206

PLASMA PHYSICS LABORATORY

Contract AT(30-1)—1238 with the
US Atomic Energy Commission

PRINCETON UNIVERSITY
PRINCETON, NEW JERSEY

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Princeton University
Plasma Physics Laboratory
Princeton, New Jersey

BPERP, BSLANT and GLOBE
Optimized Codes for the Simulation
of Plasmas in a Static Uniform
Magnetic Field

by

MASTER

Bernard Rosen, Hideo Okuda,
and Louis G. Di Massa

MATT-896

June 1972

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

AEC RESEARCH AND DEVELOPMENT REPORT

This work was supported by U. S. Atomic Energy Commission Contract AT(11-1)-3073. Reproduction, translation, publication, use, and disposal, in whole or in part, by or for the United States Government is permitted.

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Q

BPERP, BSLANT and GLOBE
Optimized Codes for the Simulation of Plasmas
in a Static Uniform Magnetic Field

Bernard Rosen*
Stevens Institute of Technology, Hoboken, N. J. 07030

and

Hideo Okuda and Louis G. Di Massa
Plasma Physics Laboratory, Princeton University,
Princeton, N. J. 08540

ABSTRACT

A description of optimized computer codes for the simulation of plasmas subject to a uniform, external, static magnetic field is given. The codes are of the particle-pushing variety. The optimization of the programs is traced through various steps including several phases of machine language coding. A brief review of the leap-frog algorithm for particle-pushing is included. The time required, per particle, per iteration, in 2 and $2\frac{1}{2}$ D simulations using an IBM 360/91 was 20 μ sec and 27 μ sec respectively in the most efficient versions of the code. This included time for the field calculations on a 32 x 32 grid.

*A visitor at Plasma Physics Laboratory

I. INTRODUCTION

We shall report, in historical order, progress that we have made in developing an efficient code for the computer simulation of plasmas in a fixed external magnetic field. The simulation is effected by "particle pushing." Applications have been made to the study of drift waves,¹ of thermal diffusion,² and of lower hybrid heating³ in 2 dimensions (BPERP) and $2\frac{1}{2}$ dimensions (BSLANT). A full 3 dimensional code (GLOBE) has also been developed and is now being applied to the study of thermal diffusion of plasmas across a magnetic field.⁴ A different version of the optimized codes which requires less computer memory for execution has also been developed. This version is quite useful for simulation of a large system; e.g. a 128×128 grid system in 2D, and its use is crucial in a 3 dimensional simulation.

These codes use the so-called leap-frog version of Newton's equations of motion. In this scheme the equations are written in difference form and involve, at any given step in the calculation, the velocities at one particular instant and the force and position one half time step later. Since the Lorentz force involves the velocity when there is a magnetic field present one must make provisions for evaluation of the velocity at the midpoint of the time interval. Two methods for doing this are given in Section 2 which is based largely upon work published by others and is presented for expletive purposes.

In the third section the development of programming improvements we have made to optimize these codes is traced. The material is presented in this quasi-historical fashion in order:

- 1) to present the methods per se;

2) to illustrate the reduction in execution time that can be realized by the use of certain techniques.

II. LEAP-FROG FORM OF NEWTON'S EQUATIONS

(This section is based largely on material in Reference 5)

The leap frog version of Newton's equations of motion is:

$$m \underline{v} (t + \delta) - m \underline{v} (t) = \alpha \underline{F} (t + \delta/2) \delta. \quad (1)$$

$$\underline{x} (t + 3 \delta/2) - \underline{x} (t + \delta/2) = \underline{v} (t + \delta) \delta. \quad (2)$$

The constant α , as explained below, is chosen so that the solution of the leap frog equations agree with those of Newton's equations for some simple force law. If $\underline{F}$ is the Lorentz force then (1) becomes

$$\underline{v} (t + \delta) - \underline{v}(t) = (e \delta/m) [\underline{E} (t + \delta/2) + \underline{v}(t + \delta/2) \times \underline{B}_0] \quad (1')$$

wherein $\underline{E}$ is the electric field and $\underline{B}_0$ is the magnetic field. Throughout the discussion $\underline{B}_0$ is considered to be a static and spatially uniform field.

