Chapter III
Computer Simulations of the Transport of Solar Cosmic Rays from the Sun to the Spacecraft

We consider transport problems in which the conditions are specified at more than one point. The method we use for solving such problems is the method of finite differences.

Introduction to Finite-Difference Methods

Suppose we have a differential equation of order greater than one, and the conditions are specified at the end points of an interval $[a, b]$. Consider first the case of an ordinary differential equation (Samuel D. Conte and Carl de Boor 1982). We divide the interval $[a, b]$ into N parts of width h. We set $x_0 = a, x_n = b$, and

$$x_n = x_0 + nh \qquad \text{for} \qquad n = 1, 2, 3, \ldots, N-1 \tag{3.1}$$

are the interior mesh points. The corresponding values of y are

$$y_n = y(x_0 + nh) \qquad \text{for} \qquad n = 0, 1, 2, \ldots, N. \tag{3.2}$$

Sometimes we have points outside $[a, b]$, called exterior mesh points. Those to the left of x_0 are denoted by

$$x_{-1} = x_0 - h, \; x_{-2} = x_0 - 2h, \; \ldots \tag{3.3}$$

and those to the right of x_n are denoted by

$$x_{n+1} = x_n + h, \; x_{n+2} = x_n + 2h, \; \ldots. \tag{3.4}$$

To solve a boundary-value problem by the method of finite-differences, every derivative appearing in the equation, as well as in the boundary conditions, is replaced by a difference approximation. The central differences lead to accurate approximations as follows:

$$\begin{aligned} y'(x_n) &\approx \frac{y_{n+1} - y_{n-1}}{2h} \\ y''(x_n) &\approx \frac{y_{n+1} - 2y_n + y_{n-1}}{h^2} \\ &\vdots \\ y^{(4)}(x_n) &\approx \frac{y_{n+2} - 4y_{n+1} + 6y_n - 4y_{n-1} + y_{n-2}}{h^4} \end{aligned} \tag{3.5}$$

We find that the error of these finite-difference approximations is of order h^2. For a linear second order differential equation

$$y''(x) + f(x)y' + g(x)y = q(x) \tag{3.6}$$

with the boundary conditions

$$\begin{aligned} y(x_0) &= \alpha \\ y(x_n) &= \beta \end{aligned} \tag{3.7}$$

these finite-difference approximations yield

$$\frac{y_{n-1} - 2y_n + y_{n+1}}{h^2} + \frac{f(x_n)(y_{n+1} - y_{n-1})}{2h} + g(x_n)y_n = q(x_n)$$
$$\text{for} \quad n = 1, 2, 3, \ldots, N-1 \tag{3.8}$$

Multiplying this by h^2 and setting $f(x_n) = f_n$, we have

$$(1 - \frac{h}{2}f_n)y_{n-1} + (-2 + h^2 g_n)y_n + (1 + \frac{h}{2}f_n)y_{n+1} = h^2 q_n$$
$$\text{for} \quad n = 1, 2, 3, \ldots, N-1 \tag{3.9}$$

Since y_0 and y_n are specified by equation (3.7), equation (3.9) represents a linear system of $N-1$ equations in $N-1$ unknowns, y_n for $n = 1, 2, \ldots, N-1$.

When we replace y_0 by α and y_n by β, the equations become

$$\begin{aligned}
(-2+h^2g_1)y_1+(1+\frac{h}{2}f_1)y_2 &= h^2q_1-(1-\frac{h}{2}f_1)\alpha \\
(1-\frac{h}{2}f_2)y_1+(-2+h^2g_2)y_2 &+ (1+\frac{h}{2}f_2)y_3 = h^2q_2 \\
(1-\frac{h}{2}f_3)y_2+(-2+h^2g_3)y_3 &+ (1+\frac{h}{2}f_3)y_4 = h^2q_3 \\
&\vdots \\
(1-\frac{h}{2}f_{n-2})y_{n-3}+(-2+h^2g_{n-2})y_{n-2} &+ (1+\frac{h}{2}f_{n-2})y_{n-1} = h^2q_{n-2} \\
(1-\frac{h}{2}f_{n-1})y_{n-2}+(-2+h^2g_{n-1})y_{n-1} &= -(1+\frac{h}{2}f_{n-1})\beta+h^2q_{n-1}
\end{aligned} \tag{3.10}$$

The coefficients in these equations can be computed, since $f(x), g(x)$ and $q(x)$ are known functions of x. Thus this linear system can be solved by any of the standard methods.

