JOURNAL OF COMPUTATIONAL PHYSICS **101**, 307–313 (1992)

Localization Schemes in 2D Boundary-Fitted Grids

THOMAS WESTERMANN

Kernforschungszentrum Karlsruhe GmbH, Abteilung für Numerische Physik, HDI-NP, P.O. Box 3640, 7500 Karlsruhe, Germany

Received November 30, 1989; revised October 22, 1990

A discussion of localization schemes in two-dimensional structured grids consisting of convex four-point meshes is presented. These algorithms are applicable to particle-in-cell codes based on two-dimensional boundary-fitted coordinates in order to localize particles inside the grid. They are fully vectorizable and two of them are directly applicable also to triangular meshes. Since all of them are exact, they avoid an overhead for a special treatment of particles near the boundary as is necessary for the approximate localization proposed by Seldner and Westermann (*J. Comp. Phys.* **79** (1988)). Hence, they are suitable for complicated geometries with outer and inner curved boundaries. Depending on the vector computer used, a speedup of 3.5 to 8 is achieved for the fastest algorithm. © 1992 Academic Press, Inc.

I. INTRODUCTION

Particle-in-cell (PIC) codes are an attractive computational tool to study kinetic phenomena, e.g., in plasma physics [5]: A grid is introduced in order to compute the electromagnetic fields, and particles carrying electric charge and mass are advanced in these fields by solving the Lorentz equation. Originally, PIC codes were developed using uniform [5] or nonuniform [9] grid zoning with grid lines parallel to the coordinate axis. However, with these concepts it was not possible to treat complicated technical devices without simplification of the geometry. An example where it is essential to model the boundaries accurately are intense light-ion-beam diodes for inertial confinement fusion [3]. In the past, most codes were not able to treat curved shapes of the emitting parts of these diodes, and thus could not adequately model the influence of these shapes on the focusing of the ion beams. It also turned out that simplifications of technical geometries, e.g., at edges and curved parts of the electrodes, lead to artificial field-enhancement and, thus, to a distorted flow of particles.

To overcome these difficulties, two-dimensional PIC codes were developed using grids fitted to the boundary of the electromagnetic devices. The concept of structured boundary-fitted coordinates is either based on triangular meshes [16, 7] or on four-point meshes [12, 6, 13]. In both cases a logical rectangular grid is introduced onto which the fitted grid in the physical space is mapped. Essentially, two techniques are then used to numerically treat the fields and particles:

The first approach as used, e.g., by Jones [6], is to solve the field equations together with the Lorentz equation in the logical grid. In the other case [13] one solves only the field equations in the logical grid but advances the particles in the physical space. In contrast to Jones, we prefer the second method, which is extendable also to unstructured grids, and standard techniques for solving the relativistic Lorentz equation can be applied. However, the drawback consists in the fact that in each time step the particles must be localized inside an irregular grid and the fields must be interpolated from irregular four-point meshes onto the particle positions.

In principle, the problems of interpolation as well as of localization in irregular 2D meshes are solved by methods introduced in [11]. These algorithms are well suited for vectorization. The basic idea of the localization scheme in [11] is to lay a fine equidistant mesh (background grid) over the boundary-fitted grid and to localize the particles inside this rectangular grid. A relationship between the equidistant mesh and the boundary-fitted grid is used to obtain the addresses of the particles with respect to the boundary-fitted grid. With this indirect method via a background grid one cannot decide exactly whether a particle near a curved boundary lies inside or outside the computational area. Particles near boundaries have to be treated separately. When simulating geometries with curved inner boundaries such as in the case of the self-magnetically insulated B_θ-diode (cf. Fig. 8) [10, 14], the corresponding additional work for the treatment of particles near boundaries produces a large overhead. Hence, this method is very efficient as long as the computational area contains no curved inner boundaries.

In this paper, three localization schemes avoiding this overhead are discussed. These algorithms are exact and fully vectorizable. After an introduction on the problem and the description of the originally used non-vectorizable algorithm in Section 2, a search algorithm based on the calculation of areas is outlined in Section 3. In Section 4 follows the discussion of a scheme using the calculation of simplices. Both methods are also applicable to triangular meshes and contain parallel structures. In Section 5, an iterative

0021-9991/92 $5.00

algorithm [15] is proposed based on a special interpolation scheme. Summary and conclusions are left to Section 6.

