# Numerical Non-neutral Plasmas

Ross L. Spencer, Grant W. Mason, and S. Neil Rasband

*Department of Physics and Astronomy, Brigham Young University, Provo, Utah 84602*

**Abstract.** For the past 15 years, or so, a set of computational tools for studying non-neutral plasmas has been developed at Brigham Young University. These codes, which include equilibrium codes (for both static plasmas and solitons), radial eigenvalue codes, 2-d eigenvalue codes, and both 2-d and 3-d particle-in-cell simulation codes, will be discussed, along with their applications to problems of interest to the non-neutral plasma physics community.

## INTRODUCTION

For many years we have been developing at Brigham Young University a set of computational tools for studying non-neutral plasmas, tools which are now being used by several different groups around the world. This paper is a brief review of these tools, the computational methods they use, and the situations in which they might appropriately be used. Since we usually work with students, we have tried to keep the methods that we use as simple as possible. In practice this means that we mostly use finite-difference methods on uniform grids, although we have done some work with non-uniform grids using finite-element methods. The codes are all written in FORTRAN, either using embedded graphics via the PLPLOT package from the magnetic fusion group at the University of Texas, Austin, or using MATLAB to produce graphics after the codes have produced data files. The all-important electrostatic field solve is handled in three different ways, depending on the number of computational dimensions. For one-dimensional problems we simply use a standard tri-diagonal Gauss elimination algorithm. For two-dimensional problems we use a banded matrix solver, similar to the tridiagonal Gauss elimination algorithm, but with a much larger bandwidth. We number the points on the grid by moving along the smallest dimension first (the $r$ direction in $(r,z)$ problems in the standard axially elongated non-neutral plasma geometry) so that the bandwidth of the sparse matrix that results from finite-differencing the Laplacian is $2N+1$, where $N$ is the number of grid points in the shortest direction. On modern computers this produces a manageable matrix problem. In three-dimensions we use a multi-grid algorithm.

In the following sections each of the codes in our suite is described.

## DRIFTK

This code[1] is a 1-dimensional radial eigenvalue code based on the drift-kinetic equation, which means that the dynamics perpendicular to the magnetic field is governed

CP692, *Non-Neutral Plasma Physics V,* edited by M. Schauer et al.

by the $\mathbf{E}\times\mathbf{B}$ drift, while parallel to the field we use kinetic theory, as embodied in the plasma dispersion function. The plasma equilibrium is assumed to be infinitely long and axisymmetric so that we may write

$$\phi_1 = \phi_1(r)\exp\left(ikz + im\theta - i\omega t\right)\ , \tag{1}$$

and the mode equation based on this model is

$$\frac{1}{r}\frac{d}{dr}\left(r\frac{d\phi_1}{dr}\right) - \frac{m^2}{r^2}\phi_1 - k^2\phi_1 - \frac{\omega_p^2(r)}{v_{th}^2}W\left(\frac{\omega - m\omega_0(r) - kv_b(r)}{kv_{th}}\right)\phi_1$$

$$+\frac{mq}{\varepsilon_0 Br(\omega - m\omega_0(r) - kv_b(r))}\frac{d}{dr}(n_0(r))\left[W\left(\frac{\omega - m\omega_0(r) - kv_b}{kv_{th}}\right) - 1\right]\phi_1 = 0\ , \tag{2}$$

where $k$ is the axial wave number, $m$ is the azimuthal wave number, $\omega_p^2(r)$ is the plasma frequency, which varies with radius because the density $n_0(r)$ varies with radius, $v_{th} = \sqrt{k_BT/m}$ is the thermal velocity, $q$ is the particle charge, $B$ is the magnetic field strength, $\varepsilon_0$ is the permittivity of free space, $\omega_0(r)$ is the equilibrium drift $\mathbf{E}\times\mathbf{B}$ rotation frequency, and where $v_b$ is the net velocity of the plasma in the $z$-direction. The function $W(z)$ is defined by

$$W(z) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\frac{xe^{-x^2/2}}{x-z}dx\ , \tag{3}$$

analytically continued from the upper-half $z$ plane, and the boundary conditions are that $\phi_1$ vanish at the conducting wall and that $\phi_1 = 0$ for $m \geq 1$ or that $d\phi_1/dr = 0$ for $m = 0$. It should also be mentioned that the code also allows for multiple species.

The algorithm to solve this equation starts by finite-differencing the differential equation on a radial grid. Since the resulting system of linear equations is homogeneous a straightforward solve will just give $\phi_1 = 0$ for any choice of $\omega$. To solve this problem we remove the equation corresponding to some radius $r_0$ from the system and replace it with the equation

$$\phi_1(r_0) = 1\ , \tag{4}$$

which now gives a system for which there is a solution for any value of $\omega$. There is, however, a kink in the solution at $r = r_0$ unless $\omega$ is one of the mode frequencies.

