

flare:  
A Unified Code for Acceleration, Transport &  
Radiation in Solar Flare Loops

Vahé Petrosian, Timothy Q. Donaghy, and Nicole M. Lloyd

April 30, 2001

## 1 General Description

This document describes the numerical routine `flare`, which is a unified code that calculates the characteristics of accelerated particles and bremsstrahlung photons, in general, with the first aim of application to a solar flare loop. The examples given below will be appropriate for electrons in a solar flare loop but the code can be applied to any problem where a simultaneous calculation of the acceleration, transport and radiation is desired. This code is a combination of three codes: An acceleration code, a particle transport code and a bremsstrahlung radiation code. The first gives the spectrum of the accelerated particles, integrated over a finite acceleration site (at the top of the flare loop), where background plasma particles are accelerated to relativistic energies through interaction with turbulent plasma waves while undergoing Coulomb and synchrotron (and/or inverse Compton) losses. The second treats the transport of particles escaping the acceleration region (along the loop field lines) into a thick target (loop footpoints located in the solar photosphere) and evaluates the spectral (and pitch angle distribution) evolution, and determines the overall so-called “thick target” or “cooled” spectrum. The transport code incorporates collisional and synchrotron losses as well as magnetic field convergence effects. The third code calculates the directivity, spectrum and polarization of the bremsstrahlung radiation from different parts of the source (along the loop and from the entire loop). The input parameters are the characteristics of the plasma in the acceleration site and in the transport (thick target) region (such as, dimensions, density, magnetic field value and geometry, temperature, and spectrum and density of plasma waves or turbulence). The output is the characteristics of the accelerated

electrons, and bremsstrahlung photons at different parts of the source and integrated over the whole source region.

## 1.1 Description of the Basic Codes

The routine flare consists of C++ codes compiled using the GNU compiler, g++. To compile flare, one needs the following set of files (also listed in Table A.2):

- **A. Primary Source and Header Files.** These are the main routines that do the necessary calculations, as well as handling the input and output of the data.
  1. `sacparticle.cc`: Main routine for the entire program. It reads the input files, runs the required routines, and prints to the output files. Table A.3 gives further details for this routine.
  2. `Particle.cc`, `Particle.h`, `ParticleDist.cc`: Main routine for calculating the stochastic acceleration electron distribution (before transport).
  3. `Fokker.cc`, `Fokker.h`, `FokkerChangcoop.cc`: The nuts and bolts for solving the Fokker-Planck equation via the Chang-Cooper method.
  4. `Outfile.cc`, `Outfile.h`: Sets up data arrays to hold the output.
  5. `Photon.cc`, `Photon.h`: Main code for transport and radiation.
  6. `ElectronFlux.cc`: Routine to solve the transport problem.
  7. `Radiation.cc`: Routine to calculate Bremsstrahlung radiation during transport.
  8. `cross.c`, `cross.h`: Routine for computation of the bremsstrahlung cross section.
- **B. Auxiliary/Support Files.** These consist primarily of libraries and “toolboxes” that contain functions and routines used in the primary programs listed above.
  1. `Functions.cc`: Code that contains some useful mathematical functions used in the acceleration, radiation and transport calculations.
  2. `toolbox.c`, `toolbox.h`: Toolbox of useful routines.
  3. `libctools.a`, `tools.h`: Library and header file for some useful routines used throughout program.

4. libcrecipes.a, recipes.h: Library and header file for Numerical Recipes routines (Press, Teukolsky, Vetterling, & Flannery, 1986).
  5. libcfinite.a, finite.h: Library and header file for routines used in solving finite difference equations.
- **C. Make and Parameter Files.** In addition to the main source code and the auxiliary files and libraries, one needs the following:
    1. Makefile: The file used to compile and produce the executable flare. [The users should make sure that the compiler flags that link to the libraries are correct for their particular systems. To avoid the need for alteration of the Makefile, we recommend placing the libraries in a /lib directory and the header files in a /include directory.]
    2. paramfile0: This contains general input parameters for the overall program, which determine the part(s) of the program to run (e.g. particle acceleration, transport, and/or radiation), the total electron energy range and the number of logarithmically spaced energy bins.
    3. paramfile1: Input parameters for the particle acceleration code, describing the spectrum of the background or injected particles, as well as the characteristics of the accelerating waves and background plasma.
    4. paramfile2: Input parameters for the transport code, describing the variation of the magnetic field and plasma density (and if needed the ionization structure) of the background plasma. The current setup is appropriate for a solar flare loop. The section below describes the physics of the codes and Table A.1 describes the variables specified by paramfile0, 1 and 2. The variables are also described in detail in §2.

### 1.1.1 Compiling and Running the Code

- To **compile** the program type the command `make` in a directory with all the source code; the output will be an executable file called `flare`. [Note: To delete all “.o” files (from previous runs) and start over, type `make clean`. Then retype `make` to compile again.]
- To **run** the program, type `./flare`. The program will read from the three paramfiles to extract the needed parameters and calculate the output.

The data is output in four files called `dataElspec`, `dataEldepth`, `dataPhspec`, and `dataPhdepth`, which are described below.