II. NOTATION AND PROBLEM

Notation. In this paper we consider two-dimensional, monoblock, structured grids consisting of arbitrary convex four-point cells. Hence, the grid is logically equivalent to a rectangular mesh which is a two-dimensional array of mesh points. In order to identify cells within a grid, each cell is assigned the addresses in the x- and y-direction of the left lower grid point of the cell as a pair of numbers (I, J). In an equidistant grid, the address of the cell, a particle with coordinates (x, y) is located in, is defined by

$$I = INT((x - x_0)/\Delta x) + 1$$

and

$$J = INT((y - y_0)/\Delta y) + 1,$$

where (x_0, y_0) are the coordinates of the left lower corner point of the grid and Δx and Δy are the mesh-sizes in the x- and the y-direction, respectively (see Fig. 1).

Non-vectorized Search Algorithm. In boundary-fitted grids, the above formulas cannot be applied anymore. For this case we consider the fact that a particle lies inside cell (I, J) with corners $P_{i,j}, P_{i+1,j}, P_{i+1,j+1}, P_{i,j+1}$ (see Fig. 2) if and only if it is

- above the lower cell boundary $P_{i,j}P_{i+1,j}$,
- on the left hand side of the right cell boundary $P_{i+1,j}P_{i+1,j+1}$,
- below the upper cell boundary $P_{i,j+1}P_{i+1,j+1}$ and
- on the right hand side of the left cell boundary $P_{i,j}P_{i,j+1}$.

This suggests the following non-vectorized search algorithm (originally proposed by Halter [3A]):

SEARCH ALGORITHM. Step 1. Start in cell $(I, J) = (I_0, J_0)$ the particle was located in at the end of the previous time step.

Step 2. If the particle is below the lower cell boundary: $J = J - 1$, goto step 2.

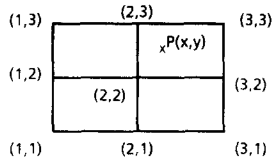

FIG. 1. Addresses of the mesh points.

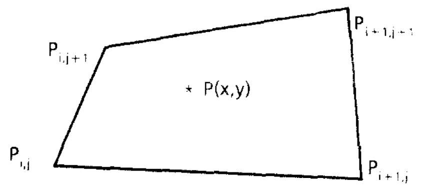

FIG. 2. Particle $P(x, y)$ in boundary-fitted cell (I, J).

Step 3. If the particle is on the right hand side of the right cell boundary: $I = I + 1$, goto step 2.

Step 4. If the particle is above the upper cell boundary: $J = J + 1$, goto step 2.

Step 5. If the particle is on the left hand side of the left cell boundary: $I = I - 1$, goto step 2.

Step 6. The particle is in cell (I, J).

Hence, there are at least four IF-clauses necessary for this direct search algorithm, even in case the particle has not moved outside the cell. This causes a very high CPU-time and, besides, the algorithm is not efficiently vectorizable.

III. A LOCALIZATION SCHEME BASED ON CALCULATION OF AREAS

In this section a localization scheme is investigated using the calculation of the area of quadrangles. In order to decide whether a particle lies inside a quadrangle $ABCD$, the area of the quadrangle is calculated and compared with the sum of the areas of the four triangles ABP, BCP, CDP, DAP (cf. Fig. 3).

A particle $P(x, y)$ lies inside quadrangle $ABCD$ if and only if the area of the quadrangle A_{ABCD} is equal to the sum of the areas of the four triangles A_{ABP}, A_{BCP}, A_{CDP}, A_{DAP}:

$$A_{ABCD} = A_{ABP} + A_{BCP} + A_{CDP} + A_{DAP}$$
$$\Leftrightarrow \text{Particle } P \text{ is inside quadrangle } ABCD.$$

If the particle is outside the quadrangle, the sum of the areas of the four triangles is larger than the area of the quadrangle (cf. Fig. 4):

$$A_{ABCD} < A_{ABP} + A_{BCP} + A_{CDP} + A_{DAP}$$
$$\Leftrightarrow \text{Particle } P \text{ is outside quadrangle } ABCD.$$

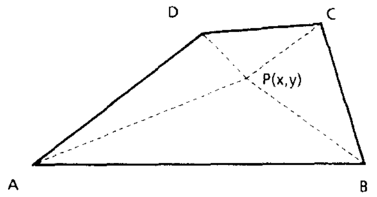

FIG. 3. Point $P(x, y)$ inside quadrangle $ABCD$.

