Chapter 3

TRISTAN

The 3-D Electromagnetic Particle Code

Oscar Buneman

3.1 Introduction

In principle, it should be possible to understand and use TRISTAN by (1) studying in depth the Hockney/Eastwood [1] and Birdsall/Langdon [2] texts, (2) learning KEMPO1 and (3) reading carefully the comments that are provided in the code itself. The following notes are intended to supply some background and guidance toward learning to apply TRISTAN to space plasma problems. As supplied, the code shows how the interaction of the solar wind with Earth's magnetic field may be simulated by TRISTAN.

Computer simulation means programming physics, not mathematics. In their most fundamental form the laws of physics are also most acceptable to computers: it is the higher-level derived equations that tend to cause problems. Maxwell's

$$\partial \boldsymbol{B}/\partial t = -\mathrm{curl}\boldsymbol{E} \qquad \text{and} \qquad \partial \boldsymbol{D}/\partial t = \mathrm{curl}\boldsymbol{H} - \boldsymbol{J} \tag{3.1}$$

as well as Newton-Lorentz'

$$\frac{\mathrm{d}m\boldsymbol{v}}{\mathrm{d}t} = q(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B}) \tag{3.2}$$

are already in typical "update" form: they are almost program statements as they stand. Only the transition from infinitesimals to finite differences calls for some tampering.

3.2 Field Update

To preserve space-time symmetries (and the reversibility of collisionless plasma physics), all finite-difference implementations of differential operators must be space- and time-centered. This also guarantees second order accuracy. $\boldsymbol{B}$ (or $\boldsymbol{H}$) data must be leap-frogged over $\boldsymbol{E}$ (or $\boldsymbol{D}$) data in time. Data must be staggered in space as illustrated in the TRISTAN comments. This illustration also demonstrates how the code really uses the most physical method of field processing, namely updating the fluxes of $\boldsymbol{B}$ and $\boldsymbol{D}$ through any cell face from the circulations of $\boldsymbol{E}$ and $\boldsymbol{H}$ respectivly around those faces, with charge flux to be included in the case of $\boldsymbol{D}$.

Exercise: Sketch a view of the complementary mesh to that shown in the comments, i.e. the mesh with cell corners and cell centers interchanged. Verify that the field component data are exactly available where needed for updating $\boldsymbol{D}$-fluxes through faces of this mesh from $\boldsymbol{H}$-circulations around those faces.

TRISTAN uses scales such that $\epsilon_o = 1$ and hence $\mu_o = 1/c^2$. This means $\boldsymbol{D} = \boldsymbol{E}$, with components denoted ex, ey, ez. Instead of recording components of $\boldsymbol{B}$ or $\boldsymbol{H}$, TRISTAN records components bx, by, bz of $c\boldsymbol{B}$ (alias $\boldsymbol{H}/c$). This makes for $e \longleftrightarrow b$ symmetry in Maxwell's equations. Throughout, TRISTAN uses a rectangular cubic grid with $\delta x = \delta y = \delta z = 1$ and time discretisation to $\delta t = 1$. Loops 7, 8 and 9 of the MAIN program then implement the field update. Loop 8 is a repeat of loop 7. Each uses $c/2$ in place of the "c" used in loop 9 which updates $\boldsymbol{E}$. This is because $\boldsymbol{B}$ must be updated in two half steps so that it is available at the same times as $\boldsymbol{E}$ for the particle update. Charge fluxes ($\boldsymbol{J}$) are subtracted from the updated $\boldsymbol{E}$- or $\boldsymbol{D}$-field later in the program.

By substituting for ex, ey, ez from bx, by, bz or vice versa one can verify that in vacuo each field component satisfies a wave equation in which second derivatives have been replaced by central second differences. This means that c must satisfy the Courant condition for stability: it must be less than $1/\sqrt{3}$. The use of $c = 0.5$ is recommended.

No display routines are provided in TRISTAN. Results for fields and particles are put out to disk at chosen time steps for post-processing as desired. When fields are to be displayed it is most important to realise that the three components of each field vector have not been recorded at the same places. To get the electric field components at location (i, j, k) one must form the following averages;

$$\begin{aligned} &\frac{ex(i-1,j,k) + ex(i,j,k)}{2} \\ &\frac{ey(i,j-1,k) + ey(i,j,k)}{2} \\ &\frac{ez(i,j,k-1) + ez(i,j,k)}{2} \end{aligned} \tag{3.3}$$

Exercise: write down the averages of b-array data which give the three b-components at location (i, j, k).

3.3 Particle Update

The time-centered finite difference version of the Newton-Lorentz particle update is:

$$\boldsymbol{v}^{new} - \boldsymbol{v}^{old} = \frac{q\delta t}{m}\langle \boldsymbol{E} + \frac{1}{2}(\boldsymbol{v}^{new} + \boldsymbol{v}^{old}) \times \boldsymbol{B}\rangle \tag{3.4}$$

$$\boldsymbol{r}^{next} - \boldsymbol{r}^{present} = \delta t \boldsymbol{v}^{new} \tag{3.5}$$

This shows that positions must be leap-frogged over velocities. As it stands, the equation for $\boldsymbol{v}^{new}$ is implicit. But since it is linear in the components of $\boldsymbol{v}^{new}$ an explicit solution can easily be written down and programmed, as was done by Hartree in 1942. Boris found a good physical interpretation of the steps in this explicit procedure:

