Kalkulator kinematyczny

Marcin Abram

WFAIS UJ
Kraków
POLAND

20 sierpnia 2009

Temat projektu

Cel

Stworzenie kalkulatora, który obliczałby maksymalny kąt wiązki produktów reakcji $A + B \longrightarrow C_1 + \ldots + C_n$.

Informacje na wejściu

- Energia i masy cząstek A i B;
- Masy cząstek $C_1, \ldots, C_n$;

Informacje na wyjściu

- Maksymalny kąt, pod którym można zaobserwować produkty reakcji;

Sposób realizacji

Stworzenie

- Obiektowej klasy `Ped` – jako czterowektorów;
- Obiektowej klasy `Pchniecie` – jako macierzy pchnięcia Lorentzowskiego;
- *Naturalnych* operacji na obiektach klasy `Ped` i `Pchniecie`:
  - dodawania/odejmowanie;
  - mnożenie skalarne (pędów);
  - składanie macierzy pchnięcia z czterowektorem i operacja odwrotna;

Dlaczego obiektowo?

- Automatyczna kontrola poprawności wprowadzanych danych (np. ujemna energia, ujemna *wartość* pędu, tj. $E^2 < m^2$);
- Ukrycie implementacji dla użytkownika (tzw. *abstrakcja danych*);
- Łatwość użycia;

Klasa Pchniecie

```
 class Pchniecie
{
   private:
      double M_[4][4];

   public:
      Pchniecie(double gamma, double x_, double y_, double z_);
      Pchniecie(Ped pa, Ped pb);
      ~Pchniecie(){}

      // wypisanie
      double gamma();
      double beta();
      void wypisz();

      // przeciazone operatory
      double& operator() (unsigned i, unsigned j);
      double operator() (unsigned i, unsigned j) const;
      Ped operator*(const Ped &p) const;
};
```

Klasa Ped

```
class Ped
{
   private:
      double p_[5];

   public:
      Ped();
      Ped(double E, double px, double py, double pz);
      Ped(double E, double m, double x, double y, double z);
      ~Ped(){}

      // wypisz
      double E() const { return p_[0]; }
      double px() const { return p_[1]; }
      double py() const { return p_[2]; }
      double pz() const { return p_[3]; }
      double pp() const { return p_[4]; }
      double p() const { return Sqrt(p_[4]); }
      double mm() const { return (p_[0]*p_[0] - p_[4]); }
      double m() const { return Sqrt(p_[0]*p_[0] - p_[4]); }
      double ax() const { return p_[1]/Sqrt(p_[4]); }
      double ay() const { return p_[2]/Sqrt(p_[4]); }
      double az() const { return p_[3]/Sqrt(p_[4]); }

      (...)
```

Klasa Ped

```
        (...)

        void wypisz();
        void losuj(double E, double m);

        // przeciazanie operatorow
        int operator==(const Ped &p2) const;
        Ped operator+(const Ped &p2) const;
        Ped operator-(const Ped &p2) const;
        Ped operator*(const Pchniecie &M) const;
        double operator*(const Ped &p2) const;
        double operator() (unsigned i) const;
};
```

Jak dokładnie przeciążać operatory:

http://pl.wikibooks.org/wiki/C%2B%2B/Przeciążanie_operatorów
http://www.intercon.pl/~sektor/cbx/advanced/overloading.html

Przykład użycia

```
Ped pa(10, 1, 1, 0, 0);     // Tworzy czteropęd o E=10, m=1
Ped pb(3, -1, 1, 0);        // Tworzy czteropęd {3, -1, 1, 0}
Ped pc = pa+pb;             // Dodawanie czteropędów

Pchniecie L(pa, pb);        // Tworzenie macierzy pchnięcia
Ped lpa = L*pa;             // Transformacja pchnięcia
Ped lpb = L*pb;
Ped lpc = lpa + lpb;

double a = lpa * lpb;       // Mnożenie skalarne WEKTORÓW pędu lpa, lpb

L.wypisz();                 // Wypisanie macierzy pchnięcia

Ped lpcl = lpc*L;           // Transformacja odwrotna
```