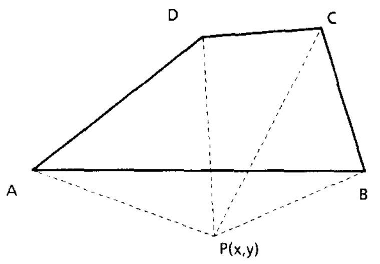

FIG. 4. Point $P(x, y)$ outside quadrangle $ABCD$.

Under the reasonable condition that during a simulation with a particle-in-cell code a particle can only cross to a neighboring cell, it is sufficient to limit the search to the cell the particle was located in at the end of the previous time step and to the surrounding cells. This yields the following search algorithm:

Search Algorithm. Step 1. Start in cell $(I, J) = (I_0, J_0)$ the particle was located in at the previous time step.

Step 2. Calculate the areas $A^{(I,J)}_{ABCD}$, $A^{(I,J)}_{ABP}$, $A^{(I,J)}_{BCP}$, $A^{(I,J)}_{CDP}$, $A^{(I,J)}_{DAP}$ of cell (I, J). Divide the sum of the four triangles by the area of the quadrangle $A^{(I,J)}_{ABCD}$.

Step 3. Repeat step 2 for the surrounding cells.

Step 4. The particle is located in the cell with the smallest area quotient.

This algorithm is applicable also to triangular grids. First, one has to select all triangles the particle can be located in and partition each of these triangles into three sub-triangles. With the same argument as before, the particle is located in the triangle with the smallest area quotient. The calculation of the areas of the quadrangles as well as of the sub-triangles can be computed independently and are, therefore, suited for parallelization.

IV. A LOCALIZATION SCHEME BASED ON CALCULATION OF SIMPLICES

In this section a localization scheme is introduced using the calculation of simplices. Let S be the convex hull of points $A(x_{00}, y_{00})$, $B(x_{10}, y_{10})$, $C(x_{01}, y_{01})$ (2-simplex):

$$S = \left\{ \lambda_1 A + \lambda_2 B + \lambda_3 C; \lambda_i \geqslant 0, i = 1, 2, 3; \sum_{i=1}^{3} \lambda_i = 1 \right\}.$$

Every element $(x, y) \in S$ is uniquely represented by a triple of real numbers (α, β, γ) such that

$$\begin{pmatrix} x \\ y \end{pmatrix} = \alpha \begin{pmatrix} x_{00} \\ y_{00} \end{pmatrix} + \beta \begin{pmatrix} x_{10} \\ y_{10} \end{pmatrix} + \gamma \begin{pmatrix} x_{01} \\ y_{01} \end{pmatrix},$$

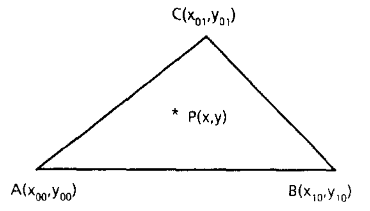

FIG. 5. Point $P(x, y)$ inside triangle ABC.

with

$$\alpha + \beta + \gamma = 1 \quad \text{and} \quad \alpha \geqslant 0, \beta \geqslant 0, \gamma \geqslant 0.$$

