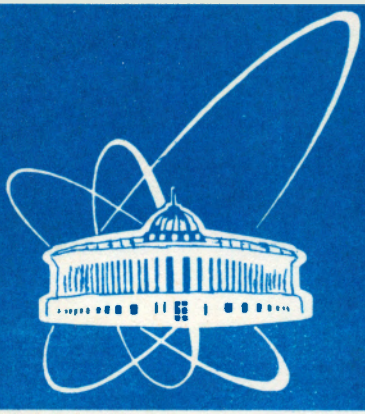

СООБЩЕНИЯ
ОБЪЕДИНЕННОГО
ИНСТИТУТА
ЯДЕРНЫХ
ИССЛЕДОВАНИЙ

Дубна

Р2-98-221

В.С.Барашенков[1], А.Г.Соловьев[2], А.Н.Соснин[3]

АЛГОРИТМ РАСЧЕТА ПЕРЕСЕЧЕНИЙ ЧАСТИЦЕЙ ЗОН В ГЕТЕРОГЕННОЙ СРЕДЕ

(Геометрические модули программы моделирования межъядерных каскадов)

[1]E-mail: barashenkov@lcta30.jinr.dubna.su
[2]E-mail: solovjev@decimal.jinr.dubna.su
[3]E-mail: sosnin@decimal.jinr.dubna.su

1998

1. Введение

В процессе расчета межъядерного каскада приходится рассматривать движение каскадной частицы в среде (мишени), состоящей из нескольких различных по составу и форме областей (зон). Траектория частицы является отрезком прямой и задается координатами ее начального и конечного положений. Границы зон, на которые разбита среда, являются поверхностями не более чем второго порядка. Требуется найти ближайшую к начальной точку пересечения траектории частицы с границей следующей зоны (если такое пересечение имеет место).

Пусть $i = 1, \ldots, N$ поверхностей описываются уравнениями:

$$\begin{array}{rcccccc} & a_{1,i}x^2 & + & a_{2,i}y^2 & + & a_{3,i}z^2 & + \\ + & 2a_{4,i}xy & + & 2a_{5,i}yz & + & 2a_{6,i}zx & + \\ + & 2a_{7,i}x & + & 2a_{8,i}y & + & 2a_{9,i}z & + \\ + & a_{10,i} & = & 0\,, & & & \end{array} \tag{1.1}$$

отрезок, соединяющий точки $\vec{r}_0 = (x_0, y_0, z_0)$ и $\vec{r}_1 = (x_1, y_1, z_1)$, задан соотношением:

$$\vec{r} \;=\; (1-t)\vec{r}_0 + t\vec{r}_1\,, \tag{1.2}$$

где параметр $0 < t < 1$.

Для i-ой поверхности задача нахождения ближайшей к $\vec{r}_0$ точки пересечения сводится к вычислению корней уравнения

$$A_i t^2 + 2B_i t + C_1 = 0 \tag{1.3}$$

на отрезке $0 < t < 1$. Коэффициенты этого уравнения вычисляются по формулам

$$\begin{aligned} A_i \;&=\; a_{1,i}(x_1-x_0)^2 + a_{2,i}(y_1-y_0)^2 + a_{3,i}(z_1-z_0)^2 + \\ &+\; 2a_{4,i}(x_1-x_0)(y_1-y_0) + 2a_{5,i}(y_1-y_0)(z_1-z_0) + \\ &+\; 2a_{6,i}(z_1-z_0)(x_1-x_0)\,, \end{aligned}$$

$$\begin{aligned} B_i \;&=\; a_{1,i}(x_1-x_0)x_0 + a_{2,i}(y_1-y_0)y_0 + a_{3,i}(z_1-z_0)z_0 + \\ &+\; a_{4,i}((x_1-x_0)y_0 + (y_1-y_0)x_0) + a_{5,i}((y_1-y_0)z_0 + (z_1-z_0)y_0) + \\ &+\; a_{6,i}((z_1-z_0)x_0 + (x_1-x_0)z_0) + \\ &+\; a_{7,i}(x_1-x_0) + a_{8,i}(y_1-y_0) + a_{9,i}(z_1-z_0)\,, \end{aligned}$$

$$\begin{aligned} C_i \;&=\; a_{1,i}x_0^2 + a_{2,i}y_0^2 + a_{3,i}z_0^2 + \\ &+\; 2a_{4,i}x_0y_0 + 2a_{5,i}y_0z_0 + 2a_{6,i}z_0x_0 + \\ &+\; 2a_{7,i}x_0 + 2a_{8,i}y_0 + 2a_{9,i}z_0 + \\ &+\; a_{10,i}\,. \end{aligned}$$

Если уравнение (1.3) не имеет корней в указанном интервале, то отрезок (1.2) не пересекает i-ю поверхность (1.1). В случае, если уравнение (1.3) имеет один

корень в этом интервале, то по формулам (1.2) находятся координаты точки пересечения с i-ой поверхностью (1.1). Если же уравнение (1.3) имеет два корня в этом интервале, то выбирается наименьший, а затем по формулам (1.2) определяются координаты искомой точки пересечения.

После того как таким образом найдены точки пересечения отрезка (1.2) со всеми N поверхностями (1.1), из них выбирается точка, ближайшая к $\vec{r}_0$.

