A New $2\frac{1}{2}$-D PIC Charging code for Low Earth Orbit

D.J.Rodgers, R.L.Kessel, A.Etemadi
Mullard Space Science Laboratory, University College London
Dorking, Surrey, RH5 6NT, UK

S.T.Brandon
Lawrence Livermore, Livermore, CA, USA

Presented at the Spacecraft Charging Technology Conference - 1989
Monterey, California, USA 31st Oct to 3rd Nov 1989

This work supported by ESA contract no. 7989/88/NL/PB(SC)

Abstract

We describe the construction of a new charging code. This is based on a $2\frac{1}{2}$-D particle-in-cell code that moves ions and electrons in a self-consistent electric field until an equilibrium is reached. The original code was designed to simulate laboratory experiments at NASA Lewis consisting of a conducting disk plus dielectric or a conducting disk alone, in an isotropic plasma.

The new code includes the effects of a non-self-consistent magnetic field and bulk plasma flow. It can also simulate more general object shapes. It is designed to simulate charging in the Low Earth Orbit environment and more general Laboratory simulations.

1 Introduction

This code, presently named PICCHARGE, is being developed at as part of a study [1] being performed for the European Space Agency, involving the Mullard Space Science Laboratory, Culham Laboratory, the Norwegian Defense Research Establishment and the Technical University, Graz. In this study, a suite of computer programs is being developed to enable the scientific investigation of spacecraft/plasma interactions as well as to provide predictions of surface charging for engineering purposes. The engineering aims are met mainly by other elements of the suite, however we expect that PICCHARGE will also be used by engineers to examine small parts of a spacecraft surface where the charging behaviour is unknown. One of the principal features of the whole study is the comparison of results from computer codes with a detailed laboratory simulation program.

2 Origins of the code

We started with a code that had been developed at the University of Kansas to compare with experimental tests being performed at NASA Lewis [2]. This is a particle-in-cell code which tracks positive and negative ions explicitly in a self-consistent electric field. It is 2-dimensional in position but 3-dimensional in velocit hence it has been described as a $2\frac{1}{2}$-D code. Figure 1 shows the 2-dimensional spatial grid. Macroparticles having the charge of hundreds or thousands of ions or electrons and of finite size are moved rather than individual ions and electrons, thus speeding up the simulation. The two dimensions of the grid are termed z and r because the space is defined to be cylindrically symmetric about an axis (the z axis). The electric potential is defined at the centre points of the grid cells and is found by 'Successive Over-Relaxation', a finite difference method which uses numerical techniques to come to a rapid convergence. The electric field is found by differentiating the potential. The particles are moved under the influence of this electric field only. When a particle reaches the edge of the simulation area, it is lost to the simulation. However, new particles are continually introduced at these edges. In this code, the new particles have an isotropic Maxwellian distribution and this forms one of the boundary conditions of the simulation. The original code was severely limited in the target objects it could include in the simulations. It would only handle metal disks, with or without a coating of a single dielectric. This disk always lay along the z=0 boundary of the simulation space.

We wanted to convert this code from one that simulated a particular laboratory configuration, to one that was more generally applicable to charging in the Low Earth Orbit (LEO) environment and to more general laboratory simulations. To do this, the following changes were chosen:

a) The mover should be replaced by one that moves particles according to a steady magnetic field as well as an electric field. The magnetic field need not be self-consistent since any induced fields are likely to be far smaller than the strong ambient field present in the Low Earth Orbit region.

b) The plasma introduced at the boundaries should be allowed to have strong drift velocities. This is because satellites in Low earth Orbit typically have velocities through the plasma of about eight times the thermal velocity of the plasma.

c) More general object shapes and positions should be allowed. This allows more complex objects to be simulated and the difference between ram and wake directions to be explored. The objects should, in addition, have multiple surface materials.

d) Dielectric properties should be made more realistic, in particular with the introduction of finite conductivity.

e) Streaming high-energy electrons should be introduced to simulate the effects of precipitating auroral electrons in polar regions.

f) Backscatter electrons should be included. These are electrons reflected from the object surface with almost their input energy. The probability for this process is material-dependent.

g) For better comparison with laboratory experiments it is desirable to be able to introduce a population of cold ions, created from the residual gases by the charge-exchange process and with essentially zero kinetic energy.

h) The original code was written in Rationalized FORTRAN, an uncommon FORTAN dialect. The new code is targeted for VAX/VMS computers but is written in essentially standard FORTRAN-77 so that it can be run almost anywhere.