The first term on the right hand side of (1') presents no problem as regards the leap-frog scheme while the second term does. This is because $\underline{v}$ is not known explicitly at the time $t + \delta/2$. As a first step in eliminating this difficulty one can replace $\underline{v}(t + \delta/2)$ by the symmetric form

$$1/2 [\underline{v}(t + \delta) + \underline{v}(t)] \quad (3)$$

a formula which is accurate to terms of order δ . Before considering exactly how one might evaluate $\underline{v} (t + \delta)$ given $\underline{v}(t)$ and the forces at the half time step later let us remark upon one attractive feature of this replacement. In the absence of an electric field, kinetic energy of each particle is conserved by the equations in this form. For an immediate consequence of

$$\underline{v}(t + \delta) - \underline{v}(t) = b \left[\underline{v}(t + \delta) + \underline{v}(t) \right] \times \underline{B}_0$$

(b an arbitrary finite scalar)

is that

$$\underline{v}^2(t + \delta) = \underline{v}^2(t)$$

Unfortunately this form of the equations cannot be used directly in a computer program because of the appearance of $\underline{v}(t + \delta)$ on the right hand side. However one can either:

- 1) expand $\underline{v}(t + \delta)$ in a Taylor series⁶ to order δ using the exact equation of motion

$$d \underline{v} / dt = (e/m) (\underline{E} + \underline{v} \times \underline{B}_0)$$

to evaluate the required derivatives. This method is straightforward but the symmetric form that led to conserved kinetic energy for a particle in a static magnetic field is preserved only to order $(e B_0 \delta / m)^2$.

- 2) solve the equation of motion in leap-frog form, with $\underline{v}(t + \delta/2)$ given by (3), algebraically.⁵ To illustrate this we write (1'), the equation of motion, as

$$\underline{v}(t + \delta) = \underline{K} + \Delta \underline{v}(t + \delta) \times \underline{B}_0 \quad (1'')$$

where Δ is $(\alpha e \delta / 2m)$ and where $\underline{K}$ contains all terms that are explicitly known and includes both $\underline{v}$ and $\underline{v}(t) \times \underline{B}_0$.

One finds after a brief calculation that:

$$\underline{v}(t + \delta) = \underline{K} + \frac{1}{1 + \Delta^2} \left[\Delta \cdot \underline{K} \times \underline{B}_0 + \Delta^2 (\underline{K} \cdot \underline{B}_0) \times \underline{B}_0 \right] \quad (4)$$

We have found the second scheme preferable and have adopted it. Hockney and Buneman⁵ have also shown that one should choose

$$\alpha = \tan (\Omega \delta/2) / (\Omega \delta/2)$$

$$\Omega = \text{cyclotron frequency} = e \underline{B} / m$$

in order that the leap-frog equations should yield

- a) correct cyclotron motion for particles in a uniform magnetic field,
- b) the correct $\underline{E} \times \underline{B}$ drift for particles subjected to uniform electric and magnetic fields.

If one wishes this can be considered to be a renormalization in e/m that compensates for the use of the discrete time step.

Using the Hockney-Buneman scheme we were able to obtain energy conservation to within one part in 10^4 using a time step of $\approx .2/\Omega$. In using based upon expansion in a Taylor series one needed a time step $1/4$ as large to obtain good energy conservation.

III. CODING DEVELOPMENTS

(For the purposes of discussion the bulk of this section is concerned with the 2D version of the codes. The iteration times quoted in this section include to compute the field quantities on a 32×32 grid.)

The execution time per iteration can be considerably reduced by the machine language coding of the innermost loops as well as by the use of certain programming tricks available at the FORTRAN level and the first coding improvements were made in FORTRAN. These included such commonly used devices as redefining the velocities and accelerations so as to include

the time step. The Fourier transform method for the determination of the electric fields was speeded up by the use of the subtracted dipole scheme (SUDES)⁷ as well as by the use of FFT2.⁸ Furthermore the proper linear combinations of the fields that are needed for incrementing the velocities in the leap frog scheme (essentially $\underline{E}_x + \underline{E}_y \Omega \delta/2$, $\underline{E}_y - \underline{E}_x \Omega \delta/2$) were formed directly from the Fourier components of the charge density by means of a redefinition of the form factors. This eliminated two multiplications and two additions in the determination of the force. At this stage the time per iteration per particle had been reduced to 58.6 microseconds. A breakdown of the execution times for various parts of the code is given in TABLE 1.

In order to reduce the program run time further we reprogrammed the particle-pushing and charge calculation in machine language. For this purpose six subroutines, listed in TABLE 2, were written.

