Учебно-методические пособия Учебно-научного центра ОИЯИ Дубна

УНЦ-2008-33

А. В. Смирнов, Д. А. Крестников

ПРОГРАММА ПО МОДЕЛИРОВАНИЮ ДИНАМИКИ ЗАРЯЖЕННЫХ ЧАСТИЦ

Учебно-методическое пособие

2008

Учебно-научный центр ОИЯИ

А. В. Смирнов, Д. А. Крестников

ПРОГРАММА ПО МОДЕЛИРОВАНИЮ ДИНАМИКИ ЗАРЯЖЕННЫХ ЧАСТИЦ

Учебно-методическое пособие

Дубна 2008

С50

Учебное пособие составлено доцентом базовой кафедры МИРЭА «Электроника физических установок» при УНЦ ОИЯИ А. В. Смирновым (ОИЯИ) и аспирантом УНЦ ОИЯИ Д. А. Крестниковым и рекомендовано к изданию экспертной комиссией УНЦ ОИЯИ и редакционно-издательским советом МИРЭА.

Смирнов А. В., Крестников Д. А.

С50 Программа по моделированию динамики заряженных частиц: Учебно-метод. пособие. — Дубна: ОИЯИ, 2008. — 25 с.

Учебное пособие содержит описание программы для моделирования динамики заряженных частиц, реализованного в виде классов на языке программирования С++. Данное пособие составлено в качестве методического материала для курсовой работы, которая выполняется студентами по предмету «Объектно-ориентированное программирование».

Пособие предназначено для студентов и аспирантов инженерных и физических специальностей, изучающих основы ускорителей заряженных частиц и объектно-ориентированное программирование на языке С++.

Smirnov A. V., Krestnikov D. A.

The program for the charged particle dynamics simulation: Textbook. — Dubna: JINR, 2008. — 25 p.

The textbook contains a description of the program code for the charged particle dynamics simulation which was written on the base of C++ classes. This textbook has been made as a methodical material for the education course which is carried out by students in a subject «Object-oriented programming».

The book is addressed to the students and post-graduate students of physical and engineering specializations studying the charged particle accelerator physics and the object-oriented programming in C++ language.

Содержание

Предисловие

Данное пособие написано как методический материал для выполнения курсовой работы по курсу лекций «Объектно-ориентированное программирование», прочитанному Смирновым А.В. в 2003-2007 гг. студентам базовой кафедры электроники физических установок Московского государственного института радиотехники, электроники и автоматики (МИРЭА), организованной в Дубне при Учебно-научном центре Объединенного института ядерных исследований, и предназначено для студентов очной формы обучения, специальность 200600 «Электроника физических установок».

Программы для расчета динамики заряженных частиц во внешних и собственных электромагнитных полях имеют большой практический интерес. Материал, представленный в данном пособии, создан на основе реальных программ по моделированию движения частиц в позитронной ловушке и расчету кристаллических пучков в накопителях заряженных частиц [1]. В случае двухмерного решения уравнений движения (без учета продольного движения) такие задачи могут быть решены в едином подходе.

Целью курсовой работы является создание программы моделирования динамики заряженных частиц под действием электромагнитных полей и собственного пространственного заряда частиц. Возможны два варианта сложности выполнения программы: двух- и трехмерный варианты. Все объекты в программе описываются в виде классов на языке C++. Курсовая работа наглядно показывает возможность построения иерархии классов для решения сложных задач. В структуре программы продемонстрированы основные возможности объектно-ориентированного программирования.

Пособие предназначено для студентов и аспирантов инженерных и физических специальностей, изучающих основы ускорителей заряженных частиц и объектно-ориентированное программирование на языке C++.

Введение

Идея получения кристаллических ионных пучков представляет большой интерес в последнее время. Достижение очень низкой температуры в системе отсчета пучка дает новые возможности в ускорительной физике, что может быть использовано для увеличения светимости в коллайдере в случае накопления короткоживущих изотопов.

Абсолютным критерием кристаллизации является состояние ионного пучка, когда средняя температура частиц в системе отсчета пучка T становится меньше потенциальной энергии межчастичного взаимодействия U [2]:

$$\Gamma = \frac{U}{T} = \frac{1}{4\pi\varepsilon_0}\frac{Z^2e^2}{aT} \geq 150 , \qquad (1)$$

где Ze – заряд частицы, ε_0 – диэлектрическая проницаемость вакуума, a – межчастичное расстояние.

