Introduction to PLUTO

PLUTO: Package for generation of particle emission.

Scope: Source : free trajectory, 2-particle collision, particle decay, fireball from heavy-ion collision

Particles : hadrons, leptons, photon
stable and unstable (including: resonances with broad mass distribution)

Properties of particles and decay channels are taken from Review of Particle Physics.

Decays : into particles or through the virtual photon
According to the relativistic kinematics + possible simple physical models

Emission : one can simulate limited detector acceptance by filtering out momenta, angles

Possibility to encode own particle / decay / model.

Form: Library linked by ROOT session. Dedicated objects usable interactively or in macro, e.g.
`PParticle, PChannel, PReaction, ...`

Compatibility: PLUTO 5 ↔ ROOT 5, PLUTO 6 ↔ROOT 6

Algorithm: Defining the emission source + emitted particle(s) + activation of specific decays
Accounting for beam energy (Lorentz transform CM → Lab)
Generating events in loop. Each step: random pulling of angles + execution of decays

Result: Set of events with particles and their momenta.

Output format: By default, TTree object in root file. Optionally: text, TNtuple, THnF .


krzysztof.piasecki@fuw.edu.pl

Modes of work:

- Building “from bricks”, i.e objects are: sources, particles, decay channels and reaction
- Dedicated scripting language

www :

- https://www-hades.gsi.de/?q=pluto : Home page, GDPR problem → most of content is private
- https://plutouser.github.io/v6.00/ : Sources, class documentation, exemplary macros

Papers:

- I. Fröhlich et al., arXiv:0708.2382v2 “Pluto: A Monte Carlo Simulation Tool for Hadronic Physics”
- I. Fröhlich et al., arXiv:0905.2568v1 “Design of the Pluto Event Generator”

Talks:

- I. Fröhlich, “The Pluto++ Event Generator” (ACAT 07 Amsterdam)
- I. Fröhlich, “Pluto: An Event Generator Framework for Hadronic Physics” (A2 CB17 Meeting)

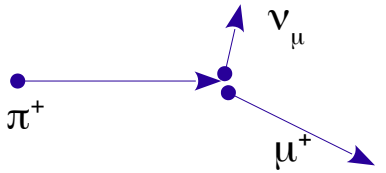
Installation

- Sources:
  ```
  wget https://plutouser.github.io/v6.00/pluto_v6.00.tar.gz
  ```
  or:
  ```
  git clone https://github.com/PlutoUser/pluto6.git
  ```
- Installation steps:
  ```
  cd pluto6 ; mkdir builddir ; cd builddir
  cmake ..
  make
  ```
- Result: Library `libPluto.so`
  I’d suggest: `export PLUTOLIBDIR={.....}/builddir`
- Activation of library:
  - in Root session: `gSystem->Load ("..../libPluto.so");`
  - in macro: `R__LOAD_LIBRARY (..../libPluto.so)`
- Neutronx computer: *please contact me*.

Example 1

Decay $\pi^+ \longrightarrow \mu^+ \nu_\mu$

(E_{kin} = 1 GeV)



Preprocessor macro that links this library

We create 1 x π^+ .
Pluto identifies hadron by name.
We give E_{kin} = 1 GeV.

We form the reaction:

1. Substrate: π^+
2. Products : μ^+, ν_μ

   Format: "Descriptor string"

3. Name for output file

We execute the loop of 10000 events.

```
R__LOAD_LIBRARY ($PLUTOLIBDIR/libPluto.so)

int pip_mupnu ()
{
  PParticle pip ("pi+", 1.0 );

  PReaction *my_reaction = new PReaction
      ( &pip ,
        "mu+ nu",
        "pip_mupnu", 1);

  my_reaction->Print ();
  my_reaction->Loop (10000, 1);

  return 0;
}
```

- **We launch the simulation**
- **and read the output file in the root format**

We need to link the library

```
$ nice root -b -q pip_mupnu.C

$ nice root -l pip_mupnu.root
{...}
Warning in <TClass::Init>: no dictionary {...}
Root [1] gSystem->Load ("..../libPluto.so")
(int) 0
```

```
$ data->Scan ("pid:GetParentId()")
$ data->Draw ("M()", "pid==8")
$ data->Draw ("E()-M()", "pid==8")
              "Rapidity()"

$ data->Draw ("Rapidity()", "pid==5")
              "Pt():Rapidity()", "pid==5")

$ data->Draw ("P():Theta()", "  Name() == \"mu+\"  ")

$ data->Draw ("Rapidity():Pt()", "pid==5")
               Beta()
               Pz()  Px()  Py()
```

