PFC/RR-89-11 DOE/ET-51013-272

Simulation and Measurement of an Electron Beam in a Wiggler Magnetic Field

Smolin, J.A.

Plasma Fusion Center
Massachusetts Institute of Technology
Cambridge, MA 02139

July 1989

This work was supported by U.S. DOE Contract DE–AC02–78ET51013.

Simulation and Measurement of an Electron Beam in a Wiggler Magnetic Field

by

John Aaron Smolin

SUBMITTED TO THE DEPARTMENT OF
PHYSICS
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE
DEGREE OF
BACHELOR OF SCIENCE IN PHYSICS

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1989

Signature of Author: ______________________________
Department of Physics
May 12, 1989

Certified by: ______________________________
Bruce G. Danly
MIT Plasma Fusion Center
Thesis Supervisor

Accepted by: ______________________________
Aron Bernstein
Chairman, Departmental Committee

Simulation and Measurement of an Electron Beam in a Wiggler Magnetic Field

by

John Aaron Smolin

Submitted to the Department of Physics
on May 12, 1989 in partial fulfillment of the
requirements for the Degree of
Bachelor of Science in Physics

ABSTRACT

Sources of high quality beams of spinning electron beams are critical to efficient free electron devices including FELs, CARMs and gyrotrons. Bifilar helical wigglers can take a beam with little perpendicular momentum and add perpendicular momentum, spinning up the beam. The effect of the electron beam self fields on the beam quality will be important. A computer simulation has been written which can simulate the behavior of electron beams in the wiggler region including the effects of the beam self-electric fields. The equations used in the code are described. Several tests of the code are presented. Results of simulation of a bifilar helical wiggler are described.

Measurement of beam parameters is also necessary. A design for a capacitive axial velocity probe is presented. The probe has been built but is still untested due to problems with a leaky flange.

Thesis Advisor: Bruce Danly
MIT Plasma Fusion Center

Contents

Acknowledgements

I would like to thank to following people for their invaluable assistance over the past year: Jonathan Wurtele, Kenneth Pendergast, Anthony DiReienzo, William Guss, and especially Bruce Danly, for help and guidance on various aspects of physics, George Yarworth and Bob Childs, for their laboratory expertise and for knowing what they were talking about, Pekka Hakkarainen and Mark London, for saving me from the nefarious computer systems which I would otherwise have had to face alone, Andrew Greene, Robert French, C.J. Smith, and the members of the Student Information Processing Board, for help in formatting my thesis in LaTeX, and Debbie Birnby and Jim Brody, who never thought their names would be mentioned in the same breath, for listening to me complain for all this time. If I have left anyone off of this list, I'm sorry and I'm sure I'll think of you sometimes.

To the Whales....

Chapter 1

Introduction

Free electron lasers (FELs), cyclotron autoresonsance masers (CARMs) and gyrotrons employ spinning beams of electrons. The fundamental principle involved is that an accelerating electric charge will give off electromagnetic waves. An electron orbiting in a magnetic field will at all times feel a centripetal acceleration towards the center of its orbit and will therefore radiate. This effect is known as syncrotron radiation. Under the right conditions the electron beam will be in resonance with the electromagnetic wave produced. The beam will be bunched by the electromagnetic wave and will then give up energy to the wave more efficiently, producing coherent radiation. This is stimulated emission between continuum states of unbound electrons, a non-quantum mechanical laser interaction.

This class of free electron devices (so named since the electrons are not bound to atoms in a gas, as in a conventional gas laser) has several properties which distinguishes it from other types of microwave and laser light sources. FELs and CARMS can be run at very high powers and at very high efficiencies as compared to other sources. Power levels as high as 1 Gigawatt and at efficiencies of up to 35% have been reported for some high-power FELs (See [13] and [7]). Efficiencies of up to 65% are possible in theory. Such devices have obvious military applications, and may prove even more useful in a variety of industrial settings from cutting steel to removing tumors. FELs, CARMS or gyrotrons may someday be used to heat plasmas in fusion reactors, or transmit power to earth from orbiting solar satellites, paving the way to cheap energy.[1] They may also be used as sources of RF power for linear accelerators.

In addition to possessing excellent power and efficiency characteristics, free electron devices are *tunable.* Tunability can be achieved over a fairly high bandwidth by varying the magnetic fields and electron beam voltages used in a device. This is a property not available from other forms of coherent sources. Potential applications include use as virtually unjamable pinpoint radars, as point to point communication systems with extremely high bandwidths, and as important tools for research in plasma physics and other fields.

These types of devices will require sources of high quality spinning electron beams. The resonance interaction which occurs depends on the beam having a well defined momentum, with a certain amount of momentum in the direction of travel and a certain amount perpendicular to the direction of travel in the spinning motion. There also must not be too much variation in the momenta of different electrons in the beam or the interaction will become less efficient, or even fail to occur at all. There are several types of sources of good-quality linear beams including electrostatic accelerators (as used in the free electron laser used in the Center for Free Electron Laser Studies at the University of California at Santa Barbara), pulse modulators (pulse forming networks connected to large transformers, as used in a variety of gyrotron, FEL and CARM experiments at MIT), and linacs (linear accelerators, both RF and induction type). Rotation can be added to the beams from these devices using a type of magnet know as the wiggler.

One type of wiggler magnet consists of two wires wrapped around each other in a bifilar helix, terminated by current loop (see Figure 1.1). Current shunts may exist which

[1]Unless cold fusion proves practical first.

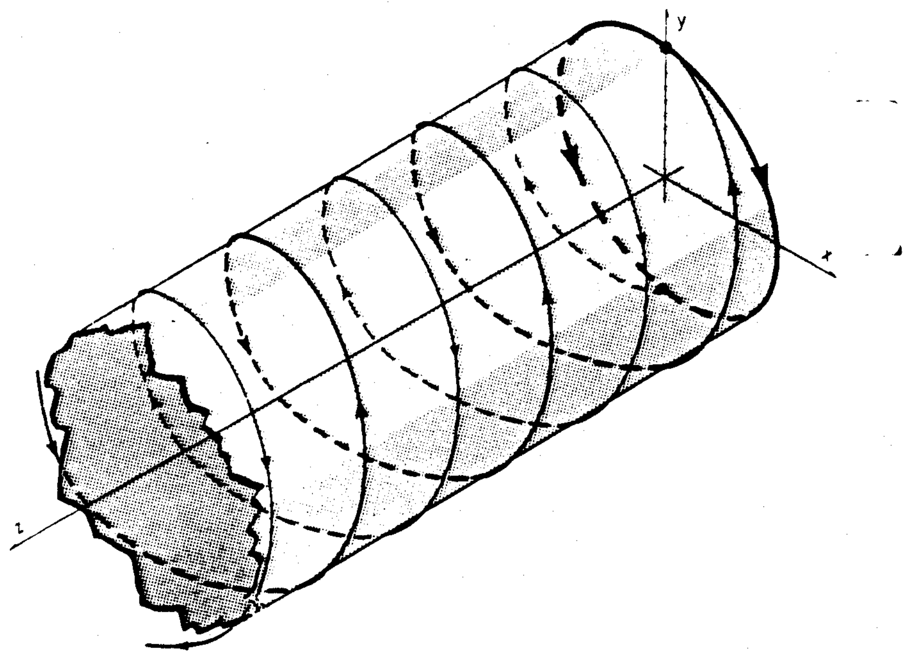


Figure 1.1: A section of a wiggler magnet. The light solid and medium solid lines are the helical windings, and the heavy solid line is the loop termination. No current shunts are shown. This figure is originally from [6].

allow input of additional current or removal of current to taper the field. The wiggler shown has a current loop termination but has no shunts. The wiggler field is a vector perpendicular to the axis of the coil which rotates around with the wires which define the helix. Usually the wiggler in enclosed in a solonoidal magnet which provides a constant axial guide field. Electrons in a certain energy range will interact will this type of field construct and gain perpendicular momentum while losing axial momentum. Thus, the wiggler 'spins-up' the beam.

Design of wigglers for use in free electron devices is no easy matter since predicting the effect of the wiggler on a given electron beam, particularly a realistic, non-idealized beam, is not possible analytically.[2] To this end computer simulations of electron beams

[2]Some recent work on analytic solutions for electron beams in wiggler fields has been done by F.

in wiggler field regions have been written. Most of these do not take into account the effects of space charge, the electrons repelling one another.[3] This effect can be quite important as we shall see.

I have taken an existing code[4] which did not account for space charge nor correctly for tapered wigglers, and added these important features. I also fixed various bugs in the original version. The code is now capable of handling different types of initial beam equilibria including the rigid-rotor equilibrium and the immersed flow equilibrium, and reports a wide variety of information about how the beam behaves throughout the wiggler. The code is written such that it is easy to add additional types of loading and additional diagnostic tests. The workings of the program are described in detail in Chapter 2. The code runs on the CRAY supercomputers at the National Magnetic Fusion Energy Computing Center at Lawrence Livermore National Laboratory.

I have also studied and designed a probe which, in theory, can measure the average axial velocity of a charged particle beam. The theory and design of the probe are described in Chapter 3. Chapter 3 also describes some problems which occurred in the construction and assembly of the probe, which will hopefully prevent others from making the same mistakes.

Hartemann at the MIT Plasma Fusion Center [8].

[3]For an example of a code which does not account for space charge see R. Jackson and C. Sedlak (1983) [10]. A code which does account for space charge has been written by M. Caplan, see [12].

[4]The original version was written by T.M. Tran, and was later worked on by Bruce Danly and Ken Pendergast, all at the MIT Plasma Fusion Center.

Chapter 2

A Particle Simulation Code

2.1 Overview

My program takes as inputs various data defining the wiggler magnet and data about the electron beam which will travel through it. The beam is represented by a number of 'macroparticles,' each of which has a charge to mass ratio the same as that of the electron, but which may have charge accounting for many electrons.[1] It then proceeds to integrate the particle's position based upon the derivatives of the position and momenta with respect to z, the axial coordinate. Various subroutines evaluate the derivatives based on the magnetic and electric fields calculated by other subroutines. During the integration loop various data are recorded by a diagnostic subroutine. After the integration has proceeded all the way through the length of the wiggler, various information is written to an output file, and a set of graph files is prepared which show such information as the average positions, perpendicular and axial velocities, and the fractional spreads in these values, all as functions of z. A copy of the code is included in appendix B.

[1]Only the relationship of charge to mass of a macroparticle is conserved, the total charge is not. The current is the important parameter, as will be discussed in section 2.1.2.

2.1.1 The Integrator

The code uses a very simple integrator. A derivative with respect to z is evaluated at a particular point. This derivative times a small step Δz is added to the original functional value. The derivative is again evaluated at the new functional value at $z + \Delta z$. The average of the two derivatives is then multiplied by Δz and added to the *original* value of the function. The value of z is incremented by Δz and the whole process is then repeated starting with the new position and functional value, until the desired final z is reached. This is a trapazoidal rule integrator, also known as an improved Euler method integrator. Equation 2.1 is the formula which has been described by this paragraph:

$$y_{n+1} = y_n + \frac{\Delta z}{2}[f(z_n, y_n) + f(z_{n+1}, u_{n+1})] \tag{2.1}$$

where y is the function being integrated and

$$u_{n+1} = y_n + \Delta z f(z_n, y_n). \tag{2.2}$$

Other integrators exist which should do a better job than the improved Euler method by allowing increased step sizes with improved accuracy. However, they do incur a cost of calculating the derivatives more times. Since most of the computer's time is spent calculating the derivatives this is not desirable. It is true that the increased step sizes allowed should more than offset this problem, but a version of the code run with a forth-order Runge-Kutta integrator performed less well than the current version. That version of the code did have other deficiencies, and it is possible that the code's performance could be improved with some work on a better integrator. For more information about numerical integration, especially the Euler and improved Euler methods, the reader is referred to [5].

2.1.2 Assumptions

Two-Dimensionality

For the code to run in a reasonable amount of time several simplifying assumptions have been made. These center around the use of a two-dimensional algorithm. A true three-dimensional code would require the use of many more particles, greatly increasing running time, especially if interactions between the particles are accounted for since these are all order N^2 processes. A two-dimensional slice of the electron beam is simulated with derivatives with respect to z. The slice moves through z in steps of Δz as discussed above. If some of the particles in the slice are moving at different axial velocities than others, they will in reality be in the slice at different times. We assume then that the beam is uniform throughout z. Thus, even though the particles get to the slice at different times there will be particles just like them in the slice at the instant the integration is taking place.

Conservation of Current and Zero Axial Space Charge

Generally the current in an electron beam is known or at least can be measured. The charge density or number density of electrons in the beam depends upon the magnetic fields of the region the beam is in as well as the velocity of the beam. The current, however, must be conserved everywhere, or a build-up of charge would occur. Consequently, it is the current that is used as an input to the code. The current is related to the charge per unit length and axial velocity of the beam by:

$$I = Qv_z \tag{2.3}$$

where I is the current of the beam, Q is the charge per unit length and $v_z = v_{||}$ is the average axial velocity of the beam. This relation also holds for an individual electron or for a macroparticle. The code assumes that the *current*, not the charge, on each macroparticle is conserved. The charge can be found using equation 2.3. A fast moving particle therefore has less charge than a slow one. Slow moving particles exert more force on other particles than fast moving ones do, and are also effected more by other particles. Intuitively one can think of this effect in three dimensions as more of the equivalent slow moving particles from nearby z steps bunching up together, increasing the total charge density of that particle, while fast moving particles spread apart, decreasing their charge density.

The calculation of space charge forces assumes no axial space charge. It would be very hard to determine the axial space charge with a two-dimensional code. The real axial space charge should be much smaller than the radial space charge for a beam which is reasonably uniform in z. For a perfectly uniform infinitely long beam the axial space charge will be zero (at any point the fields from charges on either side of the point must cancel) while the radial space charge will depend on the charge density in the beam.[2] It should be noted, however, that though the axial space charge may be small, it will certainly exist in reality. Equation 2.3 tells us that there will be a higher charge density in regions the beam is moving slowly than in regions it is moving faster. There will then be a charge density gradient and, therefore an axial space charge, in any region where the beam is experiencing a velocity gradient with respect to z. There will also be axial space charge effects near the ends of the beam since real beams are not infinitely long, either spatially or temporally.[3]

2.2 Equations of Motion

The program's goal is to determine the motion of the electrons in an electron beam in a wiggler magnetic field. It does this by numerically integrating the differential equations

[2]This can be seen directly from Gauss's law.

[3]It would be possible to construct theoretical situations where this would not be true by using external charges to counteract the space charge, but these charges do not exist in general.

of motion which can be derived from first principles. I begin with a discussion of a single electron in a magnetic field in free space. This eliminates the need, at first, to discuss the effect of the electrostatic forces between particles.

2.2.1 Charged Particle in a Magnetic Field

A particle in a magnetic field must obey the Lorentz force equation

$$\frac{d\vec{p}}{dt} = q\vec{v} \times \vec{B} \tag{2.4}$$

in M.K.S. units. Here $\vec{p} = \gamma m \vec{v}$ is the relativistic momentum of the particle, q is the electric charge on the particle, $\vec{v}$ is its velocity, $\vec{B}$ is the magnetic field at the particle's location in space, m is the particle's mass and $\frac{d}{dt}$ is the derivative with respect to time. γ is the Lorentz factor given by

$$\gamma = \sqrt{\frac{1}{1 - v^2/c^2}} = \sqrt{\frac{1}{1-\beta^2}} \tag{2.5}$$

where $c = 2.99792 \times 10^8$m/s is the speed of light and $\beta = v/c$.

We wish to write separate equations for each of the spatial coordinates of the particle, and to transform the derivatives with respect to time into a derivative with respect to z. We know that

$$\frac{dx}{dt} = v_x, \quad \frac{dy}{dt} = v_y, \quad \frac{dz}{dt} = v_z. \tag{2.6}$$

Looking at only the x coordinate it is clear from the chain rule that

$$\frac{dx}{dz} = \frac{dx}{dt}\frac{1}{\frac{dz}{dt}} = \frac{v_x}{v_z} = \frac{p_x}{p_z}. \tag{2.7}$$

Here v_x and v_z and the x and z components of the particle's velocity and $p_x = \gamma m v_x$ and $p_z = \gamma m v_z$ are the x and z components of the momentum. Similarly one can write

$$\frac{dy}{dz} = \frac{p_y}{p_z} \tag{2.8}$$

We can then write in complex form

$$\frac{d}{dz}(x+iy) = \frac{p_x + ip_y}{p_z}. \tag{2.9}$$

i is the square root of -1. Equation 2.9 is the equation of motion for the particle position as a function of z. It gives the x and y components of the position only. The need for a relation for z component of the position is obviated by the fact that the derivatives in equation 2.9 are with respect to z.

The above equation of motion is not, in itself, sufficient. Equations for p_x, p_y, and p_z are also needed. Fortunately equation 2.4, the Lorentz force equation, provides us with the relationships we need. Once again looking at only the x components from the cross product, we see

$$\frac{dp_x}{dt} = q(v_yB_z - v_zB_y) = \frac{q}{\gamma m}(p_yB_z - p_zB_y) \tag{2.10}$$

where B_x, B_y and B_z are the components of the magnetic field at the particle. To convert this to a derivative with respect to z we simply divide by v_z as above. This yields

$$\frac{dp_x}{dz} = \frac{q}{\gamma m v_z}(p_yB_z - p_zB_y) = \frac{q}{p_z}(p_yB_z - p_zB_y). \tag{2.11}$$

Following the same logic, we find for $\frac{dp_y}{dz}$ and $\frac{dp_z}{dz}$ that

$$\frac{dp_y}{dz} = \frac{q}{p_z}(p_zB_x - p_xB_z) \tag{2.12}$$

and

$$\frac{dp_z}{dz} = \frac{q}{p_z}(p_xB_y - p_yB_x). \tag{2.13}$$

It is convenient to rewrite these equations in terms of the normalized coordinates defined below:

$$p_+ = p_x + ip_y \tag{2.14}$$

$$B_+ = B_x + iB_y \tag{2.15}$$

$$\tilde{p}_+ = \frac{p_+}{mc}e^{-ik_wz} \tag{2.16}$$

$$\tilde{p}_z = \frac{p_z}{mc} \tag{2.17}$$

k_w ('kay-wiggle') is the wave number of the wiggler helix, or $\frac{2\pi}{\lambda_w}$, where λ_w (lambda-wiggle) is the wiggler period. One reason for this normalization is that it saves math in the integration loop since the constants don't need to be calculated over and over, but the main reason is to help out the integrator. Electrons in a wiggler will tend to have orbits in x and y which are one wiggler period long in z. The above transformations define a frame which is rotating around with the wiggler. Thus the electrons will have no rotation in this frame. This should help stabilize the integrator and an older version of the code which did not employ the transformations did not work as well as the latest version. A much smaller step size was needed to avoid exponential growth of particle orbits due to numerical error.

