THE APPLICATION OF THE FINITE ELEMENT LIBRARY TWODEP FOR THE NUMERICAL SOLUTION OF THE RELATIVISTIC FOKKER-PLANCK EQUATION

by

M. Shoucri*, V. Fuchs*, A. Bers

June 1987

Plasma Fusion Center
Massachusetts Institute of Technology
Cambridge, Mass. 02139 U.S.A.

*IREQ, Project Tokamak, C.P. 1000, Varennes, Quebec, Canada
JOL 2PO

Submitted for publication in: Computer Physics Communications
THE APPLICATION OF THE FINITE ELEMENT LIBRARY
TWODEPEP FOR THE NUMERICAL SOLUTION OF
THE RELATIVISTIC FOKKER–PLANCK EQUATION

M. SHOUCRI, V. FUCHS, A. BERS*
IREQ, Projet Tokamak
C.P. 1000, Varennes, Québec, Canada JOL 2P0

Permanent address:
* Plasma Fusion Center, Massachusetts Institute of Technology
Cambridge, Mass. 02139, U.S.A.

Abstract

We report on the application of a commercially available differential equation solver library called TWODEPEP for the numerical solution of the relativistic Fokker–Planck equation. Results obtained by solving the runaway problem, and on the solution of the R.F. heating problem are presented.
The interaction of radio-frequency waves with a plasma is described by a Fokker-Planck equation with an added quasilinear diffusion term, and the response of a plasma to a homogeneous ohmic electric field is described by the same Fokker-Planck equation with an advection term. There has been an extensive literature on numerical techniques to solve this equation which is of high importance in the study of lower-hybrid RF-driven currents in Tokamaks (see, for instance, the review article in Ref. [1]). In this note, we report on the application of a commercially available differential equation solver library called TWODEPEP [2] for the numerical solution of the Fokker-Planck equation.

This differential equation solver has been successfully applied in solving a wide variety of problems of fluid dynamics, diffusion, heat transfer, etc. We have previously reported results obtained using this library for the numerical solution to the non-relativistic Fokker-Planck equation [3, 5]. In this note we report on the application of this differential equation solver for the numerical solution of the Fortran code used to solve the relativistic Fokker-Planck equation. It illustrates how convenient is the library for the problem we are studying, if we notice that the preprocessor Fortran code has no more than a few dozen lines. We note, however, that a physical quantities of interest plus graphic output, are obtained from a separate code, which uses TWODEPEP results as input.

In the model Fokker-Planck equation we consider the plasma is assumed homogeneous, and azimuthally symmetric about the magnetic field. Relativistic effects are included. The model is described in sufficient detail in
Refs. [1], [6], its characteristic feature being a linear collision oper-
tor describing the interaction of fast test electrons with fixed Maxwellian
field electrons and ions. The electron distribution function \( f \) depends on
time and two momentum variables. In flux form the Fokker-Planck equation
is

\[
\frac{\partial f}{\partial t} + \mathbf{v} \cdot \mathbf{S} = 0
\]  

(1)

where \( f \) is the test-electron distribution function and \( \mathbf{S} \) the test electron
flux in the presence of an electric field \( \mathbf{E} \) and a quasi-linear diffusion coefficient \( D \):

\[
\mathbf{S} = \mathbf{E} f - D \cdot \text{grad} f + S_{\text{coll}}
\]  

(2)

We use the normalized variables

\[
t + t v \quad v + v/v_t \quad P = P/P_t
\]  

(3)

where the thermal velocity unit \( v_t \) is defined by the following relations
[6]:

\[
v_{t}^{2} = \frac{1}{3n_e} \int v^{2} f_{m}(P) d^{3}P = \frac{T_e}{m_e} \left( 1 - \frac{5}{2} \Theta + \frac{55}{8} \Theta^{2} \right)
\]  