{PID → por. s. 6}

1 entry = collection of particles from 1 reaction (TClonesArray)

Kinematics calculated in Lab

PParticle – Particle

Represents a given particle with given properties and 4-momentum.

```
$ PParticle p ("pi0")
$ p.Print ()
$ cout << p.Name() << "\t" << p.ID() << endl;

$ listParticle ("pi0")   or   ( 7 )

$ p.Set... (...)
    SetE, Px, Py, Pz,
    SetTheta (...), SetPhi(...)

$ p.E() , Px Py Pz Pt Mt Theta Phi

$ p.Boost (beta_x, beta_y, beta_z)
```

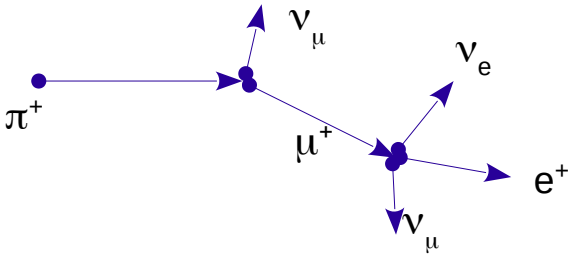
rotate the momentum vector

Lorentz Transform

Example 2

Decay $\pi^+ \rightarrow \mu^+ \nu_\mu$, $\mu^+ \rightarrow \nu_\mu e^+ \nu_e$

(E_{kin} = 1 GeV)



μ^+ is the unstable particle ($\tau \approx 10^{-6}$ s) and decays into $\nu_\mu e^+ \nu_e$.
Let's activate this decay.

String descriptor

```
"nu mu+ [e+ nu nu]"
```

(We open brackets after the mother particle and place the products therein)

Caution: *Pluto does not differentiate ν.*

Output file name

Flags

f0: Save (1–all particles / 0–only final ones)
f1: (unused)
f2: Generate production vertex (1/0 = yes/no)
f3: Generate text output file (1/0 = yes/no)

```
R__LOAD_LIBRARY ($PLUTOLIBDIR/libPluto.so)

int pip_mupnu_epnunu ()
{
  PParticle pip ("pi+", 1.0 );

  PReaction *my_reaction = new PReaction (
                    &pip ,
                    "nu mu+ [e+ nu nu]",
                    "pip_mupnu_epnunu" ,
                    1, 0, 0, 1
                  );
    my_reaction->Print ();

    my_reaction->Loop (10000, 1);
    return 0;
}
```

```
$ data->Scan ("Name() : pid : GetParentId() : GetSiblingIndex()")
$ data->Draw ("Theta()", "pid==8")
                          pid==5
                          pid==4 && GetParentId()==8
                          pid==4 && GetParentId()==5
```

Access to data on particles and decays

Particle types included into Pluto:

```
$ listParticle ()
```

← Prints out particles contained in data base

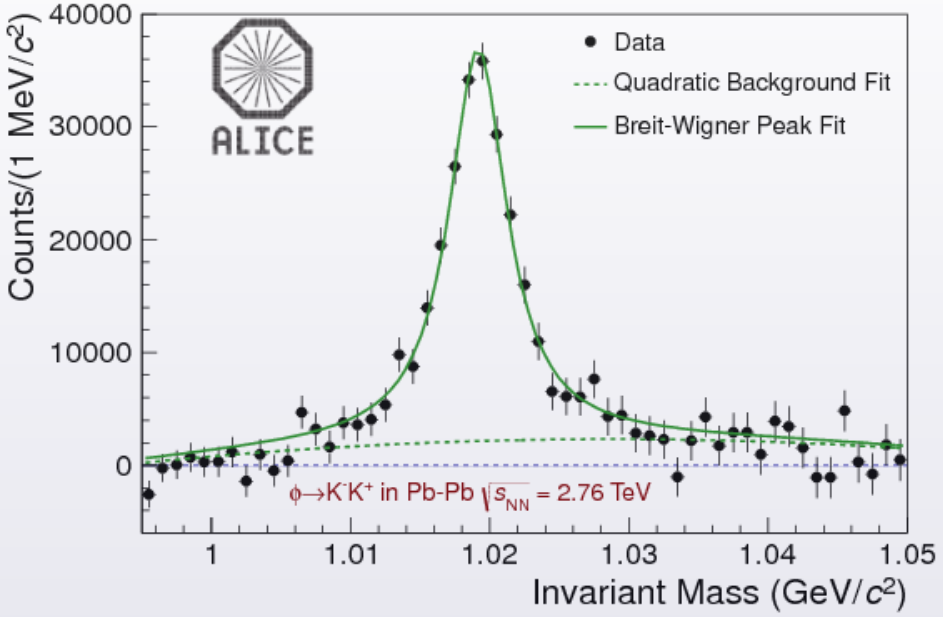
- Particles which are stable or with width Γ < 1 MeV