It is now possible to write equation 2.9 the following nice form:

$$\frac{d}{dz}(x+iy) = \frac{p_+}{p_z} = \frac{mc\tilde{p}_+}{mc\tilde{p}_z}e^{ik_w z} = \frac{\tilde{p}_+}{\tilde{p}_z}e^{ik_w z} \tag{2.18}$$

Next, combining equations 2.11 and 2.12 in complex form we obtain

$$\frac{d}{dz}(p_x + ip_y) = \frac{dp_+}{dz} = \frac{q}{p_z}[B_z(p_z - ip_x) + p_z(iB_x - B_y)]. \tag{2.19}$$

This can be rewritten in the form

$$\frac{dp_+}{dz} = -iq[\frac{B_z}{p_z}(p_x + ip_y) - (B_x + iB_y)] = -iq(\frac{B_z p_+}{p_z} - B_+) \tag{2.20}$$

To convert this into the rotating frame we note from applying the chain rule to equation 2.16 we have

$$\frac{d\tilde{p}_+}{dz} = \frac{-ik_w p_+}{mc}e^{-ik_w z} + \frac{1}{mc}\frac{dp_+}{dz}e^{-ik_w z}. \tag{2.21}$$

Plugging in the result of equation 2.20 for $\frac{dp_+}{dz}$ yields

$$\frac{d\tilde{p}_+}{dz} = -\frac{1}{mc}e^{-ik_w z}[k_w p_+ + iq(\frac{B_z p_+}{p_z} - B_+)] \tag{2.22}$$

or

$$\frac{d\tilde{p}_+}{dz} = -ik_w\tilde{p}_+ - \frac{iqB_z}{mc}\frac{\tilde{p}_+}{\tilde{p}_z} + \frac{iqB_+}{mc}e^{-ik_w z}. \tag{2.23}$$

The equation for $\frac{dp_z}{dz}$ (equation 2.13) must also be converted to the new variables. This is fairly simple since

$$\frac{d\tilde{p}_z}{dz} = \frac{1}{mc}\frac{dp_z}{dz}. \tag{2.24}$$

Therefore we find

$$\frac{d\tilde{p}_z}{dz} = \frac{q}{mcp_z}(p_x B_y - p_y B_x) \tag{2.25}$$

or

$$\frac{d\tilde{p}_z}{dz} = \frac{q}{(mc)^2\tilde{p}_z}(p_x B_y - p_y B_x) = \frac{q}{mc\tilde{p}_z}(B_y Real\{\tilde{p}_+ e^{ik_w z}\} - B_x Imag\{\tilde{p}_+ e^{ik_w z}\}) \tag{2.26}$$

where $Real\{\tilde{p}_+\}$ and $Imag\{\tilde{p}_+\}$ denote the real and imaginary parts of $\tilde{p}_+$ respectively.

The reader may by now be questioning the need for an equation for motion for p_z at all. After all, due to the cross product in the Lorentz force equation (2.4) the interaction of the particle with the magnetic field can only change the direction of the particle's momentum, while leaving it's magnitude unchanged. In other words, energy is conserved and, if the energy is known, p_z can always be calculated from p_x and p_y directly without the need for a differential equation of motion. This is true of course, however, when the additional force due to the space charge interaction is added in the next section, kinetic energy will no longer be a constant of the motion and the equation for p_z will be necessary. At that time we will need also an equation relating γ (the energy) to p_x, p_y and p_z.

It is worth noting that equations 2.23 and 2.26 depend on ratio q/m and not on either q or m individually. Since the code models electron beams it is clear that this ratio must always be $-e/m_e$ where e is the magnitude of the charge on the electron and m_e is the mass of the electron. This ratio must be preserved even if we model the beam with 'macroparticles' consisting of many electrons if we expect the equations of motion to correctly predict the particles' trajectories. We can then replace all instances of q with $-e$ and all instances of m with m_e. Thus it is natural to define the following additional normalizations:

$$\tilde{B} = \frac{e\vec{B}}{m_e c k_w} \tag{2.27}$$

$$\tilde{x} = k_w x \tag{2.28}$$

$$\tilde{y} = k_w y \tag{2.29}$$

and

$$\tilde{z} = k_w z. \tag{2.30}$$

These normalizations result in the following equations of motion, the final set:

$$\frac{d}{d\tilde{z}}(\tilde{x} + i\tilde{y}) = \frac{\tilde{p}_+}{\tilde{p}_z} e^{i\tilde{z}} \tag{2.31}$$

$$\frac{d\tilde{p}_+}{d\tilde{z}} = -i[(1 - \frac{\tilde{B}_z}{\tilde{p}_z})\tilde{p}_+ + \tilde{B}_+ e^{-i\tilde{z}}] \tag{2.32}$$

and

$$\frac{d\tilde{p}_z}{d\tilde{z}} = \frac{1}{\tilde{p}_z}[\tilde{B}_x Imag\{\tilde{p}_+ e^{i\tilde{z}}\} - \tilde{B}_y Real\{\tilde{p}_+ e^{i\tilde{z}}\}] \tag{2.33}$$

or

$$\frac{d\tilde{p}_z}{d\tilde{z}} = \tilde{B}_x Imag\{\frac{d}{d\tilde{z}}(\tilde{x} + i\tilde{y})\} - \tilde{B}_y Real\{\frac{d}{d\tilde{z}}(\tilde{x} + i\tilde{y})\}. \tag{2.34}$$

Equations 2.31 and 2.32 are the used in the original version of the code, which did not account for space charge. The equations for $\tilde{p}_z$ were not necessary, as explained above, but will be needed in the next section which will show how the space charge forces are calculated.

2.2.2 Equations of Motion Including Space Charge Effects

The equations of motion derived in the previous section are those used in the original version of the code.[4] To add the effects of space charge to the equations of motion from

[4]Though the equation for $\frac{d\tilde{p}_z}{d\tilde{z}}$ was not used because without space charge kinetic energy is conserved.

the preceding section we will need to add terms proportional to the electric field.

It is assumed for the purposes of the simulation that there is no field along the wiggler, i.e.

$$E_{\parallel} = E_z = 0. \tag{2.35}$$

The full Lorentz force equation is

$$\frac{d\vec{p}}{dt} = q(\vec{E} + \vec{v} \times \vec{B}). \tag{2.36}$$

In equation 2.4 it was assumed that $\vec{E} = 0$ everywhere. In this section we set $\vec{B} = 0$ and calculate the additional terms which will need to be added by superposition to the equations of motion calculated above. Thus, for the purposes of this section we have

$$\frac{d\vec{p}}{dt} = q\vec{E}. \tag{2.37}$$

Note that since we have assumed $E_z = 0$ it is clear from equation 2.37 that there is no contribution for the electric field to the force in the z direction. No changes need be made to equation 2.34.

Applying equation 2.37 to p_+ and multiplying by $\frac{1}{dz/dt}$ to convert the derivative with respect to time into a derivative with respect to z, we obtain

$$\frac{dp_+}{dz} = \frac{q}{v_z}(E_x + iE_y) \tag{2.38}$$

or, in normalized coordinates

$$\frac{dp_+}{dz} = \frac{qE_+}{v_z} = \frac{q\gamma E_+}{c\tilde{p}_z} \tag{2.39}$$

where E_+ has the natural definition

$$E_+ = E_x + iE_y. \tag{2.40}$$

Using equation 2.21 we see that

$$\frac{d\tilde{p}_+}{dz} = [\frac{-ik_w p_+}{mc} + \frac{1}{mc}\frac{dp_+}{dz}]e^{-ik_w z} = -ik_w\tilde{p}_+ + \frac{1}{mc}\frac{dp_+}{dz}e^{-ik_w z}. \tag{2.41}$$

The first term of equation 2.41 serves to unrotate the coordinate system. This term, which comes only from the chain rule and has nothing to do with the forces involved in the motion, has already been accounted for in equation 2.23. The second term of equation 2.41 breaks up into two parts, one from the force due to the magnetic fields, and another due to the space charge. Since the magnetic fields are all zero in this section we may replace $\frac{dp_+}{dz}$ in equation 2.41 with the result of equation 2.39, yielding

$$\frac{d\tilde{p}_+}{dz} = \frac{q\gamma E_+}{mc^2}e^{-ik_w z}. \tag{2.42}$$

Once again noting that the ratio of charge to mass of the particles in an electron beam (or a model of an electron beam) is always $-e/m_e$ it is natural to define

$$\tilde{E}_+ = \frac{eE_+}{m_e c^2 k_w}e^{-ik_w z} \tag{2.43}$$

and to rewrite equation 2.42 as

$$\frac{d\tilde{p}_+}{d\tilde{z}} = -\frac{\gamma\tilde{E}_+}{\tilde{p}_z}. \tag{2.44}$$

Note that notation makes equation 2.43 a little unclear. The first e is the magnitude of the charge on the electron; The second is the base of natural logarithms.

We may now combine equations 2.32 and 2.44 to obtain the complete form used in the latest version of the program:

$$\frac{d\tilde{p}_+}{d\tilde{z}} = -i[(1-\frac{\tilde{B}_z}{\tilde{p}_z})\tilde{p}_+ + \tilde{B}_+e^{-i\tilde{z}}] - \frac{\gamma\tilde{E}_+}{\tilde{p}_z} \tag{2.45}$$

Equations 2.31 and 2.34 remain unchanged. I repeat them here for convenience:

$$\frac{d}{d\tilde{z}}(\tilde{x}+i\tilde{y}) = \frac{\tilde{p}_+}{\tilde{p}_z}e^{i\tilde{z}} \tag{2.46}$$

$$\frac{d\tilde{p}_z}{d\tilde{z}} = \tilde{B}_x Imag\{\frac{d}{d\tilde{z}}(\tilde{x}+i\tilde{y})\} - \tilde{B}_y Real\{\frac{d}{d\tilde{z}}(\tilde{x}+i\tilde{y})\}. \tag{2.47}$$

Equation 2.46 comes only from the definitions of momentum and velocity, and is therefore unaffected by the addition of space charge forces. Equation 2.47 is not affected because we have assumed there is no axial space charge.[5]

[5]That is to say there is no E_z.

2.2.3 Calculation of γ

It should be pointed out the the γ in equation 2.45 is no longer a constant since the space charge forces can add kinetic energy to the particles in the beam. γ should be found from the definition (equation 2.5). To find γ in terms of $\tilde{p}_+$ and $\tilde{p}_z$ we start from the definition of $\vec{p}$:

$$\vec{p} = \gamma m\vec{v} = \gamma mc\vec{\beta}. \tag{2.48}$$

Then

$$|\vec{\beta}|^2 = \beta^2 = \frac{|\vec{p}|^2}{(\gamma mc)^2} = \frac{p^2}{(\gamma mc)^2} \tag{2.49}$$

or, in terms of normalized units,

$$\beta^2 = \frac{|\tilde{p}_+|^2 + \tilde{p}_z^2}{\gamma^2}. \tag{2.50}$$

By squaring both sides of equation 2.5 we obtain

$$\gamma^2 = \frac{1}{1-\beta^2}. \tag{2.51}$$

Substituting in the result of equation 2.50 for β^2 and solving for γ^2 results in

$$\gamma^2 = |\tilde{p}_+|^2 + \tilde{p}_z^2 + 1 \tag{2.52}$$

or

$$\gamma = \sqrt{|\tilde{p}_+|^2 + \tilde{p}_z^2 + 1}. \tag{2.53}$$

This is the formula for γ which should be used in the equations of motion (equation 2.45). Another form which we will find useful later on is an equation for β_z in terms of γ and $\beta_\perp = \sqrt{\beta_x^2 + \beta_y^2}$. We see that

$$\beta_z = \sqrt{1 - (1/\gamma^2) - \beta_\perp^2} \tag{2.54}$$

This equation follows directly from equation 2.52 and the definitions of the variables involved.

2.3 Calculation of the Magnetic and Electric Fields

In the above sections the equations of motion were derived in terms of the external magnetic and the self (space charge) electric fields. It was assumed that these fields were known at each particle at all times. The calculation of the fields is, however, fairly complicated. In fact, more time is spent by the computer calculating the magnetic fields than in doing anything else.

2.3.1 The External Magnetic Field

The magnetic field from any current-carrying wire can be calculated from the following equation:

$$d\vec{B} = \frac{\mu_0 I \vec{dl} \times \hat{r}}{4\pi r^2} \tag{2.55}$$

where $\mu_0 = 4\pi \times 10^{-7}$ is the magnetic permeability of vacuum, I is the current (in amps) flowing in the wire, dl is the differential element of length associated with a differential element of current, and pointing in the same direction, $\hat{r}$ is a unit vector pointing from the element of current to the position at which we wish to evaluate the magnetic field and r is the distance from the element of current to the position at which we wish to find $\vec{B}$. Equation 2.55 is known as the Biot-Savart law.

A subroutine library called COIL3, written by C.F.F. Karney, carries out a numerical integration of equation 2.55 in an efficient manner. The code is quite fast when run on a Cray supercomputer because it is fully vectorized.[6]

My program constructs a bifilar helical wiggler by dividing each wire into small segments of current[7] as an approximation to the actual wiggler magnet. The current shunts are also divided up into segments for use by COIL3. I have found 32 bars per wiggler

[6]Vectorization is a kind of optimization which allows certain array operations to progress in parallel, greatly enhancing the performance of the code.

[7]The current-carrying elements are called 'bars' by COIL3.

period and 32 bars per semi-circular shunt to produce reasonable results. The number of bars per wiggler period and per shunt is given as the program input parameter NSEG. After the current bars, as well as any constant B_z offset have been defined, whenever the program needs to know the magnetic field at a particle it simply calls the COIL3 subroutine MAGFIELD with the particle's position as an input. The subroutine returns the magnetic field. MAGFIELD can operate on an array of particle positions in its vectorized mode, and actually takes such an array as input, returning an array of field values.

2.3.2 The Electric Field Due to Space Charge

Poisson's Equation

The electric potential is calculated at any point from Poisson's equation,

$$\nabla^2\phi = \frac{\rho}{\epsilon_0} \tag{2.56}$$

where ϕ is the electric potential, ρ is the charge density (as a function of position) and $\epsilon_0 = 8.8542 \times 10^{-12}$farads/m is the electric permeability of vacuum. The electric potential is defined in terms of the electric field by

$$E = -\nabla\phi. \tag{2.57}$$

In the above equations ∇ is the differential operator defined by

$$\nabla\psi = \frac{\partial\psi}{\partial x}\hat{x} + \frac{\partial\psi}{\partial y}\hat{y} + \frac{\partial\psi}{\partial z}\hat{z} = \frac{\partial\psi}{\partial r}\hat{r} + \frac{1}{r}\frac{\partial\psi}{\partial\theta}\hat{\theta} + \frac{\partial\psi}{\partial z}\hat{z} \tag{2.58}$$

where ψ is a scalar function, $\hat{x}$, $\hat{y}$ and $\hat{z}$ are unit vectors along the coordinate axis in Cartesian coordinates, r and θ are coordinates in cylindrical coordinates, and $\hat{r}$ and $\hat{\theta}$ are unit vectors in the directions of r and θ respectively. It can also be shown that

$$\nabla^2\psi = \frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2} + \frac{\partial^2\psi}{\partial z^2} = \frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial\psi}{\partial r}) + \frac{1}{r^2}\frac{\partial^2\psi}{\partial\theta^2} + \frac{\partial^2\psi}{\partial z^2}. \tag{2.59}$$

These formulae have been quoted from Jackson (1975) [9].

Since my program is a two dimensional code assuming no axial space charge we can take $\frac{\partial\phi}{\partial z} = 0$ and $\frac{\partial^2\phi}{\partial z^2} = 0$. Thus, the last terms of the above two equations drop out, simplifying matters a bit. In the following sections all charges Q and q should be taken to mean charge per unit length. The particles, which are infinitely small circles in two dimensions, can be thought of as uniform cylinders of charge.

To solve equations 2.56 and 2.57 a boundary condition is also needed. In most wiggler magnets the beam is contained within a cylindrical metal beam tunnel of radius R_w. The potential on this wall is usually fixed at an external voltage, usually ground. Thus the needed boundary condition is

$$\phi(R_w) = \phi_{\text{external}} = 0. \tag{2.60}$$

Equations 2.56 and 2.57 and the boundary condition 2.60 are all that is needed in principle to find the electric field everywhere. However, an analytic solution to these equations for a many particle system in a conducting cylindrical tube is non-trivial, and does not exist except in certain cases involving very restrictive assumptions. Because of this problem the program solves the Poisson equation numerically. The numerical solution is done quite efficiently and might well require less time to perform the calculations than would an analytic solution in terms of Bessel functions.[8]

The subroutine HWSPLR from the SLATEC math library exists which solves the two-dimensional Poisson equation on a cylindrical grid using a finite-differencing scheme.[9] HWSPLR takes as inputs an array containing the charge density ρ on each point on the grid and also containing boundary condition information. On output it returns the

[8]Calculation of an analytic solution requires time proportional to the number of particles squared. Calculation using a grid requires some fixed time (see below) for calculations by the grid solver, plus a time proportional to the number of particles to assign them to gridpoints.

[9]The details of how HWSPLR works are beyond the scope of this paper, other than to note that HWSPLR has an execution time on the order of $MN\log(N)$ where M is the number of circular grid lines and N is the number of radial grid lines. The documentation for HWSPLR references Swarztrauber and Sweet (1975) [15].

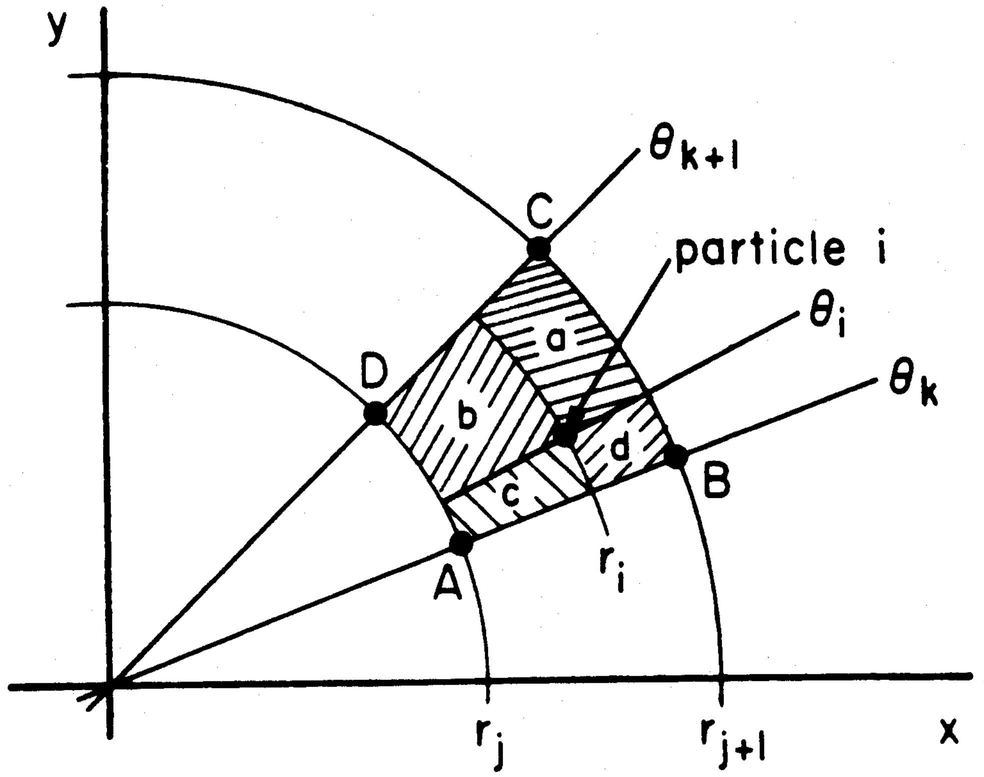


Figure 2.1: The polar coordinate grid system, showing the areas used for assigning charge to the nearest gridpoints (from Langdon and Birdsall [2])

electric potential ϕ on each point on the grid. Calculation of the charge density on the gridpoints is not as simple as it may seem. The next section describes how it is done.

Finding the Charge Density on a Cylindrical Grid

The following discussion shows how to assign the charge from a particle to a charge density at a gridpoint. This discussion follows Langdon and Birdsall (1985) [2].

Figure 2.1 shows a section of the polar coordinate grid system. The areas of regions a, b, c and d are used to weight the charge of particle i to gridpoints A, B, C, and D. The notation r_j indicates the radial coordinate of the j^{th} radial gridpoint. θ_k is similarly the angle associated with the k^{th} angular gridpoint. The following formula, quoted from

Langdon and Birdsall, gives the charge Q assigned to gridpoint A:

$$Q_A = Q_{j,k} = q_i \frac{(r_{j+1}^2 - r_i^2)(\theta_{k+1} - \theta_i)}{(r_{j+1}^2 - r_j^2)(\theta_{k+1} - \theta_k)} \tag{2.61}$$

where q_i is the charge of the i^{th} particle. It is easy to see how this formula is derived. The charge assigned to gridpoint A is simply the charge of particle i times the area of area a divided by the area of the whole region between the four gridpoints A, B, C, and D. The formulae for Q_B, Q_C and Q_D are equivalently the charge q_i times the proper area divided by the total area of the region. This is known as area weighting. It should be pointed out that at the origin these formalae become rather strange. This is a result of using polar coordinates. There is a total of N gridpoints assigned to the origin, with coordinates (j,k) of $(0,0),(0,1),\ldots,(0,k),\ldots,(0,N-1)$. Near the origin the diagram shown in Figure 2.1 becomes degenerate and both point A and D are at the origin. Equation 2.61 and the similar equations for the other three gridpoints still hold. It is only necessary to add up the charge assigned to all the origins to calculate the actual charge at the origin:

$$Q_{\text{origin}} = \sum_{n=0}^{N-1} Q_{0,k} \tag{2.62}$$

Another small oddity is that since the coordinate system wraps around a $\theta = 2\pi$, care must be taken to ensure that charge assigned to angles greater than 2π is at some time put where it belongs, i.e. back into the interval $[0, 2\pi]$.

