Time-Dependent Fokker-Planck Code Documentation

Jim McTiernan
Space Science Laboratory
University of California, Berkeley

Leon Ofman
Laboratory for Astronomy & Solar Physics
NASA/Goddard Space Flight Center
and Catholic University

Last Revised September 6, 2001

Table of Contents

General Description

FKRPLK is a code that calculates the electron velocity distribution function by solving the Fokker-Planck equation in three-dimensions (energy, pitch angle cosine, position along the field) and time.

Reference: Hamilton, Lu and Petrosian, ApJ 354, 726 (1990).

The code consists of the following files, each containing a separate subroutine:

main.f
init.f
inject.f
bfield.f
dands.f
cntridag.f
tauup.f
bcup.f
eup.f
muup.f

The include file fkrplk.h contains the common blocks of the arrays and the maximal dimensions nemax, nmumax, and ntaumax.

The Makefile uses the f90 compiler to produce the executable fkrplk.x

The code is run by executing fkrplk.x

The parameters for the run are set in main.f and init.f.

The output electron distribution function is written to fkrplk.test, and plotted using the provided IDL routine tdfp_plot.pro

Below are descriptions of the subroutines.

Description of main.f and input parameters

The parameters for the case to be solved are initialized in this subroutine. The array tr, initialized in the data statement, contains the time steps to print the output.
The number of times to report (ntr) and the maximum time of the simulation (tmax) are initialized here:

```
ntr = 8
tmax = tr(ntr)
```

Number of spatial grid points (ntauwr) and energy grid
points (newr) to write to output file.

```
ntauwr = 1
newr = 60
```

Logical variables:

symloop defines the loop symmety used in tauup.
If symmetric, we use phi(k,l,-m)=phi(k,-l,m) for m > 0.
If not symmetric, then the we use phi = 0. for m < 0

```
symloop = .true.
```

Impulsive or non-impuslive injection of the electron distribution function

```
impulsive = .false.
```

Output file: The output is written to fkrplk.test

```
open(unit=10,file='fkrplk.test')
```

Intialize the arrays by calling init

```
call init(rhomax,soft,tau)
```

Convert dimensionless time "y" to real time "t".
Assuming that ln(Lambda) = 20 and fully ionized hydrogen.

```
ytot = 1.6666667e12/rhomax
```

Description of init.f and additional input parameters

This is the initialization subroutine. The values for emin, emax,
ne, amu, nmu, taumin, taumax, and ntau are either read in or
assume the default values set here in the data statement.
This subroutine also sets dy, the time step variable. The arrays
de, dtau, beta, beta2, rb3g2, rlmbda, rlmbda2, dmu, onmu2, and
bconv are filled. The following external subroutines are needed:
dands(tau,rho,s)-- returns the electron number density rho and

spatial postion s at a given column depth tau.
bfield(s,b,dbds)-- returns the value of dln(B)/ds == dbds and b
at postion s.

Energy steps increase linearly from e(0)/4 up to emax so that
ne steps are taken.
Added calculation tau as an output parameter

The values for grid parameters are set in the following data statement:

```
      data emin,emax,ned,nmud,taumin,taumax,ntaud
     +     / 10.,1000.,60,30,1.e-5,2.e-3,30 /
```

Where emin, emax is the energy range, ned is the number of energy grid point, nmud is the number of grid point in mu, the range in tau is given by taumin and taumax, and ntaud is the number of grid points in tau. The data statement can be replaced by an appropriate read statement to read the parameters.

Description of inject.f

This subroutine injects an electron distribution which is initially a
gaussian in space, but becomes a delta function in space. It is isotropic in
pitch-angle, power-law in energy, and a triangular-shaped pulse, 2 seconds long.

Description of bfield.f

This subroutine returns the logarithmic derivative of the magnetic field
dbds = dlnB/ds and magnetic field strength b at a given spatial position s.
The magnetic field used here has the following properties:

$B = B_o(1 + s^2/L_B^2)$.

L_B is determined by the length of the corona, L_C, and the mirror ratio, r_m.

$L_B = L_C / \mathrm{sqrt}(r_m - 1)$.

xc = length of corona
rm = mirror ratio

Description of dands.f

This subroutine returns the electron number density (rho)
and spatial coordinate (s) given the "column depth" (t).

The density function used here has the following properties:
the density is a constant (rho0).

The values of the density, xkmbda (Coulomb log) and a are defined in the parameter statement

```
parameter(xlmbda=20.0, a=1.0e24, rho0=1.0e11)
```

The subroutine converts "column depth" (t) to distance s, then computes
the density.

Description of cntridag.f

This subroutine solves the tridiagonal matrix equation which results from the Crank-Nicholson (implicit) method used in subroutine muup for phi(k,l,m,out) for given k, m, and out at all l.

The implicit solver was adapted from Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T. 1986, "Numerical Recipes", (New York: Cambridge University Press), p. 40.

Description of tauup.f

In this subroutine phi(k,l,m,in) is updated to phi(k,l,m,out) for the tau term using monotonic transport. Boundary conditions for positive velocity at m=0 are either symetric or no flux comes in from m < 0; for negative velocity the boundary condition at m=ntau is that no flux comes in.
Note: Nonuniform step sizes in tau are not allowed.

Description of bcup.f

In this subroutine phi(k,l,m,in) is updated to phi(k,l,m,out) for the converging magnetic field term by Barton's monotonic transport method.

The points abs(mu) = 1 can be solved for exactly and are in this subroutine.

Description of eup.f

In this subroutine phi(k,l,m,in) is updated to phi(k,l,m,out) for the energy term by Barton's method of monotonic transport.

Description of muup.f

In this subroutine phi(k,l,m,in) is updated to phi(k,l,m,out) for the pitch angle diffusion term. The Crank-Nicholson's implicit method is employed and the subroutine cntridag is called to solve the tridiagonal matrix.

Cray J90 routines

In order to improve the execution speed of the code the routines below were rewritten to allow multitasking on the Cray J90. To run the code on the Cray J90 please use the following Makefile:

Makefile.J90

and the subroutines ending with "_par".

A sample NQS batch file for the Cray J90:

```
#QSUB -s /bin/csh
#QSUB -eo
#QSUB -o fkrplk.log
setenv NCPUS 20
cd /tmp/fpcode
set echo
set timestamp
ja ja.out
./fkrplk.x
ja -cst ja.out
```

Contacts & Acknowledgements

This program was originally developed by Russell Hamilton, Edward Lu, and Vahé Petrosian (ApJ 354, 726, 1990). Leon Ofman developed the Cray J90 version of the code.

Address questions and comments about the code to

James M. McTiernan
University of California, Berkeley
Space Science Laboratory #7450
Berkeley, CA 94720-7450

Office phone: (510) 643-9246
E-Mail: jimm@ssl.berkeley.edu

or to

Leon Ofman
NASA/Goddard Space Flight Center
Code 682
Greenbelt, MD 20771

Office phone: (301) 286-9913
E-Mail: leon.ofman@gsfc.nasa.gov