2. Описание мишени

Прежде всего введем достаточно удобное описание зон мишени. Для каждой области будем использовать 9 параметров:

$$X_1\ ,\ Y_1\ ,\ Z_1\ ,\ R_1\ ,\ X_2\ ,\ Y_2\ ,\ Z_2\ ,\ R_2\ ,\ N_{TYPE}\ .$$

Параметр N_{TYPE} характеризует тип геометрии, а остальные параметры для программно реализованных нами типов геометрии имеют следующее значение:

1. Параллелепипед. $N_{TYPE} = 1$. Область определяется неравенствами

$$X_1 \le x \le X_2\ ,\ Y_1 \le y \le Y_2\ ,\ Z_1 \le z \le Z_2\ .$$

2. Цилиндр[4] с осью, параллельной оси z[5]. $N_{TYPE} = 2.3$. Область определяется неравенствами

$$(x - X_1)^2 + (y - Y_1)^2 \le R_1^2\ ,\ Z_1 \le z \le Z_2\ .$$

3. Конус с осью, параллельной оси z[6]. $N_{TYPE} = 3.3$. Область определяется неравенствами

$$(x - X_1)^2 + (y - Y_1)^2 \le \left(R_1 + (z - Z_1)\frac{R_2 - R_1}{Z_2 - Z_1}\right)^2\ ,\ Z_1 \le z \le Z_2\ .$$

4. Шар. $N_{TYPE} = 4$. Область определяется неравенством

$$(x - X_1)^2 + (y - Y_1)^2 + (z - Z_1)^2 \le R_1^2\ .$$

Приводимый ниже алгоритм позволяет при необходимости вводить и другие типы геометрии рассматриваемых зон.

Следует отметить, что если две зоны пересекаются, то мы будем считать, что область их пересечения принадлежит зоне с бóльшим номером.

[4]Хотя это частный случай конуса, для ускорения счета его полезно рассматривать отдельно.

[5]Программно реализованы также "Цилиндр с осью, параллельной оси Ox" ($N_{TYPE} = 2.1$) и "Цилиндр с осью, параллельной оси Oy" ($N_{TYPE} = 2.2$).

[6]Программно реализованы также "Конус с осью, параллельной оси Ox" ($N_{TYPE} = 3.1$) и "Конус с осью, параллельной оси Oy" ($N_{TYPE} = 3.2$).

Например, следующая комбинация параметров

X_1	Y_1	Z_1	R_1	X_2	Y_2	Z_2	R_2	N_{TYPE}
0	0	0		3	3	3		1
1	1	0	1			3		2.3

описывает куб с длиной ребра 3, внутри которого расположен цилиндр с радиусом 1 и высотой 3, ориентированный вдоль оси z, с координатами оси (1,1).
Комбинация параметров

X_1	Y_1	Z_1	R_1	X_2	Y_2	Z_2	R_2	N_{TYPE}
1	1	0	1			3		2.3
0	0	0		3	3	3		1

описывает лишь куб — информация о цилиндре утеряна в том смысле, что подпрограмма, определяющая, в какой зоне находится частица (подпрограмма OUT), не "чувствует" этого цилиндра.

Рассмотренное описание мишени оказывается достаточно удобным и позволяет задавать весьма сложные геометрические структуры, что часто бывает необходимым при расчетах реальных систем.

3. Алгоритм

Алгоритм решения задачи организован следующим образом.

1. Для заданных описанным выше образом зон мишени вычисляются коэффициенты ограничивающих их поверхностей. Этот пункт алгоритма реализует подпрограмма GEOMETRY.

   (a) В случае параллелепипеда вычисляются коэффициенты для трех поверхностей (подпрограмма TYPE1):

   $$a_{1,i} = 1\ ,\ a_{7,i} = -\frac{X_1 + X_2}{2}\ ,\ a_{10,i} = X_1 X_2\ ,$$
   остальные нули ,
   $$a_{2,i+1} = 1\ ,\ a_{8,i+1} = -\frac{Y_1 + Y_2}{2}\ ,\ a_{10,i+1} = Y_1 Y_2\ ,$$
   остальные нули ,
   $$a_{3,i+2} = 1\ ,\ a_{9,i+2} = -\frac{Z_1 + Z_2}{2}\ ,\ a_{10,i+2} = Z_1 Z_2\ ,$$
   остальные нули ;

   (b) Для цилиндра (с осью, параллельной оси z) вычисляются коэффициенты для двух поверхностей (подпрограмма TYPE23)[7]:

   $$a_{1,i} = 1\ ,\ a_{2,i} = 1\ ,\ a_{7,i} = -X_1\ ,\ a_{8,i} = -Y_1\ ,$$

[7]Подпрограммы TYPE21 и TYPE22 — для цилиндров, ориентированных вдоль осей x и y соответственно.

$a_{10,i} = X_1^2 + Y_1^2 - R_1^2$, остальные нули ,

$a_{3,i+1} = 1$, $a_{8,i+1} = -\dfrac{Z_1 + Z_2}{2}$, $a_{10,i+1} = Z_1 Z_2$,
остальные нули ;

(c) Для конуса (с осью, параллельной оси z) вычисляются коэффициенты для двух поверхностей (подпрограмма TYPE33)[8]:

$$a_{1,i} = 1 \ , \ a_{2,i} = 1 \ , \ a_{3,i} = -\left(\frac{R_2 - R_1}{Z_2 - Z_1}\right)^2 ,$$

$$a_{7,i} = -X_1 \ , \ a_{8,i} = -Y_1 \ , \ a_{9,i} = Z_1 \left(\frac{R_2 - R_1}{Z_2 - Z_1}\right)^2 ,$$

$$a_{10,i} = X_1^2 + Y_1^2 - \left(R_1 - \frac{R_2 - R_1}{Z_2 - Z_1} Z_1\right)^2 \text{ , остальные нули ,}$$

$$a_{3,i+1} = 1 \ , \ a_{8,i+1} = -\frac{Z_1 + Z_2}{2} \ , \ a_{10,i+1} = Z_1 Z_2 \ ,$$
остальные нули ;