$\boldsymbol{v}_o \longleftarrow \boldsymbol{v}^{old}$ where $\boldsymbol{v}_o = \boldsymbol{v}^{old} + q\boldsymbol{E}\delta t/2m$: half an electric acceleration
$\boldsymbol{v}_1 \longleftarrow \boldsymbol{v}_o$ where $\boldsymbol{v}_1 - \boldsymbol{v}_o = (\boldsymbol{v}_1 + \boldsymbol{v}_o) \times q\boldsymbol{B}\delta t/2m$: a pure magnetic rotation
$\boldsymbol{v}^{new} \longleftarrow \boldsymbol{v}_1$ where $\boldsymbol{v}^{new} = \boldsymbol{v}_1 + q\boldsymbol{E}\delta t/2m$: another half electric acceleration

The equation determining $\boldsymbol{v}_1$ from $\boldsymbol{v}_o$ is still implicit but its explicit form follows from: (1) dotting with $\boldsymbol{v}_1 + \boldsymbol{v}_o$ to check that the magnetic field does no work and that the magnitudes of $\boldsymbol{v}_1$ and $\boldsymbol{v}_o$ are the same, (2) dotting with $\boldsymbol{B}$ to check that components along $\boldsymbol{B}$ are the same, (3) crossing with $q\boldsymbol{B}\delta t/2m$ and substituting back, to give:

$$\boldsymbol{v}_1 = \boldsymbol{v}_o + 2\frac{\boldsymbol{v}_o + \boldsymbol{v}_o \times \boldsymbol{b}_o}{1 + b_o^2} \times \boldsymbol{b}_o \tag{3.6}$$

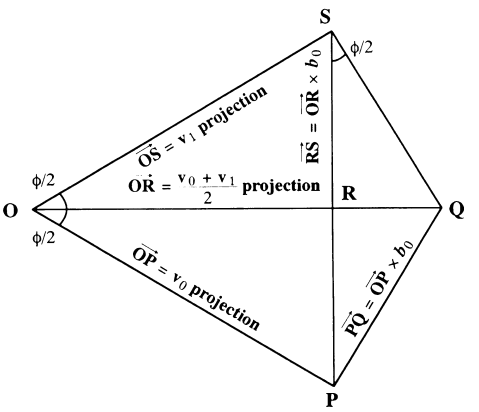
where $\boldsymbol{b}_o = q\boldsymbol{B}\delta t/2m$. See Hockney/Eastwood [1], ch 4-7-1 and Birdsall/Langdon [2], ch 4-4.

The following figure illustrates the magnetic rotation:

Figure 3.1 shows how the explicit procedure represents essentially two half-rotations in the magnetic field. The total angle of rotation, ϕ, is

$$\phi = 2\text{arctan}|\boldsymbol{b}_o| = 2\text{arctan}(qB\delta t/2m) \tag{3.7}$$

which is $qB\delta t/m$ if this is small. For large $qB\delta t/m$ the angle becomes less than $qB\delta t/m$. It maximises to π. For time steps large compared with a gyroperiod the particles perform helical motion in a constant magnetic field, or cycloidal motion in crossed constant electric and magnetic fields, but the phases will not be correct. When this is anticipated to lead to physically important errors, one must rescale so that gyroperiods are not too small.



$$RQ = RS \tan\frac{\phi}{2} = OR \tan^2\frac{\phi}{2}$$

$$OQ = OR(1+\tan^2\frac{\phi}{2})$$

$$\tan\frac{\phi}{2} = |\boldsymbol{b}_0| = \frac{\Omega_c\,\delta t}{2}$$

$$\vec{RS} = \vec{OR} \times \boldsymbol{b}_0$$

$$= \vec{OQ} \times \boldsymbol{b}_0 / (1+b_0^2)$$

$$v_1 - v_0 = \vec{PS} = 2\vec{RS}$$

$$= 2\vec{OQ} \times \boldsymbol{b}_0 / (1+b_0^2)$$

$$= 2\,\frac{\vec{OP} + \vec{OP} \times \boldsymbol{b}_0}{1+b_0^2} \times \boldsymbol{b}_0$$

$$= 2\,\frac{v_0 + v_0 \times \boldsymbol{b}_0}{1+b_0^2} \times \boldsymbol{b}_0$$

Figure 3.1: Projections into plane normal to $\boldsymbol{B}$.

The execution of the time-centered electric accelerations and magnetic rotations will be found at the end of the "MOVER" subroutine.

Exercise: Initialise only some 20 electron-ion pairs, at random positions and with random initial velocities, not exceeding 0.1 in magnitude. Initialise the fields to uniform crossed electric and magnetic fields: $ex = 0.2, ey = 0.2, ez = 0., bx = 0., by = 0., bz = 1.0$ Move the particles by calling the MOVER on them for 50 time steps without changing the fields. Record all positions at all time steps and display the orbits by the graphical method of your choice. Observe the cycloidal motions and note the angle of gyration of the electrons per time step. Use $qi = -qe = 1.0, me = 1.0, mi = 16.0, mx = 25, my = 25, mz = 25$.

3.4 Speed Limits

If a particle were to move through several cell meshes in one time step, it would miss available field information. Hence it is desirable to impose a speed limit on particles. With c already set to 0.5 we know that particles ought not to go more than half a mesh cell per time step on physical grounds. Quite apart from Einstein's speed limit, considerations of causality and stability motivate one to make sure that the particles do not outrun the fields. On looking for economical ways of applying brakes to particle motion, one finds that Einstein's method is the simplest, even if one had no other reason for making one's code relativistic. So Einstein's gammas have been incorporated in TRISTAN's particle mover.

One might argue that this is rather wasteful in the many applications of TRISTAN to plasmas which are essentially non-relativistic. However, square

roots are hardwired into many computers and are no costlier than divisions. Also, if electromagnetic effects are expected to be important, the Courant limit on $c\delta t$ will result in time scales such that the non- relativistic particles crawl through the mesh very slowly.