Next the charges must be converted into charge densities by dividing by the local differential area. The differential element of area is given by, in polar coordinates,

$$dA = r dr d\theta. \tag{2.63}$$

On a quantized grid we can replace dr and $d\theta$ by Δr and $\Delta\theta$, where Δr is the distance between adjacent circular grid lines and $\Delta\theta$ is the angular distance between adjacent radial grid lines. Thus, at all angular gridpoints with a given radius r_j we have

$$\Delta A_{j,k} = r \Delta r \Delta\theta. \tag{2.64}$$

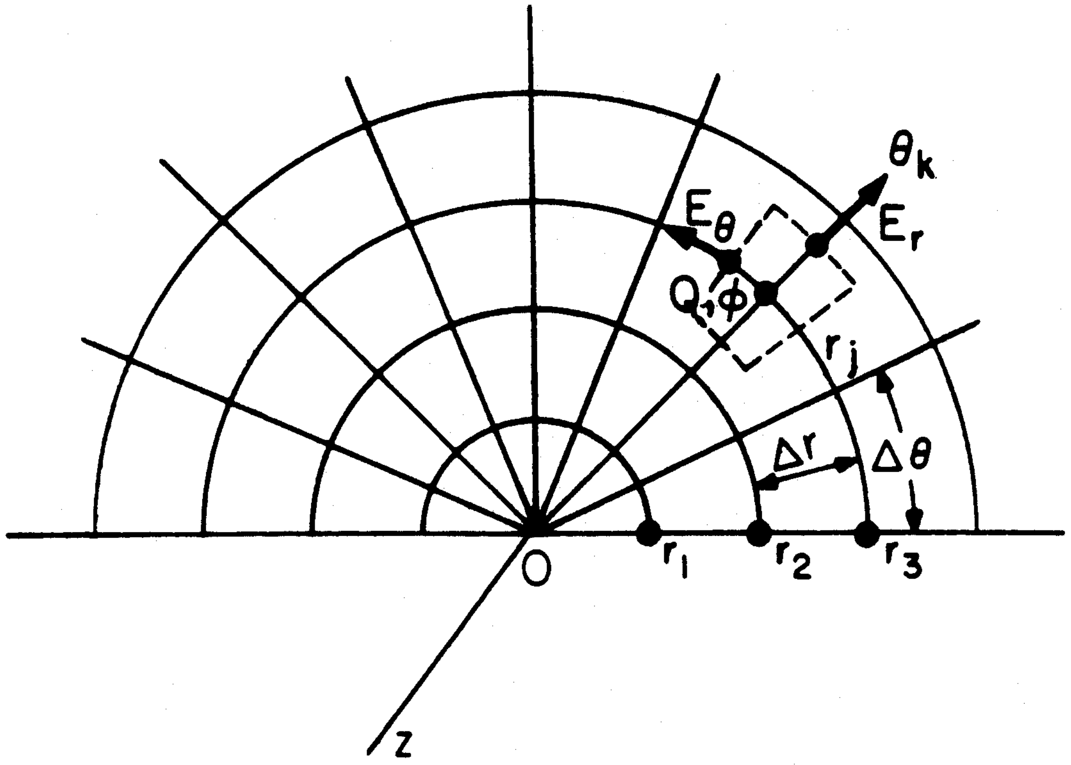


Figure 2.2: Area used in calculating charge density on the cylindrical grid except at the origin (from Langdon and Birdsall [2])

An alternate way of finding equation 2.64 is to think of ΔA as representing an area like the one shown in Figure 2.2 which surrounds a gridpoint and extends halfway to the next set of grid lines in each coordinate. This area is found by taking the difference of the areas of two circles having radii defined by outermost and innermost boundaries of the region and multiplying by the ratio of $\Delta\theta$ to 2π to account for the angular portion of the circle occupied by the region. In other words

$$\Delta A_{j,k} = \pi[(r_j + \frac{1}{2}\Delta r)^2 - (r_j - \frac{1}{2}\Delta r)^2]\frac{\Delta\theta}{2\pi} = r_j \Delta r \Delta\theta. \tag{2.65}$$

We can now use equation 2.64 to find

$$\rho_{j,k} = \frac{Q_{j,k}}{\Delta A_{j,k}} = \frac{Q_{j,k}}{r\Delta R\Delta\theta}. \tag{2.66}$$

We have now finished calculating the charge density at the gridpoints from the charge on the individual particles everywhere except for at the origin. Unfortunately, if we apply equation 2.66 with $r_j = 0$ we find that the charge density goes to infinity. This is not helpful and is incorrect. Obviously if we have a beam of uniform charge density we should not find that the charge density is infinite at the origin and nowhere else. It is better to continue in the line of thought presented in the preceding paragraph. If we think of the local area at the origin as a circle of radius $\Delta R/2$ the situation is much improved. This circle accounts for all of the area not covered by the other regions described above. The area of this circle is given by

$$\Delta A_{\text{origin}} = \pi(\frac{\Delta r}{2})^2 \tag{2.67}$$

and the charge density at the origin is then

$$\rho_{\text{origin}} = \frac{Q_{\text{origin}}}{\Delta A_{\text{origin}}} = \frac{Q_{\text{origin}}}{\pi(\Delta r/2)^2}. \tag{2.68}$$

The charge density is now ready to be used to determine the potential by solving Poisson's equation. This is handled by HWSPLR.

Converting The Electric Potential to the Electric Field

The electric field is calculated from the potential by use of equation 2.57. On our grid this means we can find the radial electric field E_r and the angular electric field E_θ by the following equations:

$$E_{r\,j+\frac{1}{2},k} = -\frac{\phi_{j+1,k} - \phi_{j,k}}{\Delta r} \tag{2.69}$$

$$E_{\theta\,j,k+\frac{1}{2}} = -\frac{\phi_{j,k+1} - \phi_{j,k}}{r_j \Delta\theta}. \tag{2.70}$$

Note that these equations calculate the fields halfway between grid lines. If we want the fields on the gridpoints we can average the surrounding pair of field values. It turns out that after a minimum of algebraic manipulation we have, with no loss of generality:

$$E_{r\,j,k} = \frac{\phi_{j-1,k} - \phi_{j+1,k}}{2\Delta r} \tag{2.71}$$

and

$$E_{\theta\,j,k} = \frac{\phi_{j,k-1} - \phi_{j,k+1}}{2r_j \Delta\theta}. \tag{2.72}$$

Here again care must be taken at the theta wrap-around and near the origin. When $k = M - 1$ the $k + 1$ in equation 2.72 should be replaced with 0. Similarly, when $k = 0$ the $k - 1$ should be replaced with $N - 1$. Near the origin equation 2.71 breaks down since $j - 1$ is completely undefined. A similar problem occurs at the outer boundary where the beam tunnel wall is. Langdon and Birdsall make the suggestion that we use at the origin

$$E_{r\,0,k} = E_{r\,\frac{1}{2},k} \tag{2.73}$$

and

$$E_{\theta\,0,k} = E_{\theta\,1,k}. \tag{2.74}$$

This solution is not entirely satisfactory as it is only approximate, but it is difficult to do better. A distinction is now made between the N origins which represents approaching arbitrarily close to the origin from each of the N directions. A similar method can be used to determine the fields at the boundary. The E_r is taken to be its value halfway from the second to last grid line to the last one. E_θ is taken to be 0. since this is the boundary condition for all electric fields at the surface of a conductor. However, this is not as important since if enough grid lines are used it is unlikely that a particle will ever be near enough to the wall for us to need to know the field there. If a particle is found at the boundary it is likely that something has gone wrong anyway.

It is now a simple matter to convert from E_r and E_θ to E_x and E_y or to $\tilde{E}_+$. If we take

$$E_{r\theta} = E_r + iE_\theta \tag{2.75}$$

it is easy to show that

$$E_+ = E_x + iE_y = E_{r\theta}e^{i\theta}. \tag{2.76}$$

2.3.3 Self Magnetic Fields

Self magnetic fields arise from the moving charges in the beam. Moving charges are currents, and a current generates a magnetic field. The effects of self magnetic fields may be important in some devices. My code does not yet calculate the self-consistant self magnetic fields. However, these fields may be calculated in a manner quite similar to that used to find the self electric fields described above. It is simply necessary to assign currents to a grid and to solve Poisson's equation three times to find the vector potential $\vec{A}$. The boundary conditions will depend on the magnetic penetration time of beam tunnel wall. I have not worked out the exact methodology for calculating the self magnetic fields, but it should not be especially difficult now that the method for finding the self electric fields is understood.

2.4 Tests

A particle simulation code is of no use if its results are not believed. To prove that the code is reliable it can be tested against the analytic solutions which exist for various simple cases. The next few sections describe some of these tests, and report how well the code fared in comparison.

2.4.1 Single Particle with Axial Magnetic Field and No Space Charge

One simple case is that of a single charged particle in an constant magnetic field ignoring space charge effects. The particle must obey the Lorentz equation (equation 2.4). Since the magnetic force is always perpendicular to the particle's direction of motion, the particle must be in a circular orbit. Any particle in a circular orbit must feel a centripetal force inwards given by:

$$F = \frac{\gamma m v_\perp^2}{r} \tag{2.77}$$

where $v_\perp$ is the velocity perpendicular to the magnetic field and r is the radius of the particle's orbit. Equating equations 2.4 and 2.77 and solving for r we find

$$r = \frac{m_e v_\perp \gamma}{eB_z} = \frac{m_e p_\perp c}{eB_z}. \tag{2.78}$$

Here we take $p_\perp = \gamma\beta_\perp$ to be a normalized perpendicular momentum. It is also easy to show that the wavelength in z of the motion is given by

$$\lambda = \frac{2\pi m_e \gamma \beta_z c}{eB_z} = \frac{2\pi m_e c}{eB_z}\sqrt{\gamma^2 - 1 - p_\perp^2}. \tag{2.79}$$

Figure 2.3 shows the result of running the code space charge disabled and with the following parameters: $\gamma = 2.12907$, $B_z = 1.7045$ Tesla, $p_\perp = 1$, and a single particle starting at the origin. Initially all the perpendicular momentum is in the x direction.

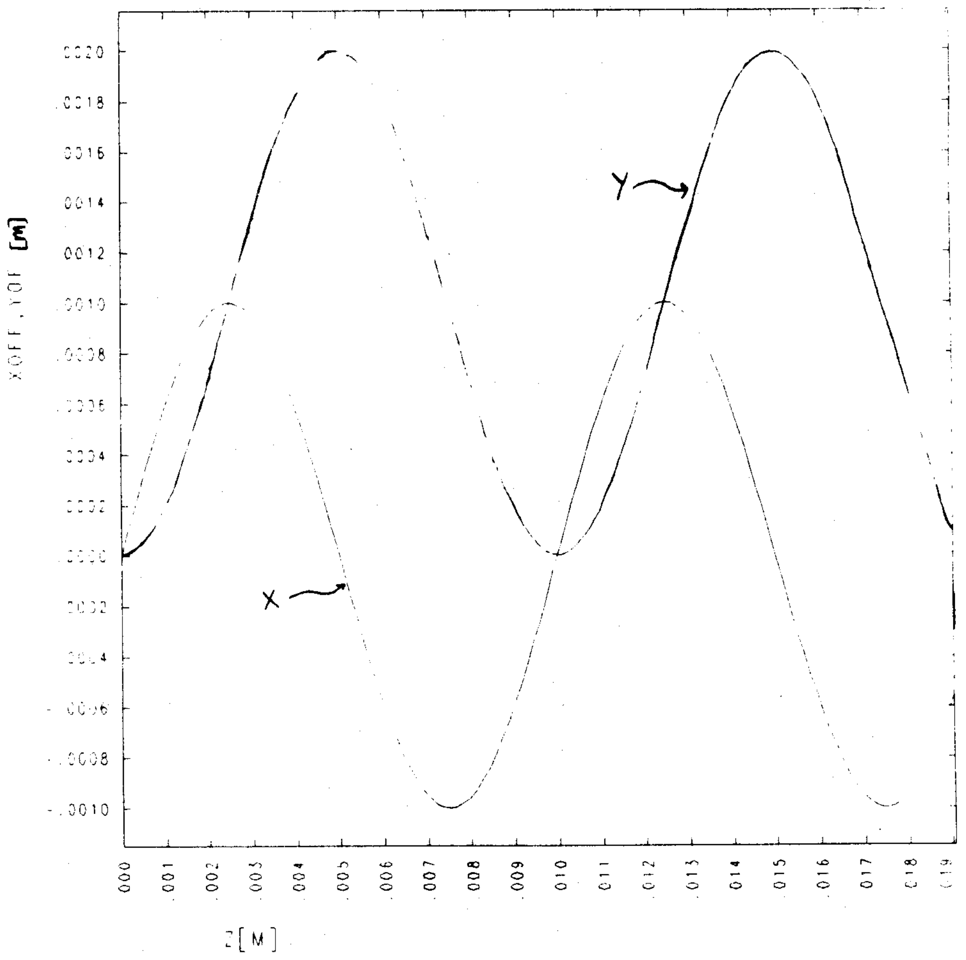


Figure 2.3: Particle orbit in an axial field with no space charge

These numbers produce the nice result of $r = .001\text{m}$ and $\lambda = .01\text{m}$. This is nicely confirmed by the results of the simulation.

2.4.2 Uniform Beam with no Magnetic Fields

Another case which can be analyzed analytically is that of a uniform beam moving in free-space with no magnetic fields. The code can easily simulate this with a beam on axis.[10] The electric field outside a uniform beam of charge per unit length Q is easily found from Gauss's law to be

$$E_r = \frac{Q}{2\pi\epsilon_0 r}. \tag{2.80}$$

This equation also applies at $r = r_{\text{beam}}$. We make the assumption that $p_\perp$ and $\beta_\perp$ are both approximately equal to zero. Then, using equation 2.37 and manipulating derivatives into a suitable form yields

$$rr'' = \frac{Qe}{2\pi\epsilon_0 m_e c^2} = 2\nu = K. \tag{2.81}$$

The factor $\nu = \frac{Qe}{4\pi\epsilon_0 m_e c^2}$ is called Budker's parameter. K is the perveance. Here I have been following Lawson (1977) [11]. We can use equations 2.3 and 2.54 to write K in terms of the current I:

$$K = \frac{Ie}{2\pi\epsilon_0 m_e c^3 \gamma}(1 - \frac{1}{\gamma^2})^{-\frac{3}{2}} \tag{2.82}$$

Equation 2.81 is not integrable in general. In the limit where $K << 1$ the solution can be taken to be

$$r^2 - Kz^2 = r_0^2 \tag{2.83}$$

where r is the outer radius of the beam and r_0 is the initial value of r. Since $K << 1$ it can be seen that $r \approx r_0$ in this limit. This condition is known as the paraxial limit.

[10] An off-axis beam would produce strange effects due to image charging on the beam tunnel walls. Another solution would be to run with an extremely large beam tunnel radius, but this is not practical in terms of computer time and quite unnecessary.

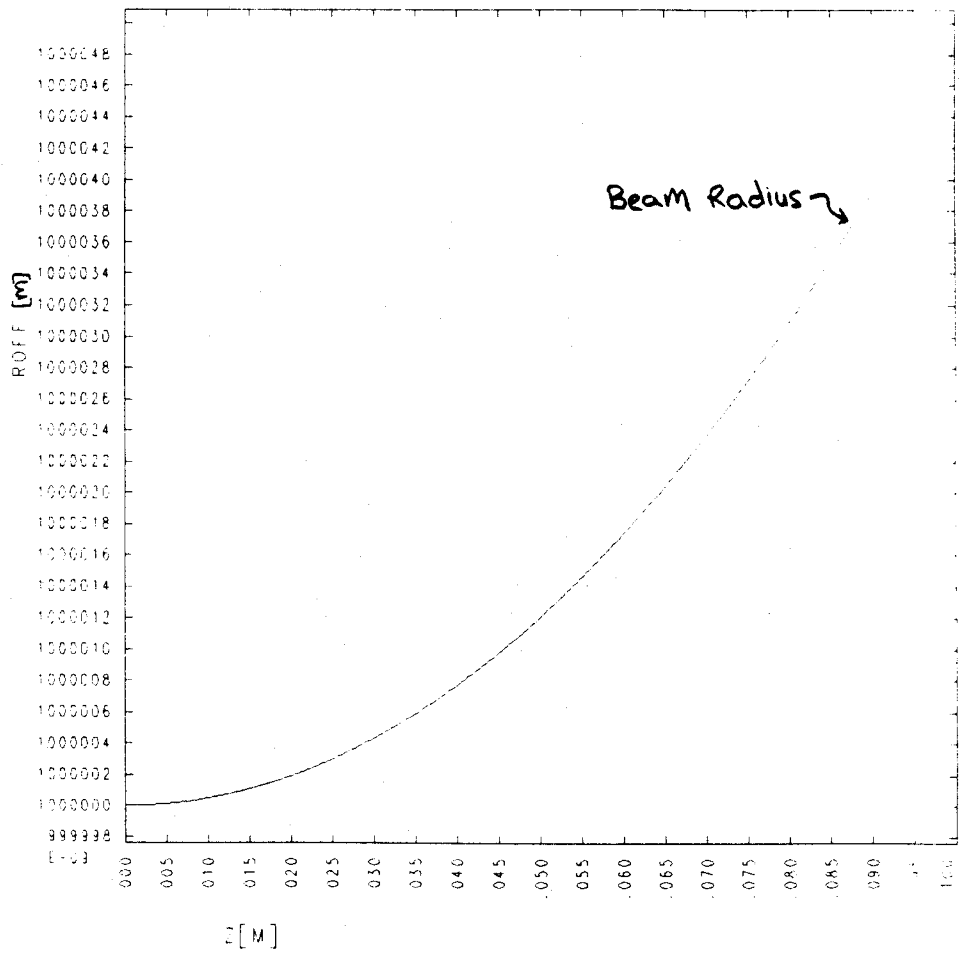


Figure 2.4: Hyperbolic beam spread in the paraxial limit

Figure 2.4 shows the result of a simulation for $K = 10^{-8}$ A V$^{-3/2}$, $\gamma = 4$, $r_0 = 1$ mm and $I = 3.0945 \times 10^{-4}$ amps. These numbers result in a perveance of $K = 10^{-8}$. At $z = 0.1$m we should find that $r - r_0 = 5.0 \times 10^{-8}$m and the results from the code are within 3% of that figure. The fact that there is any divergence at all is probably a result of numerical error.

2.4.3 Immersed Flow

A beam with space charge and no initial rotation which then moves in a uniform axial magnetic field is in a condition known as immersed flow. While no analytic solution exists for immersed flow, others have done numerical simulations and predicted various results which can be compared to those of my program. If we make the simplification that β_z is constant it is possible to derive the following differential equation for the outer electron of the beam, quoted from Brewer (1967) [4]:

$$\frac{d^2r}{d(\beta_p z)^2} - \frac{1}{2r} + r(\frac{\omega_H}{\omega_p})^2(1 - \frac{1}{r^4}) = 0 \tag{2.84}$$

where $\omega_p = \sqrt{\frac{Ie}{\epsilon_0 m_e \pi r_0^2 v_z}}$, $\omega_H = eB_z/2m_e$, and $\beta_p = \omega_p/v_z$. Note that this is a non-relativistic solution.

Brewer obtained solutions to this equation using an analog computer [3]. The particles undergo sinusoid-like oscillations around a radius slightly larger than r_0. The minimum and maximum radii can be calculated from the zeros of the integral of equation 2.84, itself a transcendental equation.