(d) Если зона имеет форму шара, то коэффициенты вычисляются для одной поверхности (подпрограмма TYPE4):

$a_{1,i} = 1$, $a_{2,i} = 1$, $a_{3,i} = 1$,
$a_{7,i} = -X_1$, $a_{8,i} = -Y_1$, $a_{9,i} = -Z_1$,
$a_{10,i} = X_1^2 + Y_1^2 + Z_1^2 - R_1^2$, остальные нули ;

2. Для определенных таким образом N поверхностей определяются N условий их ограниченности (та же подпрограмма GEOMETRY и вызываемые ею подпрограммы TYPE1, TYPE21, TYPE22, TYPE23, TYPE31, TYPE32, TYPE33, TYPE4). Эти условия в точности аналогичны приводимым выше при описании зон, т.е. из каждой поверхности вырезается часть, принадлежащая той зоне, которую эта поверхность ограничивает.

3. Для одной из N поверхностей решается уравнение (1.3) и находятся его корни, лежащие в интервале $0 < t < 1$ (подпрограмма CROSS). Если таковых нет, то выполняется переход к пункту 6. В случае если имеется один корень — переход к пункту 4. Если два корня, то выбирается меньший, после чего делается переход к пункту 4.

4. Подстановкой найденного корня уравнения (1.3) в формулы (1.2) находится точка пересечения отрезка с рассматриваемой поверхностью (подпрограмма CROSS).

5. Для найденной точки пересечения проверяются условия, установленные в пункте 2 (подпрограмма CHECK). Если они выполняются, то эта точка запоминается.

[8]Подпрограммы TYPE31 и TYPE32 — для конусов, ориентированных вдоль осей x и y соответственно.

6. Пункты 3 – 5 повторяются для каждой из N поверхностей (этот процесс реализован подпрограммой COROUT).

7. Из найденных таким образом точек пересечения (их число $\leq N$) выбирается ближайшая к $\vec{r}_0$.

Отметим, что при рассмотрении других типов геометрии нужно дополнить лишь пункты 1 и 2 описанного выше алгоритма, пункты 3 – 7 остаются неизменными. Таким образом, наш алгоритм достаточно универсален и позволяет, при необходимости, ввести новые геометрические формы зон мишени.

4. Программная реализация алгоритма

Все подпрограммы, реализующие описанный алгоритм, написаны на языке FORTRAN'77. Прежде всего, из входного файла INPUT.DAT считывается информация о зонах мишени. Соответствующий фрагмент INPUT.DAT имеет вид:

```
.....................................

C                    TARGET PARAMETERS
C                Number of zones in the target ( <31 )
                                4
C                Geometrical description of the zones.
C   (One string is used for every zone.
C   The explanation of the parameters see in subroutines OUT or GEOMETRY).
    0.      0.      0.      0.      0.      23.8    15.7    0.      2.3
    0.      0.      0.      0.      2.4     21.4    13.3    0.      2.3
    0.      0.      0.      0.      4.9     18.9    10.8    0.      2.3
    0.      0.      0.      0.      4.9     18.9     1.3    0.      2.3

.....................................
```

(В качестве примера рассмотрен случай четырех зон: это 4 цилиндра, вложенных последовательно один в другой).

Эти данные считываются подпрограммой BENPAK. Здесь же вызывается подпрограмма GEOMETRY, преобразующая эти данные в информацию о поверхностях, ограничивающих зоны. Соответствующий фрагмент BENPAK имеет вид:

```
.....................................

      READ (5,*)
      READ (5,*)
C     Number of zones in the target ( <31 )
      READ (5,*)NZONE
      READ (5,*)
      READ (5,*)
      READ (5,*)
```

```
C     Ceometrical description of the zones.
C     The explanation of parameters see in the subroutine OUT.
      DO 5 I=1,NZONE
    5 READ (5,*)(GH(J,I),J=1,9)
      CALL GEOMETRY
```

.....

Подпрограмма GEOMETRY для каждой зоны в зависимости от типа ее геометрии вызывает соответствующую подпрограмму (подпрограммы TYPE1, TYPE21 и т.п.), которая и вычисляет коэффициенты поверхностей, ограничивающих эту зону (COMMON /SURFACE/SF(10,60)), а также параметры, определяющие условия ограниченности этих поверхностей (COMMON /CONDITION/CN(9,60)). После этого определяется число поверхностей (COMMON /NSURFACES/NSF).

Вопрос о том, какой зоне принадлежит данная точка, решается подпрограммой OUT. По координатам точки, заданным в качестве аргументов, эта подпрограмма выдает либо отрицательное число, равное по абсолютной величине номеру зоны, в которой находится рассматриваемая точка (COMMON /GOOUT/GOT), либо положительное число, если точка лежит вне мишени.

Подпрограмма COROUT производит поиск первой точки пересечения частицей границы зоны. Для этого вызываются подпрограммы CROSS и CHECK. Подпрограмма CROSS решает уравнение (1.3) на отрезке $(0,1)$ и по формулам (1.2) находит точки пересечения (DIMENSION RC1(3),RC2(3)). Подпрограмма CHECK проверяет, действительно ли найденная подпрограммой CROSS точка принадлежит данной поверхности (IND>0), а не ее продолжению (IND<0). Кроме того, подпрограмма COROUT вычисляют потерю энергии заряженной частицей на данном отрезке пути.