(4)
where \( f_m(P) \) denotes the relativistic Maxwellian distribution and \( \Theta = T_e/(m_e c^2) \). We will restrict ourselves to the case where \( \Theta \ll 1 \) (for \( T_e = 5 \, \text{kV} \) we have \( \Theta = 0.01 \)) so that the thermal velocity \( v_t = (T_e/m_e)^{1/2} \). The thermal momentum is given by

\[
P_t = m v_t
\]  

and the thermal collision frequency is given by

\[
\nu = 4\pi n_e e^4 \ln A/m^2 v_t^3
\]  

with

\[
v_t = (T_e/m_e)^{1/2}
\]  

The electric field is normalized to \( E_0 = P_t v/e \) and the diffusion coefficient is normalized to \( D_0 = \nu P_t^2 \). Also, for \( \Theta \ll 1 \), the relativistic Maxwellian distribution is given in our normalized units by:

\[
f_m(P) = \frac{1}{(2\pi)^{3/2}} e^{-P^2/(\gamma+1)}
\]  

where

\[
\gamma = (1 + P^2/P_{th}^2)^{1/2}
\]  

and
\[ \beta_{th} = \frac{v_t}{c} \]  

(10)

In the spherically symmetric coordinates \((v, \mu)\), where \(\mu = P/|P| = \cos\theta\), we have:

\[ \nabla \cdot S = \frac{1}{p^2} \frac{\partial}{\partial p} (p^2 S_p) - \frac{1}{p} \frac{\partial}{\partial \mu} (\sqrt{1-\mu^2} S_\mu) \]  

(11)

where

\[ S_p = \varepsilon \mu f - D \mu \left( \mu \frac{\partial f}{\partial p} + (1-\mu^2) \frac{\partial f}{\partial \mu} \right) - (A(P) \frac{\partial f}{\partial p} + F(P)) f \]  

(12)

\[ S_\mu = (1-\mu^2) \frac{1}{2} \left[ -\varepsilon f + D \left( \mu \frac{\partial f}{\partial p} + \frac{1-\mu^2}{p} \frac{\partial f}{\partial \mu} \right) + \frac{1}{p} \left( B(P) + \frac{Z}{2v} \right) \frac{\partial f}{\partial \mu} \right] \]  

(13)

Here \(Z\) is the ion charge and the coefficients \(A(P), B(P)\) and \(F(P)\) are given in our normalized units by:

\[ A(P) = \frac{4\pi}{3} \left( \frac{\gamma^3}{P^3} \right) \int_0^P P' \frac{f(P')}{\gamma^{1/2}} \, dP' + \int_0^\infty P' \gamma f_m(P') \, dP' \]  

(14)

\[ B(P) = \frac{4\pi}{3} \left( \frac{3}{2} \right) \int_0^P P' \frac{f_m(P')}{\gamma^{1/2}} \, dP' - \frac{\gamma^3}{2p^3} \int_0^P \frac{P' h f_m}{\gamma^{1/2}} (P') \, dP' + \int_0^\infty P' \gamma f_m(P') \, dP' \]  

(15)
\[ F(P) = \frac{4\pi}{3} \left( \int_0^P P' f_m(P') \left( 3v' - v'^3 \beta_{th}^2 \right) dP' \right) + \]

\[ \frac{2P}{Y} \int_0^\infty P' f_m(P') \beta_{th}^2 dP' \]  

(16)

The finite element code TWODEPEP [2] solves the 2-D partial differential equation

\[ \frac{\partial U}{\partial t} = \frac{\partial Q_x}{\partial x} + \frac{\partial Q_y}{\partial y} + R \]  

(17)

Comparing Eq. (17) with Eq. (11) we can identify \( P + x, \mu + y, U + f \) and

\[ Q_x = P^2 S_p \]  

(18)

\[ Q_y = -P \sqrt{(1-\mu^2)} S\mu \]  

(19)

The momentum-space domain in which the equation is solved is \( 0 < P \leq P_{\text{max}} \) and \( -1 \leq \mu \leq 1 \). The condition \( S\mu = 0 \) at \( \mu = \pm 1 \) follows from Eq. (13), and represents an axisymmetric situation. At \( P = P_{\text{max}} \) we always set \( f = 0 \), although one should impose a runaway flux condition \( S_P = \varepsilon_{\mu} f \).