**We now describe each module of flare in turn.**

## 2 Stochastic Acceleration

### 2.1 Physical Motivation & Theory

In this section we give a brief review of the physical system that the particle acceleration portion of flare describes and the theory behind the numerical calculations. Stochastic acceleration of particles can be described by the Fokker-Planck diffusion equation. As long as the gyro-radii of particles,  $r_g \simeq c\beta\gamma/\Omega$ , is much smaller than the spatial variations of the magnetic field, the phase space distribution of particles can be described by four variables: time, spatial coordinate  $s$  along the magnetic field lines, energy, and pitch angle  $\mu$ . Then the equation for the distribution of the particles, which undergo stochastic acceleration by interaction with the plasma turbulence, can be written as:

$$\frac{\partial f}{\partial t} = -\mu c\beta \frac{\partial f}{\partial s} - \frac{\partial}{\partial E}(\dot{E}_L f) - \frac{\partial}{\partial \mu}(\dot{\mu} f) + \frac{\partial}{\partial \mu}(D_{\mu\mu} \frac{\partial f}{\partial \mu}) + \frac{\partial}{\partial E}(D_{EE} \frac{\partial f}{\partial E}) + \dots + \mathcal{S}(E, \mu, s, t). \quad (1)$$

where  $\beta$  is the particle velocity in units of the speed of light,  $\dot{E}_L$  and  $\dot{\mu}$  denote the energy losses and pitch angle changes respectively,  $D_{\mu\mu}$  and  $D_{EE}$  are the pitch angle and energy diffusion coefficients respectively, and  $\mathcal{S}$  is a source term. This equation can be simplified under certain assumptions. For example, if the pitch angle diffusion rate is larger than all other rates, the particle distribution becomes isotropic quickly and the third and fourth terms on the right hand side of the above equation can be ignored. The equation simplifies to:

$$\frac{\partial f}{\partial t} = \frac{\partial^2}{\partial E^2}[D(E)f] - \frac{\partial}{\partial E}[(A(E) - |\dot{E}_L|)f] - \frac{f}{T_{esc}(E)} + Q(E). \quad (2)$$

where  $D$  and  $A$  are the diffusion and systematic acceleration coefficients, and  $T_{esc}$  is the particle escape time; these parameters are related to  $D_{\mu\mu}$  and  $D_{EE}$  (or the momentum diffusion coefficient  $D_{pp}$ ). In this code, we use the parameterization which is similar to what is expected from stochastic acceleration by Alfvén or Whistler waves (see Hamilton and Petrosian, ..):

$$D(E) = \mathcal{D}\beta(\gamma\beta)^{q'} \quad (3)$$

$$A(E) = \mathcal{A}(\gamma\beta)^{q'-1} = \mathcal{D}(q' + 2)(\gamma\beta)^{q'-1} \quad (4)$$

$$T_{\text{esc}}(E) = \mathcal{T}_{\text{esc}}(\gamma\beta)^s / \beta + L / (\beta c \sqrt{2}) \quad (5)$$

where  $L$  is the size of the acceleration region. There are five independent parameters,  $\mathcal{D}$ ,  $\mathcal{T}_{\text{esc}}$ ,  $L$ ,  $s$ , and  $q'$ . The parameter  $\dot{E}_L = \dot{E}_{\text{Coul}} + \dot{E}_{\text{synch}} = -4\pi r_0^2 cn \ln \Lambda / \beta - 4r_0^2 B^2 \beta^2 \gamma^2 / 9m_e c^2$  describes the Coulomb and synchrotron energy loss rates (in units of  $m_e c^2$ ), which add two additional parameters,  $n$  and  $B$  (we set  $\ln \Lambda = 20$ ). Note that the diffusion and acceleration coefficients are parameterized in a general way such that different forms of turbulence may be accommodated. Setting  $q = 2$ ,  $s = 0$ , for example, is the parameterization for scattering off a hard sphere. These equations and parameterizations are described in detail in the following papers: Hamilton and Petrosian (1992), whistler waves; Park, Petrosian & Schwartz (1997), all three forms; for analytic solutions see Park and Petrosian (1994), and for numerical solutions Park and Petrosian (1995). For general evaluation of the Fokker-Planck coefficients see Dung & Petrosian (1994) and Pryadko & Petrosian (1997, 1998, and 1999).

## 2.2 The Routines and Sub-routines

The first part of flare consists of routines to calculate the initial stochastic acceleration of electrons, using the Chang-Cooper method, before transport effects along the loop are calculated. This portion of the program is implemented by the main routine `Particle` and the classes `Fokker`, `Outfile`, which we describe below. [Note the term “class” refers to the C++ data structures.] The idea here is that you can give a list of input parameters (see below) describing the relevant physical situation, and the program will calculate the time dependent, time integrated, or steady state electron distribution due to stochastic acceleration, as described in equations 2 - 5. The program includes pitch angle averaged synchrotron and Coulomb losses within the acceleration region. Because it is not dependent on any particular geometry, chemical composition of region, etc., the program can be used to calculate stochastic acceleration for a number of astrophysical situations, which are defined by the input parameters.

1. *Class Particle* – This class contains the main routines for calculating the loop top electron distribution, or initial stochastic acceleration electron distribution without transport. It sets up the coefficients and functions to be inputted into `Fokker`, which actually does the calculation of solving the Fokker-Planck equation. For example, the background plasma or injection particle spectrum is defined in `Particle` and

this is the part one would need to edit to add an injection function that is not a thermal or gaussian function. See Table A.4.

2. *Class Fokker* – This class contains the nuts and bolts routines for solving the Fokker-Planck equation using the Chang-Cooper finite differencing method. See Table A.6.
3. *Class Outfile* – This class is mainly responsible for setting up the data arrays that hold the accelerated electron distributions. See Table A.5.