Тексты подпрограмм приведены в Приложении.

В случае использования неописанных типов геометрии необходимо написать подпрограмму TYPE_new и включить соответствующую информацию в подпрограммы GEOMETRY, CHECK, OUT. Подпрограммы COROUT и CROSS при этом остаются неизменными.

Приложение

```
      SUBROUTINE OUT(X,Y,Z)
C-----------------------------------------------------------------------
C     DETERMINATION OF A ZONE NUMBER.
C     IF GOT > 0, THEN THE POINT IS OUTSIDE THE TARGET.
C     IF GOT < 0, THEN GOT = - ZONE NUMBER.
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /BLOK1/GH(9,30)
     *       /NZONE/NZONE
     *       /GOOUT/GOT
C-----------------------------------------------------------------------
      DO 100 I=NZONE,1,-1
C     Selection of a geometry type :
      IF(GH(9,I).EQ.1.0D0)GO TO 1
      IF(GH(9,I).EQ.2.1D0)GO TO 21
      IF(GH(9,I).EQ.2.2D0)GO TO 22
      IF(GH(9,I).EQ.2.3D0)GO TO 23
      IF(GH(9,I).EQ.3.1D0)GO TO 31
      IF(GH(9,I).EQ.3.2D0)GO TO 32
      IF(GH(9,I).EQ.3.3D0)GO TO 33
      IF(GH(9,I).EQ.4.0D0)GO TO 4
C-----------------------------------------------------------------------
 1    CONTINUE
C     Parallelepiped :
      IF(X.GE.GH(1,I).AND.X.LE.GH(2,I).AND.
     +   Y.GE.GH(3,I).AND.Y.LE.GH(4,I).AND.
     +   Z.GE.GH(5,I).AND.Z.LE.GH(6,I))GO TO 200
      GO TO 100
C-----------------------------------------------------------------------
 21   CONTINUE
C     Cylinder along x-axes :
      IF(X.GE.GH(1,I).AND.X.LE.GH(2,I).AND.
     +   DSQRT((Y-GH(3,I))**2+(Z-GH(5,I))**2).LE.GH(7,I))GO TO 200
      GO TO 100
C-----------------------------------------------------------------------
 22   CONTINUE
C     Cylinder along y-axes :
      IF(Y.GE.GH(3,I).AND.Y.LE.GH(4,I).AND.
     +   DSQRT((Z-GH(5,I))**2+(X-GH(1,I))**2).LE.GH(7,I))GO TO 200
      GO TO 100
C-----------------------------------------------------------------------
 23   CONTINUE
C     Cylinder along z-axes :
      IF(Z.GE.GH(5,I).AND.Z.LE.GH(6,I).AND.
```

```
     +   DSQRT((X-GH(1,I))**2+(Y-GH(3,I))**2).LE.GH(7,I))GO TO 200
         GO TO 100
C-------------------------------------------------------------------
 31  CONTINUE
C    Cone along x-axes :
     IF(X.GE.GH(1,I).AND.X.LE.GH(2,I).AND.
     +   DSQRT((Y-GH(3,I))**2+(Z-GH(5,I))**2).LE.
     +   GH(7,I)-GH(1,I)*(GH(8,I)-GH(7,I))/(GH(2,I)-GH(1,I)))GO TO 200
     GO TO 100
C-------------------------------------------------------------------
 32  CONTINUE
C    Cone along y-axes :
     IF(Y.GE.GH(3,I).AND.Y.LE.GH(4,I).AND.
     +   DSQRT((Z-GH(5,I))**2+(X-GH(1,I))**2).LE.
     +   GH(7,I)-GH(3,I)*(GH(8,I)-GH(7,I))/(GH(4,I)-GH(3,I)))GO TO 200
     GO TO 100
C-------------------------------------------------------------------
 33  CONTINUE
C    Cone along z-axes :
     IF(Z.GE.GH(5,I).AND.Z.LE.GH(6,I).AND.
     +   DSQRT((X-GH(1,I))**2+(Y-GH(3,I))**2).LE.
     +   GH(7,I)-GH(5,I)*(GH(8,I)-GH(7,I))/(GH(6,I)-GH(5,I)))GO TO 200
     GO TO 100
C-------------------------------------------------------------------
 4   CONTINUE
C    Sphere :
     IF(DSQRT((X-GH(1,I))**2+(Y-GH(3,I))**2+(Z-GH(5,I))**2).LE.
     +   GH(7,I))GO TO 200
     GO TO 100
C-------------------------------------------------------------------
 100 CONTINUE
     GOT=1.0D0
     RETURN
 200 GOT=-FLOAT(I)
     RETURN
     END

     SUBROUTINE COROUT(BACK,SIDE,FORV)
C-------------------------------------------------------------------
C    DETERMINATION OF THE CLOSEST INTERSECTION POINT.
C-------------------------------------------------------------------
     IMPLICIT REAL*8 (A-H,O-Z)
     COMMON /WRPRIN/WPIN(9)
    *       /WRPR/WP(9)
    *       /GOOUT/GOT
    *       /BLOK1/GH(9,30)
```



```
    *       /GHAUX/GHAUX(30)
    *       /NSURFACES/NSF
    *       /SURFACE/SF(10,60)
    *       /CONDITION/CN(9,60)
     DIMENSION RC1(3),RC2(3)
C-------------------------------------------------------------------
     BACK=0.0D0
     IF(SIDE.GT.0.0D0)RETURN
