Hauptseminarvortag

Der RAMBO (RAndom Momenta BOoster) Algorithmus bei Teilchenkollisionen

Martin Rehwald

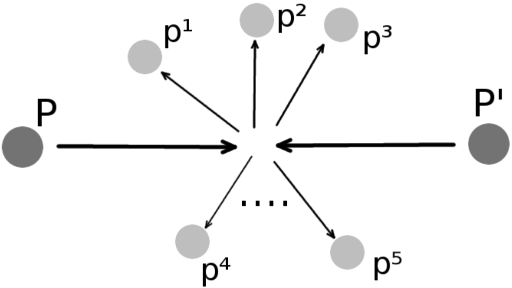
January 13, 2014

Motivation

- 2 einlaufende Teilchen im Experiment, n austretende Teilchen
- Oberservablen durch:

$$O = \int d\phi_n(P + P', p_1, ..., p_n) \frac{M(p_1, ..., p_n)}{8 \cdot K}$$

  definiert
- Ziel: Teilchenimpulse generieren



$$O = \int d\phi_n(P + P', p_1, ..., p_n) \frac{M(p_1, ..., p_n)}{8 \cdot K}$$

In der Teilchenphysik: $E >> m$
Der Lorentz-invariante Phasenraum ergibt sich aus:

$$\Phi_n(p_1, ..., p_n) = \int \prod_{i=1}^{n} \frac{d^4 p_i}{(2\pi)^3} \theta(p_i^0) \delta(p_i^2) (2\pi)^4 \delta\left(\tilde{P} - \sum_i^n p_i\right)$$

Mit $\tilde{P} = \sum$ einlaufende Impulse $= P + P'$

Bedingungen an die Impulse p:

- Viererimpulserhaltung
- Teilchen liegen auf der Massenschale

Ansatz

- 2 unterschiedliche Ansätze zur Erzeugung der auslaufenden Viererimpulse

1. **Sequenzielle Ansatz:**
    - Prinzip: Zerfall in 2-Teilchen
    - Problem: Für große Teilchenzahlen und vernachlässigbare Massen ist Algorithmus ineffizient
      z.B.: für 6 entstehende Teilchen können nur noch 20% der Ereignisse verwendet werden
2. **Demokratischer Ansatz:**
    - Nutze nach dem Phasenraumvolumen verteilte Zufallszahlen
    - Erste Umsetzung in Form des RAMBO (RAndom Momenta BOoster) Algorithmus

Grundlage des Algorithmus

- Phasenraum:

$$\Phi_n(p_1, ..., p_n) = \int \prod_{i=1}^{n} \frac{d^4 p_i}{(2\pi)^3} \theta(p_i^0) \delta(p_i^2) (2\pi)^4 \delta\left(\tilde{P} - \sum_i^n p_i\right)$$

- Wir starten mit:

$$R_n = \int \prod_i^n \frac{d^4 q_i}{(2\pi)^3} \theta(q_i^0) \delta(q_i^2) (2\pi)^4 f(q_i^0) = (2\pi)^{4-2n} \left(\int_0^\infty x f(x) dx\right)^n$$

- verbinden die q_i durch folgende Transformation (Lorentz und Skalierung) mit den p_i:

$$p_i^0 = x(\gamma q_i^0 + \vec{b}\vec{q}_i) \quad \text{und} \quad \vec{p}_i = x(\vec{q}_i + \vec{b} \cdot \vec{q}_i + a(\vec{b} \cdot \vec{q}_i)\vec{b})$$

$$Q^\mu = \sum_{i=1}^n q_i^\mu, \quad M = \sqrt{Q^2}, \quad \vec{b} = -\frac{1}{M}\vec{Q}, \quad \gamma = \frac{Q^0}{M}, \quad a = \frac{1}{1+\gamma}, \quad x = \frac{\sqrt{\tilde{P}^2}}{M}$$

- Einsetzen der Transformation führt zu:

$$R_n = \underbrace{\int \prod_{i=1}^{n} \frac{d^4 p_i}{(2\pi)^3} \theta(p_i^0) \delta(p_i^2) (2\pi)^4 \delta\left(\tilde{P} - \sum_i^n p_i\right)}_{\phi_n} \cdot S_n$$

- Für $f(x) = e^{-x}$ ist $S(\sqrt{\tilde{P}^2})$

$\rightarrow$ Durch die Lorentz-Transformation erzeugten Impulse erfüllen das Phasenraumkriterium!!!
Damit lässt sich ein Monte-Carlo Algorithmus erzeugen.

Der Algorithmus

Der eigentliche RAMBO Algorithmus wird in 2 Schritten realisiert:

1. Erzeugung von 4n Zufallszahlen (u_{i1}, u_{i2}, u_{i3} und u_{i4}) im Intervall [0,1]. Damit im Winkel gleichverteilte Zufallszahlen und Verteilung der 0-Komponente nach $x \cdot e^{-x}$:

$$c_i = 2u_{i1} - 1, \quad \varphi_i = 2\pi u_{i2}, \quad q_i^0 = -\ln(u_{i3}u_{i4})$$

$$q_i^x = q_i^0\sqrt{1-c_i^2}\cos(\varphi_i), \quad q_i^y = q_i^0\sqrt{1-c_i^2}\sin(\varphi_i), \quad q_i^z = q_i^0 c_i$$

2. Mit Transformation: $p_i^0 = x(\gamma q_i^0 + \vec{b}\vec{q}_i)$ und $\vec{p}_i = x(\vec{q}_i + \vec{b}\cdot\vec{q}_i + a(\vec{b}\cdot\vec{q}_i)\vec{b})$ werden die Impulse p_i erzeugt.

$$Q^\mu = \sum_{i=1}^{n} q_i^\mu, \quad M = \sqrt{Q^2}, \quad \vec{b} = -\frac{1}{M}\vec{Q}, \quad \gamma = \frac{Q^0}{M}, \quad a = \frac{1}{1+\gamma}, \quad x = \frac{\sqrt{\tilde{P}^2}}{M}$$

RAMBO mit Massen

- Bisher: Vernachlässigung der Massen $m \ll E$.
- Erweiterung von RAMBO auf Teilchen mit Masse ungleich Null
- Als weiterer Schritt im RAMBO Algorithmus
- Transformiere p_i zu:

$$\vec{k}_i = \xi \vec{p}_i$$

- Bestimme ξ aus:

$$\sqrt{\tilde{P}^2} = \sum_{i=1}^{n} \sqrt{m_i^2 + \xi^2 \cdot (p_i^0)^2}$$

- i.A. müssen Numerische Gleichungen gelöst werden
- Für größere Massen sinkt die Effizienz des Algorithmus deutlich

Quellen

- R Kleiss, W.J Stirling, S.D Ellis, A new Monte Carlo treatment of multiparticle phase space at high energies
- Stefan Weinzierl, Introduction to Monte Carlo methods

Ende

Jetzt zum Algorithmus!