If electromagnetic effects are expected to be ignorable and fields can be treated as being communicated instantaneously, one could always choose an electrostatic and/or magnetostatic code and pay the penalty of an expensive static field solver. Perhaps the simplest way of dealing with nonrelativistic plasmas is to use TRISTAN with an artificial velocity of light, slower than the real velocity of light, but fast enough so that the particles perceive fields as propagated instantaneously. In this way one covers electromagnetic effects qualitatatively, although not with quantitative accuracy.

3.5 Interpolation

Throughout the code, linear interpolation is employed for subgrid resolution. This means that there is no stringent lower limit to the sizes of such quantities as gyroradii or Debye lengths. For quantities recorded on the integer mesh $x = i, y = j, z = k$, this means interpolating the eight nearest entries by applying weights as follows;

$$\begin{array}{llllllll}
(1-\delta x) & (1-\delta y) & (1-\delta z) & at & i & ,j & ,k \\
\delta x & (1-\delta y) & (1-\delta z) & at & i+1 & ,j & ,k \\
(1-\delta x) & \delta y & (1-\delta z) & at & i & ,j+1 & ,k \\
(1-\delta x) & (1-\delta y) & \delta z & at & i & ,j & ,k+1 \\
(1-\delta x) & \delta y & \delta z & at & i & ,j+1 & ,k+1 \\
\delta x & (1-\delta y) & \delta z & at & i+1 & ,j & ,k+1 \\
\delta x & \delta y & (1-\delta z) & at & i+1 & ,j+1 & ,k \\
\delta x & \delta y & \delta z & at & i+1 & ,j+1 & ,k+1
\end{array} \tag{3.8}$$

where $\delta x = x - i, \delta y = y - j, \delta z = z - k$. This is the three-dimensional version of the familiar "area weighting" for two dimensions. If one draws the cubic mesh which has the integer grid points as centers (not as corners), and if one treats a particle as a uniform cube, the size of one cell, with the nominal particle coordinates as center, each entry is weighted by the volume of the rectangular block of the particle which is intercepted by the cell belonging to the meshpoint of the entry. We refer to this as "volume weighting".

If a charge density record is to be accumulated on this mesh, the amount which each particle contributes to each entry will also be exactly its charge multiplied by these volumes. If an electrostatic potential were to be constructed (assuming the code had reached some static equilibrium), the potential at the location of the particle would be obtained from the grid point entries in the same manner. Notice, however that the field components, obtained by differencing

the potentials, would then be obtained at the mid-points along their respective axes, just as shown for ex, ey and ez in the diagram in the code.

Conversely, given the field component record at the locations shown, one must first average between the entries $ex(i,j,k)$ and $ex(i-1,j,k)$, and correspondingly for ey and ez, to give the value of ex at $x = i, y = j, z = k$, before applying the weights shown above. The averaging and weighting can be conveniently combined. In the case of the $\boldsymbol{B}$-components one has to average in two dimensions before weighting. All this is done in the MOVER subroutine prior to the acceleration and rotation procedures. The interpolations take most of the computer time.

Exercise: Do the previous exercise with zero electric field ($ex = 0., ey = 0., ez = 0.$) and linearly varying magnetic field: $bx = 0., by = 0., bz(i,j,k) = -2. + 4. * i/mx$. Observe the "grad B drift" (misnomer: should be "curl $\boldsymbol{B}$ drift"). Do ions and electrons drift in the same direction? Will there be a current due to different drifts? Is this current so oriented that it could account for the curl of $\boldsymbol{B}$? Observe the occurrence of a drift even for particles that do not move out of a cell.

3.6 Charge Fluxes

TRISTAN does not employ a charge density array as such: only charge fluxes, i.e. the amounts of charge flowing through the faces of a cubic mesh, are needed. The cubic mesh in question is the complementary mesh to that of the diagram shown in the program comments. This complementary mesh is used when the fluxes of $\boldsymbol{E}$ (or $\boldsymbol{D}$) through cell faces are updated by the circulations of $\boldsymbol{B}$ (or $\boldsymbol{H}$) around them. (See the first exercise above.) According to Maxwell, the fluxes of charge must then be deducted for a full update of the $\boldsymbol{D}$-fluxes.

One finds that a particle which does not leave its cell during a move (i.e. a particle which retains its i,j,k) will transport charge through twelve faces, four for each orientation. How much charge is readily calculated assuming the particle performs a straight move (see ref. [3]). This charge is subtracted from the $\boldsymbol{E}$-components in the "DEPSIT" subroutine. If a particle moves into a neighboring cell during its move, one splits the move into two parts, one for each cell, and deposits the charge fluxes separately. This splitting is done in the three nested "SPLIT" routines before the DEPSIT routine is entered.

3.7 Smoothing

By making each particle into a cube, rather than treating it as a delta- function, we are already eliminating much of the unnatural noise created by coarse-graining the (physically very fine-grained) population of particles. As a further step toward smoothing, TRISTAN uses a primitive form of filtering, namely

averaging over adjacent neighboring cells in all three dimensions. Any source to Maxwell's equation, i.e. any charge flux added in the DEPSIT routine, is divided into four parts only two of which are kept in the cell, the other two are deposited in the cells to each side. This amounts to "convolving" the flux array with the sequence 0.25, 0.5, 0.25 and it is done in each of the three dimensions.

Translated into Fourier space, this amounts to applying a low-pass filter which eliminates the badly aliassed harmonics (the π-modes). In 3-D, each charge flux is in fact spread over $3*3*3$ neighboring cells and the 27 weights are recorded once and for all in the smoothing array sm(27) while ms(27) contains the corresponding index displacements. This accounts for the loop at the end of the DEPSIT routine. In several places it was found convenient to treat the originally three-dimensional field arrays as one-dimensional and to distinguish between the three physical dimensions by means of the index "strides" (ix, iy, iz) used by FORTRAN.