A Finite-Difference Method for this Transport Equation

The transport equation for the particles released from the Sun was solved by the finite-difference method of Ruffolo (1991, 1995). Currently, that is the only program that considers the effects of pitch-angle scattering, focusing, deceleration and convection. [Now, two Ph.D students (Roman Hatzky at Kiel, Germany and David Lario at Barcelona, Spain) are following Ruffolo (1995) to improve their programs to include these effects.] The distribution function $F(t, \mu, z, p)$ depends on t (time), μ (the cosine of the pitch-angle), z (the distance along the magnetic field from the Sun), and p (the momentum of a particle). The μ-flux is defined by

$$S_\mu(t,\mu,z,p) = \left[\frac{v}{2L(z)}\left(1+\mu\frac{v_{sw}}{v}\sec\psi-\mu\frac{v_{sw}v}{c^2}\sec\psi\right)(1-\mu^2)\right.$$
$$\left.-v_{sw}\left(\cos\psi\frac{d}{dr}\sec\psi\right)\mu(1-\mu^2)\right]F(t,\mu,z,p)$$
$$-\frac{\varphi(\mu)}{2}\frac{\partial}{\partial\mu}\left(1-\mu\frac{v_{sw}}{c^2}v\sec\psi\right)F(t,\mu,z,p). \qquad (3.11)$$

At $\mu = \pm 1$ this is equal to zero, showing that particles will not flow into the region $|\mu| > 1$.

We solve this transport equation by splitting the right hand side of equation (2.44) into individual terms. We will study the results of updating $F(t,\mu,z,p)$ from t to $t+\Delta t$ according to the following steps:

1. To study the effects of the μ-flux, related to pitch-angle scattering, focusing, and differential convection. This was calculated from the pair of neighboring grid points μ and $\mu+\Delta\mu$. The calculating we used changed the intensity of particles (F) from

$$F(\mu) \leftarrow F(\mu) - \left(\frac{\Delta t}{4n}\right)\left[\frac{S_\mu(\mu+\Delta\mu/2)-S_\mu(\mu-\Delta\mu/2)}{\Delta\mu}\right] \qquad (3.12)$$

$$F(\mu) + \left(\frac{\Delta t}{4n}\right)\left[\frac{S_\mu(\mu+\Delta\mu/2)-S_\mu(\mu-\Delta\mu/2)}{\Delta\mu}\right] \leftarrow F(\mu). \qquad (3.13)$$

In this step we check the error by repeating for shorter steps until the result changes less than the preset tolerance.

2. To study the deceleration or systematic decrease in p (momentum). In this case we can use either t or $s = vt$ as an independent variable for the simulation. We often use s for modeling the transport of solar flare particles. The equation for the deceleration part of transport can be written as

$$\frac{\partial}{\partial t}F(t,\mu,z,p) = \frac{1}{\tau_d}\frac{\partial}{\partial p}pF(t,\mu,z,p), \qquad (3.14)$$

where the deceleration time is

$$\frac{1}{\tau_d} = v_{sw}\left[\frac{\sec\psi}{2L(z)}(1-\mu^2) + \cos\psi\frac{d}{dr}\sec\psi\mu^2\right]. \tag{3.15}$$

Then we get

$$\frac{\partial}{\partial t}pF(t,\mu,z,p) = \frac{1}{\tau_d}\frac{\partial}{\partial \ln(p/p_0)}pF(t,\mu,z,p), \tag{3.16}$$

and

$$pF(t+\Delta t,\mu,z,p) = pe^{\Delta t/\tau_d}F(t,\mu,z,pe^{\Delta t/\tau_d}). \tag{3.17}$$

Then $F(t,\mu,z,pe^{\Delta t/\tau_d})$ is derived by geometric interpolation between p-grid points at constant t or constant s.

3. To study the effects of streaming and convection of particle released from the Sun. The term of streaming from the distribution at one z-grid point to another gives

$$F(i\Delta\mu,z,p) \leftarrow F(i\Delta\mu,z-i\Delta z,p), \tag{3.18}$$

where $\mu = i\Delta\mu$ and $\Delta z = \Delta\mu(v\Delta t)$. The equation for streaming of these particles is

$$\frac{\partial F(t,\mu,z,p)}{\partial t} = -\mu v\frac{\partial F(t,\mu,z,p)}{\partial z}. \tag{3.19}$$

The main steps are updated for use in simulating the cosmic rays transport. The following section details program files in the transport simulations.

wind.c

It contains the main program for simulating the transport of energetic particles. It will get the values from the user and call the other files to run for the specific cases. The wind.c file comprises the subr utines *main()*, *elements()* and *step()*.

main() gets the input values from the user and calculates basic variables for this transport simulation. Some values are kept in the form of arrays and matrices.