Items a) b) and h) of this code have been completed, integrated into the existing code and tested. Item c), which is the largest change to the code has been completed, and integrated with the existing code but has not yet been fully tested. Items d), e), f) and g) are expected to be complete soon. The following sections describe work on items a), b) and c).

3 The Mover

The movement of particles may be expressed by equations of motion. In this code these ordinary differential equations (below left) are replaced by 'finite difference' equations (below right) in which the *dt* term is replaced by a finite time difference Δt.

$$m\frac{dv}{dt} = F \qquad m\frac{v_{new} - v_{old}}{\Delta t} = F$$

$$\frac{dx}{dt} = v \qquad \frac{x_{new} - x_{old}}{\Delta t} = v$$

The mover calculates the velocity and position one half time-step out of phase. This 'Leap-Frog' method gives the code greater stability which means that larger values of Δt can be used without the code becoming inaccurate. In the new code, the equations of motion take into account the electric and magnetic fields, i.e.

$$m\frac{dv}{dt} = q(E + v \times B)$$

This equation is actually solved by a three-step process devised by Boris [3] that separates the electrostatic and magnetic effects.
Starting at time $t - \Delta t/2$, the particle is accelerated by half of the electric field:

$$v^- = v_{t-\Delta t/2} + \frac{qE\Delta t}{2m}$$

Then, the velocity vector is rotated according to the magnetic field as if there were no electric field.

$$\begin{aligned} \tan(\theta/2) &= -tan(qB\Delta t/2m) \\ v_x^+ &= \cos(\theta)v_x^- - \sin(\theta)v_y^- \\ v_y^+ &= \sin(\theta)v_x^- + \cos(\theta)v_y^- \end{aligned}$$

Finally, the particle is accelerated by the second half of the electric field to yield the velocity at time $t + \Delta t/2$.

$$v_{t+\Delta t/2} = v^+ + \frac{qE\Delta t}{2m}$$

Although this mover is performing more complex calculations than the simple electrostatic mover, most of the complicated terms can be precalculated at the start of the simulation and so the new mover is only about 10% slower than the original.

Figure 2 shows a test that was performed on the mover to see if the time step that was chosen by the code was appropriate. The figure shows three trajectories, one with time step Δt as chosen by the code, another with $10 \times \Delta t$ and another with timestep $0.1 \times \Delta t$. The trajectory produced with timestep $10 \times \Delta t$ shows significant deviation from the other two trajectories which are practically overlaid. This means that the chosen timestep is appropriate because the trajectory is accurate to far less than the grid cell dimension and yet the timestep is not unnecessarily small.

When the magnetic field was set to zero, the new mover produced trajectories that were identical to the old electrostatic mover. Figure 3 shows an example when the electric field was set to zero. Here the particle simply gyrates around the magnetic field. The gyroradius or 'Larmor radius' of such motion is easily calculated analytically. The diameter of this gyration produced by the code agrees with the theoretical value to 6 significant places, which is easily accurate enough for our simulations.

4 Introducing Drifting Plasma at the Boundaries

The code considers this process in two parts: assigning the correct velocity to a particle entering at a boundary; and calculating how many new particles will enter.

It is necessary to give the new particle a random velocity so that when many particles have been added, they will have the desired velocity distribution. Figure 4 [4] shows the technique used. The desired normalized distribution, $p(y)$ is integrated to form $F(y)$. A random number from the computer random number generator is a 'uniform deviate' i.e. it has uniform probability between 0 and 1. When such a random number value has been obtained, y is chosen such that $F(y)$ has this value. The code only evaluates $F(y)$ at the start when it creates a table of $F(y)$ values.

We consider the three velocity components of the particle separately. When a particle enters a boundary, the probability distribution for the velocity vectors parallel to that boundary will simply be the velocity distribution of the plasma, i.e. $p(y) = f(y)$ where y is the velocity in appropriate units. However, for the velocity perpendicular to the boundary the probability is altered by the fact that particles with high velocities perpendicular to the boundary are more likely to cross the boundary, i.e. $p(y) \propto y\ f(y)$.

The non-drifting 1-d Maxwellian distribution is:

$$f(y) = \left(\frac{m}{2\pi KT}\right)^{1/2} \exp\left(-y^2\right)$$

where $y = v/v_{th}$, v is the velocity and v_{th} is the thermal velocity. Integrating this for the case of velocity parallel to the boundary, to obtain $F(y)$ produces:

$$F(|y|) = \frac{2}{\pi} \int_0^y \exp\left(-y^2\right) dy \equiv \mathrm{erf}(y)$$

In the case of the drifting plasma, the same $F(y)$ can be used as long as the drift velocity v_D is added to the resulting velocity.

