithm 3 getdiffcoef/getσ/get$b$

```
procedure GETσ⃗(α_i, p_i, L, φ, t)
    E_i ← sqrt(p^2 c^2 + m^2 c^4)                    ▷ Energy of particle
    Find the corresponding L, MLT, and Time bin in Diffusion coefficients.
    find vertices of square such that α_i ∈ (α_0, α_1), E_i ∈ (E_0, E_1)
    Do bi-linear interpolation (wikipedia) to find diffusion coefficients/σ,b.
    return (D_11, D_12, D_22)
end procedure
```

Algorithm 4 get$\vec{\sigma}$

```
procedure GET σ⃗(α_i, p_i, L, φ, t)
    (D_11, D_12, D_22) ← getdiffcoef(α_i, p_i, L, φ, t)      ▷ Get diffusion coefficients
    σ_11 ← √(2D_11)/p_i                                      ▷ Xin Tao Thesis eq3.21-3.23
    σ_21 ← √2 D_12/√D_11
    σ_22 ← √(2D_22 − σ_12^2)
    return (σ_11, σ_12, σ_22)
end procedure
```

8 NOTES ON UNITS

We use the ˆ to denote units in the K2 system (I know I used prime earlier, I'll fix that in post). All other units are generally cgs.

$$\hat{t} = t/t_N,\ t_N = R/c; \quad \hat{r} = r/r_N,\ r_N = R \tag{12}$$

$$\hat{v} = v/v_N,\ v_N = c \quad \hat{p} = p/p_N,\ p_N = m_0 c \tag{13}$$

Based on this, consider the *normalised* diffusion coefficient's D provided by Dr.Ma (in units of s^{-1}). Then, we can define a time-scale T:

$$T = D^{-1},\ \hat{T} = T/t_N = D^{-1}/t_N \Rightarrow \hat{D} = D * t_N; \tag{14}$$

To get D in $[momentum]^2/[time]$, we can simply do:

$$\tilde{\hat{D}} = \hat{D}\hat{p}^2 \tag{15}$$

where $\tilde{\hat{D}}$ are the diffusion coefficients in $[momentum]^2/[time]$ in K2 units.

We can also do this for the sigma's. Note we can define a time scale τ such that $\tau^{.5} = \sigma_{11}^{-1}$. Then,

$$\hat{\tau} = \tau/t_N \to \hat{\sigma_{11}}^{-1} = \sigma_{11}^{-1}/\sqrt{t_N} \to \hat{\sigma_{11}} = \sqrt{t_N}\sigma_{11} \tag{16}$$

Similarly, we have for b_2,

$$[b_2] = [momentum]/[time] \to \hat{b_2} = \frac{b_2}{p_N} * t_N \tag{17}$$

For our time step, dt (or ds), the scaling has to be:

$$\hat{dt} = dt/t_N \tag{18}$$