Besides those that are listed there is another routine, MULT, that is used if there are two or more species of particles used in the simulation. In a purely electrostatic code MULT would be used to multiply the electric field by the appropriate mass ratio in order to scale the accelerations properly. (In order to save time the e/m for one species is included in the form factors.) In the cases considered here MULT also performs the task of forming the correct combinations of $\underline{E}_x$ and $\underline{E}_y$ referred to in the second paragraph of this section since the coefficients for these combinations of field components are mass dependent.

Some of the programming tricks that were used in these routines were:

- 1) By loading the particles so that the position was always a positive number a reference to memory did not have to be made to determine if the particle was to the right of the left hand

boundary (which was at zero). This was for both the x and y directions.

- 2) in SMOOTH and GRAD the indexing difficulties encountered at the boundaries of a periodic system when taking the differences between "adjacent" elements were handled by the use of bit masking operations which eliminated testing for the edges of the periodic system.
- 3) in BPERP the indices required to address the field components appropriate to the $I + 1$ 'th particle were calculated while the I 'th particle was being pushed. This reduces the time required to access memory for the IBM-360/91. A similar trick can be used in FORTRAN coding to advantage. Also the particle pushing was coded to minimize the accessing of the particle coordinates and velocity components.

As a result of these changes the time per particle per iteration was reduced to 27 microseconds; a breakdown is given in TABLE 3.

The next step was to rewrite the routines to take advantage of a hardware feature of the Model 91: namely, the fetching of data in this machine takes place by double words. In order to take advantage of this the coordinates were stored in memory such that the y component of the position of particle I followed the x coordinate and similarly for the velocity the field components. This representation, which was suggested by work of Fallon, Kruer, and Winsor,⁹ can be thought of as equivalent to representing the dynamical variables as complex numbers, e.g., $x + iy$.

This further reduced the execution time per particle per iteration to 24 microseconds and again, the breakdown is shown in TABLE 3, this time under the column marked Phase 2.

The final coding improvement was made by employing unnormalized additions in particle pushing. This was done so as to keep those bits which represented a given particle's cell number in fixed position within the computer word that gave that particle's position. This cell number is directly related to the indices needed to address the appropriate field components. By this means the indexing operations could be done more directly and further the subroutine TOROID could be eliminated as a separate routine.

In this form the execution per particle per iteration took 20 microseconds and the breakdown is shown under the column Phase 3 in TABLE 3.

A $2\frac{1}{2}$ D version of this code, called BSLANT, has also been written. It differs from BPERP in that, obviously, another component of the velocity is handled. Only the routine BPERP needed modification so as to include all the components of the magnetic field. A full 3-D code (GLOBE), with magnetic field in the z direction, has also been written. It is an extension of BPERP allowing the particle motion and the electric field to have components along the field lines. It uses the same leap-frog scheme in the x-y plane as does BPERP.

As mentioned above a different version of the codes has been developed for simulations in large systems. This version is essentially the same as the most efficient version with the following exceptions:

- 1) Subroutine NU22 and SMOOTH are combined into one routine in which the distribution of the dipole moment is made directly onto the neighboring grid points, particle by particle.
- 2) Subroutine GRAD is deleted and the gradient of the field is calculated directly in BPERP for each particle separately.

By deleting these two subroutines and the arrays for the dipole moments and

for the field gradients one can save a considerable amount of core storage; this makes simulation of large systems feasible. (By large we mean, e. g., 128 x 128 in 2 D and 32 x 32 x 32 in 3 D.) However, since this version includes more operations per particle than per grid compared with the faster version, it is not recommended that one use the space-conserving version for relatively small systems (smaller than 64 x 64) with many particles per cell. The choice depends on the size of the system as well as on the number of particles. Recently we have also developed a code which makes use of the potential instead of the electric field. This code is particularly useful for a 3 dimensional simulation.

IV. SUMMARY

We have presented an account of various programming techniques that we have used to optimize particle-pushing plasma simulation codes. The specific codes employed were those for simulating a plasma subject to a constant magnetic field. The results of the application of these techniques is illustrated by the reduction in execution times that we have noted.

V. ACKNOWLEDGEMENT

The authors would like to thank Prof. John M. Dawson for his encouragement of this work. One of us (B. R.) would like to thank the Plasma Physics Laboratory for the hospitality shown him.

This work was supported by U. S. Atomic Energy Commission Contract AT (30-1)-1238. One of us (B. R.) was supported in part by U. S. Atomic Energy Commission Contract AT (30-1)-3785.