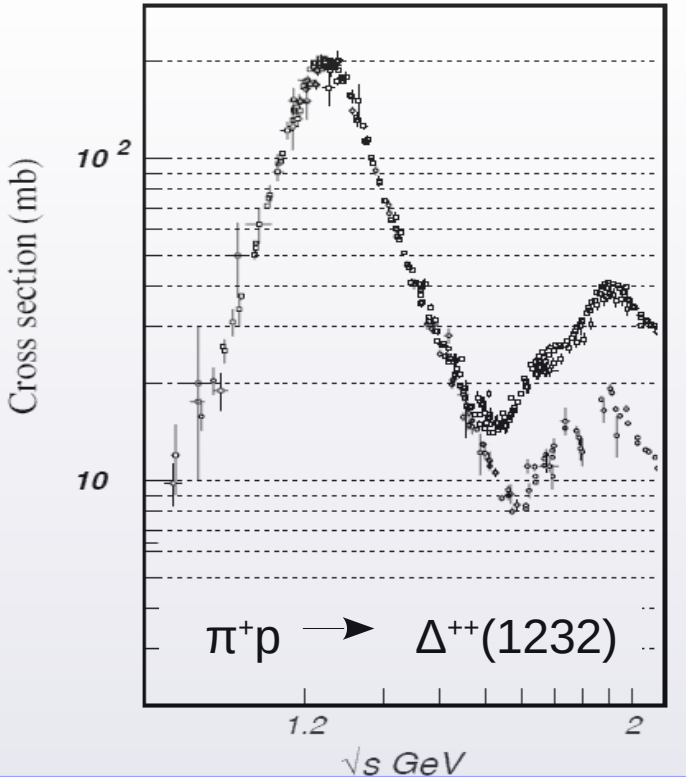
PID	Symbol	Type	M[MeV]
1	g	γ	0
2	e+	e^+	0.511
3	e-	e^-	0.511
4	nu	ν	0
5	mu+	μ^+	105.7
6	mu-	μ^-	105.7
7	pi0	π^0	135.0
8	pi+	π^+	139.6
9	pi-	π^-	139.6
10	K0L	K^0_L	497.7
11	K+	K^+	493.7
12	K-	K^-	493.7
13	n	n	939.6
14	p	$\underline{p}$	938.3
15	anti_p	$\overline{p}$	938.3
16	K0S	K^0_S	497.7
17	eta	η	547.5
18	Lambda	Λ	1115.7
19	Sigma+	Σ^+	1189.4
20	Sigma0	Σ^0	1192.6
21	Sigma-	Σ^-	1197.4
22	Xi0	Ξ^0	1314.9
23	Xi-	Ξ^-	1231.3
24	Omega	Ω	1672.5
25	anti_n	$\overline{n}$	939.6
26	anti_Lambda	$\overline{\Lambda}$	1115.7
27	anti_Sigma-	$\overline{\Sigma}^-$	1189.4
28	anti_Sigma0	$\overline{\Sigma}^0$	1192.6
29	anti_Sigma+	$\overline{\Sigma}^+$	1197.4
30	anti_Xi0	$\overline{\Xi}^0$	1314.9
31	anti_Xi+	$\overline{\Xi}^+$	1321.3
32	anti_Omega+	$\overline{\Omega}$	1672.5
33	K0S	η	497.7
53	eta'	η'	957.7
67	J/Psi	J/Ψ	3096.9
68	Psi'	Ψ'	3686.0

- Nucleons and lightest nuclei:

PID	Symbol	Type	M[MeV]
13	n	n	939.6
14	p	$\underline{p}$	938.3
15	anti_p	$\overline{p}$	938.3
25	anti_n	$\overline{n}$	939.6
45	d	d	1875.6
46	t	t	2809.3
47	alpha	^{4}He	3727.4
49	He3	^{3}He	2809.2

Resonances (particles having the mass distribution)
Excited states of group of 2 or 3 valence quarks.