With \( f = 0 \), however, the code will maintain required accuracy with considerably less computational effort in terms of required elements and memory. The \( f = 0 \) boundary condition at \( P = P_{\text{max}} \) will distort the solution only in a narrow boundary layer close to \( P_{\text{max}} \). To avoid the singularity at
P = 0, we impose the boundary condition f = f_m (f_m is defined in Eq. (8)) at P = P_{min} (P_{min} could be made very small, say 0.01). The imposition of f at P = P_{min} will slightly violate density conservation in the bulk when ε and/or D are non-zero, but this effect is very small since for the values of ε and D considered the number of particles transferred to the tail is orders of magnitude smaller than the number of particles in the bulk. Thus when ε ≠ 0, an imposed fixed boundary condition in the bulk represents an equilibrium situation in which electrons are being fed into the distribution at the same rate at which they run away. Hence we assume steady-state and solve for the equilibrium:

\[ \frac{\partial Q_x}{\partial x} + \frac{\partial Q_y}{\partial y} = 0 \] (20)

The functions Q_x and Q_y are defined in Eqs. (18) and (19) (see subroutines FXX and FXY in the Appendix, in which U + f, UX + \frac{\partial f}{\partial x} and UY + \frac{\partial f}{\partial y}). Equation (20) is a linear equation for f and TWODEPEP requires only one iteration to calculate the solution (parameter TF = 1 in the main, and NOUT = 1 denotes that the output should be given after one iteration). Also in the main NEQ = 1 indicates we are solving one equation, NTF = 9300 denotes the number of elements we are using and NDIM = 2 is a memory storage parameter. With these parameters the code would require a memory close to 5.5 million bytes on an IBM machine using double precision arithmetic. OXX and OXY are the functions Q_x and Q_y (defined in Eqs. (18) and (19)) which are calculated in the functions FXX and FXY respectively. D3EST is a function which distributes the elements in such a way as to have more elements in regions where more precision is required. UPRINT is the output solution
given by $U \equiv f$. The output solution is given on a grid defined by $x = XA + i \cdot H_x (i = 0, \ldots, NX)$ and $y = YA + j \cdot HY (j = 0, \ldots, NY)$. MWR = 8 is the file number on which the results are stored, and the INTEGRAL function integrates a function of the solution over the domain where the solution is calculated. In the Appendix presented, we are calculating the rf power absorbed defined by:

$$P_d = -2\pi \int_{-1}^{1} d\mu \int_{0}^{\infty} D \frac{\partial f}{\partial P} (P^3/\gamma) dP$$  \hspace{1cm} (21)

The remaining parameters in the main define the boundary conditions. For more details see Ref. [2]. Finally, the calculations of the integrals in Eqs. (14) - (16) has been simplified by noting that for small values of $P$, and for the small values of $H$ we are considering, the quantity $v = P/\gamma = \nu$ and the integrals in Eqs. (14) - (16) can be calculated from well-known error functions integrals [1], while for large $P$, we have the asymptotic expressions:

$$A(P) = \frac{1}{v^3}$$  \hspace{1cm} (22)

$$B(P) = \frac{1}{2v} (1 - \frac{1}{v^2})$$  \hspace{1cm} (23)

$$F(P) = \frac{1}{v^2}$$  \hspace{1cm} (24)

We present in the following two sample results obtained using the program in the Appendix.
a) The runaway distribution function:

