# Particle in Cell Simulation

F. Ebadpour and A. Navid

***Abstract*—Aim of this work is particle system simulation by the new programming language therefore we reviewed several methods for simulation of plasma systems. The particle-in-cell (PIC) method is then applied on a simple plasma system. In the PIC method ions are considered as a neutral-stationary background where electrons move around their initially occupied equilibrium states in one-dimension according to the Maxwell velocity distribution and the electromagnetic wave introduced as a disturbance into the system. The simulation starts from the point where the electromagnetic wave disturbs locations and velocities of the system particles (electrons) to reach an equilibrium state. The simulation program is written in SQL/Q-Basic. Here we demonstrated the successful use of PIC method on a wide variety of N particle plasma systems. In addition, this method could be used in three dimensional cases and with other field calculation methods, e.g. cloud-in-cell (CIC).**

***Index Terms*—Particle in cell (PIC), vlasov code, full particle code, MHD code, two fluid code, test particle code, hybrid code, P3M technique, SQL language, NGP method, particle-particle code, tree code, cloud-in-cell (CIC).**

## I. Introduction

Plasmas simulation introduce in two method, fluid simulation and kinetic simulation. Fluid simulation is done in two types of MHD and Two fluid codes. Kinetic simulation is done in four type of Vlasov, Full particle, Test particle and Hybrid codes in which ions and electrons considered as fluid and as kinetic particles (Fluid ions and kinetic electrons) [1-3].

Between these methods only Full particle code represented. In this method all of particle available into plasma interested and interaction between them modelled and hereby effect of them on together reflected as a change in position and velocity of particles. This method also divides into three types of Particle-Particle code, Tree code and particle-cell code. One of these code e.g. particle-cell will be discussed in this work.

## II. Particle-in-Cell Code

Particle in cell (PIC) is a prevailing method used in gravitational N particle systems, compressible and incompressible fluid and extremely in plasmas models. PIC simulation divided into two fundamental part "particle-particle" and "particle-mesh" that known as (P3M) technique of Hockney and Eastwood [4].In this approach, the charge of each particle is distributed among the nearest grid points, and then the electric field is calculated by solving Poisson's equation using the charge density on the grid as the source [5].

Manuscript received March 10, 2012; revised April 30, 2012.

The authors are with the Department of Sciences, Marand Branch, Islamic Azad University, Marand, Iran.

## III. An Example PIC Simulation

One dimensional assumption system for simulation is plasma consist of ions and electrons. Ions consider as a neutral base don't have motion. Original discussion cases are electrons that are elementally occupied equilibrium state and obey from Maxwell velocity distribution. Passing a electromagnetic wave from plasma create a disturbance in location and velocities of system particles (electrons). We begin simulation from this state and continue it to reach an equilibrium state.

## IV. Program Description

This program wrote in base of SQL language that combined with Q basic language. In these program 1064 particles in 32 cells simulated. An instable harmonic exert on the particles locations and consider particles as a disturbed system. Following code referred to disturbance:

```
For k = 1 To N
    i.AddNew
    i.Fields("i").Value = k
    i.Update
Next
For k = 1 To ng
    j.AddNew
    j.Fields("j").Value = k
    j.Fields("xj").Value = (l / ng) * (k - 0.5)
    j.Update
Next
```

Already particle spread as symmetric and equal a simulation space. Now exert distortion change on them.

```
For k = 0 To 1000
    t.AddNew
    t.Fields("t").Value = k * dt
    t.Update
Next
i.MoveFirst
While Not i.EOF
      it.AddNew
      it.Fields("i").Value = i.Fields("i").Value
      it.Fields("t").Value = 0
  it.Fields("x").Value = (l / N) * i.Fields("i").Value + 0.05
* (l / N) * Cos((l / N) * i.Fields("i").Value)
      it.Update
      i.MoveNext
Wend
```

These codes really exert a sinusoid disturbance on particles that its amplitude is smaller than of particles distances. Since done first step that is calculation of charge on cells. This section of program write in SQL programming language that its code as a following.

```
PARAMETERS [ts] IEEESingle;
INSERT INTO Gjt ( j, pj, t, xjj )
SELECT JT.j, nz([SumOfpj1])+nz([SumOfpj0]) AS Expr1, JT.t, JT.Xj
FROM (JT LEFT JOIN pj0 ON (JT.j = pj0.j0) AND (JT.t = pj0.t)) LEFT JOIN pj1 ON (JT.t = pj1.t) AND (JT.j = pj1.j1)
WHERE (((JT.t)=[ts]));
```