Следующее условие относится к оптической структуре накопительного кольца. Накопительное кольцо должно иметь жесткую фокусировку, и энергия ионного пучка должна быть меньше критической энергии накопителя. Количество суперпериодов накопителя должно быть как минимум в четыре раза больше максимального бетатронного числа для поперечных колебаний частиц:

$$4 \cdot \max\{Q_x, Q_y\} < \textit{периодичность} . \qquad (2)$$

Для создания численной модели кристаллических пучков используется метод «молекулярной» динамики, в котором пучок разбивается на ячейки

определенного размера. Распределение частиц в каждой из ячеек считается одинаковым. Этот метод заметно сокращает время расчета. Хорошим способом проверки работы численной модели может быть бесконечный канал с постоянной фокусировкой, который описывается гамильтонианом

$$H=\frac{1}{2}\left(p_x^2+p_y^2+\frac{p_\tau^2}{\gamma_0^2}\right)+\frac{k}{2}\left(x^2+y^2\right)+\frac{r_{ion}}{\gamma_0^2\beta_0^2}\sum_i\frac{1}{\sqrt{(x-x_i)^2+(y-y_i)^2+\gamma_0^2(z-z_i)^2}}, \quad (3)$$

где p_x, p_y, p_z, x, y, z есть нормализованные моменты и соответствующие координаты, γ_0, β_0 – релятивистские факторы, k – градиент фокусирующей структуры. Форма кристалла описывается безразмерной линейной плотностью частиц λ_{ion}, которая определена как [3]:

$$\lambda_{ion}\equiv\frac{N}{C}\left(\frac{3\,r_{ion}}{2k\gamma_0^5\beta_0^2}\right)^{1/3}=\left(\frac{3N^3r_{ion}}{8\pi^2Q^2C\gamma_0^5\beta_0^2}\right)^{1/3}, \quad (4)$$

где N – число частиц в накопителе, C – периметр накопителя, Q – бетатронное число, r_{ion} – классический радиус иона:

$$r_{ion}=\frac{1}{4\pi\varepsilon_0}\frac{Z^2e^2}{Amc^2}, \quad (5)$$

где m – масса частицы, Z и A – ее зарядовый и порядковый номера, c – скорость света.

На рис. 1 представлены результаты моделирования кристаллических пучков для бесконечного канала с постоянной фокусировкой, описанного выражением (3). Здесь наглядно показано, как форма кристалла зависит от безразмерного параметра λ_{ion} (4). А именно, если λ_{ion}<0,709, то наблюдается цепочка, при

0,709<λ_{ion}<0,964 – зигзаг, при 0,964<λ_{ion}<3,10 – спираль, а при 3,10<λ_{ion}<5,7 – фигура, состоящая из оболочки и цепочки [4].