The algorithm now varies the mode frequency $\omega$ until the removed equation is satisfied and the kind is removed (this technique is called matrix shooting.) The user chooses values for $k$ and $m$, then supplies the code with an initial guess for $\omega$. The algorithm depends sensitively on this initial guess and often blows up when the guess is inappropriately chosen, so the code also allows the user to scan $\omega$ either along the real axis or across a region of the complex $\omega$ plane to find approximately where the mode frequency lies. When an approximate value found by scanning is used as an initial guess, the code usually converges.

This code has been used to study damped quasi-modes (spatial Landau damping) and is routinely used by the experimenters at Brigham Young University and other places to identify the modes observed in experiments.

## DRIFT2D

We (S. Neil Rasband) have also developed a 2-dimensional eigenvalue code[2] for use with the $(r,z)$ cold-fluid $v_b = 0$ version of Eq. (2). In this case the perturbed potential is taken to be of form

$$\phi_1 = \phi_1(r,z)\exp(im\theta - i\omega t) \tag{5}$$

and the mode equation involves both $r$ and $z$ derivatives. This code uses a finite-element algorithm on a non-uniform rectangular mesh in the $r,z$ plane. After performing the many moment integrals of the finite element method a large linear algebra problem of the form

$$L(\omega)_{ij}\phi_j = 0 \tag{6}$$

is obtained. To avoid the trivial solution $\phi = 0$ we replace the zero on the right-hand side by some suitably chosen vector $r_i$ to obtain

$$L(\omega)_{ij}\phi_j = r_i\ , \tag{7}$$

which is directly solved by LU decomposition using a banded matrix solver. The unknown mode frequency $\omega$ is then varied until the solution vector $\phi_j$ becomes numerically infinite, indicating that a mode has been found. (The mode frequencies are the values of $\omega$ that make the operator $L$ singular.)

This code is used to find frequency shifts and eigenfunction changes due to finite length, equilibrium shape, and induced charges on nearby walls in non-neutral plasma modes.

## EQUILSOR

This code[3] is our axisymmetric equilibrium code which finds the equilibrium potential $\phi(r,z)$ and density $n(r,z)$ given electrode shapes and applied voltages. The code can model both rings at the outer edge of the cylindrical computing volume, or electrode segments specified as line segments in the $(r,z)$ plane within the cylindrical computing volume. By choosing many such segments any axisymmetric electrode arrangement may be modeled. When the electrode line segments cross the grid lines we use a short-legged version of the finite-difference approximation to the Laplacian to minimize the relatively large errors produced by a “stair-step” approximation.

The code computes three types of equilibria, corresponding to the following three forms of the non-linear equilibrium equation:

**Mid-plane density profile:**

$$\nabla^2\phi = -\frac{q}{\varepsilon_0} n_{mid}(r)\exp\left(-q(\phi(r,z) - \phi(r,0))/kT\right)\ . \tag{8}$$

This form allows the user to specify the radial density profile $n_{mid}(r)$ at the plasma mid-plane ($z = 0$) then allow the plasma to adjust its density along the magnetic field lines until equilibrium is reached.

$\int ndz$ **profile:**

$$\nabla^2\phi = -\frac{q}{\varepsilon_0} n_{mid}(r) \exp\left(-q(\phi(r,z) - \phi(r,0))/kT\right) . \tag{9}$$

This form allows the user to specify a radial profile of the line-integrated density $\int ndz$, and then the algorithm, as part of the equilibrium solve, adjusts the mid-plane density profile $n_{mid}(r)$ until the computed profile of $\int ndz$ matches the profile supplied by the user. So this form is just a slight modification of the mid-plane profile form.

**Global thermal equilibrium:**

$$\nabla^2\phi = -\frac{q}{\varepsilon_0} n_0 \exp\left(-q(\phi(r,z) - \phi(0,0))/kT + Cr^2\right) . \tag{10}$$

This version adjusts both the radial and axial profiles using the above form of the density obtained from the condition for global thermal equilibrium[4]. The constant $C$ is adjusted as the algorithm proceeds to achieve the mid-plane radius specified by the user.

The convergence rate of the code is determined by the ratio $\lambda_D/L_p$, where $\lambda_D$ is the Debye length and where $L_p$ is a measure of the size of the plasma. The code converges very quickly when this ratio is of order 1, but slows down as this ratio becomes small. In addition to the convergence problem, small values of this ratio also cause the exponential factor on the right-hand side of the equilibrium equation to blow up. This can be controlled by starting the iteration sequence at a larger value of the plasma temperature than desired, then slowly decreasing it toward the target value. This allows the equilibrium to slowly adjust to a decreasing Debye length, controlling the potentially large value of the exponential function when the temperature is small.

This code provides the input for DRIFT2D, RATTLE, and INFERNO (discussed below) and is routinely used by experimenters to turn profiles of $\int ndz$ into pictures of what the plasma equilibrium looks like.