### 2.3 How to Run this Portion of the Program.

To implement this program *independently* of the rest of flare - that is, to just calculate the particle acceleration without worrying about transport effects or calculating the photon spectrum - one simply needs to set the first line of *paramfile0* appropriately. This line allows you to enter a three bit number that specifies which/how many modules to run. The first, second and third bits are for initial acceleration, particle transport, and photon spectrum, respectively. The number 1 indicates "run", while the number 0 indicates "ignore". For example, if one wants to run the particle acceleration only, the input would be 100. If one wants to implement all three modules, the input on this first line of *paramfile1* is 111. The other input parameters specified in *paramfile0* are:

- $E_{min}$  - Minimum energy of numerical grid in MeV, line 2 column 1
- $E_{max}$  - Maximum energy of numerical grid in MeV, line 2, column 2
- $ne$  - Number of energy bins (usually of order 1000), line 2, column 3

### 2.4 Input Parameters for the Acceleration Site.

The input parameters specific to the acceleration process can be specified in *paramfile1*. Their definitions are listed below:

- Turbulence type - You may input one of the following options here, line 1:
  1. *real* - Parameterized by power law functions given in equations 3,4, and 5.
  2. *simple*
  3. *whis* - Whistler waves ( $s = q'$ )

4. hard - Hard sphere ( $s = 0, q' = 2$ )
- Turbulence time dependence - specifies if you want steady state, time integrated or time resolved; line 2:
    1. steady - steady state solution
    2. fluence - time integrated
    3. timeloop - time resolved
  - Turbulence parameters - these are the general turbulence parameters from equations 3-5; line 3:
    1. dd - Diffusion coefficient,  $\mathcal{D}$  in equation 3; line 5, column 1. It is recommended to set this parameter to unity and scale the others accordingly to match the physical parameters at hand.
    2. aa - Acceleration coefficient,  $\mathcal{A} = (q' + 2)\mathcal{D}$  in equation 4; line 3, column 2.
    3. tt - Escape time parameter,  $\mathcal{T}_{esc}$  in equation 5; line 3, column 3. (If dd is set to 1 then  $tt = \mathcal{T}_{esc}\mathcal{D}$ .)
    4. qd - Power law index  $q'$  for acceleration and diffusion in equations 3 and 4; line 3, column 4
    5. qt - Power law index  $s$  for energy dependent escape in equation 5; line 3, column 5
  - Plasma Characteristics; line 4:
    1. Size -  $L$ , the size of region in  $cm$ ; line 4, column 1, (if dd is set to 1 then  $size = L\mathcal{D}$ ).
    2. Density -  $n$ , the density of particles in  $cm^{-3}$ ; line 4, column 2 (for  $dd = 1$ , then  $density = n/\mathcal{D}$ ).
    3. Bfield -  $B$ , the magnetic field in Gauss; line 4, column 3 (for  $dd = 1$ , then  $Bfield = B/\sqrt{\mathcal{D}}$ ).
  - Source Particle Spectrum term  $Q(E)$  - specifies the spectrum of the background plasma or injected particles; lines 5 and 6:
    1. Injection type - input is either *gauss*, a gaussian spectrum, or *thermal*, a thermal spectrum; line 5. Note that you can go into the program and change the injection spectrum at the end of Particle.cc, if you'd rather inject a function not listed here.

2.  $kT$  - either the temperature of the Maxwellian or the mean energy of the Gaussian distribution in MeV; line 6, column 1.
3.  $N_{tot}$  - normalization of the injection spectrum; line 6, column 2.
4. width - the  $\sigma$  of the Gaussian distribution - set to 0 if not injecting a Gaussian; line 6, column 3.

## 2.5 Output parameters

The electron spectrum is written to the file dataElspec, which is formatted as follows:

Kinetic energy,  $E$  (MeV); corresponding velocity,  $\beta$ , in units of the speed of light; Electron flux spectrum in the acceleration site,  $\beta cf(E)$ ; flux of the electrons which escape the acceleration region,  $Lf(E)/T_{esc}(E)$ .

## 2.6 Examples - Implementing the program:

The program was originally intended to investigate particle acceleration in solar flares. The following parameters are typical for solar flares (see, e.g. Park et al. 1997, Petrosian & Donaghy 1999). The paramfiles 0 and 1 were set as shown in Table 1:

**Table 1: Example of Paramfile Settings**

- **paramile0:**

Parameter Values	Description
100	What to calculate
0.001 1000 200	Emin, Emax, # energy bins

- **paramile1:**

Parameter Values	Description
real	Turbulence Type (real, simple, whis, hard)
steady	Steady or Timedepend (steady, fluence, timeloop)
0.05 0.05 2. 1.7 1.0	Turbulence Parameters
1e9 5e10 300	Size, Density and B field
thermal	Acceleration Plasma Type (none, thermal, gauss)
0.003 1.0 0.1	Acceleration Plasma Params (kT, Ntot, width)

Paramfile2 is irrelevant here because we are only interested in the acceleration process. Figure 1 shows the output in dataElspec (and Table 2 shows a sample portion of the output file). Spectral breaks are evident at

low energies ( $\sim 0.1$  MeV), when the Coulomb loss rate is comparable to the acceleration rate, and at  $\sim 100$  MeV, above which synchrotron losses dominate.

**Table 2: Output file dataElspec for Example 1**

line 1: $dd$	$tt$	$q$	$s$ :		
0.05	2	1.7	1		
line 2: $L$	$n$	$N$	$B$	Convergence:	
1e+09	1e+11	2.284e+20	500	1	
line 3: $n$ : j=5	j=15	j=23	j=57	j=100	j=175:
6e+08	1.8e+09	2.412e+09	2.548e+09	2.72e+09	3.02e+09
line 4: $N$ : j=5	j=15	j=23	j=57	j=100	j=175:
6e+19	1.8e+20	2.388e+20	3.78e+21	1.688e+22	4.688e+22
line 5+: Energy	$\beta$	$f^*c\beta$	Fesc	ThickTar	flux(j):
0.001	0.04469	4.736e+13	1.397e+13	0	0 0 0 0 0
0.001072	0.04626	1.94e+10	5.568e+09	0	0 0 0 0 0
0.001148	0.04788	1.142e+09	3.189e+08	0	0 0 0 0 0
...	...	...	...	...	...

In Table 2,  $N$  denotes the column depth, and different values of  $j$  correspond to different depths along the loop (see §4 below). All other variables are as described in the preceding text.