     NZ=IDINT(-SIDE+0.3)
     INDIKATOR=-1
C    Looking for the closest intersection point amongst all surfaces:
     DO 1 I=1,NSF
C    Determination of intersection points for the surfase of number I:
     CALL CROSS(I,RC1,RC2,IND1,IND2)
C    Checking for these intersection points:
     DEL1=0.0D0
     IF(IND1.GT.0)CALL CHECK(I,RC1(1),RC1(2),RC1(3),IND)
     IF(IND1.GT.0.AND.IND.GT.0)
    +DEL1=DSQRT((WP(1)-RC1(1))**2+(WP(2)-RC1(2))**2+(WP(3)-RC1(3))**2)
     DEL2=0.0D0
     IF(IND2.GT.0)CALL CHECK(I,RC2(1),RC2(2),RC2(3),IND)
     IF(IND2.GT.0.AND.IND.GT.0)
    +DEL2=DSQRT((WP(1)-RC2(1))**2+(WP(2)-RC2(2))**2+(WP(3)-RC2(3))**2)
C    Choose ot the closest intersection point:
     IF(BACK.GT.DMAX1(DEL1,DEL2))GO TO 1
     BACK=DMAX1(DEL1,DEL2)
 1   CONTINUE
     IF(BACK.LE.0.0D0)RETURN
C    To avoid stoping:
     BACK=BACK-1.0D-7
     PATH=
    +SQRT((WP(1)-WPIN(1))**2+(WP(2)-WPIN(2))**2+(WP(3)-WPIN(3))**2)
C    The closest intersection point:
     WP(1)=WP(1)-(WP(1)-WPIN(1))*BACK/PATH
     WP(2)=WP(2)-(WP(2)-WPIN(2))*BACK/PATH
     WP(3)=WP(3)-(WP(3)-WPIN(3))*BACK/PATH
C    Determination of a zone number:
     CALL OUT(WP(1),WP(2),WP(3))
C    Correction of the energy for the charged particles:
     IF(WP(8).EQ.0.)RETURN
     PATH=
    +SQRT((WP(1)-WPIN(1))**2+(WP(2)-WPIN(2))**2+(WP(3)-WPIN(3))**2)
     IF(GHAUX(NZ).GT.0.)WP(7)=TNEW(PATH,SIDE)
     FORV=WP(7)
     CALL REGPPD(SIDE,WPIN(7),WP(7))
C-------------------------------------------------------------------
```

```
      RETURN
      END

      SUBROUTINE CROSS(I,RC1,RC2,IND1,IND2)
C-----------------------------------------------------------------------
C     CALCULATION OF INTERSECTION POINTS :
C  Surface:       SF(1,I) * X^2 +   SF(2,I) * Y^2 +   SF(3,I) * Y^2 +
C              + 2 SF(4,I) * XY  + 2 SF(5,I) * YZ  + 2 SF(6,I) * ZX  +
C              + 2 SF(7,I) * X   + 2 SF(8,I) * Y   + 2 SF(9,I) * Z   +
C              +   SF(10,I) = 0.
C
C  Interval:      X = (1-T) * WPIN(1) + T * WP(1),
C                 Y = (1-T) * WPIN(2) + T * WP(2),
C                 Z = (1-T) * WPIN(3) + T * WP(3),          0 < T < 1.
C
C  To find intersection points, we should solve the
C  equation:      A * T^2 + 2 B * T + C = 0,                0 < T < 1.
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /SURFACE/SF(10,60)
     *       /WRPRIN/WPIN(9)
     *       /WRPR/WP(9)
      DIMENSION RC1(3),RC2(3)
C-----------------------------------------------------------------------
      IND1=-1
      IND2=-1
      T1=0.0D0
      T2=0.0D0
      DO 1 J=1,3
      RC1(J)=0.0D0
 1    RC2(J)=0.0D0
C-----------------------------------------------------------------------
C     Coeficients for the equation:
      A=SF(1,I)*(WP(1)-WPIN(1))**2+
     +SF(2,I)*(WP(2)-WPIN(2))**2+
     +SF(3,I)*(WP(3)-WPIN(3))**2+
     +2.0D0*SF(4,I)*(WP(1)-WPIN(1))*(WP(2)-WPIN(2))+
     +2.0D0*SF(5,I)*(WP(2)-WPIN(2))*(WP(3)-WPIN(3))+
     +2.0D0*SF(6,I)*(WP(3)-WPIN(3))*(WP(1)-WPIN(1))
      B=SF(1,I)*(WP(1)-WPIN(1))*WPIN(1)+
     +SF(2,I)*(WP(2)-WPIN(2))*WPIN(2)+
     +SF(3,I)*(WP(3)-WPIN(3))*WPIN(3)+
     +SF(4,I)*((WP(1)-WPIN(1))*WPIN(2)+(WP(2)-WPIN(2))*WPIN(1))+
     +SF(5,I)*((WP(2)-WPIN(2))*WPIN(3)+(WP(3)-WPIN(3))*WPIN(2))+
     +SF(6,I)*((WP(3)-WPIN(3))*WPIN(1)+(WP(1)-WPIN(1))*WPIN(3))+
     +SF(7,I)*(WP(1)-WPIN(1))+
```



```
     +SF(8,I)*(WP(2)-WPIN(2))+
     +SF(9,I)*(WP(3)-WPIN(3))
      C=SF(1,I)*WPIN(1)**2+
     +SF(2,I)*WPIN(2)**2+
     +SF(3,I)*WPIN(3)**2+
     +2.0D0*SF(4,I)*WPIN(1)*WPIN(2)+
     +2.0D0*SF(5,I)*WPIN(2)*WPIN(3)+
     +2.0D0*SF(6,I)*WPIN(3)*WPIN(1)+
     +2.0D0*SF(7,I)*WPIN(1)+
     +2.0D0*SF(8,I)*WPIN(2)+
     +2.0D0*SF(9,I)*WPIN(3)+
     +SF(10,I)
C-----------------------------------------------------------------------
      IF(DABS(A).GT.1.0D-10)GO TO 2
      IF(DABS(B).LT.1.0D-10)RETURN
      T1=-0.5D0*C/B
      GO TO 3
C-----------------------------------------------------------------------