Every point inside a triangle $A(x_{00}, y_{00})$, $B(x_{10}, y_{10})$, $C(x_{01}, y_{01})$ can be interpreted as an element of the corresponding convex hull S (cf. Fig. 5). Hence, a particle $P(x, y)$ lies inside a triangle ABC if and only if $\alpha, \beta, \gamma \in \mathbb{R}$ exist with the properties

$$\alpha x_{00} + \beta x_{10} + \gamma x_{01} = x,$$
$$\alpha y_{00} + \beta y_{10} + \gamma y_{01} = y,$$
$$\alpha + \beta + \gamma = 1,$$

and

$$\alpha \geqslant 0, \quad \beta \geqslant 0, \quad \gamma \geqslant 0.$$

For a given point $P(x, y)$ α, β, γ can be computed using the formulas

$$\beta = \frac{(y_{01} - y_{00})(x - x_{00}) - (x_{01} - x_{00})(y - y_{00})}{(y_{01} - y_{00})(x_{10} - x_{00}) - (y_{10} - y_{00})(x_{01} - x_{00})}, \quad (1)$$

$$\gamma = \frac{-(y_{10} - y_{00})(x - x_{00}) + (x_{10} - x_{00})(y - y_{00})}{(y_{01} - y_{00})(x_{10} - x_{00}) - (y_{10} - y_{00})(x_{01} - x_{00})}, \quad (2)$$

$$\alpha = 1 - \gamma - \beta. \quad (3)$$

In order to localize a particle inside a triangle, it is sufficient to compute $\alpha, \beta, \gamma \in \mathbb{R}$ according to Eqs. (1)–(3) and check whether

$$\alpha \geqslant 0, \quad \beta \geqslant 0, \quad \gamma \geqslant 0 \quad (4)$$

or not. Only if condition (4) is satisfied, point $P(x, y)$ lies inside the triangle ABC.

In order to localize a particle inside a quadrangle $ABCD$, one has to partition the quadrangle into two triangles ABD and DBC and calculate for both triangles $(\alpha^1, \beta^1, \gamma^1)$ and $(\alpha^2, \beta^2, \gamma^2)$, respectively. The particle then lies inside the quadrangle $ABCD$ if $(\alpha^1, \beta^1, \gamma^1) \in \mathbf{I}^3 = [0, 1]^3$ or $(\alpha^2, \beta^2, \gamma^2) \in \mathbf{I}^3$.

Under the condition that during a time step of a particle-in-cell simulation a particle traverses at most over one cell boundary, the search algorithm can be limited to the cell the particle was located in at the previous time step and to the surrounding cells. By applying the simplex scheme, the following algorithm for quadrangles is obtained:

Search Algorithm. Step 1. Let $(I, J) = (I_0, J_0)$ be the cell the particle was located in at the previous time step.

Step 2. Partition cell (I_0, J_0) and the eight surrounding cells into two triangles, respectively. In each triangle solve the system of equations (1)–(3) leading to $(\alpha^1, \beta^1, \gamma^1), ..., (\alpha^{18}, \beta^{18}, \gamma^{18})$.

Step 3. Set $d_i = \min(\alpha^i, \beta^i, \gamma^i)$, $i = 1, ..., 18$.

Step 4. Determine the index i for which d_i is non-negative. Then the particle is inside the corresponding cell.

It is trivial to note that this search algorithm is directly valid for triangular meshes. Since the computation of the parameters $(\alpha^i, \beta^i, \gamma^i)$ can be computed independently of each other, this computation can be performed in parallel.

V. A LOCALIZATION ALGORITHM BASED ON AN INTERPOLATION SCHEME

The advantages of the first two localization methods are that the calculation of the areas and the values of (α, β, γ) can be computed separately and independently of each other. Therefore, this algorithm is well suited for a parallelization strategy. However, these results cannot be used in order to find a localization strategy to shorten the search. In particular, when the particles cross more than one cell during a time step, the search has to be extended to additional 16 or even more cells. In this section a localization algorithm is introduced taking provisional results into account. The algorithm is based on an interpolation scheme [15] valid for arbitrary convex quadrangles. The interpolation formulas are applied iteratively in order to localize the particles inside boundary-fitted grids.

After having finished the work, we were informed that J. U. Brackbill and H. M. Ruppel developed the same ideas for PIC calculations of fluid flows [2] and we heard from J. Ambrosiano [1] that he and R. Löhner investigated a similar search strategy in order to localize particles inside unstructured triangular meshes.

Before discussing our scheme, the generalized area-weighting method used for interpolation is briefly outlined for completeness. For a detailed discussion see Ref. [11].