The DEPSIT subroutine takes most of the computer time in each time step. This is because each particle references at least $12 * 27 = 324$ field array data. The deposit procedure cannot be vectorised in the particles (unless the particles are carefully sorted, or unless some extra storage arrays are kept). Fortunately, the smoothing loop (of length 27) can be vectorised on CRAY machines which offer the "gather-scatter" instruction. However, the compiler must be helped over its inhibitions to use this instruction by giving the directive "cdir$ ivdep" which makes it ignore the possibility of vector dependencies (which we know will not occur). Compilers on other machines ignore this directive as an apparent comment line. The same directive will be found in several places in TRISTAN. The loop over the particles in the mover is vectorised (without persuasion) by the CRAY compiler.

While TRISTAN does not keep a record of charge density, div $\boldsymbol{D}$ or div $\boldsymbol{E}$ can always be calculated as the outflow of $\boldsymbol{D}$ or $\boldsymbol{E}$ from each unit cube, $ex(i,j,k) - ex(i-1,j,k) + ey(i,j,k) - ey(i,j-1,k) + ez(i,j,k) - ez(i,j,k-1)$. However, when post-processing TRISTAN runs, one is usually most interested in the densities of ions and electrons separately. To get those, one has to go through the ion or electron array and deposit according to the volume weighting rule shown above. For this purpose, we append a DENSTY subroutine which will create each separate density array and which also applies the smoothing algorithm, in conformity with TRISTAN's smoothing of all sources of fields.

Exercise: run TRISTAN as supplied through time step 128 (with a break at or before step 64). Use DENSTY on the last particle output to get the electron and ion density distributions separately. Compare their difference with div $\boldsymbol{E}$ as obtained from the last field output. Limit your comparison to the interior (see section after next for boundary problems).

3.8 Sorting and Localisation

The TRISTAN version presented here does not keep the particles sorted. On some highly parallel machines it may become desirable to keep particles and fields in memory locations related closely to their physical locations. In this case the SPLIT routines could be modified to do the necessary re-indexing of each particle that has left its cell. It should be noted that with such ordering all data traffic in TRISTAN is strictly local: only nearest neighbor information is needed to update all quantities. Eliminating Poisson-solves was the most important step towards such localisation: the solution of Poisson's equation anywhere depends on the charges everywhere.

The present version of TRISTAN does a very limited form of sorting: it determines which particles have left the total domain allowed for particles and deletes them from memory. The decision whether a particles is "in" or "not.in" is made in the SPLIT subroutines.

3.9 Boundary Conditions: Particles

Other than absorption of particles at the boundaries, a user may want to apply periodicity or reflection (specular or inelastic). Some guidance for these alternatives will be found in the comments. An important feature of absorption is that "absorbed" particles leave their charge behind on the boundary. Such boundary charging makes up for the fact that the written-off particles are, in reality, only just a little way beyond the boundary and the simulated plasma ought to continue to "feel" them.

One finds that on average as many ions escape as electrons, so the total boundary charge remains moderate. However, the immobility of the boundary charges is somewhat non-physical and in the sample version of TRISTAN the boundaries have been made slightly conducting: see the "CONDUCT" subroutine. Surface currents proportional to the surface fields are there subtracted from those fields, with the spread, or "smoothing", which we apply to all current sources.

Boundary charging may also occur when influx of particles from the exterior is programmed. Suppose the exterior is taken to be uniform hot plasma, then the electron flux will exceed the ion flux and one cannot introduce the incoming particles as neutral pairs. Our charge-conserving code keeps track of all charge movements and does not allow the non-physical process of creation of a single charge of one sign. The code would imply the simultaneous creation of a charge of the opposite sign at the same location. Thus for each electron created at the boundary and injected inwards the code assumes that a positive charge has been created at the same spot and if this positive charge is not tracked, the code implies it to remain immobile.

So injection of charge from the boundary will cause boundary charging of

the opposite sign to absorption of charge. The imbalance due to an excess of electron inflow over ion inflow is, in general, compensated by a similar imbalance of the outflow.

In the solar wind application such an influx of solar plasma is programmed across the four "lateral" boundaries which are parallel to the direction of the wind: those fluxes are purely "thermal". The wind is very supersonic, so the main supply of solar plasma is across the sun-facing boundary and there one can create ions and electrons as pairs with the same drift but different random velocities. On the tail surface one only has absorption.

The normal velocity components of particles entering across the lateral boundaries are picked from a short pre-calculated table so that their distribution becomes half-maxwellian. Elsewhere when generating thermal velocities we use the cheap method of adding three random numbers, each uniformily distributed between $-.5$ and $+.5$, and multiplying by twice the rms velocity for each component. Users may want to add more than three randoms in order to create distributions closer to gaussian then the "three-dice" curve. The call to a uniform random number generator (random between 0. and 1.) is "ranf()" on CRAYs. The user must change this to the appropriate call on her/his machine. In some cases random number generators must be initialised with a "seed" and it is necessary to carry this seed forward through any interruptions of a run.

3.10 Boundary Conditions: Fields

The range of particle co-ordinates is $3. < x < mx - 2., 3. < y < my - 2., 3. < z < mz - 2$. This leaves buffer zones all round in which fields can be recorded and updated and thus provides room for applying field boundary conditions of one's choice, independently of the particles.

The simplest boundary condition for fields is periodicity. Programming this option into TRISTAN is left to the user: the buffer zones can then be employed for recording repeats needed when particles interpolate the fields, and for accumulating charge fluxes which spill over in the DEPSIT subroutine.