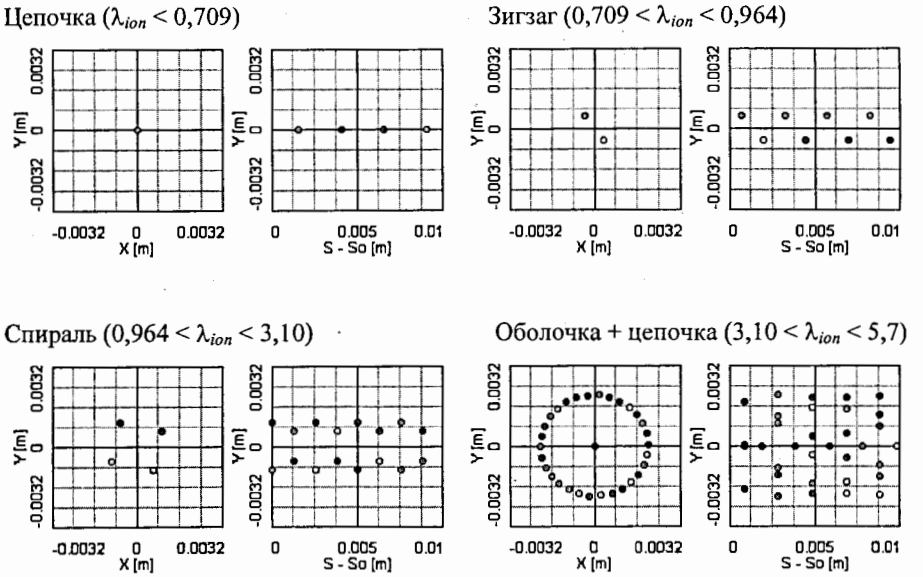

Рис.1. Кристаллические пучки в канале с постоянной фокусировкой

Материал курсовой работы

Группа студентов разбивается на подгруппы по два-три человека. Каждая подгруппа выполняет свое индивидуальное задание, которое включает в себя разработку определенного объекта, его тестирование, написание документации. Результатом выполнения курсовой работы каждой подгруппы является создание программы, состоящей из двух модулей: модуля объекта для основной программы и тестирующего модуля. Отчет включает в себя описание общей задачи, задачи для данной подгруппы, документацию к объекту, тексты программ обоих модулей, пример работы тестирующей программы.

В итоге все объекты собираются в единую программу, которая выполняет моделирование динамики частиц. Программа состоит из следующих объектов (задания для подгрупп):

1) вектор координат;
2) вектор сил;
3) массив заряженных частиц;
4) кулоновское взаимодействие частиц;
5) библиотека оптических элементов:
    а) дрейфовый промежуток,
    б) фокусирующая сила (жесткость),
    в) охлаждающая сила (вязкость),
    г) переменное продольное магнитное поле,
    д) переменное поперечное электрическое поле,
    е) пространственный заряд;
6) численные методы интегрирования:
    а) метод Эйлера,
    б) метод Рунге–Кутты;
7) интерфейс общей программы;
8) управляющий модуль программы.

Далее приводится краткое описание каждого объекта, а также пример написания классов. Непосредственно написание функций этих классов и их тестирование должно быть реализовано в ходе выполнения курсовой работы. Как правило, все объекты написаны для трехмерного случая, а в примерах рассматривается расчет динамики для двухмерного случая.

1. Вектор координат

Вектор координат для трехмерного случая состоит из трех координат и соответствующих скоростей $C = [x, Vx, y, Vy, z, Vz]$. Пример объявления класса координат, который включает в себя необходимые операции с координатами,

```
class Coordinate
{public:
  double x, Vx, y, Vy, z, Vz;
  Coordinate();
  void operator = (Coordinate);
  void operator += (Coordinate);
  Coordinate operator + (Coordinate);
};
```

2. Вектор сил

Вектор сил состоит из трех скоростей и соответствующих компонент сил $F = [Vx, Fx, Vy, Fy, Vz, Fz]$. Пример объявления класса сил, который включает в себя необходимые операции с компонентами сил,

```
class Force
{public:
  double Vx, Fx, Vy, Fy, Vz, Fz;
  Force();
  void operator = (Force);
  void operator += (Force);
  Force operator + (Force);
  Coordinate operator * (double);//умножение вектора сил на шаг
};
```

3. Вектор полей

Вектор полей состоит из компонент электрического и магнитного полей *f = [Ex, Ey, Ez, Bx, By, Bz]*. Пример объявления класса полей, который включает в себя необходимые операции с компонентами полей,

```
class Field
{
  public:
  double Ex, Ey, Ez, Bx, By, Bz;
  Fields ();
  void operator = (Field);
  void operator += (Field);
  Field operator + (Field);
};
```

Так как некоторые эффекты, использованные в программе, возвращают значение поля в определенной точке пространства, появляется необходимость пересчета значения данного поля в вектор силы, с которой оно действует на частицу. Для этого используется хорошо известная формула

$$\vec{F} = \frac{Ze}{Am}\left(\vec{E} + \left[\vec{v} \times \vec{B}\right]\right), \qquad (6)$$

где $\vec{E}$ – вектор электрического поля, $\vec{v}$ – вектор скорости частицы, $\vec{B}$ – вектор магнитного поля. При переходе к проекциям на координатные оси получаем компоненты вектора сил:

$$F_x = \frac{Ze}{Am}\left(E_x + v_y B_z - v_z B_y\right),$$
$$F_y = \frac{Ze}{Am}\left(E_y - v_x B_z + v_z B_x\right), \qquad (7)$$
$$F_z = \frac{Ze}{Am_e}\left(E_z + v_x B_z - v_y B_x\right).$$

4. Массив заряженных частиц

Класс, описывающий массив заряженных частиц, состоит из массива координат и функции для генерации начального распределения частиц. Конструктор класса должен создать динамический массив координат, а деструктор – освободить выделяемую под массив память.

```
class Particles
{ protected:
   Coordinate *array;
  public:
   int Number;
   Particles(int n) { array = new Coordinate[Number = n]; }
  ~Particles() {delete array[];}
   Coordinate& operator [] (int i) {  return array[i];  }
   void Initialization();
   void Distribution();
};
```