SQL codes are very summarized. These codes exert commands on series of information and using of repetitive commands didn't needed. This helps me have a better sense of system programming. Now use Fast Forrier Transform (FFT) to compute electric field. this transform is a discrete technique to avoid integration and differentiate. These codes are following.

```
'**************cal Ej
    Set v = CurrentDb.QueryDefs("e(k)")
    v.Parameters("ts").Value = k * dt
    Set ek = v.OpenRecordset
    Set gjt = CurrentDb.TableDefs("gjt").OpenRecordset(dbOpenDynaset)
    gjt.Filter = "t=" + CStr(k * dt)
    Set gjt = gjt.OpenRecordset
    gjt.MoveFirst
    While Not gjt.EOF
      e0 = 0
      For q = -ng / 2 To ng / 2
        e0 = e0 + (ek.Fields("ree").Value * Cos(q * gjt.Fields("xjj").Value) - ek.Fields("ime").Value * Sin(q * gjt.Fields("xjj").Value)) / l
      Next
      gjt.Edit
      gjt.Fields("ej").Value = e0
      gjt.Update
      gjt.MoveNext
  Wend
  PARAMETERS TS IEEESingle;
  SELECT Gjt.t, Sum(([k]*([l]/[ng])*Sin([k]*[l]/[ng])/(4*[epsi]*(Sin([k]*[l]/(2*[ng])))^2))*([l]/[ng])*[pj]*Sin(-[k]*[xj])) AS ReE, Sum(-([k]*([l]/[ng])*Sin([k]*[l]/[ng])/(4*[epsi]*(Sin([k]*[l]/(2*[ng])))^2))*([l]/[ng])*[pj]*Cos(-[k]*[xj])) AS ImE
  FROM Init, j INNER JOIN Gjt ON j.j = Gjt.j
  GROUP BY Gjt.t
  HAVING (((Gjt.t)=[ts]));
```

Therefore field on cell points be obtained and field on particle can be calculated. As signed earlier, weighting action again used. But in this case reverse action, exert on cells, done and each particle share form environmental cells field. Here again NGP method used but another accuracy methods can be used providing its accuracy much than charge ration method accuracy.

```
'**********cal Ei
    Set v = CurrentDb.QueryDefs("e(i)")
    v.Parameters("ts").Value = k * dt
    Set ek = v.OpenRecordset
    Set fit = CurrentDb.TableDefs("fit").OpenRecordset(dbOpenDynaset)
    fit.Filter = "t=" + CStr(k * dt)
    Set fit = fit.OpenRecordset
    fit.MoveFirst
    While Not fit.EOF
   ek.FindFirst ("i=" + CStr(fit.Fields("i").Value) + " AND t=" + CStr(k * dt))
      fit.Edit
      fit.Fields("e").Value = ek.Fields("ei1").Value
      fit.Update
      fit.MoveNext
    Wend
  PARAMETERS TS IEEESingle;
  SELECT Fit.i, Fit.t, (([x1]-[x])*[ej0]+([x]-[x0])*[ej1])*[ng]/[l] AS ei1
  FROM Init, (Fit INNER JOIN EjMax ON (Fit.t = EjMax.t) AND (Fit.i = EjMax.i)) INNER JOIN EjMin ON (Fit.t = EjMin.t) AND (Fit.i = EjMin.i)
  WHERE (((Fit.t)=[ts]));
```

While system field on each particle determined, actually all force from system on particles can be obtained. So now using of equations of motion new X and V of particles can be calculated to complete last step of circle and system go on front first stage according as follows:

```
PARAMETERS TS IEEESingle, ts1 IEEESingle;
INSERT INTO Fit ( i, t, V, xi )
SELECT Fit.i, [TS] AS Expr2, [v]+[qe]*[e]*[dt]/[me] AS Expr1, Fit.x
FROM Fit, Init
WHERE (((Fit.t)=[ts1]));
PARAMETERS i IEEESingle;
UPDATE Fit, Init SET Fit.x = [xi]+[v]*[dt]
WHERE (((Fit.t)=[i]));
```

New location of particle may be out of system limitation then deleted from system and repaired by adding other particle from symmetric side as follows:

```
UPDATE Fit, Init SET Fit.x = init.l+fit.x
WHERE (((Fit.x)<0));
UPDATE Fit, Init SET Fit.x = -init.l+fit.x
WHERE (((Fit.x)>[init].[l]));
```

continuation on simulation process for defined system, some graph obtained, explored and analyzed them.