However, because of the versatility of this program, it can be applied to many other astrophysical situations. For example, we may want to set the parameters relevant for gamma-ray bursts to investigate stochastic acceleration in this context. Here we are injecting a very narrow gaussian into a region with a large magnetic field and low density. Note that the Bfield, density, and diffusion coefficients are *scaled* from their actual values (see description of scalings in §2.4). Table 3 lists the input parameters and Figure 2 shows the output for this set of parameter values.

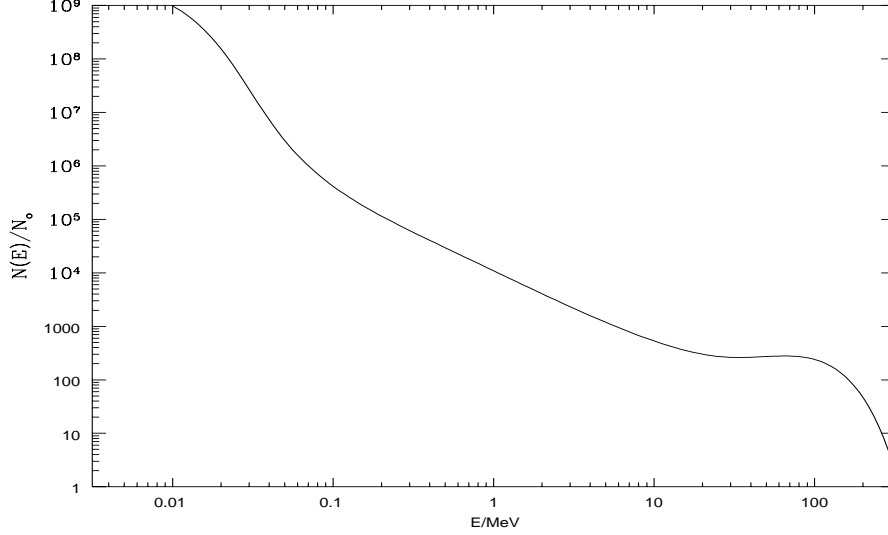


Figure 1: Electron spectrum for example parameters relevant for Solar flares

**Table 3: Paramfile 1 for Gamma-Ray Burst Example**

Parameter Values	Description
real	Turbulence Type (real, simple, whis, hard)
steady	Steady or Timedepend (steady, fluence, timeloop)
200.0 200.0 .001 1.2 0.	Turbulence Parameters
1.e9 1. 100.	Size, Density and B field
gauss	Acceleration Plasma Type (none, thermal, gauss)
1.0E10 100.0 0.1	Acceleration Plasma Params (kT, Ntot, width)

### 3 Transport

#### 3.1 Physical Motivation and Theory

The transport code solves an equation similar to equation 1 and assumes a steady state situation ( $\partial f / \partial t = 0$ ) which is valid for time scales greater than the energy loss time ( $\tau_{loss} = E/E_L$ ) and the transport time  $T_{tr} = L/(c\beta)$ , over the size  $L$  of the region; for a flare loop,  $L$  is essentially the length of the loop from the top to the transition region. For this the coefficients

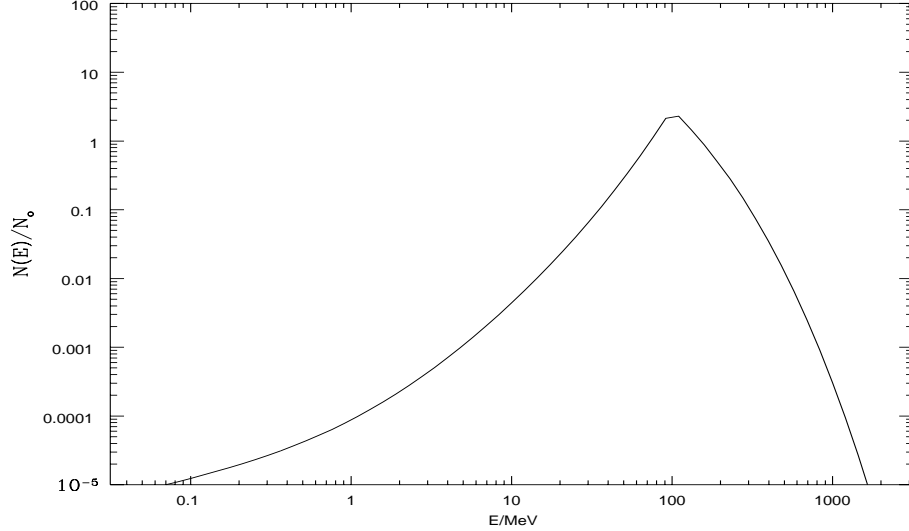


Figure 2: Electron spectrum for example parameters relevant for GRBs.

depend on pitch angle  $\mu$ , density  $n$ , and magnetic field  $B$ :

$$\dot{E}_{\text{Coul}} = -4\pi r_0^2 cn \ln \Lambda / \beta, \quad (6)$$

$$\dot{E}_{\text{synch}} = -2r_0^2 B^2 \beta^2 \gamma^2 (1 - \mu^2) / 3m_e c, \quad (7)$$

$$\dot{\mu}_{\text{synch}} = 2r_0^2 B^2 \mu (1 - \mu^2) / 3\gamma m_e c, \quad (8)$$

$$D_{\mu\mu, \text{Coul}} = -4\pi r_0^2 cn \ln \Lambda (1 - \mu^2) / (\gamma^2 \beta^3), \quad (9)$$

with other coefficients, including  $D_{EE}$  are set to zero.

The transport equation includes an additional term equal to  $(\frac{d \ln B}{ds}) \frac{\partial}{\partial \mu} ((1 - \mu^2) f)$  which accounts for systematic field variations when - like in a solar flare loop - one is not dealing with somewhat regular fields. Note that, in general,  $\frac{d \ln B}{ds}$  is not a constant. The code has the option of setting it to a constant or evaluating its variation based on an assumed model, e.g. quadratic in  $s$ . In these cases, instead of specifying  $\frac{d \ln B}{ds}$ , the ratio of the  $B$  field at the point of injection ( $s = 0$ ) to that at a fixed column depth  $N_o = \int_0^{s_o} n dS$  is used. For a solar flare,  $N_o = N_{tr}$ ,  $s_o = s_{tr}$ , where  $s_{tr}$  is the distance along the loop from the point of injection to the transition region.