Specular reflection of fields is a problematic option in a fully electromagnetic and relativistic code: it would imply infinite conductivity at the field boundary, with zero skin depth, something rather non-physical. A surface with finite conductivity could be programmed as several layers to which CONDUCT is applied and holding the fields down to zero beyomd several skin-depths.

In space applications the most relevant boundary condition for the fields is that they should be able to radiate away into space and should not be reflected at the boundaries. Lindman [4] addressed the problem of a radiation-absorbing boundary condition for the scalar wave equation. TRISTAN uses, in the SURFACE subroutine, an equivalent of Lindman's lowest-order condition for Maxwell's field equations. Simple radiation absorbing conditions cannot be exact, but Lindman's higher-order conditions are very nearly perfect.

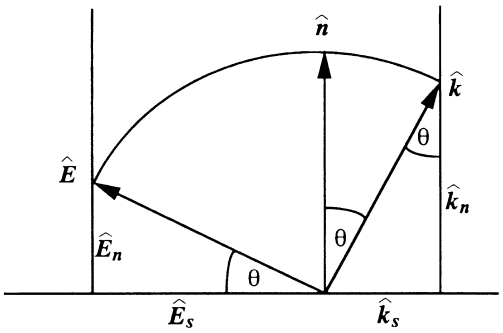


Figure 3.2: $\boldsymbol{B}$ toward viewer.

The problem is that the decision as to what is "outgoing" involves non-local information or some kind of wave-analysis of the fields. TRISTAN leaves the interpretation of field data in terms of "waves" to post-processors. Fortunately, the discrimination against "incoming" radiation can be based on local field information when one knows that the angle of incidence is not too far from normal. If the angle of incidence is exactly normal, then:

$$\frac{1}{c}\frac{\partial}{\partial t} = -\frac{\partial}{\partial n} \tag{3.9}$$

for outward signals, in application to any field variable ("n" is the direction of the outward normal).

When the angle of incidence (relative to the outward normal) is θ one starts by distinguishing between two cases, namely (i) the polarisation is such that $\boldsymbol{B}$ is in the surface and (ii) the polarisation is such that $\boldsymbol{E}$ is in the surface: see Figures 3.2 and 3.3. Our wave description assumes a phase factor $e^{i\omega t - i\boldsymbol{k}\cdot\boldsymbol{r}}$. Subscripts "n" and "s" denote normal and surface components. $\hat{\boldsymbol{E}}, \hat{\boldsymbol{B}}, \hat{\boldsymbol{k}}$ and $\hat{\boldsymbol{n}}$ are unit vectors along $\boldsymbol{E}$, $\boldsymbol{B}$, $\boldsymbol{k}$ and the normal. When $\boldsymbol{B}$ is in the surface:

$$\hat{\boldsymbol{B}}_s = \hat{\boldsymbol{B}} = \hat{\boldsymbol{k}} \times \hat{\boldsymbol{E}} = \frac{\hat{\boldsymbol{k}}_s}{\sin\theta} \times \frac{\hat{\boldsymbol{E}}_n}{\sin\theta} \tag{3.10}$$

$$k\boldsymbol{B}_s = \frac{1}{\sin^2\theta}\boldsymbol{k}_s \times \boldsymbol{E}_n \tag{3.11}$$

When $\boldsymbol{E}$ is in the surface:

$$\hat{\boldsymbol{B}}_s = -\hat{\boldsymbol{k}}_s B\cot\theta \tag{3.12}$$

$$k\boldsymbol{B}_s = -\boldsymbol{k}_s B\cot\theta = \frac{-\cos\theta}{\sin^2\theta}\boldsymbol{k}_s B_n \tag{3.13}$$

For general polarisation we add the two cases and get for the total $k\boldsymbol{B}_s$:

$$k\boldsymbol{B}_s = \frac{\boldsymbol{k}_s \times \boldsymbol{E}_n - \cos\theta \boldsymbol{k}_s B_n}{(1-\cos\theta)(1+\cos\theta)} \tag{3.14}$$

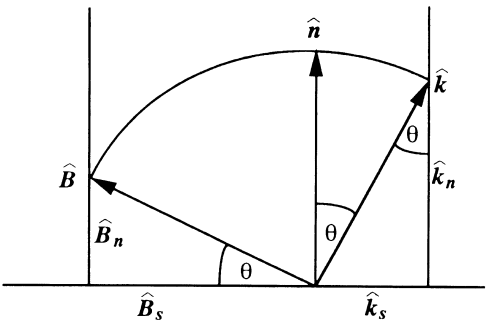


Figure 3.3: $\boldsymbol{E}$ away from viewer.

Multiply this by $i(1-\cos\theta)$ and identify $ik = i\omega/c = \frac{1}{c}\frac{\partial}{\partial t}$, $i\boldsymbol{k} = -\nabla$ and $ik\cos\theta = -\frac{\partial}{\partial n}$, then:

$$(\frac{1}{c}\frac{\partial}{\partial t}+\frac{\partial}{\partial n})\boldsymbol{B}_s = \frac{\cos\theta}{1+\cos\theta}\nabla_s B_n - \frac{1}{1+\cos\theta}\nabla_s\times\boldsymbol{E}_n = c_B\nabla_s B_n - c_E\nabla_s\times\boldsymbol{E}_n \tag{3.15}$$

where $c_B = \frac{1}{2}-\frac{1}{2}\tan^2\frac{\theta}{2}$ and $c_E = 1-c_B = \frac{1}{2}+\frac{1}{2}\tan^2\frac{\theta}{2}$. This establishes an "update" formula for $\boldsymbol{B}_s$ using only local derivatives, but it assumes knowledge of θ.