Interpolation

If a particle $P(\alpha_1, \alpha_2)$ is located in a unit square, cell (I, J), the field E_p at the particle position is calculated from the fields $E_{i,j}$, $E_{i+1,j}$, $E_{i+1,j+1}$, $E_{i,j+1}$ given at the mesh points using the standard area-weighting method [4, 8]:

$$E_p = (1-\alpha_1)(1-\alpha_2) E_{i,j} + \alpha_1(1-\alpha_2) E_{i+1,j} + (1-\alpha_1)\alpha_2 E_{i,j+1} + \alpha_1\alpha_2 E_{i+1,j+1}.$$

In order to be able to apply the area-weighting method in an arbitrary quadrangle Q the non-equidistant cell has to be transformed into the unit square $\mathbf{I}^2 = [0, 1] \times [0, 1]$. Let $(x, y) \in Q$ be the position of the particle inside cell (I, J) with corners $(x_{i,j}, y_{i,j})$, $(x_{i+1,j}, y_{i+1,j})$, $(x_{i+1,j+1}, y_{i+1,j+1})$, $(x_{i,j+1}, y_{i,j+1})$. The interpolation weights $(\alpha_1, \alpha_2) \in \mathbf{I}^2$ are given by the following formulas:

$$\alpha_2 = \frac{-p + (p^2+q)^{1/2}}{(x^s_{i+1,j+1} - 1)} \quad \text{for} \quad x^s_{i+1,j+1} \neq 1,$$
$$\alpha_2 = \frac{y^s}{1 + x^s(y^s_{i+1,j+1} - 1)} \quad \text{for} \quad x^s_{i+1,j+1} = 1, \tag{5}$$

$$\alpha_1 = \frac{x^s}{1 + \alpha_2(x^s_{i+1,j+1} - 1)}, \tag{6}$$

where

$$\begin{pmatrix} x^s \\ y^s \end{pmatrix} := \begin{pmatrix} x_{i+1,j} - x_{i,j} & x_{i,j+1} - x_{i,j} \\ y_{i+1,j} - y_{i,j} & y_{i,j+1} - y_{i,j} \end{pmatrix}^{-1} \begin{pmatrix} x - x_{i,j} \\ y - y_{i,j} \end{pmatrix},$$

$$p = \tfrac{1}{2}(1 + x^s(y^s_{i+1,j+1} - 1) - y^s(x^s_{i+1,j+1} - 1))$$

and

$$q = y^s(x^s_{i+1,j+1} - 1).$$

The computation of the square root and the IF-clauses for the case $x^s_{i+1,j+1} \approx 1$ are avoided by solving Eq. (5) iteratively,

$$\alpha_2^0 = \frac{y^s}{1 + x^s/x^s_{i+1,j+1}(y^s_{i+1,j+1} - 1)}, \tag{7}$$

$$\alpha_2^i = \frac{y^s(1 + \alpha_2^{i-1}(x^s_{i+1,j+1} - 1))}{1 + \alpha_2^{i-1}(x^s_{i+1,j+1} - 1) + x^s(y^s_{i+1,j+1} - 1)} \quad \text{for} \quad i \geqslant 1, \tag{8}$$

and evaluating Eq. (6). When using grids consisting of trapezoids (in particular orthogonal grids) formula (5) is reduced exactly to the calculation of α_2^0 in (7).

One of the characteristic features of applying the generalized area-weighting method as an interpolation scheme is that linear functions are represented exactly. This formulation has the advantage that the interpolation weights can be easily evaluated in an explicit manner.

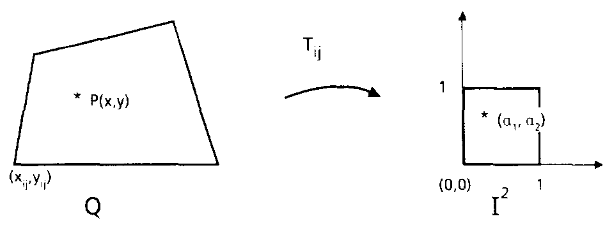

FIG. 6. Transformation of quadrangle Q onto the unit square $\mathbf{I}^2$.