The results presented in this case are with $\varepsilon = 0.04$, $D = 0$ and $\beta_{th} = 0.0442$ (i.e. 1 kV plasmas). In Fig. (1), we plot the perpendicular velocity moment of the distribution function $f$, defined by

$$ F = 2\pi \int_{0}^{\infty} dP_{\perp} P_{\perp} f $$

as a function of $P_{\parallel}$. The function $F$ is the parallel distribution function. It shows clearly the formation of a runaway tail. We show in Fig. (2) perpendicular cuts of $f$, indicating the presence of a hot tail population, a result similar to what has been recently presented for the non-relativistic case [5]. The flux $\Gamma_R$ of electrons through the surface of a sphere of radius $R$ is defined by

$$ \Gamma_R = -\frac{\partial}{\partial t} \int d^3P f = \int d^3P \text{div} = 2\pi P^2 \int_{-1}^{1} d\mu S_P $$

For the present case ($\varepsilon = 0.04$) the result for $\Gamma_R$ is $0.14 \times 10^{-5}$. This is in very good agreement with the value $0.13 \times 10^{-5}$ obtained from the expression [7]:

$$ \Gamma_R = \Gamma_{NR} \exp \left\{ -\frac{\beta_{th}^2}{8\varepsilon^2} \left[ \frac{1}{2} \frac{2}{3\varepsilon^3/2} (1 + z)^{1/2} \right] \right\} $$
where $\Gamma_{NR}$ is the runaway production rate for the non-relativistic case
($\Gamma_{NR} = 0.196 \times 10^{-5}$ for $\epsilon = 0.04$). Table I summarizes some of the
results obtained for different values of $\epsilon$ for a 1 kV plasma ($\beta_{th} = 0.0442$).
$J_R$ denotes the relativistic current (in units of $envt$)

$$J_R = \int v_4 f d^3p$$ (28)

calculated for different values of the normalized electric field $\epsilon$ and $Z$.
$\Gamma_{NR}$ and $\Gamma_R$ are the runaway production rates calculated for the non-
relativistic case and the relativistic case respectively and in the last
column $\Gamma_R$ is calculated from Eq. (27), showing a very good agreement with
the value in the previous column. Finally, we show in Fig. (3) contour-
plots for the 2-D solution $f$ obtained for $\epsilon = 0.04$.

b) Lower-Hybrid Current Drive.

We consider the case when $\epsilon = 0$ and the quasilinear diffusion term is given
by:

$$D = \begin{cases} 
3 & \text{for } v_1 < P_1/\gamma < v_2 \\
0 & \text{otherwise}
\end{cases}$$ (29)

$D \neq 0$ inside a region bounded by the hyperbolae whose equations are given
by

$$P_\parallel^2 \left( \frac{1}{v_{1,2}^2 - \beta_{th}^2} - P_\perp \beta_{th}^2 \right) = 1$$ (30)
Figure (4) shows a contour plot of $f$ obtained for the case $v_1 = 4$, $v_2 = 7$ and $\beta_{th} = 0.1$. The region where $D \neq 0$ is clearly shown to be bounded by the hyperbola whose equations is given by Eq. (30). Figure (5) shows a plot of the parallel distribution function $F$ as defined in Eq. (25). A plateau formation is apparent in the region where $D \neq 0$. The current associated with this distribution is $J_R = 0.45 \times 10^{-2}$ and the power absorbed as defined in Eq. (21) is $1.57 \times 10^{-4}$ resulting in a figure of merit $J_R/p_d = 28$.