For those who prefer analytical derivations, the formula can be obtained from Maxwell's equations, taking z along the normal as follows;

$$\frac{1}{c}\frac{\partial B_x}{\partial t} = \frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y} \qquad\qquad \frac{1}{c}\frac{\partial E_y}{\partial t} = \frac{\partial B_x}{\partial z} - \frac{\partial B_z}{\partial x} \tag{3.16}$$

Add:

$$(\frac{1}{c}\frac{\partial}{\partial t}-\frac{\partial}{\partial z})(B_x+E_y) = -\frac{\partial B_z}{\partial x} - \frac{\partial E_z}{\partial y} \tag{3.17}$$

Subtract:

$$(\frac{1}{c}\frac{\partial}{\partial t}+\frac{\partial}{\partial z})(B_x-E_y) = \frac{\partial B_z}{\partial x} - \frac{\partial E_z}{\partial y} \tag{3.18}$$

Use

$$\frac{1}{c}\frac{\partial}{\partial t}+\frac{\partial}{\partial z} = \frac{1-\cos\theta}{1+\cos\theta}(\frac{1}{c}\frac{\partial}{\partial t}-\frac{\partial}{\partial z}) = \tan^2\frac{\theta}{2}(\frac{1}{c}\frac{\partial}{\partial t}-\frac{\partial}{\partial z}) \tag{3.19}$$

and substitute from (3.17):

$$(\frac{1}{c}\frac{\partial}{\partial t}+\frac{\partial}{\partial z})(B_x+E_y) = -\tan^2\frac{\theta}{2}(\frac{\partial B_z}{\partial x}+\frac{\partial E_z}{\partial y}) \tag{3.20}$$

Add (3.18) and (3.20) and divide by 2:

$$(\frac{1}{c}\frac{\partial}{\partial t}+\frac{\partial}{\partial z})B_x = (\frac{1}{2}-\frac{1}{2}\tan^2\frac{\theta}{2})\frac{\partial B_z}{\partial x} - (\frac{1}{2}+\frac{1}{2}\tan^2\frac{\theta}{2})\frac{\partial E_z}{\partial y} \tag{3.21}$$

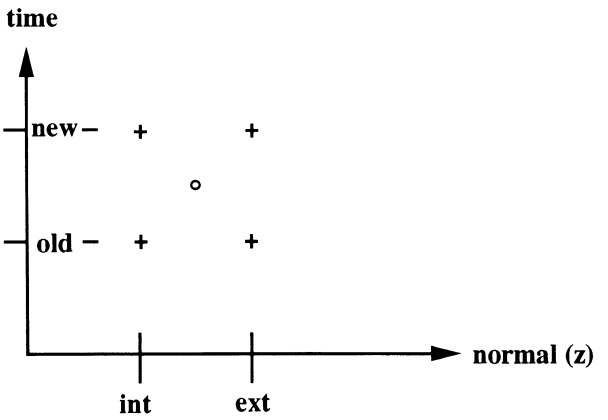


Figure 3.4: Update of $\boldsymbol{B}$.

and similarly for B_y.

Since k_s, B_n and E_n are each proportional to $\sin\theta$ the θ-dependence of c_B and c_E causes only fourth order effects for small θ. Moreover, our formula gives strictly correct answers for normal incidence, no matter what values are chosen for c_B and c_E.

While $c_B = c_E = 0.5$ would give the highest accuracy for infinitesimal θ, we have frozen c_B and c_E to their values for 45 degree incidence: $\frac{1}{2}\tan^2\frac{\theta}{2} = 1.5-\sqrt{2}$, $c_B = .4142$, $c_E = .5858$. Then the formula, while strictly correct for $\theta = 0$ and $\theta = 45$, is still surprisingly good at angles in-between. For larger angles, approaching grazing incidence, one gets unwanted reflections. These could be reduced by means of higher-order Lindman methods, translated to Maxwell's equations. Their implementation calls for maintaining extra arrays of surface data.

Note that the update of surface components requires information on the normal component of $\boldsymbol{B}$. Note also that if loops 7 and 8 were written separately for each component, one of the three indices could in each case be taken all the way to the end (instead of stopping just short of the end).The end value of the normal component is therefore available directly from Maxwell's equations. This is, in fact, carried out inside the SURFACE subroutine.

For the actual update of $\boldsymbol{B}$ we use the central-difference representation of the differentials in the formula at a layer midway between the outermost and first internal layer, and at the half-time between "old" and "new" for $\boldsymbol{B}$-data.

$$
\begin{aligned}
(\frac{1}{c}\frac{\partial}{\partial t} + \frac{\partial}{\partial z})B_x \longrightarrow \quad & \tfrac{1}{2c}(B_x^{new,ext} + B_x^{new,int} - B_x^{old,ext} - B_x^{old,int}) \\
& + \tfrac{1}{2}(B_x^{new,ext} + B_x^{old,ext} - B_x^{new,int} - B_x^{old,int})
\end{aligned}
$$

so that

$$B_x^{new,ext} = B_x^{old,int} + \frac{1-c}{1+c}(B_x^{old,ext} - B_x^{new,int})$$
$$+ \text{ term proportional to}$$
$$\text{the x} - \text{difference of } B_z$$
$$\text{at half} - \text{time}$$
$$+ \text{ term proportional to}$$
$$\text{the y} - \text{difference of } E_z$$
$$\text{midway between "ext" and "int".}$$

There is a corresponding formula for $B_y^{new,ext}$.

Surface components of $\boldsymbol{B}$ must be obtained in this manner at the top ends of the B–arrays. In the case of $\boldsymbol{E}$ one uses exactly the same method at the bottom end, $i = 1, j = 1, k = 1$. Because of the $e \longleftrightarrow b$ symmetry in our units, the same subroutine can be called for this purpose. Our method assumes that there is vacuum at and beyond the boundaries: the buffer zones permit this assumption. If one wanted to simulate an exterior uniform plasma, magnetised or not, one would have to redo the theory.