Teoria

Transformacja Lorentza pędów:

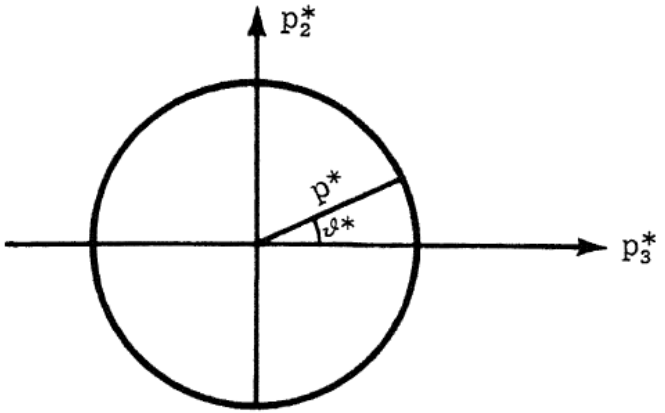
$$p_1 = p_1^\star, \tag{1}$$
$$p_2 = p_2^\star, \tag{2}$$
$$p_3 = \gamma(p_3^\star + \beta E^\star). \tag{3}$$

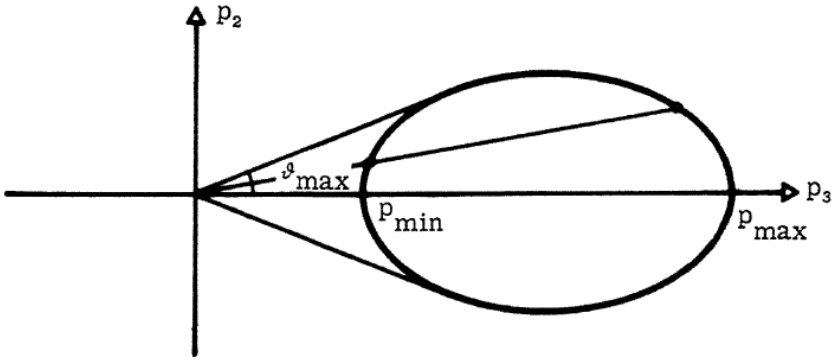
Dla układu środka masy (CM), mamy:

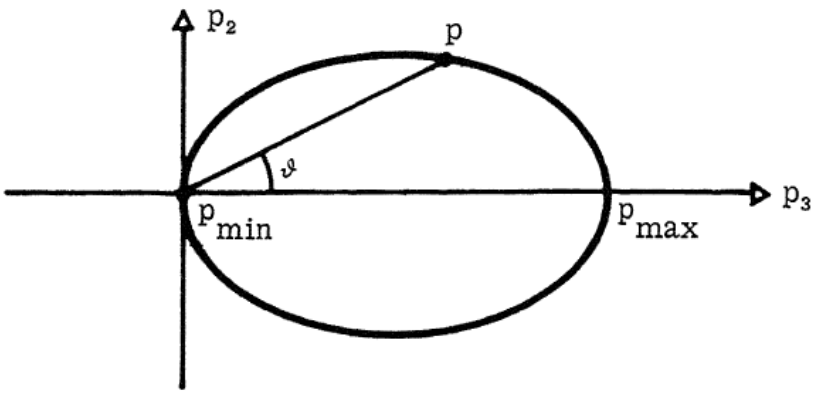
$$p_1^{\star 2} + p_2^{\star 2} + p_3^{\star 2} = p^{\star 2}. \tag{4}$$