For velocities perpendicular to the boundary, $vf(y)$ is normalized and integrated to find $F(y)$. For the non-drifting case:

$$F(y) = 1 - \exp\left(-y^2\right)$$

In the drifting case, the drifting Maxwellian must be used:

$$f(v) = \left(\frac{m}{2\pi KT}\right)^{1/2} \exp\left(-\frac{(v - v_D)^2}{v_{th}^2}\right)$$

and so

$$p(y) = (yv_{th} + v_D)\left(\frac{m}{2\pi KT}\right)^{1/2} \exp\left(-y^2\right)$$

where we now define that $y = (v - v_D)/v_{th}$. The resulting $F(y)$ is more complicated than before:

$$F(y) = \frac{1}{c}\left[-\frac{1}{2}\left(\exp(-y^2) - \exp\left(-\frac{v_D^2}{v_{th}^2}\right)\right) + \frac{v_D}{v_{th}}\frac{\sqrt{\pi}}{v_{th}}\left(\mathrm{erf}\left(\frac{v_D}{v_{th}}\right) + \mathrm{erf}(y)\right)\right]$$

$$\text{where} \quad c = \frac{1}{2}\exp\left(-\frac{v^2}{v_{th}^2} + \frac{v_D}{v_t h}\frac{\sqrt{\pi}}{2}\left(1 + \mathrm{erf}\left(\frac{v_D}{v_{th}}\right)\right)\right)$$

When $v_D = 0$, this equation becomes identical to the non-drifting case.

The expected flux Γ of new particles is found by integrating the differential number flux $\gamma(v)$ over all velocities.

$$\gamma(v) = nvf(v)$$

For a non-drifting Maxwellian:

$$\gamma(v) = \frac{nv}{\sqrt{\pi}v_{th}}\exp\left(-\frac{v^2}{v_{th}^2}\right)$$

$$\Gamma = \int_0^\infty \frac{nv}{\sqrt{\pi}v_{th}}\exp\left(-\frac{v^2}{v_{th}^2}\right)\mathrm{d}v = \frac{nv_{th}}{2\sqrt{\pi}}$$

For a drifting Maxwellian:

$$\gamma(v) = \frac{nv}{\sqrt{\pi}v_{th}}\exp\left(-\frac{(v - v_D)^2}{v_{th}}\right)$$

$$\Gamma = \frac{nv_{th}}{2\sqrt{\pi}}\exp\left(-\frac{v_D^2}{v_{th}^2}\right) + \frac{nv_D}{2}\left(\mathrm{erf}\left(\frac{v_D}{v_{th}}\right) + 1\right)$$

Again, when $v_D = 0$ this drifting case becomes identical to the non-drifting case.

There were numerous tests performed to see that these complicated algorithms produced sensible results in the code. Perhaps the most useful are tests of the stability of the number of particles in the population. In one test where the drift velocity was set to zero, a calibration mode was used in which there was no object in the simulation and particles were reflected from the lower boundary. In this case, the number of particles in the simulation remained stable over a large number of timesteps.

A similar test was performed with a drift velocity 8 times the thermal velocity in the -z direction. The same calibration mode was no longer appropriate and so particles were absorbed along the z=0 boundary by a conducting disk held at zero potential. The fact that no particles could enter at this boundary was not important because the drift velocity was sufficiently high compared to the thermal velocity that essentially no particles would enter from this boundary anyway. In this case too, the population remained stable over a large number of time steps.

5 A More General Shape Definition

The existing grid characteristics were left unchanged when we introduced a more general way of defining objects. This was to minimize the effects of the new changes on the existing code.

Previously, the target object was always a disk lying on the z=0 axis. A new method of defining objects has been defined. In this, an object may be built up from blocks the size of the grid cells. The object may be placed anywhere within the simulation grid as long each of its corners lies at the centre of a grid cell and its edges are parallel the r and z axes. Different surface materials may be defined beginning at one grid cell centre and extending to another. An example of a simple object is shown in figure 5. This illustrates the one exception to the rule that the corners must lie at the centre of grid cells. If the left hand edge of an object is defined at the centre of the first cell in the r direction, then the code automatically assumes that the user intends the object to extend all the way to the r=0 axis. This is necessary to ensure that objects without central holes may be defined.