On the edges of the rectangular boundary box one runs into a few problems which are most easily dealt with by advancing the edges in two stages, by means of PRELEDGE before updating interior and surfaces and then by POSTEDGE afterwards.

One great benefit from the radiation absorbing boundaries is the fact that one can create static fields dynamically. One just switches on the sources and lets the transients escape to inifinity. Switching the sources on gradually and smoothly will reduce the transients themselves. In the supplied version of TRISTAN the Earth's field is created in this manner. A simple ring current of strength "o" is made to circulate around the Earth's center. The current is smoothed, like all current sources, in space. It is ramped up very smoothly in time, following a cubic, by building its differences $o1, o2, o3$, until o levels off to its final value. After that, the current is held fixed. The Earth's static field then remains while small electromagnetic transients take, typically, less than 100 steps to dissipate.

Static electric fields due to a distribution of charges with nett total charge value zero can be created similarly: one dissociates the charges and moves them into their positions gradually. One then keeps the charges in their fixed locations and lets electromagnetic transients radiate away. The static electric field due to the charges remains.

Exercise: Run TRISTAN without particles, injecting only the Earth's ring current as in the version supplied (62 steps rise time). Continue for another 200 steps and record the fields. Check that the magnetic field has its proper dipole structure and that the electic fields are very small, even in the central

region where the ring current is flowing. Check also that the magnetic field is essentially a gradient field except in the location of the ring current.

Exercise: Run TRISTAN without particles and inject, instead of the ring current, a current which zig-zags along mesh edges from $(i1, j1, k1)$ to $(i2, j2, k2)$. Make this current rise and decay gently, following a bell-shaped function of time (a quartic is simplest). After the current has stopped, run for another 200 steps. Now record the fields and check that one has the electrostatic field due to a negative charge at $(i1, j1, k1)$ and a positive charge at $(i2, j2, k2)$. Check that in this case the magnetic field is very small and that the electric field is a gradient field.

3.11 Initialisation

Poisson's equation is not explicitly solved while running TRISTAN: it is only "carried forward" by making the changes of $\boldsymbol{D}$ exactly consistent with the changes of the charge density. Therefore TRISTAN must be started from a state in which Poisson's equation is strictly satisfied. Also, $\mathrm{div}\boldsymbol{B} = 0$ must be ensured initially: TRISTAN conserves this condition by ensuring that only curls (of $\boldsymbol{E}$) are used to change $\boldsymbol{B}$.

Fortunately, most applications call for some simple initial state in which these divergence condidtions are trivially satisfied. Initially neutral plasmas in uniform electric and magnetic fields are typical examples. The uniform fields can be initialised as such and the neutral plasma can be set up by creating electrons and ions as pairs in the same locations but with different initial velocities so that dissociation takes place in the first step.

In the TRISTAN version supplied, initially uniform solar wind plasma is created in zero initial electric and magnetic fields. A uniform magnetic field of any orientation could be initialised instead, to simulate an IMF. The initial solar wind is a uniform thermal plasma with drift (which is "supersonic"). When initialising particle velocities, it is important to make sure that they are less than c. For a properly relativistic plasma it is best to create maxwellian distributions of momenta (by means of the "three-or-more-dice" technique) and then derive the velocities from the initial momenta.

As shown above, fields due to static currents or charges are most simply created from nothing by building up the sources gradually: that is how Earth's magnetic field is created in the TRISTAN version supplied. This immediately blows a hole in the solar wind until finer detail of the magnetopause develops, with penetration of solar plasma through the "cusps".

A version of TRISTAN is available in which a self-consistent current column with Bennett [5] profile is initialised. The initial magnetic field is there derived from a vector potential to ensure zero divergence. The boundary conditions are mixed in this version: periodic in the direction of the current, absorbing

laterally. This version is useful for studying beam-plasma interactions. It can be mailed electronically upon request.

The choice of the important parameters $mx, my, mz, maxprt, qe, qi, me, mi$ is governed by the following considerations: the proportions between the field array dimensions, $mx : my : mz$, can be chosen according to the physics to be simulated. Their absolute values are governed by core size or core allocation. Field and particle information should be comparable, so $maxprt$ should be of the same order of magnitude - or preferably a little larger than - the product $mx * my * mz$. Typically, for the simulation of a uniform plasma one would want at least one electron-ion pair per cell.

Regarding qe, qi, me, me: the ratios qe^2/me and qi^2/mi are important. The absolute values of qi and/or qe essentially only control the scales on which the fields are measured. Mostly $qe = -qi$ is a convenient choice because then a pair of computer-electrons and computer-ions is neutral. The ratio qe^2/me, multiplied by the electron density (numbers per cell), equals ω_{pe}. With $\delta t = 1$, we know that ω_{pe} must be less than 2 to avoid a familiar numerical instability. Since one knows the highest density of the initial plasma, one can choose qe^2/me accordingly. If, in the course of a run, the plasma develops higher density spots, this need not become a cause of worry: the density would have to be high over an extended region for the numerical instability to occur.