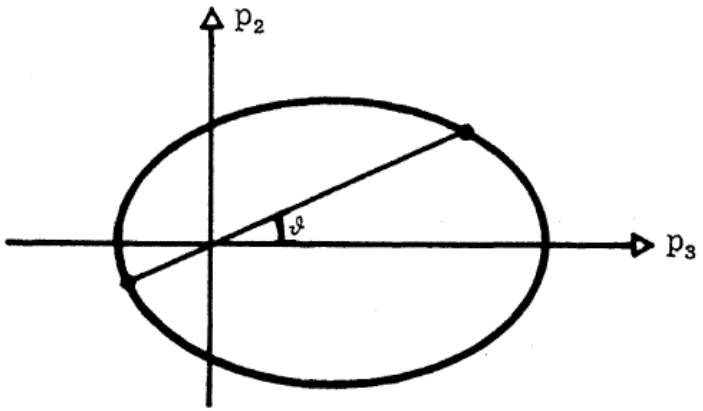
Po transformacji Lorentza do układu laboratoryjnego (LAB):

$$\frac{p_1^2}{p^{\star 2}} + \frac{p_2^2}{p^{\star 2}} + \frac{(p_3 - \beta\gamma E^\star)^2}{(\gamma p^\star)^2} = 1. \tag{5}$$


p_2^*
p^*
ϑ^*
p_3^*


p_2
ϑ_{max}
p_{min}
p_{max}
p_3


p_2
p
ϑ
p_{min}
p_{max}
p_3


p_2
ϑ
p_3

Ponieważ $p_1^{\star 2} + p_2^{\star 2} + p_3^{\star 2} = p^{\star 2}$ oraz $p_3 = \gamma(p_3^\star + \beta E^\star)$, to:

$$p_{3,max} = \gamma(p^\star + \beta E^\star), \tag{6}$$

$$p_{3,min} = \gamma(-p^\star + \beta E^\star). \tag{7}$$

Wprowadzamy:

$$v^\star \equiv \frac{p_0^\star}{E^\star} = \frac{p_0^\star}{\sqrt{p_0^{\star 2} + m^2}}. \tag{8}$$

Wtedy:

$$p_{3,min} = \gamma(-p^\star + \beta E^\star) = \gamma p^\star \left(\frac{\beta}{v^\star} - 1\right) = 0, \tag{9}$$

jest warunkiem na to, aby elipsa przechodziła przez punkt $(0, 0)$.
Jest to jednoznaczne z warunkiem $\beta = v^\star$.

Maksymalny kąt

Bez szkody dla ogólności możemy położyć $p_1 = 0$. Wówczac:

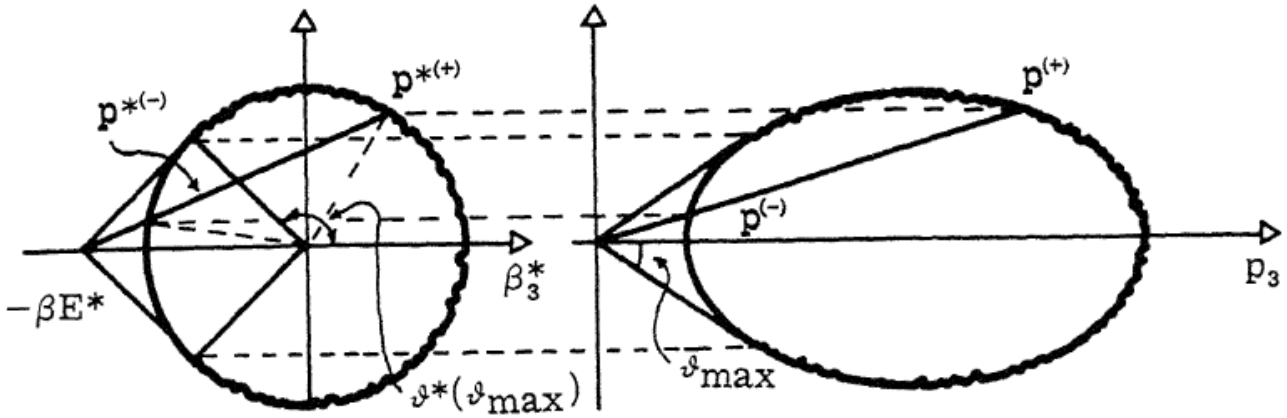
$$p_2 = p_3 \,\mathrm{tg}\phi. \tag{10}$$

Podstawiając do równania na elipsę dostajemy:

$$p_3^2 \,\mathrm{tg}^2\phi + \frac{(p_3 - \beta\gamma E^\star)^2}{\gamma^2} = p^{\star 2}. \tag{11}$$