The source term  $S(E, \mu, s) = F_{esc}(E) \delta(s) \Theta(\mu)$  assumes a steady state injection of electrons at  $s = 0$  (a point at the boundary of the acceleration site), isotropically away from this site (in the forward  $\mu > 0$  direction) and

with an energy spectrum equal to that of the electrons escaping from the acceleration site  $F_{esc} = (f_{ac}(E)/T_{esc}(E))L$ . For a solar flare loop whose particles can stream down both sides of the loop, we solve the equation for one leg of the loop. Electrons, which either because of Coulomb collisions or field convergence, return back to the acceleration site (with  $\mu < 0$ ) are reflected back into the transport region at  $s = 0$  with equal but positive  $\mu$ . The transport code, in reality solves the equation for  $s > 0$  where the source term  $S = 0$ , and uses  $F_{esc}(E)$  as the boundary condition for the flux  $F(E, s, \mu) = c\beta f(E, s, \mu)$  at  $s = 0$ . The flux here refers to the flux of along a bundle of loop (i.e. integrated over the cross section of the loop) so that  $f$  has units of  $cm^{-1}$  and  $F$  units of  $s^{-1}$ .

### 3.2 The Routines

Class Photon – The code for both the transport and radiation calculations is contained in the Photon class which deals with all aspects of taking the output of the acceleration code, setting up the loop arrays and performing the Fokker-Planck calculation. Table A.7 described the principal data structures and routines in the class. For further details see Leach & Petrosian (1981, 1983), Leach (1984), Mctiernan Hamilton, Lu & Petrosian (1990).

### 3.3 How to Run this Portion of the Program

This program requires setting the parameter values in all of the paramfiles (0,1, and 2) with 110 (or 111 if the photon spectrum is desired) for the first row in paramfile0. It also will use the energy range and bin numbers from this file, as well as the output  $F_{esc}(E)\Theta(\mu)$  from the acceleration portion for the initial spectrum and pitch angle dependence at  $s = 0$ . It is possible to run this portion of the program by itself for the types of injected spectra that can be specified in paramfile1. One then must choose an energy independent (i.e.  $s = 0, 1/2$  for relativistic and non-relativistic particles, respectively) and relatively low value for the escape time ( $tt \ll 1/dd$  and  $\tau_{loss} = E/E_{loss}$ ), relative to the other timescales (i.e. the acceleration and or loss times). This way the injected spectrum emerges unaffected from the acceleration site and becomes the input in the transport code.

### 3.4 Input Parameters

In addition to the parameters discussed above, this portion of the code requires a set of parameters which describes the variation of the plasma density, magnetic field, and ionization structure along the loop. The density

variation enters only in the details of the distribution of the emission along the loop. This distribution has a simple form if expressed in terms of the column depth  $N(s) = \int_0^s n ds$ , but could be more complex, e.g. in a solar flare when particles penetrate below the chromosphere when  $n$  increases strongly. The present code assumes a stratified solar atmosphere. The ionization structure affects only the Coulomb logarithm and is not a major effect. The field geometry also can be chosen to be converging, diverging, semicircular, or straight. There are parameters that allow one to turn on and off the effects of collisional losses, synchrotron losses, and field convergence.

### 3.5 Output Parameters

The output of this portion of the code is the electron spectrum and pitch angle (cosine) distribution at different heights,  $f(E, \mu, s)$ , and the total spectrum  $f_{tot}(E, \mu) = \int_0^\infty f(E, \mu, s) ds$ , that one needs to evaluate the thick target (or total) emission. This is sometimes referred to as the cooling spectrum. The current code gives the pitch angle integrated spectrum:  $\int_{-1}^1 f(E, \mu, s) d\mu = f(E, s)$ , so that  $f_{tot}(E) = \int f(E, s) s = \frac{1}{\beta \dot{E}_L} \int_E^\infty F_{esc}(E) dE$ .

### 3.6 Examples

A portion of the output file `dataElspec` is shown in Table 2. The first line gives the values of the parameters  $dd$ ,  $tt$ ,  $q'$  and  $s$ ; the second line lists the values of the size  $L$  in *cm*, density  $n$  in  $cm^{-3}$ , the column depth to the transition region, the  $B$  field at  $s = 0$ , and the convergence rate (i.e. the ratio of the  $B$  field at  $s = 0$  to that in the transition region); the third and fourth lines give the values of distance  $s$  or the column depth  $N(s)$  along the loop for which the electron spectrum is given (the last six columns give the specified depth bin numbers  $j$ ); the next line describes the column density, energy in  $mc^2$  units,  $\beta = (1 - (E + 1)^2)^{-1/2}$ , the electron flux  $F = f(E) \times c\beta$  in the acceleration site, the flux  $F_{esc} = L f(E) / T_{esc}$  of escaping electrons, the equivalent thick target flux  $\int F_{esc} / \dot{E}_L dE$  and the electron flux at the specified depths.

Table 4 shows the output from `dataEldepth` - the same data but now in finer detail as a function of depth at a few representative energies (or velocities) specified in lines 3 and 4. The rest of the structure of this table is the same as in Table 2. Figure 3 shows examples of electron spectrum at different depths and energies for a solar flare case.