REFERENCES

¹H. Okuda, B. Rosen, and J. M. Dawson, Sherwood Theoretical Meeting, NYU, March 1971.

²J. M. Dawson, H. Okuda and R. N. Carlile, Phys. Rev. Letters, 27, 491 (1971). Also H. Okuda and J. M. Dawson, Princeton Plasma Physics Laboratory PPL-AP52.

³H. Okuda, J. M. Kindel and J. M. Dawson, Proceedings of the Fifth Conference on Numerical Simulation of Plasmas, University of Iowa, Iowa (1971).

⁴H. Okuda and J. M. Dawson, Princeton Plasma Physics Laboratory PPL-AP54.

⁵R. W. Hockney, Stanford Univ. Tech. Rept. SUIPR No. 53, May 1966. Also O. Buneman, J. Compt. Phys., 1, 517 (1967).

⁶R. N. Carlile, Proceedings of the Fourth Conference on Numerical Simulation of Plasmas, Office of Naval Research, Arlington, Va. (1970).

⁷B. Rosen, W. L. Kruer and J. M. Dawson, Proceedings of the Fourth Conference on Numerical Simulation of Plasmas, Office of Naval Research, Va. (1970).

⁸FFT is a machine language version of the subroutine HARM contained in IBM's Scientific Subroutine Package. FFT2 is a version of FFT applicable to REAL data.

⁹H. Fallon, N. Winsor and W. L. Kruer, Bull. Am. Phys. Soc., 14, 1025 (1969).

TABLE 1

NUMBER OF ELECTRONS	NUMBER OF IONS	GRID SIZE	TIME PER ITERATION PER PARTICLE (MICROSECONDS)	
			FORTRAN	IMPROVED FORTRAN
4096	4096	32 x 32	131	69

TIME (PER/ITERATION) FOR PARTS OF IMPROVED FORTRAN CODE
ON AN IBM 360/91

FORTRAN DO LOOPS	SECONDS	PER PARTICLE (MICROSECONDS)
BRINGS PARTICLES BACK WITHIN BOUNDARIES	0.140	17.1
CALCULATES THE CHARGE AND DIPOLE ARRAYS	0.134	16.4
CALCULATES FIELDS	0.003	
DISTRIBUTES DIPOLE MOMENTS	0.008	
COMPUTES GRADIENTS OF THE FIELDS	0.012	
MOVES THE PARTICLES	0.206	25.1
FFT2 (3 CALLS)	0.039	

PARTICLE PUSHING

58.6 MICROSECONDS

TABLE 2

SUBROUTINE	FUNCTION
TOROID	Brings particles back within boundaries.
NU22	Calculates the charge and dipole arrays.
SMOOTH	Distributes dipole moments according to subtracted dipole scheme.
FIELDS	Calculates fields from charge density in Fourier space.
GRAD	Computes gradients of the fields by differencing.
BPERP*	Moves the particles according to Newton's equations of motion with Lorentz force in leap frog fashion.

*The name BPERP is used for both the program and for a subroutine.
The context should indicate which one is meant.

TABLE 3

NUMBER OF ELECTRONS	NUMBER OF IONS	GRID SIZE	TIME PER ITERATION PER PARTICLE (MICROSECONDS)			
			PHASE I	PHASE II	PHASE III	BSLANT
4096	4096	32 x 32	27	24	20	24.8

TIME (PER/ITERATION) FOR PARTS OF CODE IN SECONDS AND MICROSECONDS PER PARTICLE
ON AN IBM 360/91

SUBROUTINE	PHASE I	PHASE II	PHASE III	BSLANT
TORIOD	0.036 .	0.033 .		SECONDS/
	. 4.4	. 4.0		MICROSECONDS PER PARTICLE
NU22	0.044 .	0.042 .	0.036 .	0.035 .
	. 5.4	. 5.1	. 4.4	. 4.3
FFT2	0.039			
SMOOTH	0.002			
FELDER	0.002			
GRAD	0.006			
BPERP	0.068 .	0.062 .	0.070 .	0.098 .
	. 8.3	. 7.6	. 8.5	. 12.0

PARTICLE PUSHING (MICROSECONDS)

PHASE I	PHASE II	PHASE III	BSLANT
18.1	16.7	12.9	16.3

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, express or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission to the extent that such employee or contractor prepares, handles or distributes, or provides access to, any information pursuant to his employment or contract with the Commission.