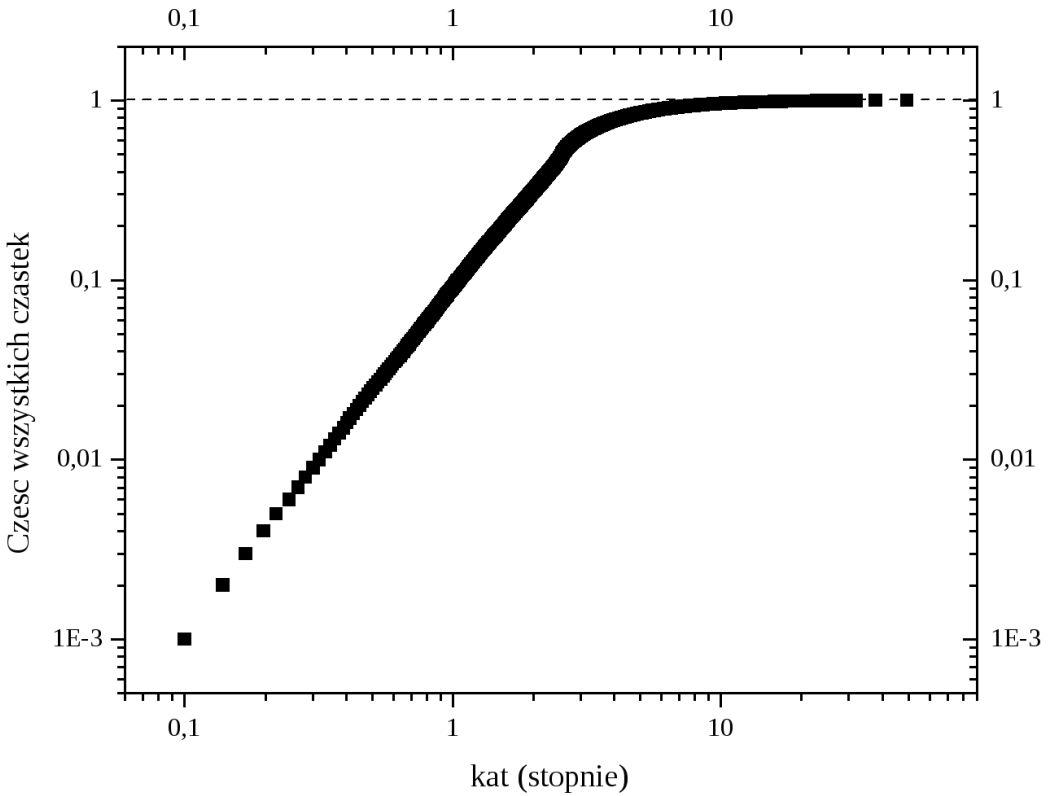
Rozwiązując ze względu na p_3 dostajemy:

$$p_3^{(\pm)} = \frac{\beta\gamma E^\star \pm \sqrt{\beta^2\gamma^2 E^{\star 2} - (1 + \gamma^2 \,\mathrm{tg}^2\phi)(\beta^2\gamma^2 E^{\star 2} - \gamma^2 p^{\star 2})}}{1 + \gamma^2 \,\mathrm{tg}^2\phi}. \tag{12}$$



Maksymalny kąt dla zerowania się wyrażenia pod pierwiastkiem. Dostejemy:

$$\mathrm{tg}^2\phi_{max} = \frac{v^{\star 2}}{\gamma^2(\beta^2 - v^{\star 2})}. \tag{13}$$



Rysunek: Część wszystkich cząstek, które mogą być zarejestrowane przez detektor o danej szerokości kątowej (w płaszczyźnie wiązki wejściowej).

Bibliografia

B. Hagedorn,
Relativistic Kinematics,
New York, Amsterdam, W. A. Benjamin, Inc., 1964.

Sektor van Skijlen,
C++ bez cholesterolu (wersja internetowa)
http://www.intercon.pl/∼sektor/cbx/