**Table 4: Output file dataEldepth**

line 1: $dd$	$tt$	$q$	$s$ :			
0.05	2	1.7	1			
line 2: $L$	$n$	N	$B$	Convergence:		
1e+09	1e+11	2.284e+20	500	1		
line 3: $E$ : j=49	j=59	j=69	j=79	j=92	j=110	j=143:
0.02951	0.05888	0.1175	0.2344	0.5754	1.995	19.5
line 4: $\beta$ : j=49	j=59	j=69	j=79	j=92	j=110	j=143:
0.2377	0.3288	0.4463	0.5863	0.7727	0.9426	0.9988
line 5+: $y$ [cm]	coldp	El flux(j):				
0	0	8.e-09, 3.e-10,	6.e-11, 2.e-11,	5.e-12, 9.e-13,	7.e-14	
0	0	2.e-11, 7.e-13,	9.e-14, 1.e-14,	3.e-15, 2.e-16,	1.e-18	
...	...	...				

## 4 Bremsstrahlung Radiation

Given the spectrum and pitch angle distribution of the electrons, the third part of the code evaluates the bremsstrahlung photon spectrum emitted in a direction  $\theta$  with respect to the local magnetic field using equations given in Koch & Motz (1959). At each point along the loop, the exact angular distribution at a given detector, however, depends on the geometry of the magnetic field lines. Thus, a possibly more useful spectrum is the average (over  $\theta$ ) spectrum which is

$$J(k, \tau) = \int dE F(E, \tau) n \frac{d\sigma}{dk}(E, k), \quad (10)$$

where  $\tau$  is a measure of depth,  $d\tau = 4\pi r_0^2 \ln \Lambda n(s) ds$  and  $F(E, \tau)$  is the number of electrons per energy and depth. The spectrum at the acceleration site is obtained with  $F(E, \tau) = f(E) \times c\beta$  and thick target spectrum with  $F(E, \tau)$  set equal to  $F_{thick}(E) = \int_0^\infty F(E, \tau) d\tau$ .

The only parameter setting for this portion of the program is the one required for its running, which means that column 3 of line 1 in paramfile0 should be set to 1.

The output files are essentially identical to those for electrons from the transport part; instead of the electron spectrum, the output here gives the photon (energy not number) spectrum,  $EF(E, \tau)$  at the acceleration site ( $\tau = 0$ ), at different depths, and the overall thick target spectrum. Figure 4 shows some examples of the photon spectrum at different depths and energies for a solar flare case.

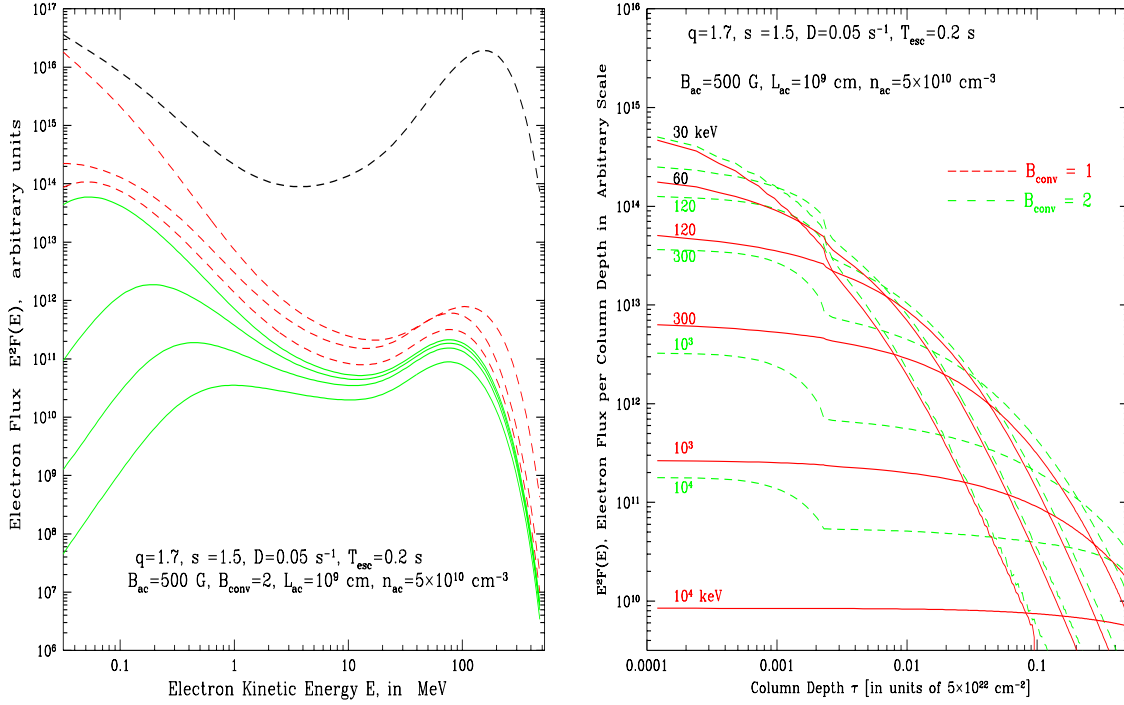


Figure 3: **Left Panel:** Electron spectral flux (multiplied by  $E^2$  for convenience) for the specified model parameters and several depths.  $B_{conv}$  is the ratio of the magnetic field at the transition region to that at the loop top. The heavy dashed line is the spectrum at the acceleration site and the three dashed lines from top to bottom are spectra just outside this region, and at column depths equal to 1/3 and 2/3 of the column depth to the transition region, which is at the depth  $N_{tr} = 1.14 \times 10^{20} \text{ cm}^{-2}$ , or  $\tau_{tr} = 0.0023$ . The four solid lines are for regions below the transition regions at depths of  $\tau = 0.0024, 0.038, 0.17$  and  $0.47$ , respectively from top to bottom. —**Right Panel:** Same as left panel except now the flux is plotted as a function of depth at the specified values of energies.