5. Кулоновское взаимодействие частиц

Класс для вычисления пространственного заряда массива частиц создается как производный класс на основе двух базовых классов: класса массива заряженных частиц и базового класса оптических элементов. Перегруженная виртуальная функция GetForce вычисляет силу Кулона, которая может быть получена из гамильтониана (1):

$$F_x = \frac{Z^2e^2}{4\pi\varepsilon_0}\sum_{i\neq j}^{N}\frac{x_i - x_j}{\left((x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2\right)^{3/2}},$$

$$F_y = \frac{Z^2e^2}{4\pi\varepsilon_0}\sum_{i\neq j}^{N}\frac{y_i - y_j}{\left((x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2\right)^{3/2}}, \qquad (8)$$

$$F_z = \frac{Z^2e^2}{4\pi\varepsilon_0}\sum_{i\neq j}^{N}\frac{z_i - z_j}{\left((x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2\right)^{3/2}}.$$

```
class Beam : public Particles, public Optics
{ public:
  virtual Initialization() { ; }
  Force GetForce(Coordinate, double t);
};
```

6. Библиотека сил

Класс Optics – это базовый класс для библиотеки эффектов, вычисляющих либо вектор сил, либо вектор полей. Для определения библиотеки используется базовый класс с виртуальными функциями GetField и GetForce. Причем в

производных классах можно переопределить любую из этих функций. Если будет переопределена функция GetForce, то она будет использована из производного класса в динамике расчета. Если переопределена только GetField, то будет вызвана функция GetForce из базового класса, которая пересчитает вектор полей в вектор сил в соответствии с (7).

```
class Optics
{ public:
  bool use;
  Optics() { use = false; }
  virtual Initialization() { ; }
  virtual Field GetField(Coordinate C, double t)
  { Field f;
    return f;
  }
  virtual Force GetForce(Coordinate C, double t)
  { Field f;
    f = GetField(C,t);
    Force F;
    F.Fx = Z*e / (A*m)*(f.Ex + C.Vy*f.Bz – C.Vz*f.By);
    F.Fy = Z*e / (A*m)*(f.Ey - C.Vx*f.Bz + C.Vz*f.Bx);
    F.Fz = Z*e / (A*m)*(f.Ez + C.Vx*f.By – C.Vy*f.Bx);
    retrun F;
  }
};
```

а) Дрейфовый промежуток

Как правило, каждый оптический элемент включает в себя дрейфовый промежуток, который описывает движения частицы без внешних сил. Именно

этот элемент библиотеки осуществляет изменение координат частиц, все другие оптические элементы меняют только скорости частиц.

```
class Drift : public Optics
{ public:
  Drift () { use = true; }
  Force GetForce(Coordinate C, double t)
  {  Force F;
    F.Vx = X.Vx;
    F.Vy = X.Vy;
    F.Vz = X.Vz;
    return F;
  }
};
```

б) Фокусирующая сила (жесткость)

Фокусирующая сила имеет линейную зависимость от координаты частицы, что описывает простые колебания материальной точки (пружина, маятник). В накопителях заряженных частиц это соответствует фокусировке по всем координатам:

$$F_x = -k_x x,$$
$$F_y = -k_y y, \qquad (9)$$
$$F_z = -k_z z.$$

```
class Rigidity : public Optics
{ public:
  double kx, ky, kz;
  Rigidity ();
  void Initialization();
  Force GetForce(Coordinate, double t);
};
```


в) Охлаждающая сила (вязкость)

Охлаждающая сила имеет линейную зависимость от скорости и приводит к затуханию колебаний частиц. Данная сила описывает простейший вариант силы охлаждения. Обратите внимание, что фокусирующая сила есть функция от координаты (9), а сила вязкости – функция скорости:

$$F_x = -\lambda_x V_x,$$
$$F_y = -\lambda_y V_y, \qquad (10)$$
$$F_z = -\lambda_z V_z.$$

```
class Viscosity : public Optics
{ public:
  double Lx, Ly, Lz;
  Viscosity ();
  void Initialization();
  Force GetForce(Coordinate C, double t);
};
```