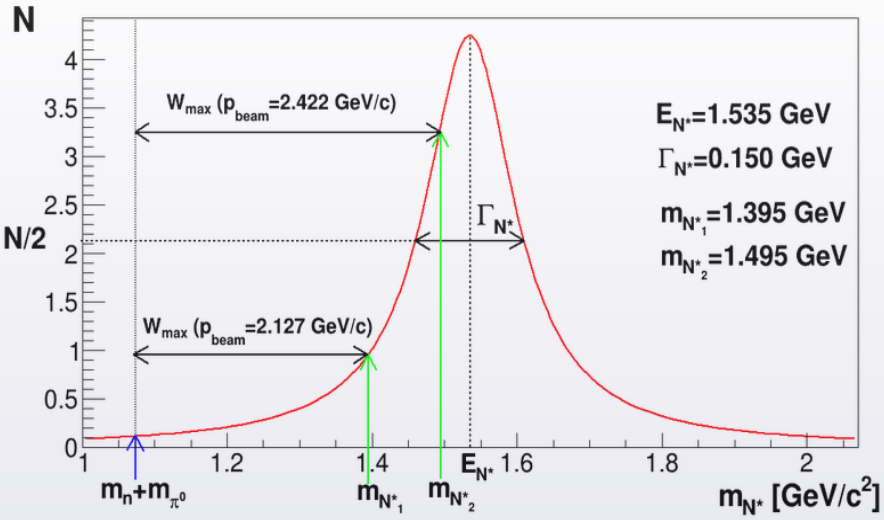
Model: relativistic Breit-Wigner profile

$$g(m) = A \frac{m^2 \Gamma^{\mathrm{tot}}(m)}{(M_{\mathrm{R}}^2 - m^2)^2 + m^2 (\Gamma^{\mathrm{tot}}(m))^2}$$

$$\Gamma^{\mathrm{tot}}(m) = \sum_{\mathrm{k}}^{N} \Gamma^{\mathrm{k}}(m)$$

Γ^{tot} : total half-width,
sum over contributions to decay channels, Γ^k

Probing interval: $m \in [\, M_R - 2\Gamma\ ,\ M_R + 12\ \Gamma\,]$



- Resonances in data base of Pluto:

Mesons

PID	Symbol	Type	M[MeV]	Γ [MeV]
41	rho0	ρ^0	769.9	150.7
42	rho+	ρ^+	769.9	150.7
43	rho-	ρ^-	769.9	150.7
52	w	ω	781.9	8.43
54	sigma	σ	600.0	500.
55	phi	ϕ	1019.4	4.43

Baryons

PID	Symbol	Type	J[ħ]	M[MeV]	Γ[MeV]
34	D0	$\Delta(1232)^0$	3/2	1232	120
35	D++	$\Delta(1232)^{++}$	3/2	1232	120
36	D+	$\Delta(1232)^+$	3/2	1232	120
37	D-	$\Delta(1232)^-$	3/2	1232	120
38	NP11+	$N(1440)^+$	1/2	1440	350
39	ND13+	$N(1520)^-$	3/2	1520	120
40	NS11+	$N(1535)^-$	1/2	1535	150
56	DP330	$\Delta(1600)^0$	3/2	1600	350
57	DP33++	$\Delta(1600)^{++}$	3/2	1600	350
58	DP33+	$\Delta(1600)^+$	3/2	1600	350
59	DP33-	$\Delta(1600)^-$	3/2	1600	350
60	DS310	$\Delta(1620)^0$	1/2	1620	150
61	DS31++	$\Delta(1620)^{++}$	1/2	1620	150
62	DS31+	$\Delta(1620)^+$	1/2	1620	150
63	DS31-	$\Delta(1620)^-$	1/2	1620	150
64	NP110	$N(1440)^0$	1/2	1440	350
65	ND130	$N(1520)^0$	3/2	1520	120
66	NS110	$N(1535)^0$	1/2	1535	150

- Nature has many more resonances, in particular those with higher masses.
- One can add new particles to data base; see page 23.

Access to data on particles and decays, cont.

```
$ listParticle (34)
```

← Data on $\Delta(1232)^0$:

- centroid and Γ of mass distribution
- Decay channels + contributions (BR ≡ branching ratio)

Direct access:

```
$ PStaticData* sd = makeStaticData() ;

$ sd->GetParticleMass (7)
      GetParticleID ("p")
      GetParticleName (7)
      GetParticleTotalWidth (34)
      GetParticleSpin (34)
      GetParticleParity (34)
      GetParticleIsospin (34)

$ sd->GetParticleNChannels (7)  ---> 2

$ sd->PrintParticle (7)  lub ("pi0")

$ sd->PrintDecayByKey (143)

Database key=143
Database name=pi0 --> photon + photon
Decay index=3
Branching ratio=0.988000
Decay product 1->Database name=g
Decay product 2->Database name=g

$ sd->GetDecayBR (3)   ---> 0.988
```

← "Manager" object containing the data base

```
(...)
Database key=143
Database name=pi0 --> photon + photon
Decay index=3
Branching ratio=0.988000
Decay product 1->Database name=g
Decay product 2->Database name=g

Database key=144
Database name=pi0 --> dilepton + photon
                                    (Dalitz)
Decay index=4
Branching ratio=0.012000
Decay product 1->Database name=g
Decay product 2->Database name=dilepton
```

Access to data on particles and decays, cont.