It is important to note that the transformation T (cf. Fig. 6) of the quadrangle Q onto the unit square $\mathbf{I}^2$ which maps each point $P(x, y)$ of the quadrangle onto a point (α_1, α_2) depends on the geometrical structure of the quadrangle, e.g., the transformation is not the same for different quadrangles.

Localization

In the following, the interpolation scheme is used in order to find the particle position with respect to the grid: After particle P has been advanced the interpolation weights (α_1, α_2) are computed by transforming the cell (I_0, J_0) the particle was located in at the previous time step. With knowledge of these interpolation weights (α_1, α_2) one can decide whether the particle still lies inside the same cell or whether it is outside the cell:

$$(\alpha_1, \alpha_2) \in \mathbf{I}^2 \Leftrightarrow \text{Particle } P \text{ still is inside cell } (I_0, J_0).$$
$$(\alpha_1, \alpha_2) \notin \mathbf{I}^2 \Leftrightarrow \text{Particle } P \text{ has left cell } (I_0, J_0).$$

In particular, when using an equidistant grid, and considering that the particle can only move to a neighboring cell, the following holds:

$$\alpha_1 > 1, \alpha_2 > 1 \Rightarrow \text{Particle in cell } (I_0 + 1, J_0 + 1)$$
$$\alpha_1 \in \mathbf{I}, \alpha_2 > 1 \Rightarrow \text{Particle in cell } (I_0, J_0 + 1)$$
$$\alpha_1 < 0, \alpha_2 > 1 \Rightarrow \text{Particle in cell } (I_0 - 1, J_0 + 1)$$
$$\alpha_1 > 1, \alpha_2 \in \mathbf{I} \Rightarrow \text{Particle in cell } (I_0 + 1, J_0)$$
$$\alpha_1 \in \mathbf{I}, \alpha_2 \in \mathbf{I} \Rightarrow \text{Particle in cell } (I_0, J_0)$$
$$\alpha_1 < 0, \alpha_2 \in \mathbf{I} \Rightarrow \text{Particle in cell } (I_0 - 1, J_0)$$
$$\alpha_1 > 1, \alpha_2 < 0 \Rightarrow \text{Particle in cell } (I_0 + 1, J_0 - 1)$$
$$\alpha_1 \in \mathbf{I}, \alpha_2 < 0 \Rightarrow \text{Particle in cell } (I_0, J_0 - 1)$$
$$\alpha_1 < 0, \alpha_2 < 0 \Rightarrow \text{Particle in cell } (I_0 - 1, J_0 - 1).$$

However, when using a non-equidistant grid (cf. Fig. 7), it is not guaranteed that the particle is found by calculating the weights corresponding to the old cell addresses. When the particle has left the cell, this algorithm must be applied

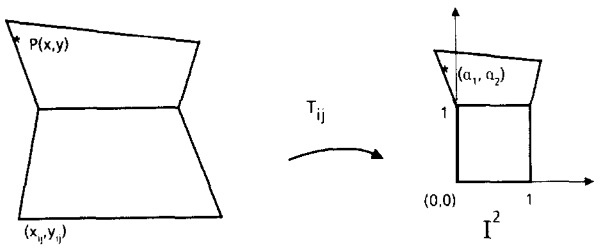

FIG. 7. Transformation of point P with respect to cell (I, J).

iteratively until the correct interpolation weights $(\alpha_1, \alpha_2) \in \mathbf{I}^2$ are found. The search algorithm is then

SEARCH ALGORITHM. Step 1. Let $(I, J) = (I_0, J_0)$ be the cell the particle was located in at the previous time step.

Step 2. Compute the interpolation weights (α_1, α_2) with respect to cell (I, J).

Step 3. Add the interpolation weights to the cell address: $(I + \alpha_1, J + \alpha_2)$.

Step 4. Set $(I, J) = (INT(I + \alpha_1), INT(J + \alpha_2))$.

Step 5. Repeat steps 2 to 4 $k \in \mathbb{N}$ times.

The parameter k depends on the structure of the grid. In case of an equidistant grid k can be chosen to be 1. (In this case, however, it is advisable to determine the cell in a direct manner.) Our experience indicates that usually $k = 3$ is sufficient when using a grid with convex four-point meshes. It also turned out that for the purpose of localization it is sufficient to compute the new cell addresses of the particles only using α_2^0 (cf. Eq. (7)).