An individual particle knows whether it is inside the obstacle or not by means of a 'logical grid'. In this system, each grid cell is divided up into 4 quarters that may be individually set to be 'metal' or 'vacuum', i.e. part of the object or not. Figure 6 illustrates the different combinations of metal and vacuum that can be found in one cell. Figure 7 shows what a simple cylinder looks like under this system. The main purpose of this system is for identifying particle impacts on the object. If a particle lies in a grid cell which is not all vacuum, then its trajectory is analysed to see if it has struck the object, and if so where. The charge of the macroparticle is then added to the surface and secondaries emitted as appropriate.

The potential solution is still performed using successive over-relaxation, but the solver has been rewritten. The basic problem is to solve the Poisson equation:

$$\nabla^2 \Phi = -4\pi\rho$$

or in cylindrical coordinates: $$\frac{\partial^2 \Phi}{\partial r^2} + \frac{1}{r}\frac{\partial \Phi}{\partial r} + \frac{\partial^2 \Phi}{\partial z^2} = -4\pi\rho$$

The finite element representation of this equation is:

$$4\Phi(r,z) - \Phi(r,z+h) - \Phi(r,z-h) - \left(1 + \frac{h}{2r}\right)\Phi(r+h,z) - \left(1 - \frac{h}{2r}\right)\Phi(r-h,z)$$

$$= 4\pi h^2 \rho(r, z)$$

where h is the grid cell length. The code takes account of the presence of an object inside the simulation space, by solving this equation first in a box enclosing the object and then in the space above, below and to either side of the object. The entire calculation is iterated many times until the required accuracy is achieved. Figure 8 shows an example of a potential solution around a simple cylinder. Although this shows the qualitative behaviour we expect, a more rigorous validation of this potential solver is being sought.

6 Summary

We started with a $2\frac{1}{2}$-D particle-in-cell code of known value in simulating a particular NASA Lewis laboratory charging experiment. In creating PICCHARGE, we have extended this code to treat the effects of magnetic fields and drifting plasmas, which are prominent features of the LEO environment. We have introduced a flexible and more general method of defining objects, so that a greater range of objects may be studied, with more surface materials. In addition, we can now examine differences in wake and ram characteristics. More, extensions to the code are planned to improve the simulation in terms of auroral electrons and material properties and to make comparisons with laboratory experiments easier.

7 References

[1] Martin A.R and Latham P.M ' Current European activities in Spacecraft/Plasma Interactions in Low Earth Orbit', This Conference.

[2] S.T.Brandon 'Numerical Simulations of Positively-Biased and Dielectric-Conductor Disks in a Plasma', Thesis, University of Kansas, 1984

[3] J.P.Boris, 'Relativistic Plasma Simulation - Optimization of a Hybrid Code', Proc. 4th Conf rence on Numerical Simulation of Plasmas, NRL, Washington D.C., 1970

[4] W.H.Press, B.P.Flannery, S.A.Teukolsky and W.T.Vetterling 'Numerical Recipes', Cambridge University Press, 1986

CAPTIONS

Figure 1. The 2-D spatial grid of the original code, showing one macroparticle and the conductor and dielectric at the lower edge of the simulation space. From Brandon [1].

Figure 2. Three trajectory plots with timesteps $10 \times \Delta t$, Δt and $0.1 \times \Delta t$. $\Delta t = 3.7 \times 10^{-9}$s. The trajectory with points most widely spaced had the longest timestep.

Figure 3. Calculated trajectory of a particle gyrating in a magnetic field. Input parameters: R-velocity $= -6 \times 10^7$cm/s, Z-velocity $= 3 \times 10^7$cm/s, Bz $= 6 \times 10^{-5}$T. The diameter of this gyration is 11.3713cm, compared with a theoretical value of 11.37136cm.