C     Determinant of the equation:
 2    D=B*B-A*C
      IF(D.LE.0.0D0)RETURN
      DET=DSQRT(D)
C     Roots of the equation:
      T1=-B/A-DET/A
      T2=-B/A+DET/A
C-----------------------------------------------------------------------
 3    IF(T1.GT.0.0D0.AND.T1.LT.1.0D0)IND1=+1
C     It T1 is in (0,1), then the first intersection point:
      IF(IND1.GT.0)RC1(1)=(1.0D0-T1)*WPIN(1)+T1*WP(1)
      IF(IND1.GT.0)RC1(2)=(1.0D0-T1)*WPIN(2)+T1*WP(2)
      IF(IND1.GT.0)RC1(3)=(1.0D0-T1)*WPIN(3)+T1*WP(3)
      IF(T2.GT.0.0D0.AND.T2.LT.1.0D0)IND2=+1
C     It T2 is in (0,1), then the second intersection point:
      IF(IND2.GT.0)RC2(1)=(1.0D0-T2)*WPIN(1)+T2*WP(1)
      IF(IND2.GT.0)RC2(2)=(1.0D0-T2)*WPIN(2)+T2*WP(2)
      IF(IND2.GT.0)RC2(3)=(1.0D0-T2)*WPIN(3)+T2*WP(3)
C-----------------------------------------------------------------------
      RETURN
      END

      SUBROUTINE CHECK(I,X,Y,Z,IND)
C-----------------------------------------------------------------------
C     CHECK: DOES THE INTERSECTION POINT BELONG TO A ZONE OF NUMBER I ?
C     IF IND > 0, THEN YES.
C     IF IND < 0, THEN NO.
C-----------------------------------------------------------------------
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /CONDITION/CN(9,60)
C-----------------------------------------------------------------------
      IND=+1
C     Selection of a condition type :
      IF(CN(9,I).EQ.1.0D0)GO TO 1
      IF(CN(9,I).EQ.2.1D0)GO TO 21
      IF(CN(9,I).EQ.2.2D0)GO TO 22
      IF(CN(9,I).EQ.2.3D0)GO TO 23
      IF(CN(9,I).EQ.3.1D0)GO TO 31
      IF(CN(9,I).EQ.3.2D0)GO TO 32
      IF(CN(9,I).EQ.3.3D0)GO TO 33
      IF(CN(9,I).EQ.4.0D0)GO TO 4
C-----------------------------------------------------------------------
 1    CONTINUE
C     Parallelepiped :
      IF(X.GT.CN(1,I)-1.0D-10.AND.X.LT.CN(2,I)+1.0D-10.AND.
     +   Y.GT.CN(3,I)-1.0D-10.AND.Y.LT.CN(4,I)+1.0D-10.AND.
     +   Z.GT.CN(5,I)-1.0D-10.AND.Z.LT.CN(6,I)+1.0D-10)RETURN
      GO TO 100
C-----------------------------------------------------------------------
 21   CONTINUE
C     Cylinder along x-axes :
      IF(X.GT.CN(1,I)-1.0D-10.AND.X.LT.CN(2,I)+1.0D-10.AND.
     +   DSQRT((Y-CN(3,I))**2+(Z-CN(5,I))**2).LT.CN(7,I)+1.0D-10)RETURN
      GO TO 100
C-----------------------------------------------------------------------
 22   CONTINUE
C     Cylinder along y-axes :
      IF(Y.GT.CN(3,I)-1.0D-10.AND.Y.LT.CN(4,I)+1.0D-10.AND.
     +   DSQRT((Z-CN(5,I))**2+(X-CN(1,I))**2).LT.CN(7,I)+1.0D-10)RETURN
      GO TO 100
C-----------------------------------------------------------------------
 23   CONTINUE
C     Cylinder along z-axes :
      IF(Z.GT.CN(5,I)-1.0D-10.AND.Z.LT.CN(6,I)+1.0D-10.AND.
     +   DSQRT((X-CN(1,I))**2+(Y-CN(3,I))**2).LT.CN(7,I)+1.0D-10)RETURN
      GO TO 100
C-----------------------------------------------------------------------
 31   CONTINUE
C     Cone along x-axes :
      IF(X.GT.CN(1,I)-1.0D-10.AND.X.LT.CN(2,I)+1.0D-10.AND.
     +   DSQRT((Y-CN(3,I))**2+(Z-CN(5,I))**2).LT.
     +   CN(7,I)-CN(1,I)*(CN(8,I)-CN(7,I))/(CN(2,I)-CN(1,I))+1.0D-10)
     +RETURN
      GO TO 100
```



```
C-----------------------------------------------------------------------
 32   CONTINUE
C     Cone along y-axes :
      IF(Y.GT.CN(3,I)-1.0D-10.AND.Y.LT.CN(4,I)+1.0D-10.AND.
     +   DSQRT((Z-CN(5,I))**2+(X-CN(1,I))**2).LT.
     +   CN(7,I)-CN(3,I)*(CN(8,I)-CN(7,I))/(CN(4,I)-CN(3,I))+1.0D-10)
     +RETURN
      GO TO 100
C-----------------------------------------------------------------------
 33   CONTINUE
C     Cone along z-axes :
      IF(Z.GT.CN(5,I)-1.0D-10.AND.Z.LT.CN(6,I)+1.0D-10.AND.
     +   DSQRT((X-CN(1,I))**2+(Y-CN(3,I))**2).LT.
     +   CN(7,I)-CN(5,I)*(CN(8,I)-CN(7,I))/(CN(6,I)-CN(5,I))+1.0D-10)
     +RETURN
      GO TO 100
C-----------------------------------------------------------------------
 4    CONTINUE
C     Sphere :
      IF(DSQRT((X-CN(1,I))**2+(Y-CN(3,I))**2+(Z-CN(5,I))**2).LT.