Due to the local character of the transformation it is possible that at the first iteration cycle of the localization the values of the weights α_1 and α_2 are to high if the grid zoning changes rapidly. In order to avoid such an "overshooting" of the weights they should be restricted (for example, to be between -1.99 and 2.99). In particular, by restricting the interpolation weights possible singularities occuring in exceptional cases are eliminated.

One advantage of this scheme compared with the two previous ones is that the search is not limited to the cell the particle was located in at the previous time step and to the surrounding cells, since this algorithm chooses its own strategy for each iteration step according to the results of the iteration step before.

Depending on the grid, a particle can cross several cells (if this is permitted by the underlying physics and numerics) and is found with only few iterations. However, due to the local character of the transformation, the number k of iteration steps depends on how far the particles are allowed to move inside the grid and on the variation of the size and the shape of the zones. In particular, when using a non-smooth grid and particles are permitted to move over more than one cell, this scheme becomes inefficient. In this case it is appropriate to combine the indirect localization method via a background mesh (cf. [11]) with the iterative method. The indirect scheme delivers a good initial guess which can be taken as the starting point for an iterative search. By applying this combination $k = 2$ should always be sufficient.

VI. RESULTS

As a matter of fact, all three localization schemes are exact. Hence, when performing simulations with our BFCPIC code [13], the numerical results are independent of the algorithm used.

The localization schemes were introduced in order to reduce the CPU-times of the BFCPIC code to a reasonable length. In order to obtain realistic CPU-times for the localization schemes, a typical run of the BFCPIC code in case of the self-magnetically insulated B_θ-diode was performed [14]. For the purpose of localization it is sufficient to limit the discussion to the grid model of this diode. A numerical model of the self-magnetically insulated B_θ-diode together with a boundary-fitted grid (41 × 65 grid points) is shown in Fig. 8. For the simulation about 10,000 electrons and 22,000 ions were used. The numerical simulations were performed on the vector computers Fujitsu VP 50 and VP 400.

Since in our code the time step is limited in such a way that the particles can only move to a neighboring cell per cycle, $k = 3$ is sufficient for the iterative method. (When the particles are allowed to cross two cells, $k = 5$ must be chosen for this special kind of grid in order to guarantee that all particles are found.)

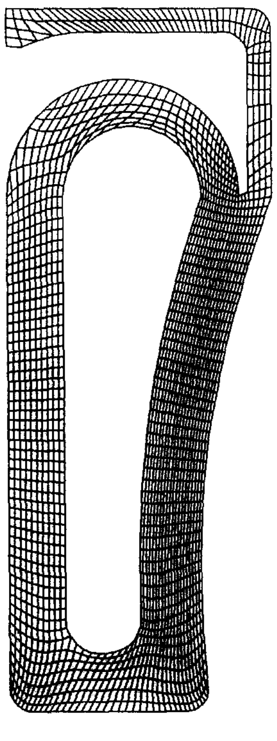
FIG. 8. Boundary-fitted grid of the self-magnetically insulated B_θ-diode, 41 × 65 grid points.

TABLE I

CPU-Times per Particle of the Localization Algorithms on Fujitsu Vector Computers Using $k = 3$

CPU-time [μs]	VP 50/400 (scalar mode)	VP 50 (vector mode)	VP 400 (vector mode)
Non-vectorized method	18.4	18.1	17.8
Area method	45.3	10.8	9.3
Simplex method	42.7	12.4	10.2
Iterative method	19.7	5.0	2.3

In Table I the CPU-times for the localization schemes are given together with the ones for the non-vectorizable search algorithm. Listed is the average time required to localize one particle.

As assumed, the original, non-vectorized algorithm is the fastest one on the scalar units of the computers, but it is not vectorizable. Due to the larger amount of arithmetic operations of the area method and the simplex method, these schemes are not efficient on the scalar units. However, since the vectorization degree is about 99%, they run faster on the vector units. On the scalar units of the VP computers, the CPU-time of the iterative method is comparable with the non-vectorizable algorithm. But on the vector units of the VP 50 and VP 400, a speedup is achieved of 3.7 and 8, respectively.

