JOURNAL OF COMPUTATIONAL PHYSICS **61**, 519–522 (1985)

Note

Vector Calculation of Particle Code

I. Introduction

The development of a vector computer requires the modification of the algorithm into a suitable form for vector calculation. Among many algorithms, the particle code is a typical example which has suffered damage in the calculation on supercomputers owing to its possibility of recurrent data access in collecting cell-wise quantities from particle quantities [1].

In this article, we report a new method to liberate the particle code from recurrent calculations. But it should be noticed that the method may depend on the architecture of the supercomputer, and works well on FACOM VP-100 and VP-200 [2]: the indirect data accessing must be vectorized and its speed should be fast.

II. Algorithm

The particle code uses fixed Eulerian grids (cells) and moving Lagrangian particles. The physical quantities attached to the particles such as mass, momentum, and so on are convected by the particles from cell to cell and are collected into cells. The problem intrinsic in the particle code occurs in this final stage as shown in the following program:

(a) *Program* 1

```
      DO 100 IP = 1,IPMAX
      IX = X(IP)
      DX = X(IP) - IX
      F(IX) = F(IX) + DX
      F(IX + 1) = F(IX + 1) + 1 - DX
  100 CONTINUE
```

where the number of particles is *IP*MAX and those locations are stored in the array

* Permanent address: Fujitsa Limited, Shinkamata, Ota-ku, Tokyo, Japan.

0021-9991/85 $3.00

variable X. If some particles are located within a cell, i.e., $\mathrm{INT}(X(IP1)) = \mathrm{INT}(X(IP2)) = \cdots = IX$, the recurrent addressing of F occurs at the calculation of the sum of DX and $(1 - DX)$ into F and inhibits the vector operation. In order to avoid this difficulty, we propose the following algorithm:

(b) *Program* 2

```
         DO 1000 IY = 1,LVEC
         DO 1000 IX = 0,IMESH + 1
         FT(IX, IY) = 0.0
    1000 CONTINUE
         DO 1100 IK = 1,IPMAX,LVEC
         DO 1100 IV = IK,MIN(IK + LVEC - 1,IPMAX)
         L = IV - IK + 1
         IX = X(IV)
         DX = X(IV) - IX
         FT(IX,L) = FT(IX,L) + DX
         FT(IX + 1,L) = FT(IX + 1,L) + 1 - DX
    1100 CONTINUE
         DO 1200 IV = 1,LVEC
         DO 1200 IM = 0,IMESH + 1
         F(IM) = F(IM) + FT(IM,IV)
    1200 CONTINUE
```

Here, the particles are divided into some groups whose size is $LVEC$. In the innermost loop of the loop 1100, the particles in one group are put into calculation. The particle's quantity, which is unity in this case, is collected into temporal two-dimensional array variable FT with a fraction of DX or $(1\text{-}DX)$. Even if some particles are located within a cell and have the same IX, the index L in $FT(IX, L)$ is different for each particle and hence the quantities are stored into different addresses. There is no overlapping of address and vector operation is possible. This operation is repeated until all groups are put into calculation. In the loop 1200, this temporal value FT is re-collected into F, where the vector operation is also possible for variable IM which represents the mesh number. Here $IMESH$ is the total mesh number used in the system.

III. Test Runs

In Fig. 1, we show the results of test runs on a VP-100 computer at the Institute of Plasma Physics, Nagoya University. The results show that the efficiency of the

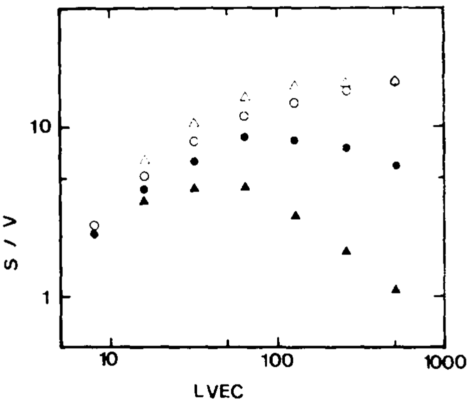

FIG. 1. The results of the test runs on VP-100 supercomputer. Solid triangles and circles show the ratios of execution CPU time of scalar calculation of Program 1 and vector calculation of Program 2 for 10 and 100 particles per cell, respectively. Open triangles and circles show the ratios of execution CPU time of scalar and vector calculations of Program 2, for 10 and 100 particles per cell, respectively.

new scheme depends on the number of particles per cell and vector length *LVEC*. In the figure, the solid triangles and circles show the ratios of execution CPU time of Programs 1 and 2 for 10 and 100 particles per cell, respectively. The open triangles and circles show the ratios of execution CPU time of scalar and vector calculations of Program 2 for 10 and 100 particles per cell, respectively. The speed ratio has a peak at some vector length. This optimum vector length is larger for larger numbers of particles per cell; for example, in the case of 10 particles per cell, the optimum vector length is 32–64, and in the case of 100 particles per cell, it is 64–128. This is because the increase of the number of particles makes the fraction of the loop 1000 and 1200 negligibly small, and makes the ratios of CPU time of Programs 1 and 2 approach only those of the loop 100 in Program 1 and loop 1100 in Program 2. Program 2 is four times faster and nine or more times faster than the Program 1 for 10 particles per cell and 100 particles per cell, respectively, on VP-100. The operation count of Program 1 was 1.0 μs/particle and that of Program 2 was 0.22 μs/particle for 10 particles per cell and 0.11 μs/particle for 100 particles per cell on VP-100.

In summary, we have proposed a new algorithm which can liberate the particle code from recurrent data accessing. The only problem that arises is the memory requirements. For example, the additional memory required is Min($LVEC \times IMESH \times$ the number of physical quantities, $LVEC \times IMESH + IPMAX \times 2$) in one dimension: $IPMAX \times 2$ is required to store IX and DX for each particle in the Program 2. This requirement may limit the use of the scheme.

ACKNOWLEDGMENT

The authors would like to thank Dr. Y. Abe at the Institute of Plasma Physics, Nagoya University, for his encouragement and valuable comments.

REFERENCES

1. B. L. BUZBEE, *NATO Adv. Study Inst. Ser. F* 7 (1984), 417.
2. K. MIURA AND K. UCHIDA, *NATO Adv. Study Inst. Ser. F* 7 (1983), 127.

RECEIVED: August 8, 1984

A. NISHIGUCHI
S. ORII*
T. YABE

Institute of Laser Engineering, Osaka University, Yamada-oka, Suita, Osaka, Japan

* Permanent address: Fujitsu Limited, Shinkamata, Ota-ku, Tokyo, Japan.