To use an actual physical ratio for mi/me would be unwise because of the enormous disparity of electron- and ion-time scales. In most applications one's interest is in the behaviour of one species in the presence of the other and provided their time scales differ enough to distinguish who is doing what, the magnitude of the disparity of the scales will be unimportant. Useful results have been obtained with the low mass ratio of 16:1, as in the TRISTAN version supplied. This could be interpreted as simulation with electrons which are "heavy", yet light enough compared with the ions to exhibit such features of electrons as being tightly tied by magnetic lines and very mobile along the lines - thus providing rapid neutralisation. Such electrons can also exhibit instabilities and create turbulence or heating, yet on a coarser scale than real electrons. Alternatively, one may interpret the choice of a 16:1 mass ratio as a simulation with ions which are "light", yet heavy enough to create only a slowly varying background to electron phenomena. Such ions have a large enough gyro-radius to transport information across the magnetic lines. They also serve to maintain any non-uniformities along the magnetic lines. Some users may want to indulge in the luxury (in terms of computer time) of a mass ratio 32:1, but the expense of a mass ratio 64:1 can rarely be justified. Incidentally: there is no reason why this ratio should be a power of 2.

3.12 Postprocessing

When running TRISTAN one finds that space allocation is more of a problem than time allocation. "Space" here means both core space and disk space, not physical space! As with physical observations, data acquisition is only the first stage: data storage, data analysis, graphical display and intepretation remain as post-simulation tasks.

It was found expedient to keep control of the volume of output while running TRISTAN by specifying the time step for the next full data dump and by re-starting the run from each of these dumps. This offers the opportunity of taking at least a superficial look at the output before continuing. A full dump each timestep is quite impractical. If one knows beforehand what particular information one wants out of a run - typically orbits of some chosen particles - one could do the necessary partial dumps at fixed intervals. However, first runs are often full of surprises and re-running with a new choice of output frequency and output material is then the most economical method.

Two-dimensional displays of particle densities (such as generated by the DENSTY subroutine) in a few chosen slices has been found useful and re-assuring as regards the credibility of the simulation. Projections of the magnetic vectors in the middle of those slices was helpful, too. But TRISTAN output is a happy hunting ground for a variety of advanced graphics methods, particularly in view of the three-dimensionality of the material.

Our physics education has taught us to think in terms of waves and their interaction. We are familiar with dispersion, growth and decay, reflection and absorption - features of linear systems. We try to describe non-linear phenomena by wave-wave interaction, i.e. as small perturbations of linear behavior. Wave analysis of data is expensive: TRISTAN does not use transforms since they are non-local and costly. Transforming in time is only possible as a post-processing operation and it requires a record for each time step.

If one wants to see waves in the output of a simulation one must first decide what variable(s) to subject to wave analysis. One must allocate core space for performing the spatial transforms of the chosen variables at each time step and put out the wanted spatial harmonics at each time step. Then one can post-process by transforming each spatial harmonic in time and studying the amplitudes and phases of the waves in the different variables. The necessary additions to TRISTAN are possible, but not simple.

When a simulation code reproduces observed phenomena (such as the formation of our magnetosphere and magnetotail from solar wind impinging on Earth's dipole) one can conclude that there is no longer any mystery in what we see, but whether we have "explained" or "understood" the phenomena remains arguable. The purpose of simulation is not to "explain" but to offer a facility for controlling input parameters and making predictions, rather than having to wait for variation of input to occur naturally. Moreover, simulation

offers the opportunity of going over the same data with a variety of different diagnostics or, if needed, repeating a run with the same input but recording some different variables as continuous output.

Appendix: The density subroutine

```
      subroutine DENSTY(n1,n2,maxptl,x,y,z,mx,my,mz,rh)
      dimension x(maxptl),y(maxptl),z(maxptl),rh(mx,my,mz)
      do 2 k=1,mz
      do 2 j=1,my
      do 2 i=1,mx
    2 rh(i,j,k)=0.
C  Volume weighting:
      do 1 n0=n1,n2
      i=x(n0)
      dx=x(n0)-i
      cx=1.-dx
      j=y(n0)
      dy=y(n0)-j
      cy=1.-dy
      k=z(n0)
      dz=z(n0)-k
      cz=1.-dz
C  Smoothing with the (.25,.5,.25) profile in each dimension:
      sl=.5
      do 1 l=-1,1
      sl=.75-sl
      sm=.5
      do 1 m=-1,1
      sm=.75-sm
      sn=.5
      do 1 n=-1,1
      sn=.75-sn
      s=sl*sm*sn
      rh(i+l  ,j+m  ,k+n  )=rh(i+l  ,j+m  ,k+n  )+s*cx*cy*cz
      rh(i+l+1,j+m  ,k+n  )=rh(i+l+1,j+m  ,k+n  )+s*dx*cy*cz
      rh(i+l  ,j+m+1,k+n  )=rh(i+l  ,j+m+1,k+n  )+s*cx*dy*cz
      rh(i+l  ,j+m  ,k+n+1)=rh(i+l  ,j+m  ,k+n+1)+s*cx*cy*dz
      rh(i+l  ,j+m+1,k+n+1)=rh(i+l  ,j+m+1,k+n+1)+s*cx*dy*dz
      rh(i+l+1,j+m  ,k+n+1)=rh(i+l+1,j+m  ,k+n+1)+s*dx*cy*dz
      rh(i+l+1,j+m+1,k+n  )=rh(i+l+1,j+m+1,k+n  )+s*dx*dy*cz
    1 rh(i+l+1,j+m+1,k+n+1)=rh(i+l+1,j+m+1,k+n+1)+s*dx*dy*dz
```

```
      return
      end
```

References

Bennett, W. H., *Phys. Rev.*, *45*, 890, 1934

Birdsall, C. K., and A. B. Langdon, *Plasma Physics via Computer Simulation* McGraw-Hill, first edition, 1985

Hockney, R. W., and J. W. Eastwood, *Computer Simulation using Particles* McGraw-Hill, first edition, 1981.

Lindman, E. C., *J. Comput. Phys.*, *18*, 66, 1975.

Villasenor, J., and O. Buneman, Rigorous Charge Conservation for Local Electromagnetic Field Solvers, *Comp. Phys. Comm.*, *69*, 306-316, 1992.