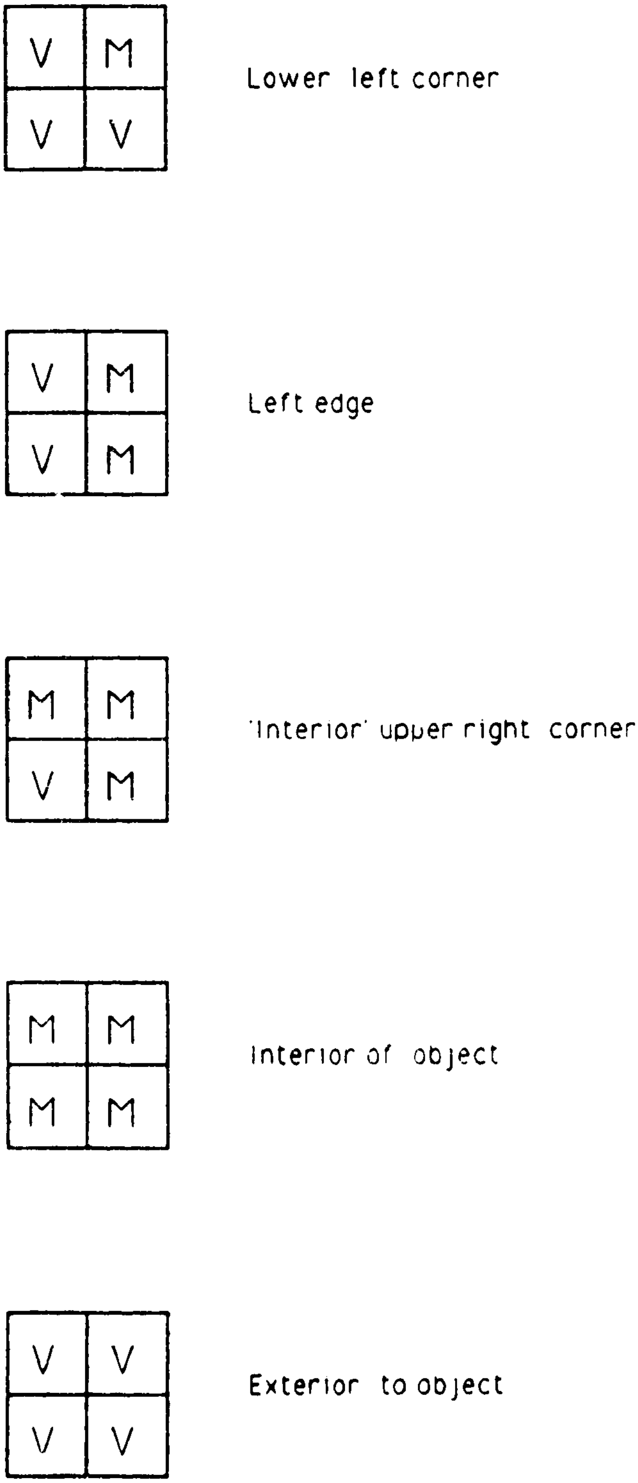
Figure 4. Tranformation methodfor generating a random deviate y from a known probability distribution $p(y)$. The indefinite integral of $p(y)$ must be known and invertible. A uniform deviate x is chosen between 0 lnd 1. Its corresponding y on the definite-integral curve is the desired deviate. (Adapted from [4].)

Figure 5. A simple object in the simulation grid. The coordinates describing the corners are shown.

Figure 6. Examples of the combinations of 'metal' and 'vacuum' that can occur in a single cell in the logical grid.

Figure 7. A simple cylinder as it appears to the logical grid system. Each surface cell has an associated number.

Figure 8. An example of a potential solution around a simple object.