Fig. 1a Shows graph of velocity with respect to time. This graph shows that how particle velocity changes when distortion exert on system. If we attend on minus peak of velocity, it will be understood that amplitude of this peak decrease along time and this indicate that system is reaching to equilibrium.

Fig. 1b. Shows tribulation effect on particles in which turn away them from their region. In fact energy of distortion on particle caused that they run away from deby distance and turned away but after a few time this changes decreased and went to horizontal line.

As seen in Fig. 1c. field get a pendulous shape but after a few time this progress be slowed and neared to equilibrium. But must attended that this field referred to one sample

particle and in the case of other particle may be had different shape. However it is possible to analyze effects of disturbance on them.

Average field on whole particle must always be zero but this deviation from actual that seen in fig. 1d indicated a large error on simulation and caused by using of NGP method to charge and field ration.

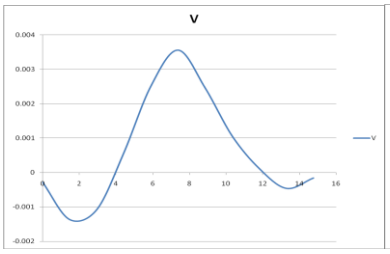


Fig. 1a

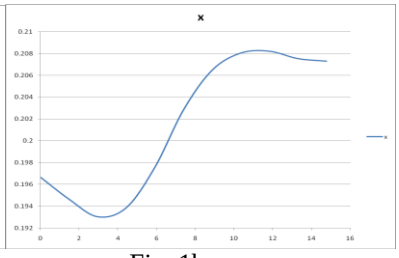


Fig. 1b

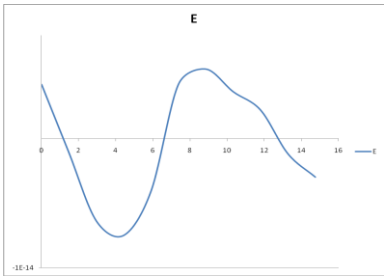


Fig. 1c

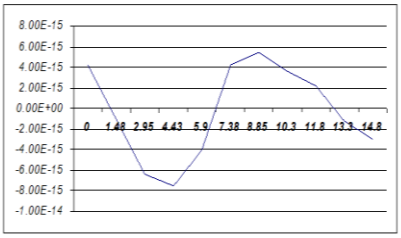


Fig. 1d

Fig. 1. Grafs of simulation results

## I. Summaries

According to indicted case we can use this method to spread area of systems and see result of them. This needs a correct understanding about systems and corrects modeling of them. By used this method to three dimensional systems and suitable ration action and adding some subprograms to correct errors, can obtain actual and accurate results. Difference between this method and other PIC simulation programs is on using SQL language and capability of its query tools. In this method we can create a data structure between tables that save data on time. Using these tables and structure help to access quickly to result of simulation and take diagrams and reports of them.

## References

[1] J. A. Heikkinen and J. L önnroth 2007 Plasma Phys. Control. Fusion 49 B465

[2] Francis F. Chen (2006). *Introduction to Plasma Physics and Controlled Fusion*, 2nd ed. Springer. ISBN 0306413322.

[3] Nicholas Krall and Alvin Trivelpiece (1986). *Principles of Plasma Physics*. San Francisco Press. ISBN 0911302581

[4] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*, IOP Publishing Ltd. (1988).

[5] C. K. Birdsall and A. B. Langdon, *Plasma Physics via Computer Simulation*, IOP Publishing Ltd. (2005)

[6] J. P. Verboncoeur, "Particle simulation of plasmas: review and advances," *Plasma Phys. Control. Fusion* vol. 47, no. 2005, pp. A231-A260.