Figure 2.5 shows the result of a run of my code with $I = 0.01$ amps, $B_z = 4.9408$ Tesla, $r_0 = 1$ mm, and $\gamma = 1.001$ to insure a non-relativistic case. Initially there was no perpendicular momentum. First notice the important qualitative result that all the beam scalloping is outwards from the initial radius, a result which should always be true when there is no initial perpendicular momentum. The maximum outer particle excursion is 1.057 mm and the period of oscillation is 10 mm. These numbers are in

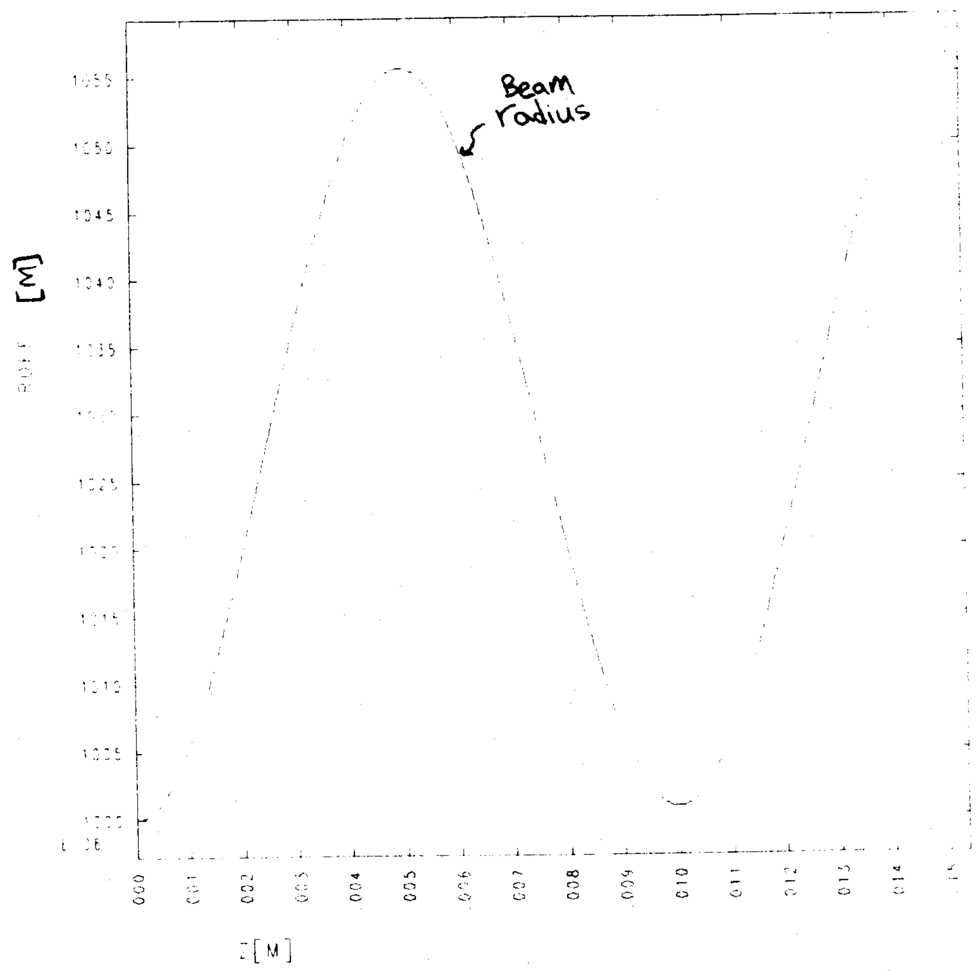


Figure 2.5: Immersed flow beam scalloping

fairly good agreement with Brewer's results of 1.07 mm maximum outer particle excursion and a period of 9.7 mm. There are several possible sources of the small discrepancies. Equation 2.84 assumes that β_z is constant; my code does not. This effect should be minimized but not eliminated in the small current limit. Another source of error may be Brewer's use of an analog computer over thirty years ago. An analog computer may well be less accurate than five or ten per cent. Perhaps the most important source of disagreement, however, is that I have only graphs of Brewer's results to work with, and have to estimate his numerical values by eye.

2.4.4 Brillouin Flow

When an electron beam enters a region of an axial magnetic field from a region of zero axial magnetic field, it will pick up a rotation given by $\omega = \omega_h$. The beam will be rotating like a rigid-rotator, i.e. each electron has a tangential velocity proportional to its radius, and has no radial velocity. It is also possible to show that $\omega_h/\omega_p = 1/sqrt2$ (see Brewer 1967 [4]). This condition is known as 'perfect' Brillouin flow. The particles should simply rotate around as they move forward in z, with no radial variation whatsoever.

Figure 2.6 shows the result of running my code with the following parameters, which should produce a rigid-rotor equilibrium: $I = 0.01$ amps, $B_z = 1.7468 \times 10^{-2}$ Tesla, $r_0 = 1\,\text{mm}$, and $\omega = \omega_h = 1.5362 \times 10^7$ radians/second. The beam envelope, despite small variations, is about what is expected. The beam just flows along and does not change size. However, the radius of the particle that initially had the largest radius varies wildly. It should simply stay at the edge of the beam, behaving as a rigid-rotator. Why it does not as well as how the beam envelope works out correctly without the individual particles behaving correctly, remains a mystery.

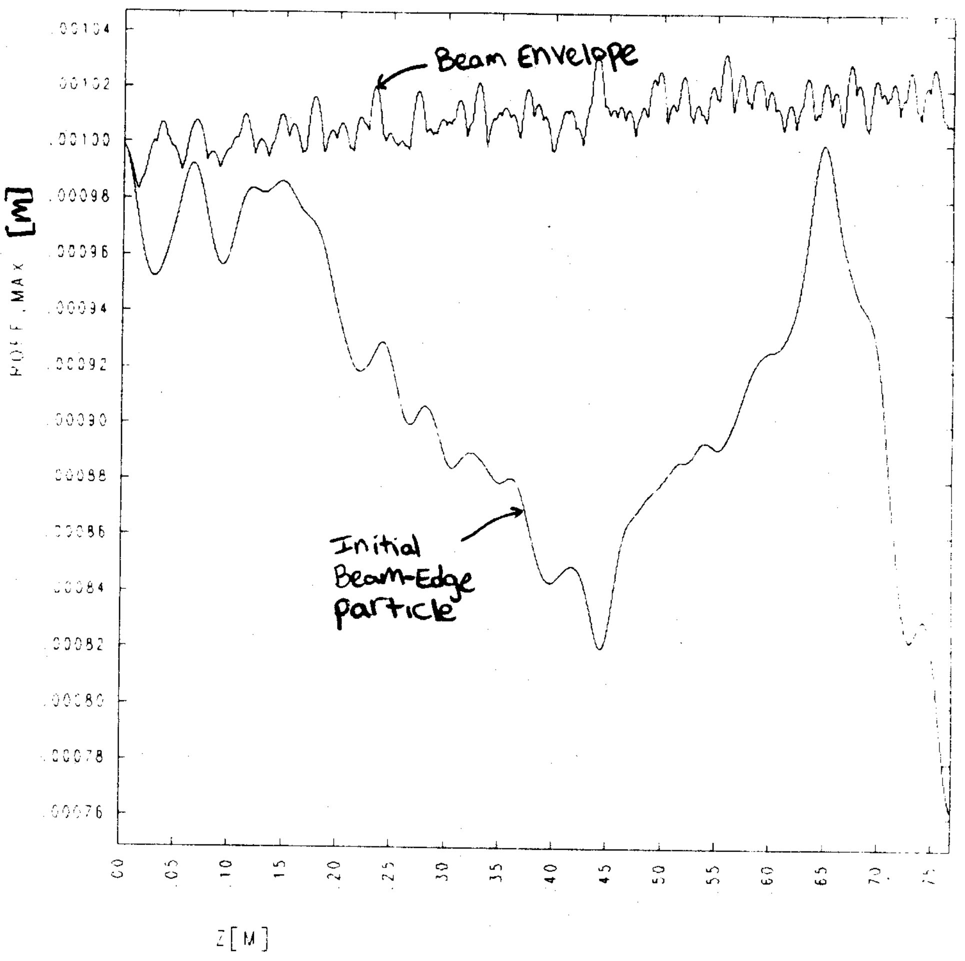


Figure 2.6: Brillouin flow beam envelope and initial off-axis particle radius. These are not the same for unknown reasons.

2.5 Simulation of an Actual Wiggler

My code has been used to model the wiggler for the 35 Ghz CARM at the MIT Research Laboratory of Electronics (RLE). I did simulations both with and without accounting for space charge effects. Figure 2.7 shows the magnitude of the tapered magnetic field profile for this wiggler, as calculated by my code using COIL3. There is also an axial guide field of 0.7 Tesla in the direction of increasing z. The beam tunnel wall has a radius of 0.8 cm and the wiggler helix has a radius of 2.24 cm and a period of 7 cm. The maximum current in the tapered wiggler is 3750 amps. The beam itself has a radius of 0.2 cm, and initially has a vanishingly small perpendicular momentum. The electrons in the beam begin with $\gamma = 4$. The beam current is 300 amps.

Figure 2.8 shows the calculated mean value of $\alpha = \beta_{\perp}/\beta_{\|}$ without accounting for space charge. Figure 2.9 show this value accounting for space charge. The space charge has increased the mean value of α from 0.32 to 0.34. It has also increased the spread in α, defined as the standard deviation of α over the average value of α, from less than 1.5% to nearly 24%. The spread with space charge can also be written as $\Delta\gamma_{\|}/\gamma_{\|} = 20\%$. This is a very large effect on the spread. In fact, according to work done by Bekefi, DiRienzo, Leibovitch, and Danly (1989) [1], a spread as large as the one calculated including space charge should result in CARM efficiencies for the RLE CARM of only about 3%. Ignoring the spread, which is about the same as ignoring space charge since most of the spread is due to the effects of space charge, they predicted efficiencies of up to 15%. Experimentally they have found the CARM efficiency with an older wiggler to be about 3%. Unfortunately my results indicate that the new wiggler will not perform significantly better. Space charge effects are clearly of primary concern in wiggler design.

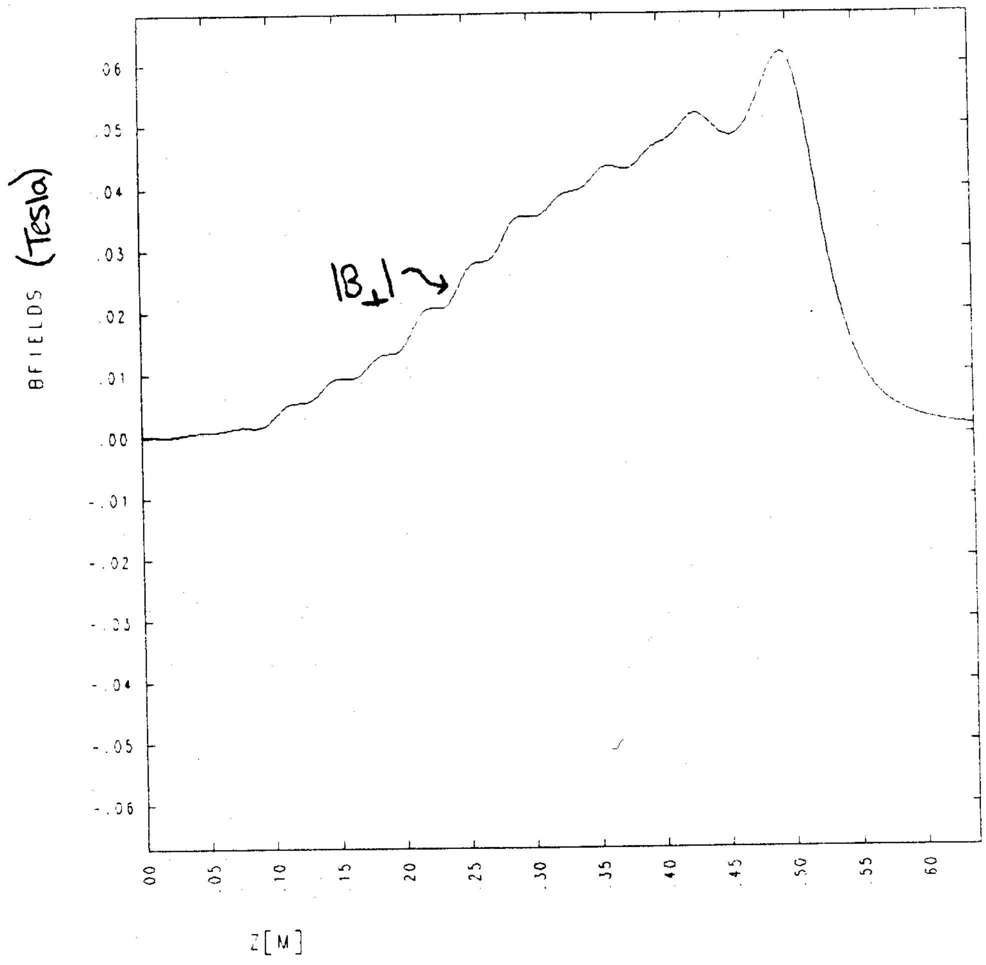


Figure 2.7: The magnetic field magnitude profile for the RLE CARM.

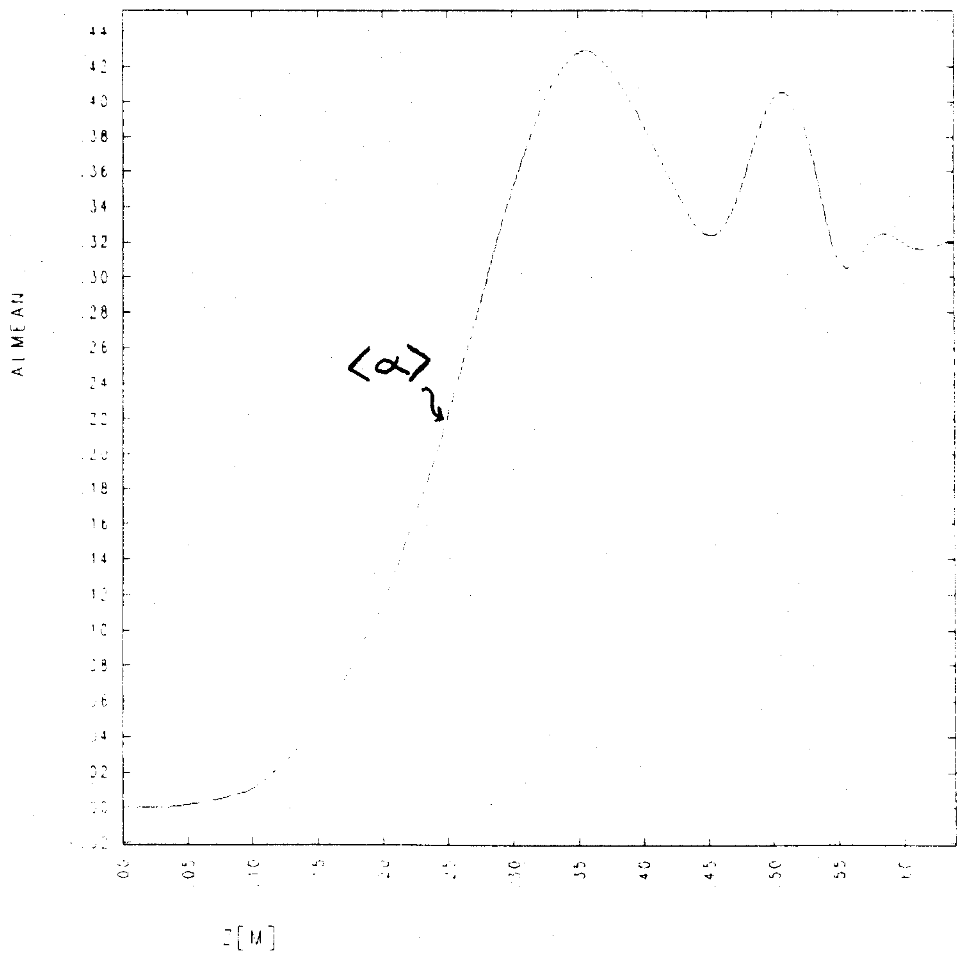


Figure 2.8: The mean value of α calculated without space charge effects.

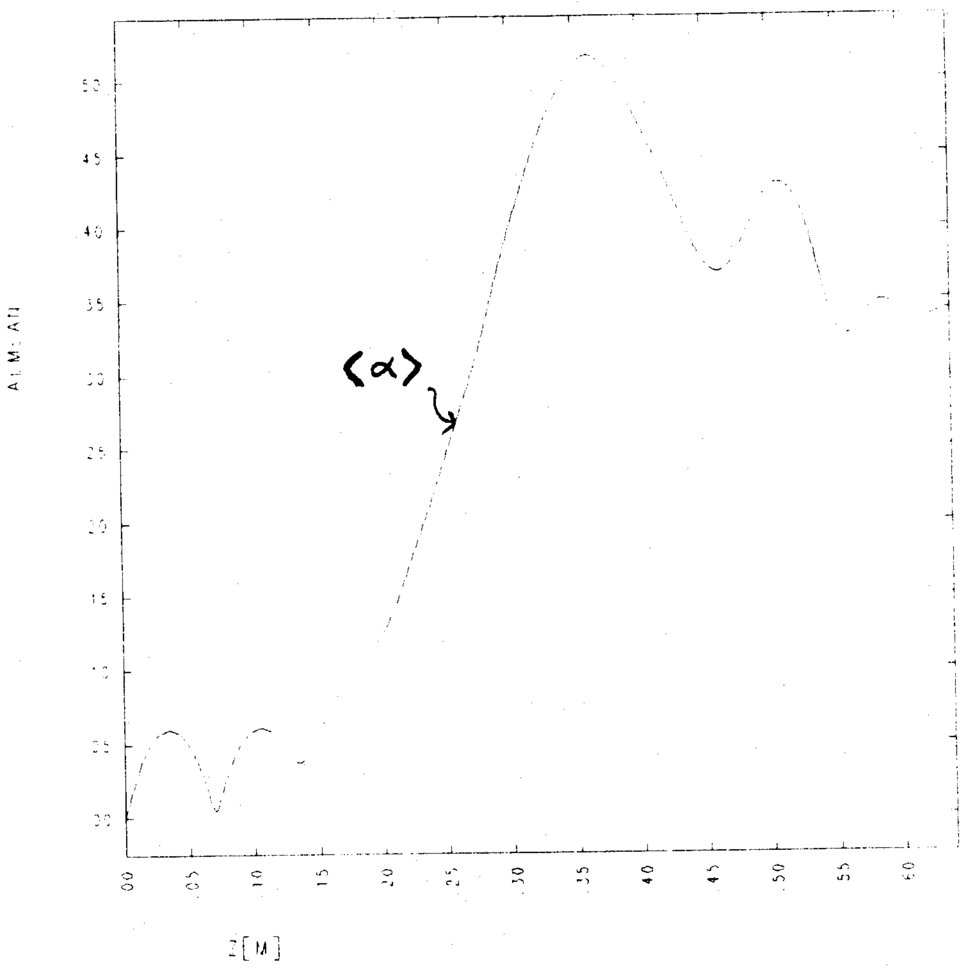


Figure 2.9: The mean value of α as calculated including space charge effects.

Chapter 3

A Capacitive Velocity Probe

Now that there is a code which can predict the motion of an intense relativistic (or non-relativistic) electron beam in a wiggler magnet, it is desirable to check the results of the program experimentally. Such experimental results are also useful in the absence of a code, but are all the more helpful when a simulation adds understanding to empirical data.