VII. CONCLUSIONS

Localization schemes for particle-in-cell codes based on boundary-fitted coordinates were introduced. These algorithms are fully vectorizable. Due to the fact that the schemes are exact, the particles near the boundaries have not to be treated separately as in the scheme presented in [11]. Hence, all three methods avoid the resulting overhead of an approximate localization for particles near boundaries. The localization based on the interpolation scheme leads to a speedup of 3.7 to 8, depending on the vector computer used. The advantages of this iterative method compared with the other ones are that the search is not limited to the cell the particle was located in at the previous time step and to neighboring cells and that the particle interpolation weights are evaluated without additional operations. Moreover, when using grids with almost squarelike shaped cells, one to two iteration steps are sufficient. When using grids with rapidly variing zones and with particles moving more than one cell per cycle, it is appropriate to combine the indirect localization method via a background grid with the iterative search.

ACKNOWLEDGMENTS

The author acknowledges the stimulating discussions with C. P. Hugelmann, E. Rudolph, and D. Seldner in particular for their engagement in finding the correct iteration formula (8), and the careful testing of the algorithms by A. Müller, which he appreciates very much. Thanks also to F. Schmitz for his help in implementing the code on the Fujitsu VP vector computers. The author also thanks the reviewers for their useful comments.

REFERENCES

1. J. Ambrosiano and R. Löhner, in *Proceedings, Thirteenth Conference on the Numerical Simulation of Plasmas, Santa Fé, Sept. 17–20, 1989*, p. IW20; R. Löhner and J. Ambrosiano, *J. Comput. Phys.* **91**, 22 (1990).
2. J. U. Brackbill and H. M. Ruppel, *J. Comput. Phys.* **65**, 314 (1986).
3. D. L. Cook, M. P. Desjarlais, S. A. Slutz, T. R. Lockner, D. J. Johnson, S. E. Rosenthal, J. E. Bailey, R. S. Coats, R. J. Leeper, J. E. Maenchen, T. A. Mehlhorn, T. D. Pointon, J. P. Quintenz, C. L. Ruiz, R. W. Stinnett, W. A. Stygar, J. P. Van Devender, in *Proceedings, Seventh Conference on High-Power Particle Beams, Karlsruhe, 1988*, p. 35.

3A. E. Halter, private communication.

4. F. H. Harlow and W. M. Evans, Los Alamos Report LA-2139, 1959 (unpublished).
5. R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles* (McGraw–Hill, New York, 1981).
6. M. E. Jones, in *Proceedings, Twelfth Conference on the Numerical Simulation of Plasmas, San Francisco, Sept. 20–24, 1987*, p. IM3.
7. M. Matsumoto and S. Kawata, in *Proceedings, Seventh Conference on High-Power Particle Beams, Karlsruhe, 1988*, p. 581; *J. Comput. Phys.* **87**, 488 (1990).
8. R. L. Morse and C. W. Nielson, *Phys. Fluids* **14**, 830 (1971).
9. J. P. Quintenz, *J. Appl. Phys.* **49**, 4377 (1978).
10. W. Schimassek, O. Stoltz, and A. Citron, in *Proceedings, Seventh Conference on High-Power Particle Beams, Karlsruhe, 1988*, p. 76.
11. D. Seldner and T. Westermann, *J. Comput. Phys.* **79**, 1 (1988).
12. J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin, *J. Comp. Phys.* **47**, 1 (1982); E. Halter, *Die Berechnung elektrostatischer Felder in Pulsleistungsanlagen*, Kernforschungszentrum Karlsruhe GmbH, KfK 4072, Karlsruhe, 1986.
13. T. Westermann, *Nucl. Instrum. Methods* **A263**, 271 (1988).
14. T. Westermann, *Nucl. Instrum. Methods* **A281**, 253 (1989).
15. T. Westermann, in *Proceedings, Thirteenth Conference on the Numerical Simulation of Plasmas, Santa Fé, Sept. 17-20, 1989*, p. IM3.
16. A. M. Winslow, *J. Comput. Phys.* **2**, 149 (1967).