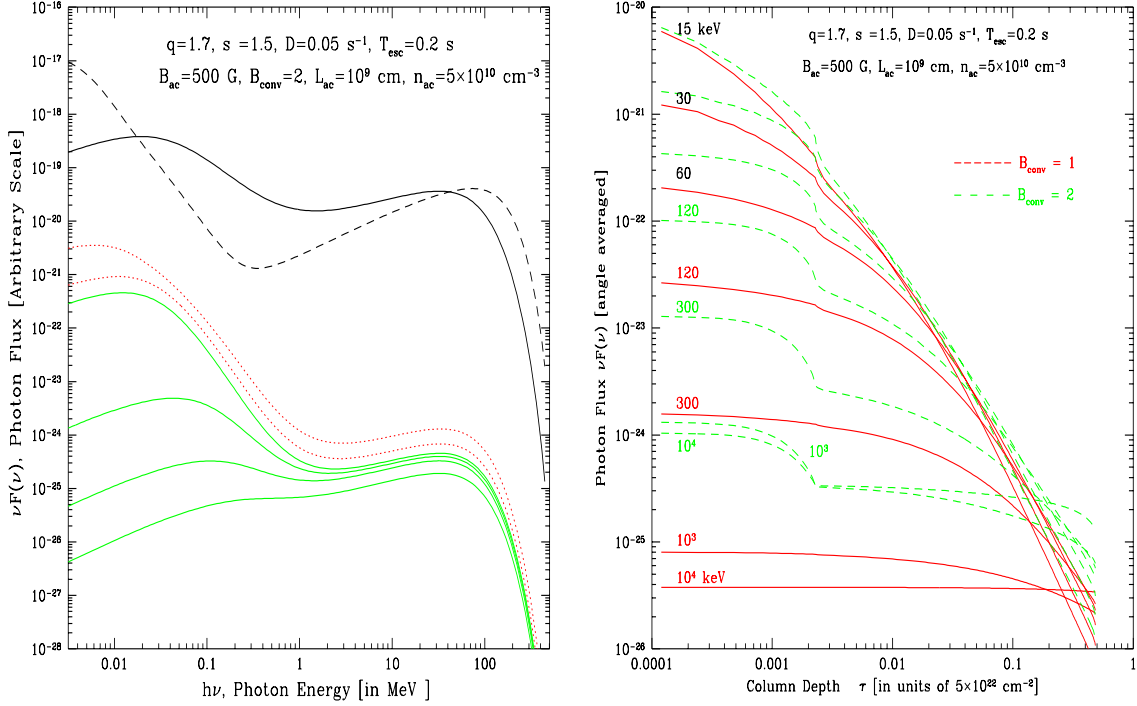


Figure 4: **Left Panel:** Same as left panel of Figure (3) but for the photon spectra at different locations. The heavy dashed line is the spectrum at the acceleration site or the loop top. The two dotted lines and the four thin solid lines are for depths  $\tau = 0.0006, 0.0018, 0.0024, 0.038, 0.17$  and  $0.47$ . The heavy solid line is the total emission from below the transition region or the footpoints. **Right Panel:** Same as the left panel except now the flux is plotted as a function of distance along the loop at the specified values of energies.

## A APPENDIX

In the following tables we tabulate the functions, routines and parameters introduced above in greater detail.

Table 1: Variables specified in paramfile0, 1, and 2 and their descriptions.

Variable	Description
outCode	3-bit string describing what to calculate and output <b>1st bit</b> —calculate loop top electron spectra <b>2nd bit</b> —calculate transport electron spectra <b>3rd bit</b> —calculate all photon spectra
emin	Minimum Energy (in units of $m_e c^2$ )
emax	Maximum Energy (in units of $m_e c^2$ )
ne	Number of logarithmically spaced energy bins
coefMode	Turbulence Model Type <b>simple</b> — <b>hard</b> —scattering off hard-spheres <b>whis</b> —scattering off whistler waves <b>real</b> —general coefficient model
fluenceMode	Time-dependence Model Type <b>steady</b> —steady-state solution <b>fluence</b> —calculate particle fluence <b>timeloop</b> —calculate time-evolution
dd	Fokker-Planck Diffusion Coefficient
aa	Fokker-Planck Systematic Acceleration Coefficient
tt	Fokker-Planck Escape-Time Coefficient
qd	Acceleration index
qt	Escape-Time index
length	Size of the Acceleration Region
density	Loop top number density of electrons
bfield	Loop top magnetic field strength
injectMode	Acceleration Plasma Type <b>thermal</b> —Thermal distribution <b>gauss</b> —Gaussian distribution
einj	Injected energy scale (in units of $kT$ )
ninj	Number of injected particles (normalization)
sigma	width scale (for Gaussian only)
khs	0—no mag conv, 1—vertical field, 2—semicircular field
kab	0— $d \ln B / ds = const$ , 1—powerlaw B, 2—quadratic B
kbc	Boundary Conditions, 0—zero, 1—reflecting
b2	Bfield convergence, $B_{tr} / B_{top}$
b3	Number of depth bins to the transition region
kb	1,2—add Coulomb energy losses
kb1	1,2—add diffusion losses
ksy	1,2—add synchrotron losses
ksmu	1,2—add synchrotron p.a. diffusion

Table 2: Classes and source code for flare.

Class	Header	Source
sacparticle	–	sacparticle.cc
Fokker	Fokker.h	Fokker.cc, FokkerChangcoop.cc
Outfile	Outfile.h	Outfile.cc
Particle	Particle.h	Particle.cc, ParticleDist.cc
Photon	Photon.h	Photon.cc, ElectronFlux.cc, Radiation.cc, Functions.cc
–	cross.h	cross.c
–	toolbox.h	toolbox.c

Table 3: *Class sacparticle* – local variables and routines.