3.1 Theory of the Capacitive Velocity Probe

3.1.1 Basic Theory

One possible result to check is the average $v_{\|}$ of the beam. A probe to measure this has been described by Shefer, Yin and Bekefi (1983) [14]. The probe is essentially a cylindrical capacitor made out of two concentric cylinders (see Figure 3.1). The idea is that when a charged particle beam passes through the probe there will be a voltage induced between the two cylinders (later referred to as capacitor 'plates') due to the presense of charge inside. The voltage can be calculated by integrating the electric field between the plates.

$$V = -\int_a^b E_r dr \tag{3.1}$$

where a and b are the radii of the inner and outer plates. Now, assuming an infinitely long capacitor and an infinitely long beam with uniform charge density in z, we can use Gauss's law to calculate E_r:

$$E_r(r) = \frac{Q}{2\pi\epsilon_0 r}. \tag{3.2}$$

This equation is true everywhere outside the beam (in vacuum at least). Q is the charge per unit length in the beam. We can then plug into equation 3.1 and find:

$$V = -\int_a^b \frac{Q}{2\pi\epsilon_0 r} dr = \frac{Q}{2\pi\epsilon_0} \ln(b/a). \tag{3.3}$$

We can then solve for Q yielding

$$Q = \frac{2\pi\epsilon_0 V}{\ln(b/a)}. \tag{3.4}$$

We also know that the charge per unit length of the beam Q is related to the average parallel velocity $v_{||}$ by equation 2.3, repeated here:

$$I = Qv_z = Qv_{||}. \tag{3.5}$$

Solving for $v_{||}$ and plugging in the result of equation 3.4 for Q we find

$$v_{||} = \frac{I \ln(b/a)}{2\pi\epsilon_0 V}. \tag{3.6}$$

But we also know that the capacitance per unit length C of a cylindrical capacitor is given by

$$C = \frac{Q}{V} = \frac{2\pi\epsilon_0}{\ln(b/a)}. \tag{3.7}$$

This is easily derived if we take equation 3.4 to be the charge on the plates of the capacitor[1] and divide by V since the definition of capacitance is $C = Q/V$. We then have the nice result

$$v_{||} = \frac{I}{CV} \tag{3.8}$$

[1]Here we are simply placing a voltage V on the plates and noticing that the derivation of the charge

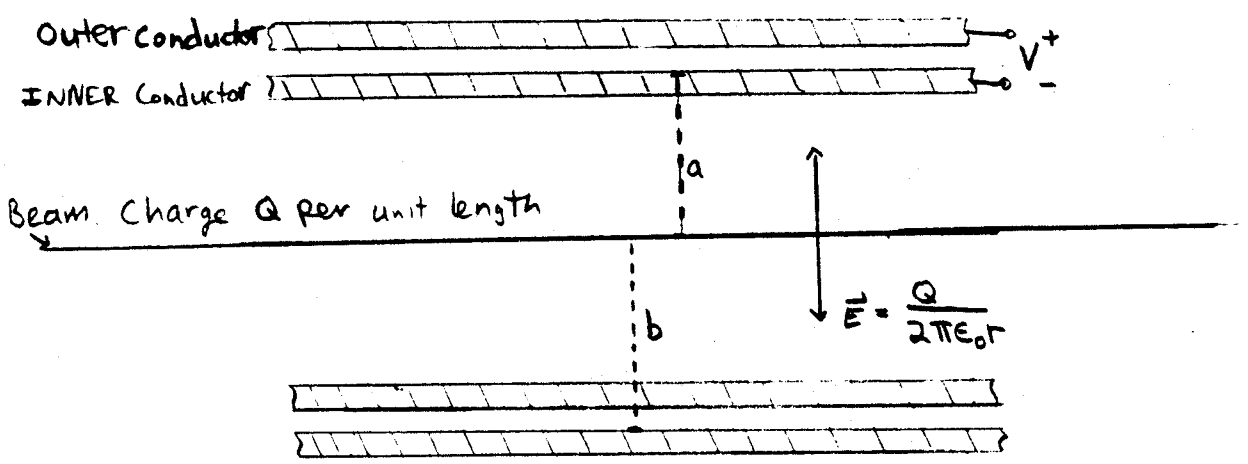

Outer conductor
INNER Conductor
V
+
-
a
Beam Charge Q per unit length
b
$\vec{E} = \frac{Q}{2\pi\epsilon_0 r}$

Figure 3.1: An infinitely long probe

or

$$V = \frac{I}{Cv_{\|}}. \tag{3.9}$$

One nice feature of this result is to notice that if we solve for V in terms of $v_{\|}$ the voltage is independent of the actual length of the capacitor, at least ignoring end effects. Thus a very short cylinder (a ring) will do and the probe need not take up much space in an experiment. The voltage is also independent of the cross section of the inner conductor, so long as its outer radius remains equal to a, because all of the derivation using Gauss's law remains unchanged.

To actually calculate $v_{\|}$ using such a probe will require an accurate knowledge of the current I and the capacitance C. The current can be measured using a collector and a current-viewing resistor after the beam has passed through the probe, or by using a Rogowski coil. The capacitance can be measured with a precision capacitance bridge. Note that the length of the capacitor must be also be known. Even though the final result is independent of the length, a capacitance bridge will measure the total capacitance which is proportional to the length. Thus we must divide by the length to eliminate this dependence and find the capacitance per unit length.

3.1.2 Other Considerations

The above discussion makes analysis of data from the probe seem extremely simple. All is not so simple. Terry Grimm, a graduate student at MIT, has built such a probe and found the analysis to be much less straightforward. One problem is that it is unclear how to handle end effects. For a capacitor of finite length the field lines from the beam

Q proceeds as it did when the voltage was induced by the beam rather than imposed by an external voltage source. In fact, the charge on the plates of the capacitor when a beam is passing though is the same as the charge in the beam. This is to satisfy Gauss's law both inside the inner plate (where $E = 0$ and just outside it where the flux must be the same as if the inner conductor didn't exist. Thus a charge of $-Q$ is induced on the inner surface of the inner plate and a charge of $+Q$ on the outer surface. This equivalence is what causes the nice result we are about to find.

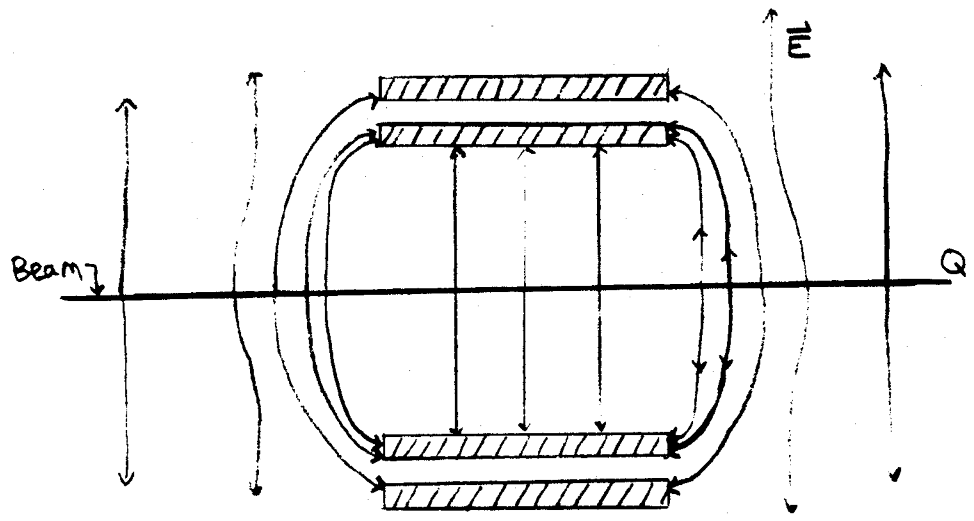


Figure 3.2: The bending of the field lines by a probe of finite length.

will bend in towards the ends of the probe as in Figure 3.2. This is because of the boundary condition at the surface of a conductor which states that the field lines must all be perpendicular to the surface. These additional field lines will distort the above discussion of Gauss's law by adding flux. This will certainly change the capacitance of the probe, and will perhaps have other effects on equations 3.8 and 3.9.

It is not clear whether the result of this field-line distortion will be simply that the actual value of the capacitance for the probe must be used instead of the theoretical result obtained for an infinitely long one, or if some other effects occur. The situation is further complicated by other conducting surface which may exist in a beam tunnel designed for an experiment other than to simply measure the velocity of a beam. It would seem that the basic functional form of equations 3.8 and 3.9 should be preserved to within a constant of proportionality based on the geometry of the probe and beam tunnel since the basic derivation depends on Gauss's law. The electric fields calculated from Gauss's law still must depend on the total charge enclosed in a gaussian surface. Hopefully then, there will be no effects on the voltage proportional to anything other than the charge (or $1/v_{\|}$). Even this, however, is unclear at this time.

Terry Grimm has been working on solving this problem analytically or numerically. A numerical solution is not wholly satisfactory for a probe which is supposed to provide an experimental rather than theoretical result. Another solution is to send through a beam with a known $v_{\|}$ to obtain a calibration. Such a beam can be generated with some electron guns by running at very low current. This will minimize space charge effects which might add perpendicular velocity and subtract parallel velocity.[2] Since the perpendicular velocity is very low and the gun accelerating voltage is known, $v_{\|}$ can easily be found. Naturally the wiggler field will have to be off while such a calibration is being done as a wiggler field would also add perpendicular velocity. In some experiments this is not practical due to the use of a fixed permanent magnet wiggler. Fortunately, however, it is usually possible to run with a γ far out of resonance with the wiggler so not much perpendicular velocity will be induced.

Another problem with the simple analysis of the preceding section is that the assumption that the beam is of uniform charge per unit length along its axis. In a wiggler region, not to mention in the interaction regions of many interesting types of devices (such as free electron lasers, gyrotrons, cyclotron autoresonance masers, etc.) the electron beam will be slowing down or speeding up. This necessarily results in a non-uniform charge per unit length due to equation 3.5. This effect could have significant implications on the accuracy of the capacitive velocity probe.

Still another consideration is that of the effect of the probe on the beam itself. Shefer, Yin, and Bekefi call the probe a "nonperturbing diagnostic," but this is not really true in

[2]This is a relativistic effect. If the perpendicular velocity increases the parallel velocity must decrease to avoid a resultant total velocity greater than the speed of light.

most practical applications. The inner conductor of the probe will generally reach several hundred volts or even higher with respect to the beam and beam tunnel walls (the walls are often at ground). The presence of such a region of high voltage may very well interfere with the beam. Even without high voltage the geometry of the probe will be important. If it is not carefully designed it could act as a resonant cavity for a spinning electron beam wreaking havoc with the expected behavior of the device it is in. Probably both of these effects can be minimized with careful design of the probe, but the probe is by no means non-interfering under all conditions. A discussion improvements to the design of such a probe beyond those outlined above is beyond the scope of this paper.

3.2 Design of a Probe for the Gyro-BWO

Figure 3.3 shows the design drawings of a capacitive velocity probe for Bill Guss's Gyro-Backwards Wave Oscillator (Gyro-BWO) experiment. The stainless steel ring is the inner conductor of the capacitor. It fits inside the larger piece of macor. The smaller piece of macor fits in the end of the larger piece as an end cap, providing insulation. Corning macor ceramic was chosen for the insulating pieces due to its high dielectric strength ($\sim 1000 - 3000$ volts/mil, depending on the frequency of the applied voltage), its relative machinability, and its fairly constant dielectric constant over a wide range of frequencies (5.92 at 10kHz, 5.68 at 8.6GHz).

The outer conductor of the capacitor is a tube, much longer than the rest of the probe. The tube is actually the beam tunnel which holds beam scrapers as well. The beam scrapers are designed to minimize the chance of the beam finding a resonant cavity before it reaches the Gyro-BWO interaction region. The beam scrapers alternate between insulating macor and conducting stainless steel. The macor parts of the probe are designed to have the same shape as the macor beam scrapers. The entire probe is designed to take the place of three beam scrapers in the beam tunnel (two macor scrapers

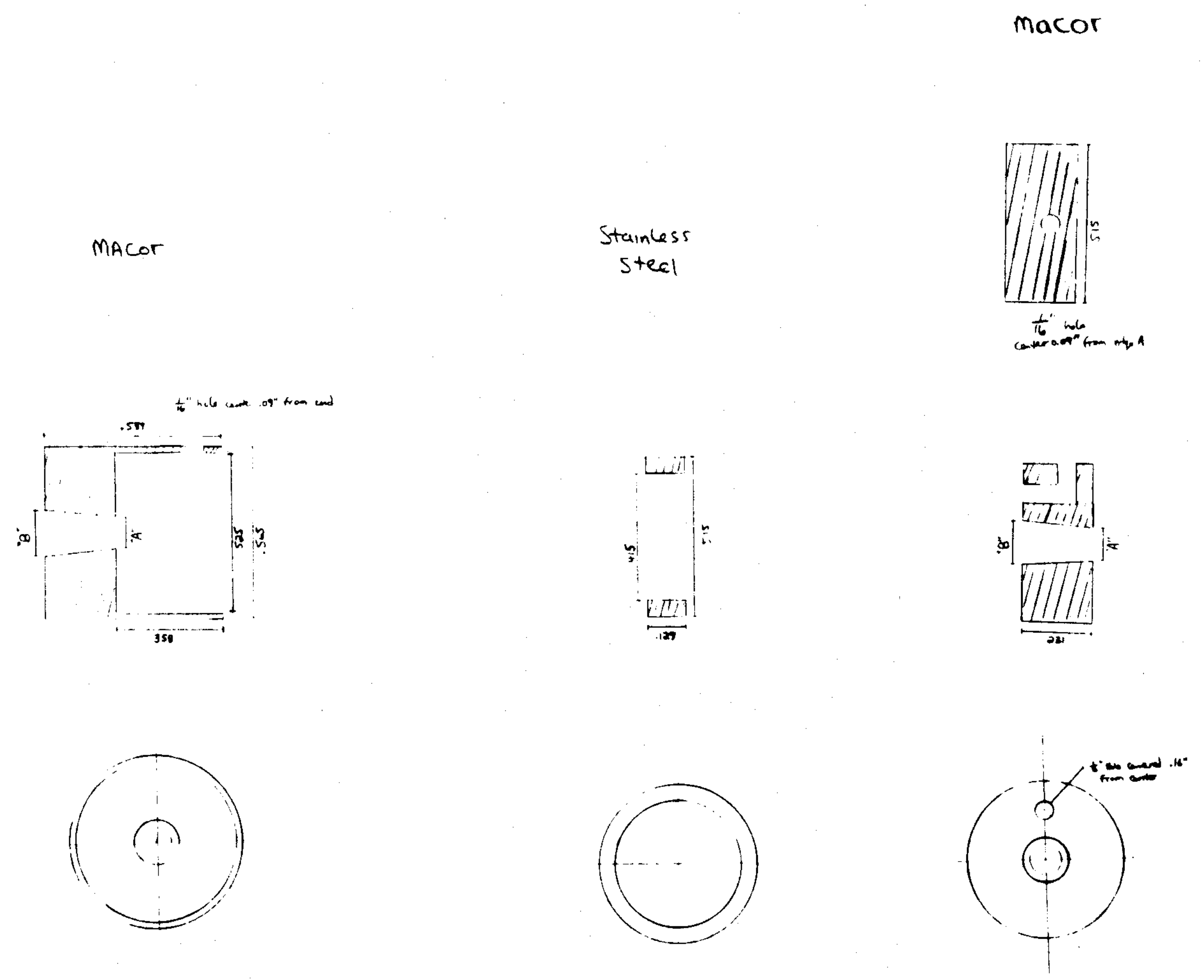


Figure 3.3: Design Drawings of Probe for the Gyro-BWO

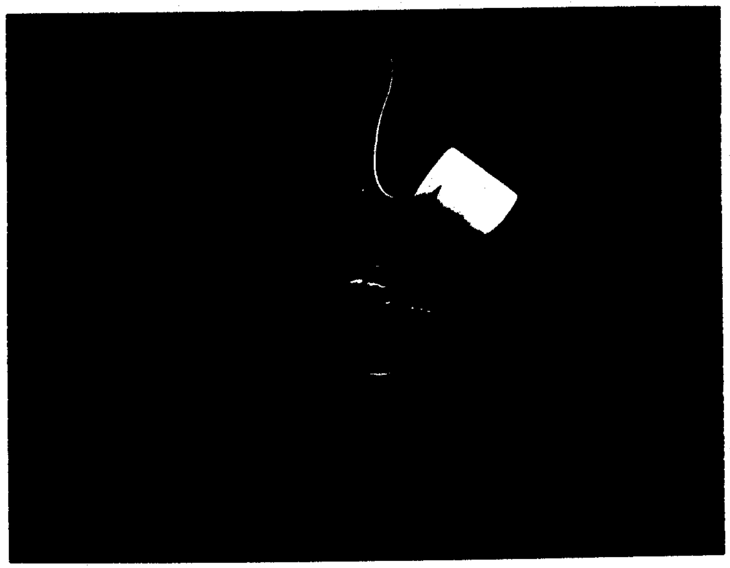


Figure 3.4: The Probe for the Gyro-BWO. The nickel shows scale.

and the stainless steel scraper between them).

A thin, single conductor teflon wire is spot-welded onto the inner conductor of the probe. It runs through the axial hole in the smaller macor pieces, turns and runs out radially, and then goes through the matching radial hole in the larger macor piece. It next runs down a slot in beam tunnel until it finally reaches a vacuum feed-through mounted in a flange. It is spot-welded to the copper conductor of the feed-through. Spot-welding was chosen instead of solder to avoid outgassing which could ruin the high vacuum. The teflon insulation on the wire was chosen for the same reason. Figure 3.4 is a photograph of the assembled probe before it is put in the beam tunnel. The nickel is there for size comparison.

The Gyro BWO runs with $\gamma = 1.2$ and a current of about 4 amps. Using equation 2.54, and assuming that $\vec{\beta} = \vec{\beta}_{\|}$ we find that $v_{\|} = 1.66 \times 10^{8} m/s$. Solving equation 3.6 for V yields

$$V = \frac{I \ln(b/a)}{2\pi \epsilon_0 v_{\|} \epsilon_{\mathrm{macor}}}. \tag{3.10}$$

The ϵ_{macor} in the denominator is the dielectric constant of macor, which has been added because the capacitance of a capacitor with a dielectric between the capacitor plates has its capacitance multiplied by the dielectric constant of the material.

From Figure 3.3 we can see that $a = 0.258$ inches $= 0.654\,\text{cm}$ and $b = 0.288$ inches $=$ $.730\,\text{cm}$. The expected voltage is then found to be $V = 8.2$ Volts. This answer is only approximate since it assumes the probe is infinitely long and suffers from the all the problems discussed above. Still, it can serve as a guide in designing the probe. The probe was designed to have the capacitor plates quite close together to keep the voltage down to a minimum. A mistake made when converting from CGS to MKS units caused the mistaken belief that the voltage would be 100 times higher than it is. The probe was designed with this in mind, causing the macor piece to be much thinner than it needs to be. Future probes need not have such thin walls which should facilitate their manufacture and improve structural strength. Strength is an important consideration, as will be seen in the next section.

Another consideration for a real probe is that of the capacitance of the electric leads. The capacitance of the leads of order 10 picofarads/foot. The capacitance of the probe is given by the result of equation 3.7 multiplied by the length of the probe in meters. The inner conductor measures 0.127 inches or 0.323 cm. The capacitance is found to be approximately 1.6 picofarads. This is smaller than that of even very short leads! This is a major problem, especially considering that it will be nearly impossible to measure the capacitance of just the probe since leads will have to connect it to the capacitance bridge. It is even harder to measure this quantity with the probe installed in the beam tunnel which is where it really should be measured since the beam tunnel geometry will have a perhaps major effect on the probe's capacitance, as discussed above.

What now is the expected voltage at the end of the leads? Charge will flow off the probe and onto the leads, which behave as a capacitor in parallel to the probe. The measured voltage at the ends of the leads will simply be the result of distributing a charge Q over the combined capacitance of the system. Thus the expected voltage is simply multiplied by the factor $C_{\text{probe}}/(C_{\text{probe}} + C_{\text{leads}})$. This voltage will appear after an RC time constant has allowed the original expected voltage to fall off exponentially as charge flows onto the leads. The time constant, however, is very small since the capacitances involved are on the order of picofarads. The time constant will be less than a 10 picoseconds if we take the resistance of the leads to be less than about 1 ohm. Most devices operate on pulse lengths significantly longer than this.