To conclude, using TWODEPEP library and a Fortran code of no more than a few dozen lines, we have obtained results for the numerical solution of the linearized Fokker-Planck equation which compares favourably with what has been previously reported in the literature for the numerical solution of this problem [1, 6]. We are now in the process of extending these results to higher plasma temperature and to velocity spectra extending to higher relativistic speeds. In this latter case, however, we have been limited by the computer memory which for the present runs is close to 5.5 million bytes. Typical runs would take 30 min CPU time on the IBM 3084. Higher relativistic speeds will require a larger number of elements and consequently also a larger memory for the library. The use of cubic elements will require an even larger memory.
<table>
<thead>
<tr>
<th>$\varepsilon$</th>
<th>Z</th>
<th>$10^2 J_R$</th>
<th>$\Gamma_{NR}$</th>
<th>$\Gamma_R$</th>
<th>$\Gamma_R$ (EQ. 27)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.005</td>
<td>1</td>
<td>1.9</td>
<td>$0.76 \times 10^{-30}$</td>
<td>$0.46 \times 10^{-38}$</td>
<td>$0.3 \times 10^{-36}$</td>
</tr>
<tr>
<td>0.01</td>
<td>1</td>
<td>3.8</td>
<td>$0.13 \times 10^{-16}$</td>
<td>$0.18 \times 10^{-18}$</td>
<td>$0.18 \times 10^{-18}$</td>
</tr>
<tr>
<td>0.03</td>
<td>1</td>
<td>12.7</td>
<td>$0.83 \times 10^{-7}$</td>
<td>$0.5 \times 10^{-7}$</td>
<td>$0.45 \times 10^{-7}$</td>
</tr>
<tr>
<td>0.04</td>
<td>1</td>
<td>18.8</td>
<td>$0.196 \times 10^{-5}$</td>
<td>$0.14 \times 10^{-5}$</td>
<td>$0.13 \times 10^{-5}$</td>
</tr>
<tr>
<td>0.04</td>
<td>4</td>
<td>8.25</td>
<td>$0.158 \times 10^{-6}$</td>
<td>$1.02 \times 10^{-7}$</td>
<td>$0.94 \times 10^{-7}$</td>
</tr>
</tbody>
</table>
REFERENCES


Appendix

```
1 9300 2
QXX
FXX(X,Y,U+UX,UY,FCT)
QXY
FXY(X,Y,U+UX,UY)
D3EST
WEIGHT(X,Y)
UPRINT
U
NX  200
NY  50
HX  0.30
HY  0.04
XA  0.1
YA  -1.
NOUT 1
TF  1
HWR  8
INTEGRALFNCT(X,Y,U+UX,UY,NOUT,T)
ARC=-1
FB1  FB(X,Y)
ARC=-3
FB1  0.
ARC=2
GB1  0.
ARC=4
GB1  0.
XGRID 0.1,60.1
YGRID -1.,1.
IX  -1,-3
IY  2,4
ADD.

DOUBLE PRECISION FUNCTION FXX(X,Y,U+UX,UY,FCT)
IMPLICIT REAL*8(A-H,O-Z)
COMMON/D1/ZC,PI,EPS,DO,VP1,VP2,FACT,BTH,DEN,TAU
GAMA=DSQRT(1.+XXNBTHZBTH)
VEL=X/GAMA
S=VEL/DSQRT(2.DO)
VPAR=X*Y/GAMA
D=DO
IF(VPAR.LT.VP1.OR.VPARGT.VP2) D=0.DO
SD=D*(YUX+(1.-Y*Y)*UY/X)
SE=EPS*U
FU=(S*DEXP(-S*S)+Z*S*DEXP(-S*S*C*C)/C)*2./DSQRT(3.1416D0)
IF(X.LE.3.2)
*BU=DERF(S)-FU+Z*DERF(C*S)/C**2
IF(X.GT.3.2) BU=1.
AA=BU/VEL**3
FF=BU/VEL**2
FXX=X**Y*(SE-SD)-(AAUX+FFUX)*XX

 Define Qx in Eq.(18)
```