V	M
V	V

Lower left corner

V	M
V	M

Left edge

M	M
V	M

'Interior' upper right corner

M	M
M	M

Interior of object

V	V
V	V

Exterior to object

Figure 6

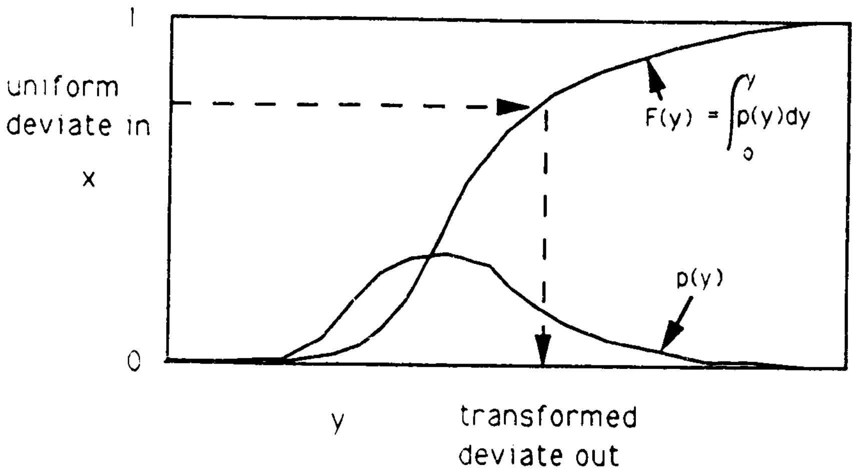