## RATTLE

This code[5, 6] is an axisymmetric particle-in-cell simulation in $(r,z)$ geometry. Note that axisymmetry means that each "particle" is actually a ring of charge. The Larmor radius is assumed to be infinitesimally small, so that the radial position of each particle remains unchanged. For simplicity, particles are constrained to lie on the $z =$ const grid lines of the simulation, so that the interpolations that produce the density $n(r,z)$ and the axial electric field $E_z(r,z)$ only take place in $z$. This code reads an equilibrium file produced by EQUILSOR and uses the density $n(r,z)$ from this file to load the particles onto the simulation grid. RATTLE has the same varied electrode shape capability as EQUILSOR. The code has a time step constraint given by the usual electrostatic Courant condition:

$$\omega_p \tau < 1 . \tag{11}$$

For most plasmas this constraint means that the code is useful for modeling plasma loading, plasma dumping, and Gould-Trivelpiece waves, e.g., processes that occur on a timescale of $1/\omega_p$ or $L/v_{th}$, where $L$ is the system size and $v_{th}$ is the thermal velocity. The code is not very useful for studying transport processes because the collisions are constrained to axial exchanges of energy along the field lines (which means that the axial distribution function $f(v_z)$ can't evolve due to collisions) and because transport processes are slow.

The basic cycle that occurs during each time step is:

(1) The particles are moved using the standard leapfrog algorithm in which velocities and positions are known at different times:

$$v_{n+1/2} = v_{n-1/2} + \frac{q}{m} E_z(\mathbf{r}_n)\tau \qquad ; \qquad z_{n+1} = z_n + v_{n+1/2}\tau \,, \tag{12}$$

with $E_z(\mathbf{r})$ found by linear or quadratic interpolation on the grid.

(2) The particle positions are then used to create density at each grid point, using a matching form of interpolation to the one used in step (1). If they don't match, a single particle will exert a force on itself.

(3) Using the density from step (2) the potential $\phi(r,z)$ is found by LU-decomposition and back-substitution on the large banded matrix that represents the Laplacian operator. This direct solve is quite manageable on modern PCs. For instance, a $100 \times 500$ grid makes a banded matrix of size $[100 \times 500, 201]$ which requires only 80 Megabytes of memory.

This code is used to interpret the meaning of observed mode frequencies in experiments, to simulate plasma loading and dumping, to study nonlinear waves (including solitons), etc..

## INFERNO

This code[7] (Grant Mason) is a 3-dimensional particle-in-cell simulation in which the motion in $z$ is governed by Newton's second law, but the motion in the $x,y$ plane is governed by the $\mathbf{E} \times \mathbf{B}$ drift. We compute in a cylindrical geometry, but use a Cartesian grid in $(x,y)$. The reason for this somewhat awkward choice is that the pain of finite differencing the Laplacian near a circular boundary that crosses Cartesian grid lines is easier to bear than that associated with the $r = 0$ singularity in a cylindrical coordinate system. In addition, the small errors in $\phi$ caused by the mismatched boundary are of very short wavelength, and hence decay exponentially from the wall into the plasma. This code is suitable for simulating non-axisymmetric Gould-Trivelpiece modes, diocotron modes, loading and dump effects, etc.. Due to the same limitations on the modelling of particle collisions as in RATTLE, and due to the increased run time required with a 3-dimensional grid, it is not very useful for studying transport problem.

INFERNO has the same basic 3-step cycle as RATTLE, except that particles must be moved in the $(x, y)$ plane as well as in $z$. To follow the drift motion in the $(x, y)$ plane we use a predictor-corrector algorithm discussed by Tajima[8].

The field solve must also be handled differently because in 3-dimensions the banded matrix form of the Laplacian is too large for LU decomposition, so we use a multi-grid algorithm for the field solve.

This code has been used to study the instability of the $m = 1$ diocotron mode for hollow density profiles, and has been used more recently to simulate the trapped-particle asymmetry modes discovered by Kabantsev, *et al*[9].

## CONCLUSION

These codes occupy a middle ground between experiment and theory, and are therefore useful for connecting the two. They can be used to interpret experimental signals, to test theoretical ideas, and can be usefully thought of as highly diagnosable simplified experiments in which the laws of physics can be adjusted to see the relative importance of various effects. They are available upon request for use by anyone who is interested.

## REFERENCES

1. Grant W. Mason and Ross L. Spencer, Phys. Plasmas **9**, 3217 (2002).
2. S. Neil Rasband and Ross L. Spencer Phys. Plasmas **10**, 948 (2003).
3. R. L. Spencer, S. N. Rasband, and Richard R. Vanfleet, Phys. Fluids B, **5**, 4267, (1993).
4. S. A. Prasad and T. M. O'Neil, Phys. Fluids **22**, 278 (1979).
5. M. D. Tinkle, R. G. Greaves, C. M. Surko, R. L. Spencer, and G. W. Mason, Phys. Rev. Lett. **72**, 352 (1994).
6. G. W. Mason, R. L. Spencer, and J. A. Bennett, Phys. Plasmas **3**, 1502 (1996).
7. Grant W. Mason and Ross L. Spencer Phys. Plasmas **9**, 3217 (2002).
8. *Computational Plasma Physics, with Applications to Fusion and Astrophysics*, Toshiki Tajima, (Addison-Wesley, New York, New York, 1989), Chapter 7.
9. A. A. Kabantsev, C. F. Driscoll, T. J. Hilsabeck, T. M. O'Neil, and J. H. Hu, Phys. Rev. Lett. **87**, 225002 (2001).