     +CN(7,I)+1.0D-10)RETURN
      GO TO 100
C-----------------------------------------------------------------------
 100  IND=-1
      RETURN
      END

      SUBROUTINE GEOMETRY
C-----------------------------------------------------------------------
C     ANALYSIS OF ZONES' SURFASES
C     AND CALCULATION OF THEIR COEFFICIENTS.
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /BLOK1/GH(9,30)
     *       /NZONE/NZONE
     *       /NSURFACES/NSF
C-----------------------------------------------------------------------
      K=1
C     Selection of a geometry type and
C     calculation of coeficients for boundary surfaces:
      DO 1 I=1,NZONE
C     Parallelepiped:
      IF(GH(9,I).EQ.1.0D0)CALL TYPE1(I,K)
C     Cylinder along x-axes:
      IF(GH(9,I).EQ.2.1D0)CALL TYPE21(I,K)
C     Cylinder along y-axes:
```

```
      IF(GH(9,I).EQ.2.2D0)CALL TYPE22(I,K)
C     Cylinder along z-axes:
      IF(GH(9,I).EQ.2.3D0)CALL TYPE23(I,K)
C     Cone along x-axes:
      IF(GH(9,I).EQ.3.1D0)CALL TYPE31(I,K)
C     Cone along y-axes:
      IF(GH(9,I).EQ.3.2D0)CALL TYPE32(I,K)
C     Cone along z-axes:
      IF(GH(9,I).EQ.3.3D0)CALL TYPE33(I,K)
C     Sphere:
      IF(GH(9,I).EQ.4.0D0)CALL TYPE4(I,K)
 1    CONTINUE
C     Number of surfaces:
      NSF=K-1
C-----------------------------------------------------------------------
      RETURN
      END

      SUBROUTINE TYPE1(I,K)
C-----------------------------------------------------------------------
C     PARALLELEPIPED: GH(9,I)=1.0 .
C     CALCULATION OF BOUNDARY SURFACES COEFICIENTS.
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /BLOK1/GH(9,30)
     *       /SURFACE/SF(10,60)
     *       /CONDITION/CN(9,60)
C-----------------------------------------------------------------------
C     The first surface:
      SF(1,K)=1.0D0
      SF(2,K)=0.0D0
      SF(3,K)=0.0D0
      SF(4,K)=0.0D0
      SF(5,K)=0.0D0
      SF(6,K)=0.0D0
      SF(7,K)=-0.5D0*(GH(1,I)+GH(2,I))
      SF(8,K)=0.0D0
      SF(9,K)=0.0D0
      SF(10,K)=GH(1,I)*GH(2,I)
C     The second surface:
      SF(1,K+1)=0.0D0
      SF(2,K+1)=1.0D0
      SF(3,K+1)=0.0D0
      SF(4,K+1)=0.0D0
      SF(5,K+1)=0.0D0
      SF(6,K+1)=0.0D0
```



```
      SF(7,K+1)=0.0D0
      SF(8,K+1)=-0.5D0*(GH(3,I)+GH(4,I))
      SF(9,K+1)=0.0D0
      SF(10,K+1)=GH(3,I)*GH(4,I)
C     The third surface:
      SF(1,K+2)=0.0D0
      SF(2,K+2)=0.0D0
      SF(3,K+2)=1.0D0
      SF(4,K+2)=0.0D0
      SF(5,K+2)=0.0D0
      SF(6,K+2)=0.0D0
      SF(7,K+2)=0.0D0
      SF(8,K+2)=0.0D0
      SF(9,K+2)=-0.5D0*(GH(5,I)+GH(6,I))
      SF(10,K+2)=GH(5,I)*GH(6,I)
C-----------------------------------------------------------------------
C     Conditions:
      DO 1 J=1,9
      CN(J,K)=GH(J,I)
      CN(J,K+1)=GH(J,I)
 1    CN(J,K+2)=GH(J,I)
C-----------------------------------------------------------------------
      K=K+3
C-----------------------------------------------------------------------
      RETURN
      END

      SUBROUTINE TYPE23(I,K)
C-----------------------------------------------------------------------
C     CILINDER ALONG Z-AXES: GH(9,I)=2.3 .
C     CALCULATION OF BOUNDARY SURFACES COEFICIENTS.
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /BLOK1/GH(9,30)
     *       /SURFACE/SF(10,60)
     *       /CONDITION/CN(9,60)
C-----------------------------------------------------------------------
C     The first surface:
      SF(1,K)=1.0D0
      SF(2,K)=1.0D0
      SF(3,K)=0.0D0
      SF(4,K)=0.0D0
      SF(5,K)=0.0D0
      SF(6,K)=0.0D0
      SF(7,K)=-GH(1,I)
      SF(8,K)=-GH(3,I)
```

```
      SF(9,K)=0.0D0
      SF(10,K)=GH(1,I)**2+GH(3,I)**2-GH(7,I)**2
C     The second surface:
      SF(1,K+1)=0.0D0
      SF(2,K+1)=0.0D0
      SF(3,K+1)=1.0D0
      SF(4,K+1)=0.0D0
      SF(5,K+1)=0.0D0
      SF(6,K+1)=0.0D0
      SF(7,K+1)=0.0D0
      SF(8,K+1)=0.0D0
      SF(9,K+1)=-0.5D0*(GH(5,I)+GH(6,I))
      SF(10,K+1)=GH(5,I)*GH(6,I)