### Define boundary curve and boundary conditions

```
DOUBLE PRECISION FUNCTION FXY(X,Y,U,UX,UY)
IMPLICIT REAL*8(A-H,O-Z)
COMMON/D1/Z,C,PI,EPS,D0,VP1,VP2,FACT,BTH,DEN,TAU
GAMA=DSQRT(1.+X*X*BTH*BTH)
VEL=X/GAMA
S=VEL/DSQRT(2.*D0)
VPAR=X*Y/GAMA
D=D0
IF(VPAR.LT.VP1.OR.VPAR.GT.VP2) D=0.D0
SD=D*Y*UX+(1.-Y*Y)*UY/X)
SE=EPS*U
FU=(S*DEXP(-S*S)*FACT+Z*ERF(C*S)*FU2C+FU/(2.*S*S))
FU2=1./(2.*S*S)
FU2C=1./(2.*S*S+C*C)
IF(VEL.LE.3.2) AU=DERF(S)*FU2*FACT+Z*ERF(C*S)*FUZC+FU/(2.*S*S)
IF(VEL.GT.3.2) AU=1.-1./VEL*Z+Z
BB=AU/VEL*3
FXY=-X*(1.-Y*Y)*(BB*UY/X-SE+SD)
RETURN

DOUBLE PRECISION FUNCTION FNCT(X,Y,U,UX,UY,NOUT,T)
IMPLICIT REAL*8(A-H,O-Z)
COMMON/D1/Z,C,PI,EPS,D0,VP1,VP2,FACT,BTH,DEN,TAU
FNCT=0.D0
NOU=NOUT+1
NOUM=NOU+1
GAMA=DSQRT(1.+X*X*BTH*BTH)
VPAR=X*Y/GAMA
VEL=X/GAMA
S=VEL/DSQRT(2.*D0)
FU=(S*DEXP(-S*S)+Z*DEXP(-S*S*C*C)/C)*2./DSQRT(3.141600)
IF(VEL.LE.3.2) BU=DERF(S)-FU+Z*ERF(C*S)/C*2.
IF(VEL.GT.3.2) BU=1.
BB=BU/VEL*3
FXY=-X*(1.-Y*Y)*(BB*UY/X-SE+SD)
RETURN

Define Qy in Eq. (49)

Function to be integrated at the end
DOUBLE PRECISION FUNCTION FB(X,Y)
IMPLICIT REAL*8(A-H,O-Z)
COMMON/D1/Z,C,PI,EPS,DO,VP1,VP2,FACT,BTH,DEN,TAU
GAMA=DSQRT(1. +X*X*BTH*BTH)
FB=0.0634936*DEXP(-X*X/(1.+GAMA))
RETURN
END

DOUBLE PRECISION FUNCTION WEIGHT(X,Y)
IMPLICIT REAL*8(A-H,O-Z)
COMMON/D1/Z,C,PI,EPS,DO,VP1,VP2,FACT,BTH,DEN,TAU
V1=VP1-1.
V2=VP2+1.
IF(EPS.NE.0.) VCRIT=DSQRT(1./EPS)
GAMA=DSQRT(1.+X*X*BTH*BTH)
VPAR=X*Y/GAMA
WD0=2.
WE0=2.
WD=0.
WE=0.
IF(EPS.NE.0..AND.VPAR.GT.VCRIT) WE=WE0
IF(DO.NE.0..AND.(VPAR.GT.V1.AND.VPAR.LT.V2)) WD=WD0
WEIGHT=1.+WD+WE
RETURN
END

BLOCK DATA
IMPLICIT REAL*8(A-H,O-Z)
COMMON/D1/Z,C,PI,EPS,DO,VP1,VP2,FACT,BTH,DEN,TAU
DATA Z/1.00/,'C/42.900/,'PI/3.14159265D0/,'FACT/1.000/
DATA EPS/0.00/,'DO/3.000/,'VP1/4.0000/,'VP2/16.000/,'BTH/0.05400/
DATA DEN/0.500/,'TAU/15.00-3/
END
END
Figure 1 The perpendicular velocity moment of the distribution function $f$, defined by Eq. (25), for the runaway problem.

Figure 2 Perpendicular cuts of the distribution function $f$ at different values of $P_\parallel$, for the runaway problem.

Figure 3 Contourplots of the distribution function $f$, for the runaway problem.

Figure 4 Contourplots of the distribution function $f$ for the RF problem.

Figure 5 The perpendicular velocity moment of the distribution function $f$, defined by Eq. (25), for the RF problem.