3.3 Problems with the Probe

The probe was built, thus beginning a long series of mishaps (well, two anyway). The first problem was that as drawn there is no way to assemble the probe. Once the wire is attached to the inner conductor and threaded through the holes in the two macor pieces there is no way to insert the smaller macor piece into the larger. The wire gets mashed between them. To solve this problem a notch was filed carefully from the edge of the larger piece of macor to the radial hole. Unfortunately, this apparently weakened the piece such that when it was placed in the ultrasonic cleaner (which was used to keep the probe extremely clean before installation in the high vacuum of the Gyro BWO beam tunnel) it developed a crack from the radial hole outward in a sort of a bite-like shape. The crack is clearly visible in Figure 3.5. Fortunately the damage should have little effect on the functioning of the probe. It is in such a location as to not decrease the probe's resistance to arcing since the easiest path is the one the wire takes and all path lengths through the cracked region are at least as long.

The other main problem with the probe at the time of this writing is that the feed-through flange had a leak. When the whole assembly was installed, the system would not stay at high vacuum and the leak was eventually traced to the weld which connects the feed-through to the flange. The flange has not yet been successfully repaired. It is

Figure 3.5: The Probe for the Gyro-BWO showing the cracked area.

unlikely that it will be completed in time for useful data to appear in this document.

The experience gained from the design of this probe should help with the design of future probes. Probes may eventually be installed on the 35 GHz CARM at the MIT Research Laboratory of Electronics and the 140 GHz CARM at the MIT Plasma Fusion Center.

Appendix A

The Need for Space Charge

A good question to ask is whether space charge will be of importance at all in a wiggler magnet where one might expect the magnetic field interaction to eclipse any other effects. However, the space charge forces are proportional to the current density of the beam, and many free electron devices are run at extremely high currents over small areas.

Our main equation of motion is the Lorentz force equation, equation 2.36, repeated here:

$$\frac{d\vec{p}}{dt} = q(\vec{E} + \vec{v} \times \vec{B}). \tag{A.1}$$

When the first term is of comparable magnitude to the second space charge effects will be important. The magnetic fields in a wiggler magnet are typically around 0.1 Tesla and the electrons are traveling close to the speed of light. The second term will then have a magnitude of order 5×10^{-12} Newtons.

The electric field at the edge of a uniform beam with charge per unit length Q was given by equation 2.80 to be

$$E_r = \frac{Q}{2\pi\epsilon_0 r_{\text{beam}}}. \tag{A.2}$$

The can be put in terms of the current by using equations 2.3 and 2.54, yielding

$$E_r = \frac{I}{2\pi\epsilon_0\beta_z c} = \frac{I}{2\pi\epsilon_0 c\sqrt{1-(1/\gamma)^2}} \tag{A.3}$$

if we assume $\beta_\perp = 0$. A typical beam might have a radius of 1mm, $\gamma = 4$ and a current of 100 amps. The first term of equation A.1 then has an order of magnitude of 10^{-12} Newtons as large as the factor from the magnetic field contribution. Clearly such a large factor should not be ignored.

Appendix B

The Actual Code

The following is the actual code. It is written as Historian source code. Historian is a program which preprocesses FORTRAN source to ease the burden of writing common blocks repeatedly and to facilitate making changes. The code needs to be linked to the following libraries: SLATEC, TV80LIB, DISSPLA, and COIL3. TV80LIB and DISSPLA are needed for the graphics output which may not be compatible with all machines. COIL3 is probably not available on all machines.

```
*cd param
c-----------------------------------------------------------------------
        implicit complex(c)
        parameter(nzmax=8192, npmax=512, pi=3.14159265359,
     +  esubmc=586.6655,twopi=2.*pi,nhpmax=40,
     +  numrmax=128,nthmax=128,esubmc2=1.95751e-06,
     +  epsilon0=8.854187818e-12,echarge=1.6021892e-19,
     +  lspeed=2.99792458e+08)
*cd combla
c-----------------------------------------------------------------------
        common//  xarry(3,npmax), barry(3,npmax), px(npmax),
     +            py(npmax), pz(npmax), alpha(npmax),pp(npmax),
     +        betap(npmax),gammas(npmax),acurrent(npmax)
*cd comwig
c-----------------------------------------------------------------------
        common/comwig/  wlamda, nwigg, rhelix, xcur, xcur0, nseg, bz0,
     +                  epsbz,  xlamdf, f(nhpmax), xcs(nhpmax),
     +  xch(nhpmax),phiwall,rpipe
*cd combea
c-----------------------------------------------------------------------
        common/combea/ npart,rbeam,gamma,delpp,ild,beamcur,omega
*cd comint
c-----------------------------------------------------------------------
       common/comint/ zmin,delz,nz,z0,z1,bint1,bint2,
     +     numr,ntheata
*cd comdia
c-----------------------------------------------------------------------
        common/comdia/ zp(nzmax), bx(nzmax), by(nzmax), bperp(nzmax),
     +                 bxref(nzmax), byref(nzmax), bpref(nzmax),
     +                 xref(3,1), bref(3,1), ipaxis, ipoff,
     +                 palpha(nzmax), ppz(nzmax), pdelal(nzmax),
     +                 bzmean(nzmax), bpmean(nzmax), pdelbp(nzmax),
     +                 pdelpz(nzmax), pxaxis(nzmax), pyaxis(nzmax),
     +                 pxoff(nzmax), pyoff(nzmax), rgaxis(nzmax),
     +                 rgoff(nzmax), pdelbz(nzmax), bpaxis(nzmax),
     +                 bpoff(nzmax), thaxis(nzmax), thoff(nzmax),
     +                 rax(nzmax), roff(nzmax), thzbar(nzmax),
     +                 thtar1(nzmax), thtar2(nzmax), xxm(nzmax),
     +  yym(nzmax),pertrbs(nzmax),phi(numrmax),phi2(nthmax+1),
     +  esave(numrmax),phi3(nthmax+1),beamenv(nzmax),beamenip(nzmax),
     +  ptrace(10,2,nzmax),ntraces(10),ptemp(nzmax),ntrace
*dk main
c
c  WIP-----Wiggler Integration Program
c
c Authors:
c       T.M. Tran    Orginal Version, August 1987
c       B.G. Danly and K.D. Pendergast,
c                    Revisions: Wiggler shunts, more graphics
c       J.A. Smolin  Major Revision, April 1989:
c                       Self-electric fields,
c                       self-consistant space charge calculation
c------------------------------------------------------------------
c Runs on Cray II and Cray XMP
c Compile with CFT77 version 2.X until version 3.0 is upgraded
c (version 3.0 works on Cray II, hangs on XMP)
c Link to the following libraries:  SLATEC, TV80LIB, DISSPLA, COIL3.
c COIL3 by C.F.F. Karney should be available somewhere at NMFECC.
c
```

```
c--------------------------------------------------------------
c    Electron trajectories in the wiggler field
c
*ca param
*ca combla
*ca comwig
*ca combea
*ca comint
*ca comdia
c----------------------------------------------------------------------
        character*1   lmore
        dimension  cy0(2*npmax), cy1(2*npmax), cyp1(2*npmax),
     +       cyp2(2*npmax),pzp1(npmax),pzp2(npmax),pzold(npmax)
        namelist /newrun/ wlamda, rhelix, xcur, nseg, gamma,
     +                   bz0, epsbz,  xlamdf, npart, rbeam, ild,
     +      delpp,zmin,delz,nz,f,nhalfp,nplots,ntrace,
     +      numr,ntheata,spacchrg,phiwall,rpipe,beamcur,
     +      omega
c----------------------------------------------------------------------
c
      CALL LINK(  "i=tty, unit5=(i,open),
     +             o=tty, unit6=(o,hardcopy),
     +             e=tty, unit59=(e,text)//" )

       call gfsize (3,127000)
       call fr80id ('film-only',0,1)
       call keep80 (1,3)
       call dders (-1)

c----------------------------------------------------------------------
cl                        1.  Read input
c
c...Default input values
c    Wiggler
        wlamda=0.06
        nhalfp=13
        do 23 i=1,nhpmax
  23    f(i)=0.
        rhelix=0.02
        xcur=400.
        nseg=32
        bz0=0.38
```

```
          epsbz=0.0
          xlamdf=1.0
c      Electron beam
          npart=32
          rbeam=0.005
          gamma=2.37
          delpp=0.001
          ild=0
          omega=0.
          beamcur=4.
c      Integration parameters
          zmin=-0.08
          delz=0.005
          nz=128
          nplots=0
            ntrace=0
c        Poisson solver parameters
            numr=64
            ntheata=64
            phiwall=0.
            rpipe=0.
c         spacchrg=0. means no space charge calculated
c         spacchrg .NE. 0. means space charge calculated
            spacchrg=1.
   10     continue
c
c     Read input data
          read(5,newrun)
          iflag=0
          do 11 i=1,nhalfp
             if(f(i).ne.0) iflag=1
   11     continue
          if(iflag.eq.0) f(1)=1.
          if(iflag.eq.0) f(nhalfp)=-1.
          write(6,newrun)

          nwigg=(nhalfp-1)/2

          beta=sqrt(1.-1./gamma**2)
          bt=107.174*gamma*beta/wlamda

c------------------------------------------------------------------------
```

```
cl                         2. Define wiggler
c
c   Set units
        call setunt('B')
c
c   Define the (const.) guide field
        call dfcnst(0.,0.,bz0)
c
c   Define the wiggler geometry
c
        zt=0.
        xcs(1)=xcur*f(1)
        xch(1)=xcs(1)
        call termin(zt,rhelix,xcs(1),nseg,0.)
        call dhelix(zt,rhelix,wlamda,nwigg,nseg,0.,xch(1))
        call dhelix(zt,rhelix,wlamda,nwigg,nseg,pi,-xch(1))
        zt=zt+wlamda/2.
        do 51 i=2,nhalfp-1,2
        xcs(i)=xcur*f(i)
        xch(i)=xch(i-1)+xcs(i)
        if(f(i).eq.0.) goto 40
        call termin(zt,rhelix,xcs(i),nseg,pi)
   40   call dhelix(zt,rhelix,wlamda,nwigg,nseg,0.,xch(i))
        call dhelix(zt,rhelix,wlamda,nwigg,nseg,pi,-xch(i))
        zt=zt+wlamda/2.
c
        xcs(i+1)=xcur*f(i+1)
        xch(i+1)=xch(i)+xcs(i+1)
        if(f(i+1).eq.0.) goto 41
        call termin(zt,rhelix,xcs(i+1),nseg,0.)
   41   if(xch(i+1).eq.0.) goto 50
        call dhelix(zt,rhelix,wlamda,nwigg,nseg,0.,xch(i+1))
        call dhelix(zt,rhelix,wlamda,nwigg,nseg,pi,-xch(i+1))
   50   zt=zt+wlamda/2.
   51   continue

c------------------------------------------------------------------------
cl                         3.  Initialization
c
c   Initialize the particles
        call inital
        call pack(cy1)
```

```
c--------------------------------------------------------------------
cl                      4.  Integration loop
c
c   Integrate along z-direction
       delzn=twopi/wlamda * delz
          delzh=0.5*delzn
       cdelz=cmplx(delzn,0.)
       cdelzh=0.5*cdelz
       npt2=2*npart
       nplt=1000
       if(nplots.ne.0) nplt=int(nz/nplots)
       iiz=1
c
       zp(1)=z1
       do 99 ij=2,nz
          zp(ij)=zp(ij-1)+delz
  99   continue
       do 100 iz=1,nz-1
         if(iiz.ge.nplt.and.nplots.ne.0) then
       call psplot(iz,1)
       call psplot(iz,2)
       call psplot(iz,3)
       call psplot(iz,4)
            iiz=0
         end if
         iiz=iiz+1
         call diagno(iz)
         z0=z1
         z1=z0+delz
         call ccopy(npt2,cy1,1,cy0,1)
          call ccopy(npart,pz,1,pzold,1)
          call parsim(z0,cy1,cyp1,pzp1)
          if (spacchrg .eq. 1.) then
             call poisson(z0,iz,cyp1)
          endif
         call caxpy(npt2,cdelz,cyp1,1,cy1,1)
          do 105 ip=1,npart
             pz(ip)=pz(ip)+delzn*pzp1(ip)
 105      continue
         call unpack(cy1)
         call bfield(z1)
          call parsim(z1,cy1,cyp2,pzp2)
         if (spacchrg .eq. 1.) then
            call poisson(z1,iz,cyp2)
         endif
         do 110 i=1,npart
            cy1(i)=cy0(i)+cdelzh*(cyp1(i)+cyp2(i))
            cy1(i+npart)=cy0(i+npart)+cdelzh*(cyp1(i+npart)+
     +       cyp2(i+npart))
            pz(i)=pzold(i)+delzh*(pzp1(i)+pzp2(i))
 110     continue
        call unpack(cy1)
        call bfield(z1)
  100   continue
        call diagno(nz)
c------------------------------------------------------------------------
cl                        9.  Output
c
c   Ouput on listing
        write(6,'(2(a,f12.5))') ' Betaz =',ppz(nz),
     +          '  rel. spread [%] =',pdelbz(nz)*100.
        write(6,'(2(a,f12.5))') ' Alpha =',palpha(nz),
     +          '  rel. spread [%] =',pdelal(nz)*100.
        write(6,200) 'transition field  BT = ',bt,'[G]'
  200   format (10x,a23,f8.2,1x,a3)
        call cputim
c   Graphics
        call dhist('Z[m]','xoff,yoff',pxoff,pyoff)
c       call histry('Z[m]','pertrbs',pertrbs)
        call plotgen(phi,numr,0.,rpipe,'R[m]','phi(R)')
        call plotgen(phi2,ntheata+1,0.,twopi,'theta','phi2(R)')
        call plotgen(phi3,ntheata+1,0.,twopi,'theta','phi3(R)')
        call plotgen(esave,numr,0.,rpipe,'R[m]','eperp(R)')
        call plotfp('Z[m]','Bfields')
        call histry('Z[m]','bpmean',bpmean)
        call histry('Z[m]','pdelbp',pdelbp)
        call histry('Z[m]','bzmean',bzmean)
        call histry('Z[m]','pdelbz',pdelbz)
        call histry('Z[m]','almean',palpha)
        call histry('Z[m]','pdelal',pdelal)
        call dhist('Z[m]','bpax,bpoff',bpaxis,bpoff)
        call dhist('Z[m]','rgax,rgoff',rgaxis,rgoff)
        call dhist('Z[m]','thax,thoff',thaxis,thoff)
        call dhist('Z[m]','x,y mean',xxm,yym)
```

```
       call dhist('Z[m]','rax,roff',rax,roff)
       call dhist('Z[m]','roff,maxr',roff,beamenv)
       call histry('Z[m]','Beam Envelope',beamenv)
       call histry('Z[m]','ipMax',beamenip)
       if (ntrace.NE.0) then
          do 350 i=1,ntrace
            do 300 j=1,nz
               ptemp(j)=sqrt(ptrace(ntraces(i),1,j)**2+
     +           ptrace(ntraces(i),2,j)**2)
 300        continue
            call histry('Z[m]','rtrace',ptemp)
 350      continue
       endif


c   Call graphics terminating routines

       call donepl

       call exit(1)

       end
*dk dhelix
c====================================================================
       subroutine dhelix(zw1,rhelix,wlamda,nwigg,nseg,zphi,xcur)
c====================================================================
c   Define a helical current
c
*ca param
       real cur
c------------------------------------------------------------------------
c   Initialization
       xkw=2.*pi/wlamda
       delzw=wlamda/nseg/2.
       zb=zw1
       xb=rhelix*cos(xkw*zb-zphi)
       yb=rhelix*sin(xkw*zb-zphi)
c
c   Define the helix
       do 100 iz=1,nseg
          za=zb
          xa=xb
```

```
             ya=yb
             zb=za+delzw
             xb=rhelix*cos(xkw*zb-zphi)
             yb=rhelix*sin(xkw*zb-zphi)
             call dfbara(xa,ya,za,xb,yb,zb,xcur)
  100   continue
c------------------------------------------------------------------------
        return
        end
*dk termin
c====================================================================
        subroutine termin(z,rhelix,xcur,nseg,thb)
c====================================================================
c
c   Termination loop
c
*ca param
        real cur, ccur
c------------------------------------------------------------------------
        dth=2.*pi/nseg
        xb=rhelix*cos(thb)
        yb=rhelix*sin(thb)
        cur=-xcur/2.
        do 100 i=1,nseg
          tha=thb
          xa=xb
          ya=yb
          thb=tha+dth
          xb=rhelix*cos(thb)
          yb=rhelix*sin(thb)
          call dfbara(xa,ya,z,xb,yb,z,cur)
          if(i.eq.nseg/2)  cur=-cur
  100   continue
        return
        end
*dk inital
c====================================================================
        subroutine inital
c====================================================================
c   Initialize the particles
c
*ca param
```

```
*ca combla
*ca comwig
*ca combea
*ca comint
*ca comdia
c------------------------------------------------------------------------
        dimension p1(2*npmax), p2(2*npmax)
c------------------------------------------------------------------------
c set up radius of beam tunnel
        if(rpipe.EQ.0.) then
          rpipe=rhelix
        endif
c   Load a homogeneous cylindrical electron beam
        if(ild.eq.0) then
c load in a square, then only keep those in circle
c  ratio of area of square to area of circle is 4/pi
        ratio=4./pi
c npload is number of paticles loaded into square
        npload=nint(ratio*npart)
        call loduni(1,npload,p1)
        call loduni(2,npload,p2)
        do 100 i=1,2*npart
          p1(i)=2.*p1(i)-1.
          p2(i)=2.*p2(i)-1.
  100   continue
        ip=0
        do 110 i=1,npload
          r2d=p1(i)**2+p2(i)**2
          if(r2d.lt.1.)  then
            ip=ip+1
            xarry(1,ip)=rbeam*p1(i)
            xarry(2,ip)=rbeam*p2(i)
          end if
  110   continue
        npart=ip
        write(6,*) 'Actual number of particles used ',npart
        else
           xarry(1,1)=0.
           xarry(2,1)=0.
           xarry(1,2)=rbeam/sqrt(2.)
           xarry(2,2)=rbeam/sqrt(2.)
        end if
```

```
c  Initialize all particle gammas to gamma
        do 120 i=1,npart
           gammas(i)=gamma
 120    continue
c
c   Fluctuations in (px,py)
        if(omega.NE.0.0) then
c Rigid-Rotor loading
c
c namelist parameter omega is rotation rate of beam
c all particles get bz determined by gamma and particle on axis
c which is the paticle with zero perp momentum
        bztemp=sqrt(1-(1/gamma)**2)
        do 175 i=1,npart
             bxtemp=-xarry(2,i)*omega/lspeed
             bytemp=xarry(1,i)*omega/lspeed
             gammas(i)=1/sqrt(1-bxtemp**2-bytemp**2-bztemp**2)
             px(i)=gammas(i)*bxtemp
             py(i)=gammas(i)*bytemp
             pz(i)=gammas(i)*bztemp
 175       continue
        else if(delpp.eq.0.0) then
c            Zero perp. velocity beam
            call resetr(npart,px,0.0)
            call resetr(npart,py,0.0)
            call resetr(npart,alpha,0.0)
        do 150 i=1,npart
           pz0=sqrt(gammas(i)**2-1.0)
           pz(i)=pz0
 150    continue
        else
           call loduni(3, 2*npart, p1)
           call loduni(5, 2*npart, p2)
           do 200 i=1,2*npart
             p1(i)=2.*p1(i)-1.
             p2(i)=2.*p2(i)-1.
  200      continue
           ip=0
           do 210 i=1,2*npart
             r2d=p1(i)**2+p2(i)**2
             if(r2d.lt.1.)  then
               ip=ip+1
```

```
                 px(ip)=delpp*p1(i)
                 py(ip)=delpp*p2(i)
                 pp(ip)=sqrt(px(ip)**2+py(ip)**2)
          betap(ip)=pp(ip)/gammas(ip)
          pz(ip)=sqrt(gammas(ip)**2-1.0-px(ip)**2-py(ip)**2)
                 alpha(ip)=sqrt(px(ip)**2+py(ip)**2)/pz(ip)
                 if(ip.eq.npart)  goto 211
               end if
  210       continue
            npart=ip
  211       continue
          end if
c ***************************************
c  Assign uniform current to each particle.   Total current
c  is beamcur.  Also normalize the current.  multiply by e/mccubedkw
c     first make beamcur negative since we are using electrons in the
c     beam
          beamcur=-beamcur
          do 250,i=1,npart
             acurrent(i)=beamcur*esubmc2*wlamda/(npart*lspeed*twopi)
 250      continue
c
c   The magnetic fields
          z1=zmin
          call bfield(z1)
c
c   Indexes of on axis and off axis particles
          zrmin=1.e10
          zrmax=0.0
          do 300 ip=1,npart
            zr2=xarry(1,ip)**2+xarry(2,ip)**2
            if(zr2.gt.zrmax) then
              ipoff=ip
              zrmax=zr2
            end if
            if(zr2.lt.zrmin)  then
              ipaxis=ip
              zrmin=zr2
            end if
  300     continue
c
c indexes of other particles to trace
```

```
c this works because particles' inital x's are uniform and in order
c   because of the way LODUNI works on the 1st Hammersley sequence
        if (ntrace.NE.0) then
           do 400 i=1,ntrace
              ntraces(i)=i*npart/(2*(ntrace+1))
 400       continue
        endif
c  Initialize the diagnostics
        thaxis(1)=pi/2.
        thoff(1)=thaxis(1)

        return
        end
*dk pack
c====================================================================
        subroutine pack(cy)
c====================================================================
c   Pack simulations variables
c
*ca param
*ca combla
*ca combea
*ca comwig
c------------------------------------------------------------------------
        dimension cy(*)
c------------------------------------------------------------------------
        xk=twopi/wlamda
        cexpo=cexp(cmplx(0.,-xk*z1))
        do 100 ip=1,npart
        cy(npart+ip)=cmplx(px(ip),py(ip))*cexpo
          cy(ip)=cmplx(xarry(1,ip),xarry(2,ip))*xk
  100   continue
        return
        end
*dk unpack
c====================================================================
        subroutine unpack(cy)
c====================================================================
c   Unpack simulations variables
c
*ca param
*ca combla
```

```
*ca combea
*ca comwig
*ca comint
c-----------------------------------------------------------------------
        dimension cy(*)
c-----------------------------------------------------------------------
        xk=twopi/wlamda
        cconst=cexp( cmplx(0.,xk*z1) )
        do 100 ip=1,npart
          xarry(1,ip)=real(cy(ip))/xk
          xarry(2,ip)=aimag(cy(ip))/xk
          px(ip)= real( cconst*cy(npart+ip) )
          py(ip)=aimag( cconst*cy(npart+ip) )
          pp(ip)=sqrt(px(ip)**2+py(ip)**2)
        gammas(ip)=sqrt(pz(ip)**2+pp(ip)**2+1)
          alpha(ip)=pp(ip)/pz(ip)
        betap(ip)=pp(ip)/gammas(ip)
  100   continue
        return
        end
*dk parsim
c===================================================================
        subroutine parsim(z,cy,cyp,pzp)
c===================================================================
c   Define equations of motion
c
*ca param
*ca combla
*ca comwig
*ca combea
c-----------------------------------------------------------------------
        dimension pzp(*),cy(*),cyp(*)
        complex cexpo,cpplus
c-----------------------------------------------------------------------
        zbar=twopi/wlamda * z
        cexpo=cmplx(cos(zbar),sin(zbar))
c
c   The derivatives
        do 200 ip=1,npart
        cpplus=cexpo*cy(npart+ip)/pz(ip)
        cyp(ip)=cpplus
        cyp(npart+ip)=cmplx(0.,-1.)*((1.-barry(3,ip)/pz(ip))*
     +   cy(npart+ip)+conjg(cexpo)*cmplx(barry(1,ip),barry(2,ip)))
        pzp(ip)=barry(1,ip)*aimag(cpplus)-barry(2,ip)
     +   *real(cpplus)
  200   continue
c
        return
        end
*dk bfield
c===================================================================
        subroutine bfield(z)
c===================================================================
c   Calculate the mag. fields using COIL3
c
*ca param
*ca combla
*ca comwig
*ca comdia
*ca combea
c-----------------------------------------------------------------------
c   Calculate the magnetic fields
        do 100 ip=1,npart
  100   xarry(3,ip)=z
        call magfld(npart,xarry,.false.,.false.,barry,dummy,dummy)
c
c   Simulate the fluctuations in Bz
        if(epsbz.ne.0.) then
        zcos=epsbz*bz0*cos(twopi*z/xlamdf)/xlamdf*pi
        zsin=epsbz*bz0*sin(twopi*z/xlamdf)/xlamdf*pi
        zdb=epsbz*bz0*sin(twopi*z/xlamdf)
        do 110 ip=1,npart
          barry(1,ip)=barry(1,ip)-zcos*xarry(1,ip)
          barry(2,ip)=barry(2,ip)-zsin*xarry(2,ip)
          barry(3,ip)=barry(3,ip)+zdb
  110   continue
        end if
c
c   Normalize the fields
        fnorm=esubmc/(twopi/wlamda)
        do 200 i=1,3
        do 200 ip=1,npart
  200   barry(i,ip)=fnorm*barry(i,ip)
c
```

```
        return
        end
*dk diagno
c===================================================================
        subroutine diagno(iz)
c===================================================================
c   Calculate the mag. fields using COIL3
c
*ca param
*ca combla
*ca comwig
*ca combea
*ca comdia
*ca comint
c-----------------------------------------------------------------------
        dimension xx(512),yy(512)
        real gmmamean,pzmean
c-----------------------------------------------------------------------
c
c   The electrons
        xkw=twopi/wlamda
        call xmean(alpha,npart,almean)
        palpha(iz)=almean
        call stdef(alpha,npart,alsig)
        if (almean.EQ.0.) then
           almean=1.e-12
        endif
        pdelal(iz)=alsig/almean
        call xmean(pz,npart,pzmean)
        call xmean(gammas,npart,gmmamean)
        bzmean(iz)=pzmean/gmmamean
        bpmean(iz)=1.-bzmean(iz)**2-1./gmmamean**2
        if (bpmean(iz).LT.0.) then
           bpmean(iz)=0.
        endif
        bpmean(iz)=sqrt(bpmean(iz))
        bztemp=pz(ipoff)/gammas(ipoff)
        call stdef(betap,npart,bpsig)
        if (bpmean(iz).EQ.0.) then
           bpmean(iz)=1.e-12
        endif
        pdelbp(iz)=bpsig/bpmean(iz)
        call stdef(pz,npart,pzsig)
        pdelbz(iz)=pzsig/pzmean
        pxaxis(iz)=xarry(1,ipaxis)
        pyaxis(iz)=xarry(2,ipaxis)
        pxoff(iz)=xarry(1,ipoff)
        pyoff(iz)=xarry(2,ipoff)
        rax(iz)=sqrt(pxaxis(iz)**2+pyaxis(iz)**2)
        roff(iz)=sqrt(pxoff(iz)**2+pyoff(iz)**2)
        do 6 i=1,npart
           xx(i)=xarry(1,i)
           yy(i)=xarry(2,i)
   6    continue
        call xmean(xx,npart,xxmean)
        call xmean(yy,npart,yymean)
        xxm(iz)=xxmean
        yym(iz)=yymean

c Find farthest off-axis particle.  This is not the same as
c    ipoff since that particle only started out farthest off-axis.
        beamtemp=0.
        do 50 i=1,npart
           beamt2=xarry(1,i)**2+xarry(2,i)**2
           if (beamt2.GT.beamtemp) then
              beamtemp=beamt2
              beamenip(iz)=i
           endif
 50     continue
        beamenv(iz)=sqrt(beamtemp)
c
c Save information about traced particles
        if (ntrace.NE.0) then
           do 60 i=1,ntrace
              ptrace(ntraces(i),1,iz)=xarry(1,ntraces(i))
              ptrace(ntraces(i),2,iz)=xarry(2,ntraces(i))
 60        continue
        endif
c  The guiding centers

        crg=cmplx(pxaxis(iz),pyaxis(iz))
        if(bz0.ne.0.)
     +    crg=crg+cmplx(0.,1./(esubmc*bz0))
     +        *cmplx(px(ipaxis),py(ipaxis))
```

```
      rgaxis(iz)=cabs(crg)
      crg=cmplx(pxoff(iz),pyoff(iz))
      if(bz0.ne.0.)
   +    crg=crg+cmplx(0.,1./(esubmc*bz0))
   +        *cmplx(px(ipoff),py(ipoff))
      rgoff(iz)=cabs(crg)

c  Obtain B-field for particle on,off axis

      xref(1,1)=xarry(1,ipaxis)
      xref(2,1)=xarry(2,ipaxis)
      xref(3,1)=zp(iz)
      call magfld(1,xref,.false.,.false.,bref,dummy,dummy)
      bpaxis(iz)=sqrt(bref(1,1)**2+bref(2,1)**2)
      xref(1,1)=xarry(1,ipoff)
      xref(2,1)=xarry(2,ipoff)
      xref(3,1)=zp(iz)
      call magfld(1,xref,.false.,.false.,bref,dummy,dummy)
      bpoff(iz)=sqrt(bref(1,1)**2+bref(2,1)**2)

c  Calculate slowly-varying phase for innermost and outermost particles

      if(iz.gt.1) then
      z=(iz-1)*delz+zmin
      zbar=xkw*z
      if(px(ipaxis).ne.0.) then
         targ1=atan(py(ipaxis)/px(ipaxis))
        else
         targ1=pi/2.
      end if
      if(px(ipoff).ne.0.) then
         targ2=atan(py(ipoff)/px(ipoff))
        else
         targ2=pi/2.
      end if
      xl=nwigg*wlamda
      if(z.lt.0.) zbar=0.
      if(abs(z-xl).lt.delz) zbar1=zbar
      if(z.gt.xl) zbar=zbar1
      th1=zbar-targ1+pi/2.
      th2=zbar-targ2+pi/2.
      psi1=th1-int(th1/twopi)*twopi
```

```
        psi2=th2-int(th2/twopi)*twopi
        if(psi1.ge.pi) psi1=psi1-pi
        if(psi1.le.0.) psi1=psi1+pi
        if(psi2.ge.pi) psi2=psi2-pi
        if(psi2.le.0.) psi2=psi2+pi
        thaxis(iz)=psi1
        thoff(iz)=psi2
        thzbar(iz)=zbar
        thtar1(iz)=-targ1
        thtar2(iz)=-targ2
        end if

c  Obtain B-field for reference on z-axis
        if(iz.eq.nz) then
           xref(1,1)=0.
           xref(2,1)=0.
           do 100 i=1,nz
              xref(3,1)=zp(i)
              call magfld(1,xref,.false.,.false.,bref,dummy,dummy)
              bxref(i)=bref(1,1)
              byref(i)=bref(2,1)
              bpref(i)=sqrt(bxref(i)**2+byref(i)**2)
  100      continue
        end if
c
        return
        end
*dk loduni
        SUBROUTINE LODUNI(NBASE,N,Y)
c       ----------------------------
c
c   Load an uniform distribution using the Hammersley's sequence.
c   (NBASE=0 ==> Random sampling !)
c
c-------------------------------------------------------------------------
        DIMENSION       Y(*)
c-------------------------------------------------------------------------
c
c   Random and Quasi-Random Loading
c
      IF(NBASE.EQ.0)  THEN
c
```