7. Библиотека электромагнитных полей

а) Переменное продольное магнитное поле

Продольное магнитное поле приводит к циклотронному движению заряженных частиц в поперечном направлении. Можно дополнительно задать изменение амплитуды поля во времени:

$$B_z = B_0 \cos(\omega t + \varphi) \qquad (11)$$

```
class MagneticField : public Optics
{ public:
  double Bo, omega, phi;
  MagneticField ();
  void Initialization();
  Field GetField(Coordinate C, double t);
};
```

б) Переменное поперечное электрическое поле

Дополнительное поперечное электрическое поле вращается со временем вокруг центра координат. Такое вращающееся поле может быть задано разрезными электродами, к которым приложено переменное электрическое поле со сдвигом на определенную фазу:

$$E_x = E_0 \cos(\omega t),$$
$$E_y = E_0 \sin(\omega t). \qquad (12)$$

```
class ElectricField : public Optics
{ public:
  double Eo, omega;
  ElectricField ();
  void Initialization();
  Field GetField (Coordinate C, double t);
};
```


в) Поле пространственного заряда (радиальное электрическое поле)

В отличие от прямого вычисления взаимодействия заряженных частиц между собой можно задать простую модель пространственного заряда частиц, которая представляет собой радиальное электрическое поле, зависящее от заданной плотности частиц n:

$$F_x = \frac{2\pi e x n}{\varepsilon_0},$$
$$F_y = \frac{2\pi e y n}{\varepsilon_0}, \qquad (13)$$
$$F_z = \frac{2\pi e z n}{\varepsilon_0}.$$


```
class Space : public Optics
{
  public:
    double n;
    Space();
    void Initialization();
    Fields getField(Coordinates C, double t);
};
```

8. Численные методы интегрирования

Возможны несколько численных методов для реализации динамики движения частиц. Простейшим является метод Эйлера, когда на каждом шаге интегрирования вычисляется сумма сил от активных оптических элементов. Пример такого метода представлен в описании класса Dynamics.

Более точный метод – метод Рунге–Кутты, когда на каждом шаге интегрирования сумма сил вычисляется несколько раз в окрестности данной точки. Однако эти методы дают довольно большую ошибку для моделирования замкнутых систем, когда сохраняется полная энергия системы. В нашем случае, когда мы используем силу вязкости, энергия системы не сохраняется, и использование этих методов интегрирования вполне допустимо.

а) Метод Эйлера

Метод Эйлера является простым вариантом реализации метода Рунге-Кутты первого порядка. В векторной форме его можно записать в следующем виде:

$$\vec{C} = \vec{C}_0 + \Delta t \sum_{i=0}^{n} \vec{F}_i(\vec{C}_0, t_0), \qquad (14)$$

где $\vec{C}$ – вектор координат, $\vec{F}$ – вектор сил. Суммирование вектора сил производится по всем активным элементам библиотеки сил. Как правило, дрейфовый участок присутствует при суммировании вектора сил.

б) Метод Рунге–Кутты

Для реализации метода Рунге–Кутты необходимо иметь функцию, которая возвращает сумму сил от активных элементов в зависимости от координаты частицы и времени интегрирования. В классе координат нужно еще определить функцию умножения вектора координат на число:

$$t_1 = t_0 + \Delta t/2, \quad t_2 = t_0 + \Delta t,$$

$$\vec{C}_1 = \Delta t \sum_{i=0}^{n} \vec{F}_i(\vec{C}_0, t_0), \quad \vec{C}_2 = \Delta t \sum_{i=0}^{n} \vec{F}_i(\vec{C}_0 + \vec{C}_1/2, t_1), \qquad (15)$$

$$\vec{C}_3 = \Delta t \sum_{i=0}^{n} \vec{F}_i(\vec{C}_0 + \vec{C}_2/2, t_1), \quad \vec{C}_4 = \Delta t \sum_{i=0}^{n} \vec{F}_i(\vec{C}_0 + \vec{C}_3, t_2),$$

$$\vec{C} = \vec{C}_0 + \frac{\vec{C}_1 + 2\vec{C}_2 + 2\vec{C}_3 + \vec{C}_4}{6}.$$