Class sacparticle		
Variable	Type	Description
ne	private int	number of energy grid-points
emin	private double	min energy
emax	private double	max energy
outCode	private int	binary code describing desired output products
pt	public Particle	Particle class, calculates LT acceleration
ph	public Photon	Photon class, calculates transport+radiation
Routine	Type	Description
SacParticle()	public	Class constructor
run()	public	main routine
readParams()	public	reads parameters from paramfiles
displayParams()	public	prints class parameters
output()	public	prints output data to data files

Table 4: *Class Particle* – local variables and routines.

Class Particle		
Variable	Type	Description
bfield		
collisions		
turbulence, betaAlfven	private float	turbulence ratio, Alfven beta
omegae	private float	electron cyclotron frequency
alphape	private float	ratio of omegap/omegae = 1/betaAlfven
gammaMax, kmin	private float	maximum electron energy, minimum resonant wave
aaa, ddd, ttt	public float	advection rate, diffusion rate, escape time
qd, qt	public float	diffusion index, escape time index
density, bfield	public float	loop top number density, magnetic field
length	public float	length of acceleration region
coulombEscape	public int	1= add escape time due to coulomb collisions
injtype	public int	integer flag describing particle injection type
coefMode	public int	integer flag describing coefficient type
fluenceMode	public int	integer flag describing fluence type
einj, ninj, sigma	public float	injection energy, num particles, width
Routine	Type	Description
Particle()	public	Class constructors
initParticle()	private	initializes class parameters
calcParticle()	public	do the calculation
calcSteady()	private	calculates steady-state solution
calcFluence()	private	calculates the particle fluence
timeLoop()	public	option to calculate at many times
setInject()	private	sets injection type (thermal, gaussian)
setCoef()	private	sets coefficient type (real, whis, hard, simp)
realCoef()	private	inits real Coeff model
whisCoef()	private	inits whistler Coeff model
simpleCoef()	private	inits Simple Coeff model
hardsphereCoef()	private	inits hardsphere model
addCoulomb()	private	adds coulomb collision terms
addSync()	private	adds synchrotron terms
eBreakCoulomb()	private	returns Coulomb collision break energy
eBreakSync()	private	returns synchrotron break energy
displayParams()	public	displays class parameters
updateParams()	public	updates class parameters

Table 5: *Class Outfile* – local variables and routines.

Class Outfile		
Variable	Type	Description
ne, ntot	public int, float	number of energy elements, total particles
ntrep	public int	number of time distributions
itrep, trep	public int	index of time report, report time
iter	public int	finite difference iteration count
type	public int	output: 1=phi, 2=flux, 3=both
e	public float	energy grid
phi	public float	particle distribution
ephi	public float	escaping particle distribution
phiflux	public float	flux distribution
phitflux	public float	"true" flux (use for debugging)
Routine	Type	Description
Outfile()	public	Class constructor
initVectors()	public	Allocate space for energy vectors
readDist()	public	get ne distribution from the specified unit number
writeHeader()	public	Write out the header information
writeDist()	public	Write current distribution(s)
writeAll()	public	Write both header and the distribution
displayParams()	public	displays class parameters

Table 6: *Class Fokker* – local variables and routines.

Class Fokker		
Variable	Type	Description
ae, de, qe	protected float	coefficients of equation
aa, bb, cc, rr, u8	protected float	tridiagonal matrix
ep, nep	protected float	energy grid, number of energy points
pe, betae, gammae	protected float	derived energy grids
Routine	Type	Description
Fokker()	protected	Class constructor
weighting()	private	get upwind or downwind weighting
changcoop()	private	calc the tridiagonal coeffs using ChangCooper
changcoopSteady()	private	steady state version of ChangCooper
fokkertq()	protected	Integrate Fokker-Planck by one time step
fokkerSteady()	protected	Solve the steady state problem
escapeFlux()	protected	Return the escaping flux
calcNtot()	protected	Calc total particles in specified distribution

Table 7: *Class Photon* – local variables and routines.

Class Photon		
Variable	Type	Description
jmax	private int	number of points in depth array
k0, k1	private int	last positive, first negative pitch angle
zone		
y, coldp	private double	depth & column depth arrays
x, z	private double	pitchangle array, energy array
ftop	private double	loop top distribution
flux	private double	final particle distribution
dn, noc, bf	private double	density profiles, Bfield profile
rad	private double	radiation from sides of loop
LTph, Fesc	public double	loop top photon distribution, escaping electrons
nc, nh	public int	number of atm constituents, profiles
kmax, imax	public int	number of pitchangle bins, energy bins
kb, kb1	public int	Coulomb energy loss, diffusion flags
khs, kab, kbc	public int	field curvature, convergence, boundary conditions
ksy, ksmu	public int	synchrotron losses, p.a. diffusion
adp, adpm	public double	numerical convergence tolerances
b2, b3	public double	bfield convergence ratio, depth to transition zone
za, zn, ei	public double	atomic number, weight, ion energy arrays
Routines	Type	Description
Photon()	public	Class constructor
calcTransport()	public	calc the transported electron flux
calcRadiation()	public	calc thin target accel region, each depth bin
calcDists()	private	calc loop arrays from paramfile info
InitY()	private	sets y[] given ydy[]
Findk0k1()	private	sets k0, k1 given x[] and kmax
InitAtmosphere()	private	initialize atmosphere arrays
InitBField()	private	initialize Bfield structure
electronFlux()	private	main routine for solving transport problem
FormXList()	private	initializes x-dependent arrays
FormBCterms()	private	calc magnetic field and coulomb terms
factor()	private	factor tridiagonal array
ringer()	private	find the solution
trim, trim1, trim2	private	trim bad solution
DoThin()	private	calc thin target bremsstrahlung
Beta(), Gamma()	private	calc beta, gamma given energy
tesc()	private	calc escape time
displayParams()	public	display class parameters
printElspec()	public	print electron spectra at different depths
printPhspec()	public	print photon spectra at different depths
printElheight()	public	print electron dist as a function of depth
printPhheight()	public	print photon dist as a function of depth