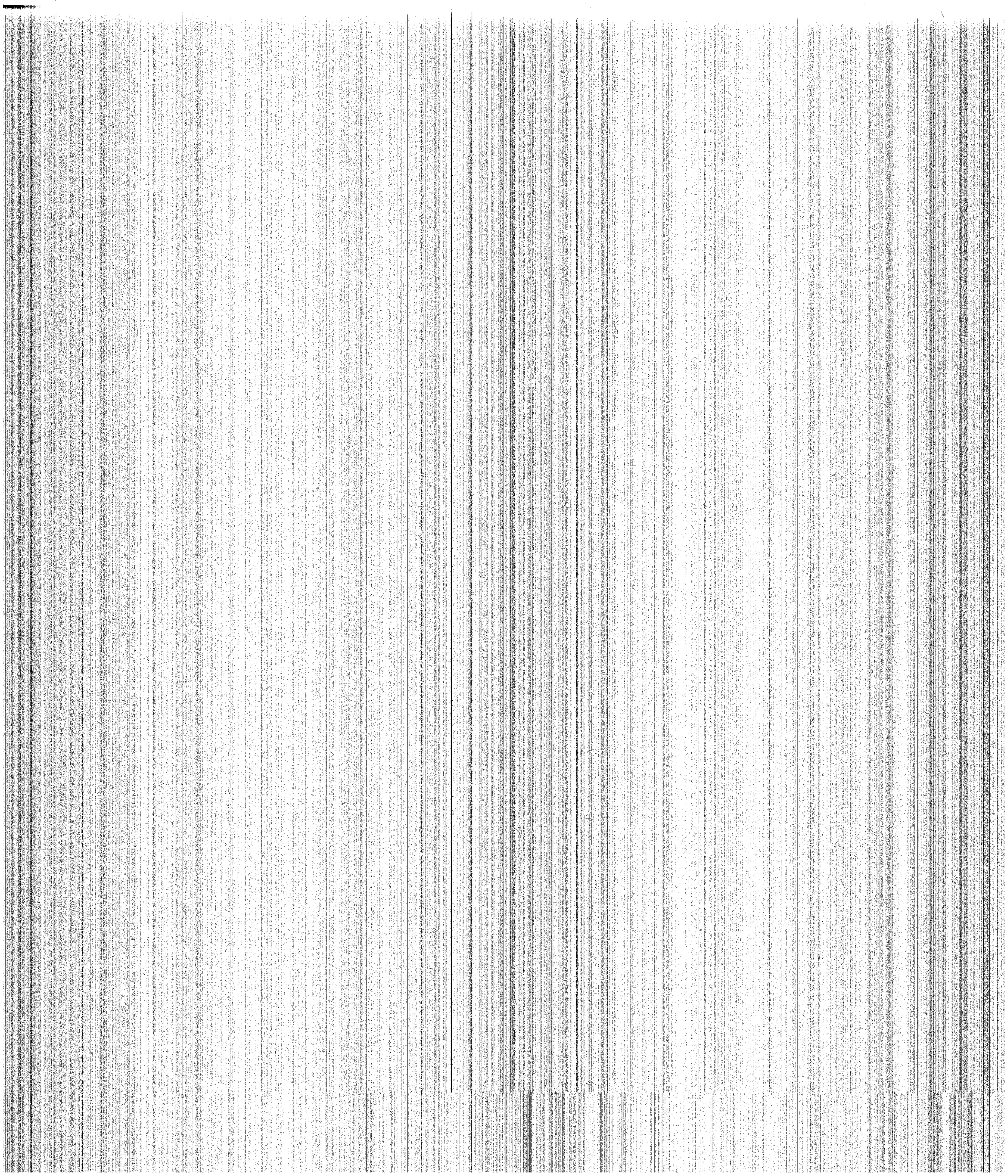
```
c         Random
        DO 100 I=1,N
  100   Y(I) = RANF(DUMMY)
c
c         First elemenent of Hammersley's sequence
      ELSE IF(NBASE.EQ.1) THEN
        DO 110 I=1,N
  110   Y(I) = (I-0.5)/N
c
c         Radical-inverse Function in base NBASE
      ELSE IF(NBASE.GT.1)  THEN
        DO 120 I=1,N
        XS = 0.
        XSI = 1.0
        J2 = I
    1   XSI = XSI/NBASE
        J1 = J2/NBASE
        XS = XS + (J2-NBASE*J1)*XSI
        J2 = J1
        IF( J2.GT.0 )  GOTO 1
  120   Y(I) = XS
       END IF
c
        RETURN
        END
*dk resetr
        SUBROUTINE RESETR(KN,PA,PVALUE)
c       -------------------------------
c
c   Set real array PA to PVALUE.
c
c---------------------------------------------------------------------------
        DIMENSION  PA(*)
c---------------------------------------------------------------------------
        DO 100 K=1,KN
          PA(K) = PVALUE
  100   CONTINUE
c
        RETURN
        END
*dk reseti
        SUBROUTINE RESETI(KN,KA,KVALUE)
```

```
c       -------------------------------
c
c   Set integer array KA to KVALUE.
c
c-----------------------------------------------------------------------------
        DIMENSION  KA(*)
c-----------------------------------------------------------------------------
        DO 100 K=1,KN
          KA(K) = KVALUE
  100   CONTINUE
c
        RETURN
        END
*dk resetc
        SUBROUTINE RESETC(KN,CA,CVALUE)
c       -------------------------------
c
c   Set complex array CA to CVALUE.
c
c-----------------------------------------------------------------------------
        COMPLEX  CA(*),  CVALUE
c-----------------------------------------------------------------------------
        DO 100 K=1,KN
          CA(K) = CVALUE
  100   CONTINUE
c
        RETURN
        END
*dk cputime
c=====================================================================
      SUBROUTINE CPUTIM
c=====================================================================
c
c   Display cpu-time used so far
c
      CALL SECOND(USETIM)
      CALL RVAR('CPU-TIME USED SO FAR ',USETIM)
      RETURN
      END
*dk daytim
c=====================================================================
      SUBROUTINE DAYTIM
SUBROUTINE DAYTIM
```

```
c======================================================================
c
c   Display current time and date
c
      CALL TIMEDATE(TIM,DAT,MACH,SUFIX)
      WRITE(6,9900) DAT, TIM, MACH, SUFIX
 9900 FORMAT(5X,'CURRENT DATE AND TIME',A10,2X,A10,2X,
     F       A1,'/',A1)
      RETURN
      END
*dk rvar
c======================================================================
      SUBROUTINE RVAR( LABEL, PVAR )
c======================================================================
c
      CHARACTER*(*)  LABEL
c
      WRITE(6,'(5X,A,E12.4)')  LABEL, PVAR
c
      RETURN
      END
*dk plotfp
c======================================================================
      subroutine plotfp(xlabel,ylabel)
c======================================================================
c
*ca param
*ca comdia
*ca comint
c
          character*(*) xlabel,ylabel
c
c-------------------------------------------------------------------------

          call aminmx(bxref,1,nz,1,ymin1,ymax1)
          call aminmx(byref,1,nz,1,ymin2,ymax2)
          call aminmx(bpref,1,nz,1,ymin3,ymax3)
          ymin=ymin1
          ymax=ymax1
          if(ymin.ge.ymin2) ymin=ymin2
          if(ymin.ge.ymin3) ymin=ymin3
          if(ymax.le.ymax2) ymax=ymax2
          if(ymax.le.ymax3) ymax=ymax3
          if(ymin.eq.ymax) then
             ymin=0.95*ymax
             ymax=1.05*ymax
          end if
          ymin=ymin-(ymax-ymin)*.05
          ymax=ymax+(ymax-ymin)*.05

          call maps(zmin,zp(nz),ymin,ymax)
          call setch(15.,1.,0,0,2,0,0)
          call crtbcd(xlabel,1)
          call setch(0.5,40.,0,0,2,1,0)
          call crtbcd(ylabel,1)
          call setpch(1,0,3,0,100)
          call tracep(zp,bxref,nz)
          call tracep(zp,byref,nz)
          call trace(zp,bpref,nz)
          call frame
      RETURN
      END
*dk histry
c========================================================================
        subroutine histry(xlabel,ylabel,y)
c========================================================================
c
*ca param
*ca comdia
*ca comint
c
          character*(*) xlabel,ylabel
          dimension y(*)
c------------------------------------------------------------------------

          call aminmx(y,1,nz,1,ymin,ymax)
          if(ymin.eq.ymax) then
             ymin=0.95*ymax
             ymax=1.05*ymax
          end if
          ymin=ymin-(ymax-ymin)*.05
          ymax=ymax+(ymax-ymin)*.05

          call maps(zmin,zp(nz),ymin,ymax)
      call maps(zmin,zp(nz),ymin,ymax)
```

```
          call setch(15.,1.,0,0,2,0,0)
          call crtbcd(xlabel,1)
          call setch(0.5,40.,0,0,2,1,0)
          call crtbcd(ylabel,1)
          call setpch(1,0,0,0,1)
          call trace(zp,y,nz)
          call frame
      RETURN
      END
*dk dhist
c=======================================================================
        subroutine dhist(xlabel,ylabel,x1,y)
c=======================================================================
c
*ca param
*ca comdia
*ca comint
c
          character*(*) xlabel,ylabel
          dimension y(*),x1(*)
c-----------------------------------------------------------------------

          call aminmx(x1,1,nz,1,xmin,xmax)
          call aminmx(y,1,nz,1,ymin,ymax)
          if(xmax.ge.ymax) ymax=xmax
          if(xmin.le.ymin) ymin=xmin
          if(ymin.eq.ymax) then
             ymin=0.95*ymax
             ymax=1.05*ymax
          end if
          ymin=ymin-(ymax-ymin)*.05
          ymax=ymax+(ymax-ymin)*.05

          call maps(zmin,zp(nz),ymin,ymax)
          call setch(15.,1.,0,0,2,0,0)
          call crtbcd(xlabel,1)
          call setch(0.5,40.,0,0,2,1,0)
          call crtbcd(ylabel,1)
          call setpch(1,0,0,0,1)
          call trace(zp,x1,nz)
c At least a temporary change so that both graph linesc
c are visable.  Dotted lines don't come out on QMS printers
```