elements() calculates elements of matrices used in step() for pitch-angle scattering at each point.

step() carries out one time step in the evolution of F.

decel.c

It is used to update $F(t, \mu, z, p)$ for the effects of deceleration. If s is used as a dependent variable, we calculate the intersection of characteristics with curve of constant s on a plot of $\ln(p)$ versus t. Along the characteristics, which are straight lines, pF is constant (Ruffolo 1995). This file can have the two routines decel() and ci().

ci() is a routine for calculating the momentum at the characteristic-intersection point by Newton's method.

field.c

It contains routines that depend on the configuration of the interplanetary magnetic field. It has a lot of versions to facilitate changes in these routines for different configurations or diffusion coefficients. The subroutines in field.c are:

dzz() calculates an effective spatial diffusion coefficient to show the strength of the pitch-angle scattering. This is used to calculate A for the user's value of λ.

diffcoeff() calculates the scattering coefficient.

antideriv() contains the antiderivative of the inverse of the diffusion coefficient divided by $A(1-\mu^2)$, when the index q is not equal to 2.

mudot() calculates the rate of increase of μ due to adiabatic

focusing and solar wind effects.

arclength() finds the pathlength along the field from the Sun to the given radius.

radius() returns the radius for a given pathlength, and is the inverse of arclength().

cospsi() finds the cosine of the field's angle from the radial direction.

dsecdz() calculates $d\sec\psi/dz$.

zenith() finds the zenith angle relative to the flare site.

decelrate() calculates the rate of deceleration.

initial.c

It is used to determine in the initial value of the distribution function $F(p, \mu, z)$.

inject.c

It to calculates the injection of the particles after the start of the simulation.

nrutil.c

It contains routines to reserve and unreserve the memory for arrays.

printout.c

It prints out the desired data.

stream.c

It updates F for the streaming and convection of particles.

tridag.c

It solves a tridiagonal matrix equation, and is copied from Press et al. (1988)

How to Use This Program

Compiling and linking this program requires nine files: wind.c, decel.c, field.c, initial.c, inject.c, nrutil.c, printout.c, stream.c and tridag.c. Some files have several versions, as follows:

field.c:

f_arc.c for an Archimedean field.

f_elong.c includes an extra "elongation factor" in the definition of z to account for a windy magnetic field.

f_exp.c for an exponential field. The constant focusing length is specified by defining the constant L.

f_morsca.c is used for more scattering.

f_quiet.c has no scattering or focusing to test the deceleration.

f_vl.c is used for a variable λ. This only affects diffcoeff().

initial.c

ini_bie.c sets the initial condition to that of Bieber et al. (1980).

ini_del.c sets $F(t=0,\mu,z,p)=0$ except at the first z value, and the highest μ value. There F is set so that $\iint F\mathrm{d}\mu\mathrm{d}z \equiv 1$. This simulates direct injection from a solar flare.

ini_lowz.c sets the flux to be non-zero only at the lowest value of z.

ini_one.c sets $F(t=0,\mu,z,p)\equiv 1$.

ini_read.c reads the initial values from the dump.dat file.

ini_zero.c sets $F(t = 0, \mu, z, p) = 0$. This is used for simulating a neutron decay proton flare.

inject.c

inj_nde.c is used for studying the injection for neutron decay electron events.

inj_ndem.c is used for determining decay electrons from different neutron energies.

inj_ndp.c is used for neutron decay proton events.

inj_null.c does nothing.

printout.c

p_3d_av.c prints output averaged over a range of z-values, and output for 3D plots.

p_all.c prints all values of $F(z, \mu, p)$.

p_fav2.c prints only the average F at a given z for all μ.

p_i2_av.c prints out the intensity and anisotropy in the fixed frame.

p_muave.c prints out z and $\langle \mu \rangle$.

p_null.c gives no output.

p_zsumf.c prints out z and the sum of $F(z, \mu, p)$ for every value of μ.

stream.c

s_abs.c assumes absorbing boundary conditions at $z = 0$ and $z =$ length. The particles are not reflected.

s_noconv.c assumes that the particles experience no convection.

s_null.c does nothing.

s_refl.c imposes reflecting boundary conditions.

We must select the appropriate versions for the transport problem of interest. After we have the files for running, we compile and link them and run the executable program. Then we must input the initial values:

1. initial value of t or s,
2. final value of t or s,
3. step size,
4. range for printing,
5. number of μ points,
6. length of the simulation region,
7. number of momentum points,
8. momenta (in MeV/c),
9. particle mass,
10. solar wind speed divided by c,
11. scattering mean free path,
12. scattering power law index, and
13. whether to print extra diagnostic information (0 = no, 1 = yes).