9. Интерфейс общей программы

Интерфейс программы должен обеспечить ввод начальных данных и инициализацию всех объектов программы, запуск алгоритма динамики частиц и вывод на экран во время интегрирования движения координат частиц: *y* от *x*, *Vx* от *x*, *Vy* от *y*. Полезно также выводить на экран графики зависимости поперечных эмиттансов от времени, которые описываются следующими формулами:

$$\varepsilon_x = \sqrt{< x^2 >< V_x^2 > - < xV_x >^2},$$

$$\varepsilon_y = \sqrt{< y^2 >< V_y^2 > - < yV_y >^2}. \qquad (16)$$

Здесь угловые скобки означают усреднение по всем частицам, т.е. для горизонтальной координаты имеем

$$< x^2 >= \frac{1}{N}\sum_{i=1}^{N} x_i^2, < V^2 >= \frac{1}{N}\sum_{i=1}^{N} V_i^2, < xV >= \frac{1}{N}\sum_{i=1}^{N} xV_i. \qquad (17)$$

10. Управляющий модуль программы

Управляющий модуль с помощью интерфейса производит инициализацию всех объектов и организует работу всей программы. В конструкторе инициализируется массив библиотеки оптических элементов, а в деструкторе осуществляется освобождение памяти, выделенное под библиотеку оптических элементов. Перед началом интегрирования движения необходимо инициализировать все объекты в программе. Константа LIBNUM определяет количество элементов библиотеки полей.

```
#define LIBNUM 7
class Dynamics
{ public:
  Optics* Lib[LIBNUM];
  double dt;
  double t;
  void Initialization();
  void Run();
  Dynamics()
  { Lib[0] = new Drift;
    Lib[1] = new Rigidity;
    Lib[2] = new Viscosity;
    Lib[3] = new MagneticField;
    Lib[4] = new ElectricField;
    Lib[5] = new Space;
    Lib[6] = new Beam;
  }
 ~Dynamics()
  { for (int i = 0; i < LIBNUM; i++)
      delete Lib[i];
  }
};
```



```
void Dynamics::Initialization()
{ t = 0;
  for (int i = 0; i < LIBNUM, i++)
    Lib[i]->Initialization();
}

void Dynamics::Run()
{ for (int i = 0; i < Particles.Number; i++)
  { Force F;
    for (int j = 0; j < LIBNUM; j++)
    if (Lib[j]->use)
      F += Lib[j]->GetForces(Particles[i], t);
    Particles[i] += F * dt;
  }
  t += dt;
}
```

Примеры расчетов

Далее приводятся результаты работы программы, созданной студентами в ходе выполнения курсовой работы. Эти расчеты были сделаны для двухмерного случая, когда не учитывается продольное движение частиц. Получение подобных результатов является хорошим критерием правильности работы программы.

а) Частица в магнитном поле с учетом пространственного заряда

Частица в магнитном поле при наличии радиального поля вследствие пространственного заряда движется по циклоиде (рис.2). Траектория состоит из двух движений: движения по ларморовской спирали и дрейфового движения вокруг центра координат.

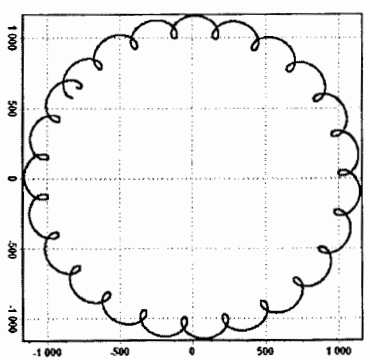

Рис.2. Движение частицы в продольном магнитном и радиальном электрическом полях

Скорость вращения частицы вокруг центра координат описывается дрейфовой частотой вращения:

$$\omega = \sqrt{\frac{\omega_B^2}{2} - \alpha} - \omega_B \sqrt{\frac{\omega_B^2}{2} - \alpha}\,, \qquad (18)$$

где $\alpha = \frac{2\pi Z^2 e^2 n}{\varepsilon_0}$ – коэффициент, зависящий от заряда частицы и плотности частиц на единицу объема, $\omega_B = \frac{ZeB_z}{Amc^2}$ – циклотронная частота, определяемая продольным магнитным полем и массой частицы.

б) Частица в магнитном и вращающемся электрическом полях

Включение дополнительного вращающегося поперечного электрического поля приводит к изменению дрейфового движения частицы. Если выбрать частоту вращающегося поля кратной частоте дрейфового движения, то можно получить разнообразные замкнутые траектории (рис. 3). Причем форма таких траекторий зависит от выбора направления вращения переменного электрического поля.