e are visable. Dotted lines don't come out on QMS printers



```
c       call tracep(zp,y,nz)
        call trace(zp,y,nz)
*ad
          call frame
      RETURN
      END
*dk psplot
c=======================================================================
        subroutine psplot(iz,icontrol)
c=======================================================================
c
*ca param
*ca combla
*ca combea
*ca comdia
*ca comint
c-----------------------------------------------------------------------
          dimension yy(4096),xx(4096)
c-----------------------------------------------------------------------

          zout=zp(iz)
        if(icontrol.EQ.1) then
          do 105 i=1,npart
             xx(i)=xarry(1,i)*1000.
             yy(i)=xarry(2,i)*1000.
  105     continue
           xlabel='x [mm]'
           ylabel='y [mm]'
        else if(icontrol.EQ.2) then
          do 200 i=1,npart
            xx(i)=xarry(1,i)**2+xarry(2,i)**2
            xx(i)=sqrt(xx(i))*1000.
            yy(i)=sqrt(py(i)**2+px(i)**2)/gammas(i)
 200      continue
          xlabel='R [mm]'
          ylabel='BetaP'
        else if(icontrol.EQ.3) then
          do 300 i=1,npart
            xx(i)=xarry(1,i)**2+xarry(2,i)**2
            xx(i)=sqrt(xx(i))*1000.
            theata=atan2(xarry(2,i),xarry(1,i))
            yy(i)=cos(theata)*py(i)-sin(theata)*px(i)
          yy(i)=yy(i)/gammas(i)
300     continue
        xlabel='R [mm]'
        ylabel='BetaTh'
      else if(icontrol.EQ.4) then
        do 400 i=1,npart
          xx(i)=xarry(1,i)**2+xarry(2,i)**2
          xx(i)=sqrt(xx(i))*1000.
          theata=atan2(xarry(2,i),xarry(1,i))
          yy(i)=cos(theata)*px(i)+sin(theata)*py(i)
          yy(i)=yy(i)/gammas(i)
400     continue
        xlabel='R [mm]'
        ylabel='BetaR'
      endif

        call aminmx(xx,1,npart,1,xmin,xmax)
        call aminmx(yy,1,npart,1,ymin,ymax)
        if (xmin.eq.xmax) then
           xmin=0.5*xx(1)
           xmax=3.*xmin
         else
           xmin=xmin-(xmax-xmin)*.05
           xmax=xmax+(xmax-xmin)*.05
        end if
        if (ymin.eq.ymax) then
           ymin=0.5*yy(1)
           ymax=3.*ymin
         else
           ymin=ymin-(ymax-ymin)*.05
           ymax=ymax+(ymax-ymin)*.05
        end if
        call maps(xmin,xmax,ymin,ymax)
        call setch(15.,1.,0,0,2,0,0)
        call crtbcd(xlabel,1)
        call setch(0.5,40.,0,0,2,1,0)
        call crtbcd(ylabel,1)
        call setpch(1,0,0,0,1)
        call pointc('*',xx,yy,npart)
        call setch(10.,.5,0,0,1,0,0)
        write(100,1000) zout
        call frame
```

```
 1000     format (35x,'Z = ',e10.3,'m')

          RETURN
          END
*dk stdef
c=======================================================================
      subroutine stdef(x,n,sigma)
c=======================================================================
c
      dimension x(*)
c-----------------------------------------------------------------------
c
c  First calculate the arithmatic mean
c
      call xmean(x,n,y)

c  Now calculate the variance
c
      varsum=0.
      do 200 i=1,n
         varsum=varsum+(x(i)-y)**2
  200 continue
      var=varsum/(n-1)

      sigma=sqrt(var)

      RETURN
      END
*dk xmean
c=======================================================================
      subroutine xmean(x,n,xm)
c=======================================================================
c
      dimension x(*)
c-----------------------------------------------------------------------
c
      xm=0.
      do 100 i=1,n
         xm=xm+x(i)
  100 continue
      xm=xm/n
      RETURN
      END


*dk poisson
c==================================================
          subroutine poisson(z,iz,cyp)
c==================================================
c     Calculate space charge effects by solving
c       Poisson's equaition on a polar grid using
c       HWSPLR in the SLATEC3 library.
c     Operates on xarray (lab-frame coords) so UNPACK should
c       be run before use
c     Returns array cyp filled with normalized coordinate
c      derivitaves of paricle momentum with respect to z
c      (adds the effects from space charge forces to whatever
c         is already in cyp)
c
c  note:   numr is number of concentric grid lines including origin and wall
c      ntheata is number of radial grid lines.  note that HWSPLR thinks there
c      are ntheata+1 radial grid lines, but line number ntheata+1
c      overlaps line number 1 becuase of theata wrap-around.  This
c      leads to some differences in the way r and theata are handled.
c      be careful.  Look at formulae for deltar and dtheata to see what
c      I mean....
*ca param
*ca comdia
*ca combea
*ca comwig
*ca comint
*ca combla
        real z,drsquard
        integer i,j,k,n,ierror,iz,partjk(2,npmax)
        dimension cyp(*)
c   can I dimension grid(numr,ntheata) on the fly or must it
c    be static?
        real grid(numrmax,nthmax+1),rtheata(2,npmax),deltar
        real dtheata,dummy,workspc(3500)
c real workspc(4*(nthmax+1)+(13+
c     +    int(log2(nthmax)))*numrmax)
        real pertrb,origin,area,subxk,zbar,charge,roe
        complex cegrid(numrmax,nthmax+1),certh(npmax),cexpo
```

```
c cegrid(j,k)=Eradial at r=j, theata=k plus i* Etheata at r=j,theata=k
c
c First assign the charge from each particle to the
c   surrounding grid points
c
c initialize the grid
         do 2 j=1,numr+1
            do 4 k=1,ntheata+1
               grid(j,k)=0.
 4          continue
 2       continue
c
c figure out distance between grid points
        deltar=rpipe/(numr-1)
c no -1 next to ntheata since theata wraps around
        dtheata=twopi/ntheata
c change to r-theata coordinates
        do 10 i=1,npart
           rtheata(1,i)=sqrt(xarry(1,i)**2+xarry(2,i)**2)
           rtheata(2,i)=atan2(xarry(2,i),xarry(1,i))
        if (rtheata(2,i).LT.0.) then
           rtheata(2,i)=rtheata(2,i)+twopi
        endif
c atan2 returns value -pi<ang<=pi
c I want things in the form 0<=ang<2*pi
c The above makes the form  0-<=and<pi
c which will be ok whne the next couple of lines
c makes the angle into an integer grid point k
 10      continue
c
c This loop actually assigns charge to gridpoints
        drsquard=deltar*deltar
c       tcurrent=0.
c       tcharge=0.
        do 20 i=1,npart
c
c Convert from r-theata to j-k (integer) coords
c do I need the plus ones?
c does the HWSPLR want from 0 to n-1 or from 1 to n?
           j=int(rtheata(1,i)/deltar)+1
           k=int(rtheata(2,i)/dtheata)+1
c save particle grid coords
```

```
           partjk(1,i)=j
           partjk(2,i)=k
c
c calculate charge density from current on each particle
           charge=acurrent(i)*gammas(i)/pz(i)
c           tcurrent=tcurrent+acurrent(i)
c           tcharge=tcharge+charge
c   minus sign since delsquared phi= minus roe/e0  right??
c
c assign normalized charge to grid points based on
c area. see page 337 of Birdsall and Langdon
c Plasma Physics via computer simulation
c normalized charge is e/mcsquared per particle
           area=drsquard*(j*j-(j-1)*(j-1))*dtheata
c
           grid(j,k)=grid(j,k)+charge*(j*j*drsquard-
     +       rtheata(1,i)*rtheata(1,i))*(k*dtheata-
     +       rtheata(2,i))/area
           grid(j+1,k)=grid(j+1,k)+(rtheata(1,i)*
     +       rtheata(1,i)-(j-1)*(j-1)*drsquard)*
     +       (k*dtheata-rtheata(2,i))/area*charge
           grid(j,k+1)=grid(j,k+1)+charge*(j*j*
     +       drsquard-rtheata(1,i)*rtheata(1,i))
     +       *(rtheata(2,i)-(k-1)*dtheata)/area
           grid(j+1,k+1)=grid(j+1,k+1)+charge*
     +       (rtheata(1,i)*rtheata(1,i)-(j-1)*(j-1)*
     +       drsquard)*(rtheata(2,i)-(k-1)*dtheata)
     +       /area
 20       continue
c
c
c Now fix things up.  All angles at r=0 should get
c total charge density at r=0 as per HWSPLR instructions and
c charge assigned to k=ntheata+1 should get added in
c to k=1 row since the angle really wraps around.
c Also, charge density assigned to j,ntheata+1 should be
c made equal to that at j,1.
c This should be faster than a MOD ntheata in the above loop.
c
        origin=0.
        do 30 k=1,ntheata+1
           origin=origin+grid(1,k)
 30     continue
c
c       do 40 k=1,ntheata+1
c          grid(1,k)=origin
c40     continue
c
        do 50 j=1,numr
           grid(j,1)=grid(j,1)+grid(j,ntheata+1)
           grid(j,ntheata+1)=grid(j,1)
 50     continue
c
c Now convert all of this to charge density from charge
        do 55 j=2,numr-1
           do 58 k=1,ntheata+1
              grid(j,k)=-grid(j,k)/(epsilon0*drsquard*(j-1)*
     +           dtheata)
 58        continue
c origin is done differently
 55     continue
        grid(1,1)=-origin*4/(epsilon0*pi*drsquard)
        do 59 k=2,ntheata+1
           grid(1,k)=grid(1,1)
 59     continue
c
c Put in boundary conditions
        do 60 k=1,ntheata+1
           grid(numr,k)=phiwall
 60     continue
c
c All set to call HWSPLR
c
c
c Call HWSPLR to solve for phi (potential)
        call hwsplr(0.,rpipe,numr-1,5,dummy,dummy,0.,
     +     twopi,ntheata,0,dummy,dummy,0.,grid,numrmax,
     +     pertrb,ierror,workspc)
c Now check for errors, etc.
        if (ierror .ne. 0) then
           write(6,*) 'warning *****'
           write(6,*) 'IERROR=',ierror
           write(6,*) 'in HWSPLR'
        endif
```

```
c save pertrb to be graphed later to check for largeness
c see HWSPLR instructions and think about this somemore
        pertrbs(iz)=pertrb
c save a copy of phi along a radius at start
c to help see if this is working ok
        if (z .eq. zmin) then
          do 70 j=1,numr
            phi(j)=grid(j,1)
 70       continue
          do 72 k=1,ntheata+1
            phi2(k)=grid(8,k)
            phi3(k)=grid(48,k)
 72       continue
        endif
c Calculate e from phi
        call phitoe(grid,cegrid)
c
c save a copy of E along a radius to help see if this is working
c ok.
        if (z .eq. zmin) then
           do 75 j=1,numr
              esave(j)=real(cegrid(j,1))
 75        continue
        endif
c
c
c Assign E's to particles based on area
c first initialize array
        do 80 i=1,npart
           certh(i)=0.
 80     continue
c now loop on particles and assign the E's
        do 90 i=1,npart
           drsquard=deltar*deltar
c retrieve particle grid coords
           j=partjk(1,i)
           k=partjk(2,i)
c now assign the fields
           area=drsquard*(j*j-(j-1)*(j-1))*dtheata
cc       write(6,*) 'i,area',i,area
cc       write(6,*) 'j,k',j,k
           weight1=(j*j*drsquard-
     +        rtheata(1,i)*rtheata(1,i))*(k*dtheata-
     +        rtheata(2,i))/area
            weight2=(rtheata(1,i)*
     +        rtheata(1,i)-(j-1)*(j-1)*drsquard)*
     +        (k*dtheata-rtheata(2,i))/area
            weight3=(j*j*drsquard-rtheata(1,i)*rtheata(1,i))
     +        *(rtheata(2,i)-(k-1)*dtheata)/area
            weight4=(rtheata(1,i)*rtheata(1,i)-(j-1)*(j-1)*
     +        drsquard)*(rtheata(2,i)-(k-1)*dtheata)
     +        /area
            certh(i)=weight1*cegrid(j,k)+weight2*
     +          cegrid(j+1,k)+weight3*cegrid(j,k+1)+
     +          weight4*cegrid(j+1,k+1)
 90     continue
c Convert Er, Etheata to normalized E in rotating frame (Epluswiggle)
c Epluswiggle is e/mcsquared (Ex+iEy)exp(-ikwz)
c and figure out effects on dp/dz 's
c add this result into cyp()
        subxk=wlamda/twopi
        zbar=z/subxk
        do 100 i=1,npart
           cexpo=cexp(cmplx(0.,rtheata(2,i)-zbar))
           cyp(npart+i)=cyp(npart+i)-cexpo*certh(i)*
     +        gammas(i)/pz(i)
100     continue
c all done believe it or not
        return
        end
*ad
*dk phitoe
c
c*********************************************
        subroutine phitoe(grid,cegrid)
c*********************************************
c calcule E from phi.  E=-grad phi
c the equations for this in polor coords from
c pages 332-335 of Birdsall and Langdan Plasma Physics
c via computer simulation
c
*ca param
*ca combla
*ca comwig
```

```
*ca comint
        dimension grid(numrmax,nthmax+1)
        complex cegrid(numrmax,nthmax+1)
        integer i,j,k
        real deltar,twodr,dtheata,twodth,er,eth
c
        deltar=rpipe/(numr-1)
        dtheata=twopi/ntheata
        twodr=deltar*2
        twodth=dtheata*2
        dr2dth=twodth*deltar
c
c question** does HWSPLR rerturn grid(1,k) all equal to phi(0)?
c do it for all grid points except near origin and wrap-arounds
        do 10 j=2,numr-1
           do 20 k=2,ntheata-1
              cegrid(j,k)=cmplx((grid(j-1,k)-grid(j+1,k))/twodr,
     +        (grid(j,k-1)-grid(j,k+1))/(dr2dth*(j-1)))
 20        continue
 10     continue
c
c fixup fields near theata wrap-around
        do 30 j=2,numr-1
           cegrid(j,1)=cmplx((grid(j-1,1)-grid(j+1,1))/twodr,
     +       (grid(j,ntheata)-grid(j,2))/(dr2dth*(j-1)))
           cegrid(j,ntheata)=cmplx((grid(j-1,ntheata)-grid(
     +        j+1,ntheata))/twodr,(grid(j,ntheata-1)-grid(
     +        j,1))/(dr2dth*(j-1)))
 30     continue
c
c fixup Er amd Etheata near origin and near boundary
        do 40 k=1,ntheata
           cegrid(1,k)=cmplx((grid(1,k)-grid(2,k))/deltar,
     +       aimag(cegrid(2,k)))
           cegrid(numr,k)=cmplx((grid(numr-1,k)-grid(numr,k))
     +       /deltar,0.)
 40     continue

c copy fields at j,k=1 to j,k=ntheata+1 so to avoid MODs when assigning
c E's from grid to particles.
        do 50 j=1,numr
           cegrid(j,ntheata+1)=cegrid(j,1)
 50     continue
c all done
        return
        end
*ad
*dk plotgen
c
c**********************************************************
        subroutine plotgen(y,numx,xmin,xmax,xlabel,ylabel)
c**********************************************************
c plots array y from xmin to xmax.  numx is number of points
c to plot
*ca param
        character*(*) xlabel,ylabel
        dimension y(*)
        real xmin,xmax,r(nzmax),ymax,ymin
        integer numx
c
        call aminmx(y,1,numx,1,ymin,ymax)
        if (ymin .eq. ymax) then
           ymin=0.95*ymax
           ymax=1.05*ymax
        endif
        ymin=ymin-(ymax-ymin)*.05
        ymax=ymax+(ymax-ymin)*.05
c       write(6,*) 'ymin,ymax',ymin,ymax
        r(1)=xmin
        deltar=(xmax-xmin)/(numx-1)
        do 10 i=2,numx
c         write(6,*) y(i)
           r(i)=r(i-1)+deltar
10      continue
        call maps(xmin,r(numx),ymin,ymax)
        call setch(15.,1.,0,0,2,0,0)
        call crtbcd(xlabel,1)
        call setch(0.5,40.,0,0,2,1,0)
        call crtbcd(ylabel,1)
        call setpch(1,0,0,0,1)
        call trace(r,y,numx)
        call frame
        return
        end
```

Bibliography

[1] G. Bekefi, A. DiRienzo, C. Leibovitch, and B. Danly. 35 ghz cyclotron autoresonance maser amplifier.. *Appl. Phys. Lett.*, 54(14), April 1989.

[2] Charles K. Birdsall and A. Bruce Langdon. *Plasma Physics via Computer Simulation.* McGraw-Hill Book Company, New York, 1985.

[3] George R. Brewer. *IRE, Trans. Profess. Group Electron Devices*, 4(2):132–140, 1957.

[4] George R. Brewer. Focusing of high-density electron beams. In Albert Septier, editor, *Focusing of Charged Particles*, volume II, chapter 3.3, pages 73–95. Academic Press, 1967. Hughes Research Laboratory, Malibu California.

[5] C.H. Edwards, Jr. and David E. Penny. *Elementary Differential Equations with Applications.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985.

[6] J. Fajans. End effects of a bifilar magnetic wiggler. *Journal of Applied Physics*, 55(1), January 1984.

[7] Henry P. Freund and Robert K. Parker. Free electron lasers. *Scientific American*, 260(4):84–89, April 1989.

[8] F. Hartemann. Paramagnetic and diamagnetic corrections to the electron dynamics in a free-electron laser with guide field. MIT Plasma Fusion Center, Permanent address: Thomson-CSF/DTE, 78141 Vélizy, France., 1989.

[9] J. D. Jackson. *Classical Electrodynamics.* J. Wiley, second edition, 1975.

[10] R. H. Jackson and C. A. Sedlak. Gyrotron beam generation with helical magnetic fields. Final Report MRC/WDC-R-061, Mission Research Corporation, Alexandria, Virginia, August 1983.

[11] J.D. Lawson. *The Physics of Charged-Particle Beams.* Clarendon Press, Oxford, 1977.

[12] J. Neilson, M. Caplan, N. Lopez, and K. Felch. Simulation of space-charge effects on velocity spread in gyro devices. Technical Report CH2252-5/85/0000-0184, IEDM, 1985.

[13] T.J. Orzechowski, B.R. Anderson, J.C. Clark, W.M. Fawley, A.C. Paul, D. Prosnitz, E.T. Scharlemann, and S.M. Yarema. High-efficiency extraction of microwave radiation from a tapered-wiggler free-electron laser. *Phys. Rev. Lett.*, 58:2172–2175, 1986.

[14] R. E. Shefer, Y.Z. Yin, and G. Bekefi. Velocity diagnostics of mildly relativistic, high current electron beams. *Journal of Applied Physics*, 54(11):6154–6159, November 1983.

[15] P. Swarztrauber and R. Sweet. Efficient fortran subprograms for the solution of elliptic equations. Technical Report TN/IA-109, NCAR, June 1975. 138 pp.