Figure 4

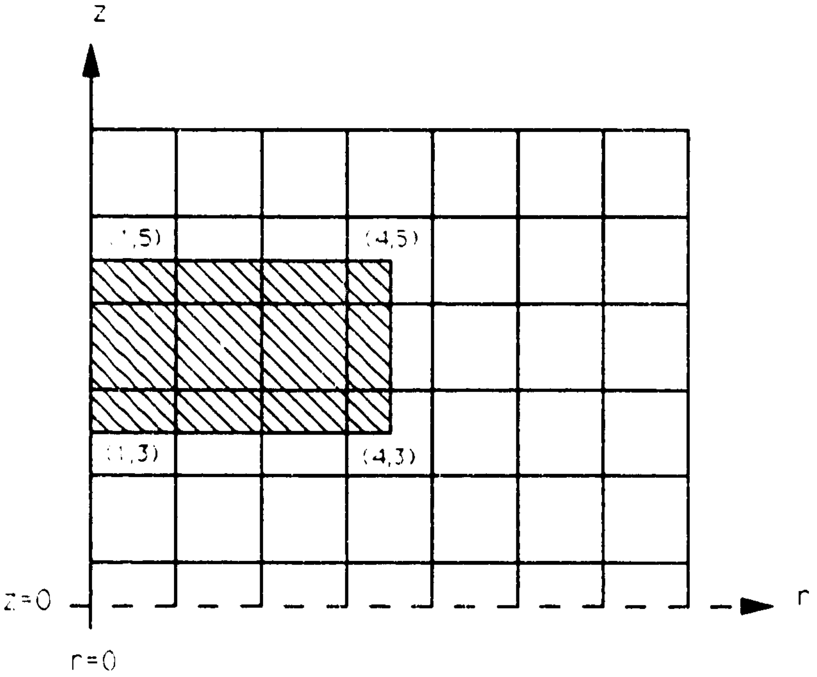


Figure 5

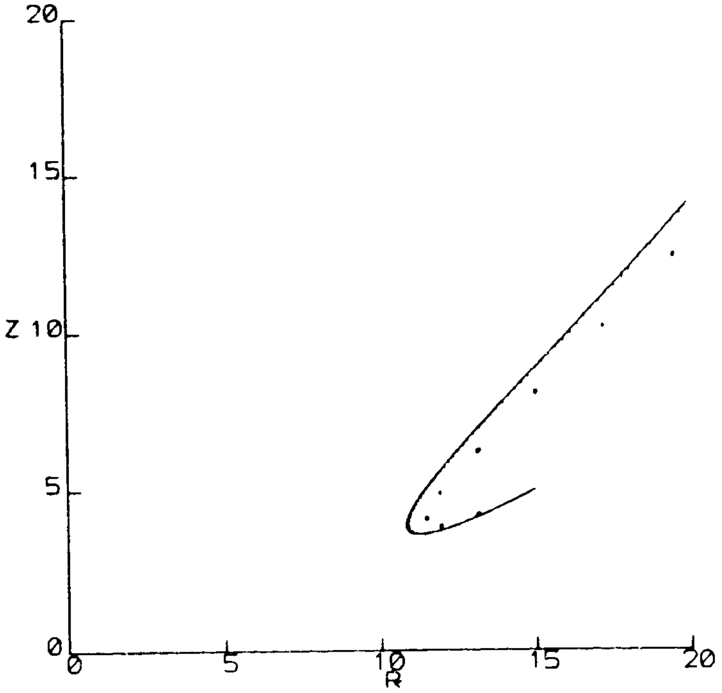


Figure 2

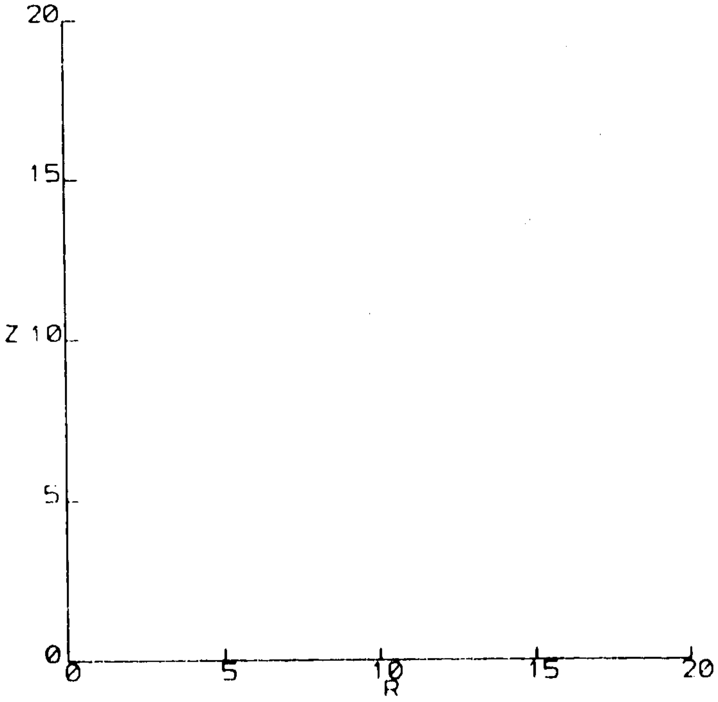


Figure 3