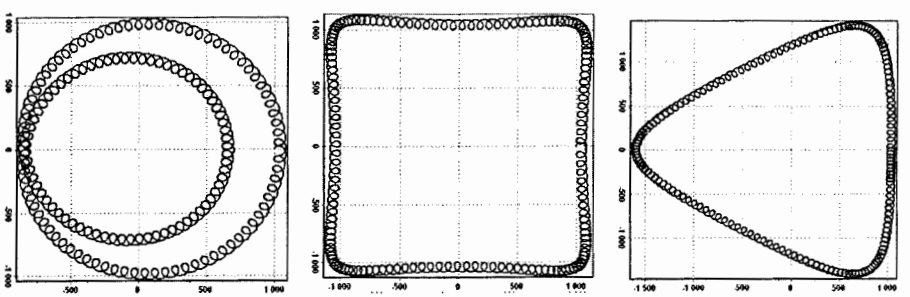

Рис.3. Траектории частицы при дополнительном вращающемся электрическом поле, когда его частота кратна частоте дрейфового движения

в) Кристаллы заряженных частиц

Кристаллы заряженных частиц, как было указано выше, получаются при совокупном действии фокусирующей и охлаждающей сил, которое сопровождается кулоновским расталкиванием частиц. На рис.1 были приведены результаты расчета кристаллического состояния частиц для трехмерного случая. В рамках курсовой работы достаточно получить кристаллы частиц для двухмерного случая.

Примеры кристаллов для разного количества частиц приведены на рис. 4 для семи, одиннадцати и двадцати частиц соответственно.

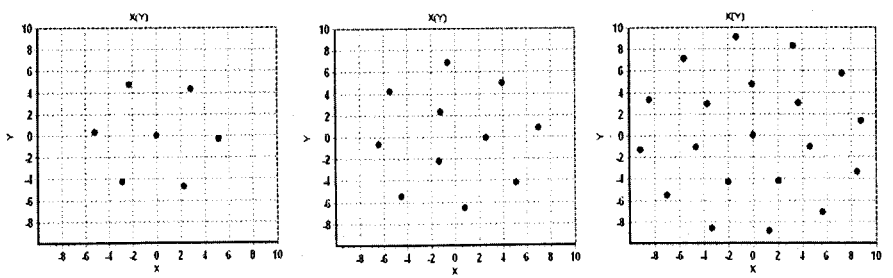

Рис.4. Кристаллы заряженных частиц для двухмерного случая

При моделировании кристаллов заряженных частиц необходимо правильно выбрать соотношение между фокусирующей и охлаждающей силами. Если силу охлаждения выбрать слишком большую, то частицы «замерзнут», не успев выстроиться в кристалл. Если охлаждающая сила будет слишком мала, то частицы будут колебаться относительно центра координат очень длительное время. На форму кристаллов также большое влияние оказывает соотношение между горизонтальным и вертикальным коэффициентами жесткости и вязкости.

Литература

[1] Мешков И.Н., Сидорин А.О., Смирнов А.В., Сыресин Е.М., Трубников Г.В. Кристаллические пучки в накопителях заряженных частиц //Письма в ЭЧАЯ. (2004) №3 (120). http://lepta.jinr.ru/betacool

[2] Wei J., Draeseke A., Sessler M., Li X. Diverse topics in crystalline beams //Proceedings of the 31st Workshop of the INFN Eloisatron Project. Erice, Italy, November 12-21 (1995) p.229.

[3] Hasse R., Schiffer J. The Structure of the Cylindrical Confined Coulomb Lattice //Annals of Physics. **203** (1990) p.419.

[4] Meshkov I., Sidorin A., Smirnov A., Syresin E., Katayama T., Tsitsui H., Mohl D. Simulation Study of Ordered Ion Beams. Prepirnt RIKEN-AF-AC-42, July 2003.

Учебное издание

Александр Валентинович СМИРНОВ,
Дмитрий Алексеевич КРЕСТНИКОВ

Программа по моделированию динамики заряженных частиц

УНЦ-2008-33

Редактор *Е. В. Сабаева*

Получено 20.05.2008. Подписано в печать 08.09.2008.
Формат 60 × 90/16. Бумага офсетная. Печать офсетная.
Усл. печ. л. 1,75. Уч.-изд. л. 1,79. Тираж 240 экз. Заказ № 56293.

Издательский отдел Объединенного института ядерных исследований
141980, г. Дубна, Московская обл., ул. Жолио-Кюри, 6.
E-mail: publish@jinr.ru
www.jinr.ru/publish/