C-----------------------------------------------------------------------
C     Conditions:
      DO 1 J=1,9
      CN(J,K)=GH(J,I)
 1    CN(J,K+1)=GH(J,I)
C-----------------------------------------------------------------------
      K=K+2
C-----------------------------------------------------------------------
      RETURN
      END

      SUBROUTINE TYPE33(I,K)
C-----------------------------------------------------------------------
C     CONE ALONG Z-AXES: GH(9,I)=3.3 .
C     CALCULATION OF BOUNDARY SURFACES COEFICIENTS.
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /BLOK1/GH(9,30)
     *       /SURFACE/SF(10,60)
     *       /CONDITION/CN(9,60)
C-----------------------------------------------------------------------
C     The first surface:
      SF(1,K)=1.0D0
      SF(2,K)=1.0D0
      SF(3,K)=-((GH(8,I)-GH(7,I))/(GH(6,I)-GH(5,I)))**2
      SF(4,K)=0.0D0
      SF(5,K)=0.0D0
      SF(6,K)=0.0D0
      SF(7,K)=-GH(1,I)
      SF(8,K)=-GH(3,I)
      SF(9,K)=GH(5,I)*((GH(8,I)-GH(7,I))/(GH(6,I)-GH(5,I)))**2
      SF(10,K)=GH(1,I)**2+GH(3,I)**2-
     *(GH(7,I)-GH(5,I)*(GH(8,I)-GH(7,I))/(GH(6,I)-GH(5,I)))**2
```



```
C     The second surface:
      SF(1,K+1)=0.0D0
      SF(2,K+1)=0.0D0
      SF(3,K+1)=1.0D0
      SF(4,K+1)=0.0D0
      SF(5,K+1)=0.0D0
      SF(6,K+1)=0.0D0
      SF(7,K+1)=0.0D0
      SF(8,K+1)=0.0D0
      SF(9,K+1)=-0.5D0*(GH(5,I)+GH(6,I))
      SF(10,K+1)=GH(5,I)*GH(6,I)
C-----------------------------------------------------------------------
C     Conditions:
      DO 1 J=1,9,1
      CN(J,K)=GH(J,I)
 1    CN(J,K+1)=GH(J,I)
C-----------------------------------------------------------------------
      K=K+2
C-----------------------------------------------------------------------
      RETURN
      END

      SUBROUTINE TYPE4(I,K)
C-----------------------------------------------------------------------
C     SPHERE: GH(9,I)=4.0 .
C     CALCULATION OF BOUNDARY SURFACE COEFICIENTS.
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /BLOK1/GH(9,30)
     *       /SURFACE/SF(10,60)
     *       /CONDITION/CN(9,60)
C-----------------------------------------------------------------------
C     The only surface:
      SF(1,K)=1.0D0
      SF(2,K)=1.0D0
      SF(3,K)=1.0D0
      SF(4,K)=0.0D0
      SF(5,K)=0.0D0
      SF(6,K)=0.0D0
      SF(7,K)=-GH(1,I)
      SF(8,K)=-GH(3,I)
      SF(9,K)=-GH(5,I)
      SF(10,K)=GH(1,I)**2+GH(3,I)**2+GH(5,I)**2-GH(7,I)**2
C-----------------------------------------------------------------------
C     Conditions:
      DO 1 J=1,9
```

```
 1    CN(J,K)=GH(J,I)
C-------------------------------------------------------------------------
      K=K+1
C-------------------------------------------------------------------------
      RETURN
      END
```

Рукопись поступила в издательский отдел
27 июля 1998 года.

Барашенков В.С., Соловьев А.Г., Соснин А.Н.
Р2-98-221

Алгоритм расчета пересечений частицей зон
в гетерогенной среде
(Геометрические модули программы моделирования
межъядерных каскадов)

Приведены алгоритм и его программная реализация для моделирования пересечения траекторией частиц границ зон в гетерогенной среде. Границы зон описываются поверхностями не выше второго порядка. Приведены иллюстративные примеры.

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна, 1998

Перевод авторов

Barashenkov V.S., Solov'ev A.G., Sosnin A.N.
P2-98-221

An Algorithm to Calculate Zone Boundaries Crossing Points
by a Particle Trajectory in Heterogeneous Medium
(Geometrical Modules of the Program
to Simulate Internuclear Cascades)

An algorithm and a computer program to calculate a zone boundary crossing point of a particle trajectory in heterogeneous medium are discussed. Zone boundaries are described by surfaces with the order not higher than the second. Illustrations are presented.

The investigation has been performed at the Laboratory of Computing Techniques and Automation, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna, 1998