THE SIMULATION GRID

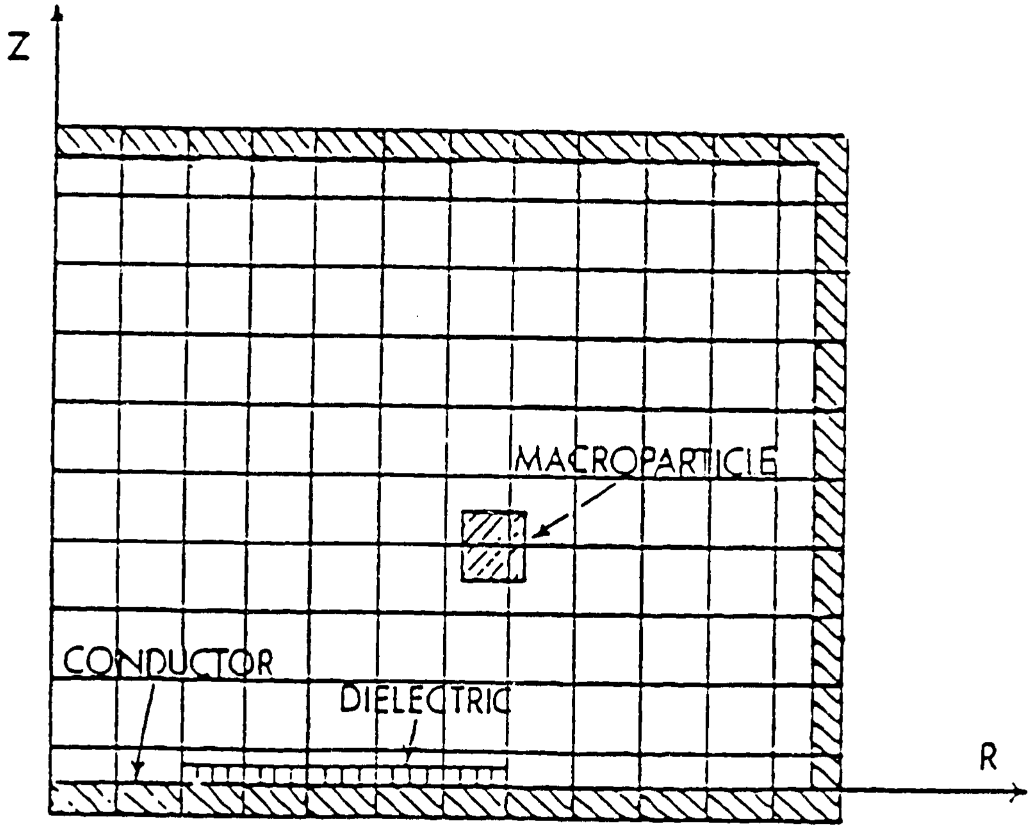


Figure 1

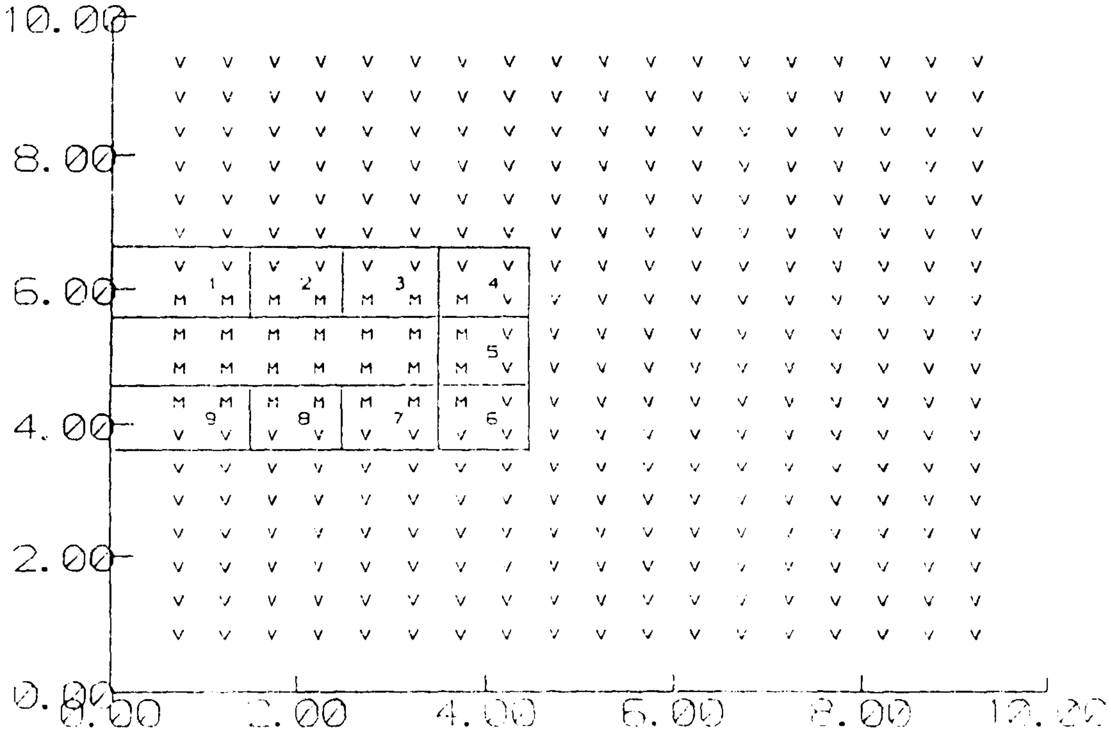


Figure 7

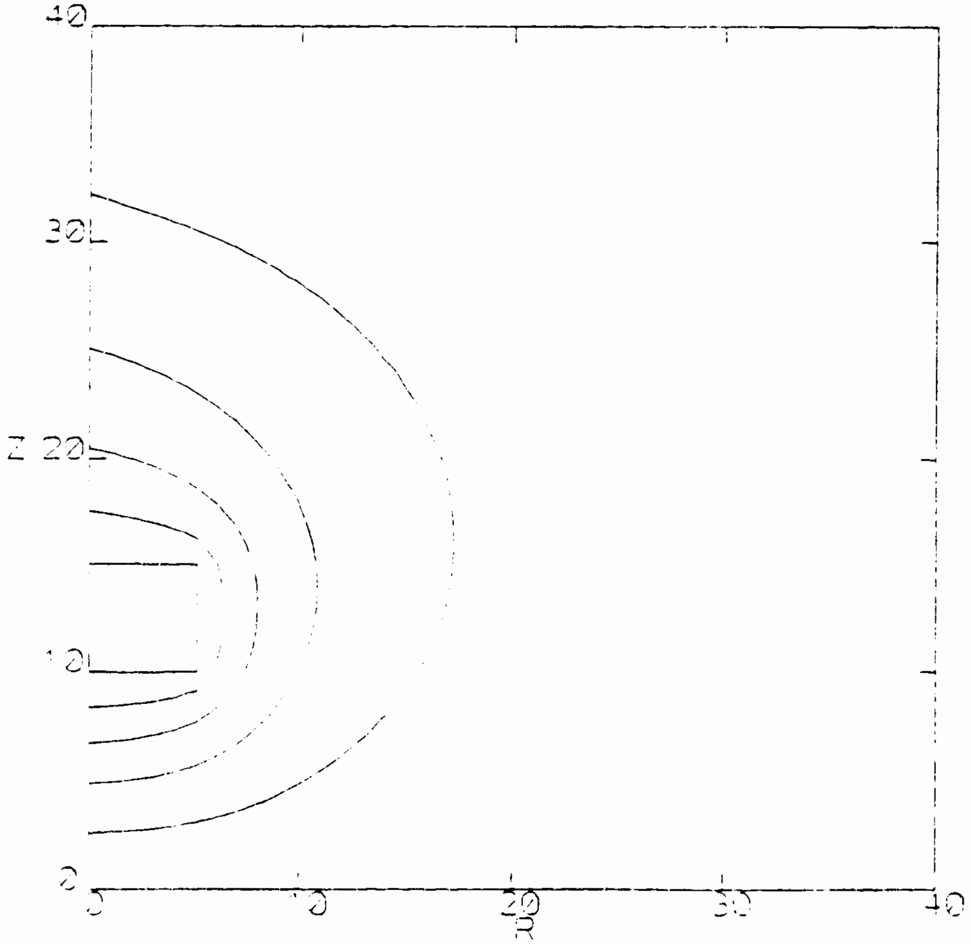


Figure 8