Access to models of decays, distributions:

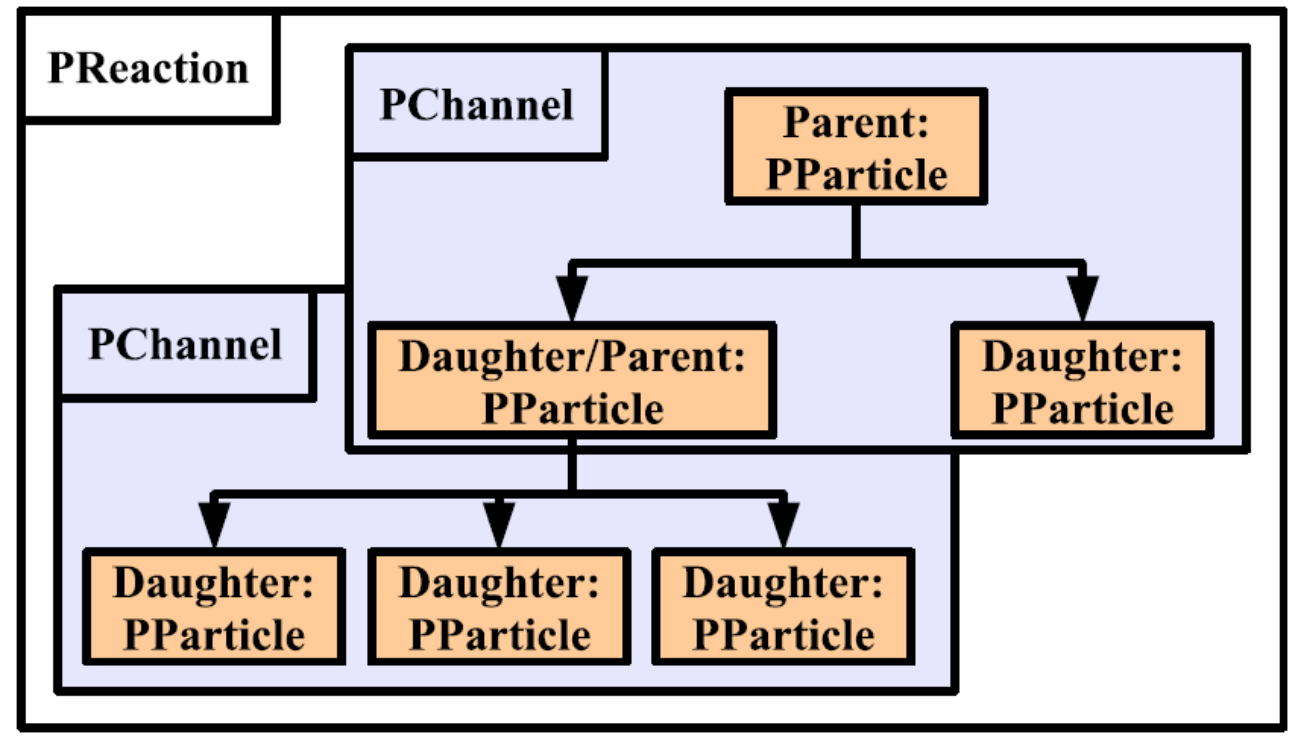
```
PDistributionManager* pdist = makeDistributionManager ()
pdist->Print ()
pdist->Print ("decay_models")
pdist->Print ("pi0_fixed_g_g")
```

nb. GenBod: cernlib.web.cern.ch/cernlib/mc/genbod.html

Example 3 Decay $\pi^+ \longrightarrow \mu^+ \nu_\mu$. This time, let's build the reaction from bricks.

(E_{kin} = 1 GeV)

Structural scheme:



- **Algorithm in our case:**

1. Create 3 particles: π^+, μ^+ and ν , represented by object of `PParticle` class.
2. Group the particles: create the array, in which we place the addresses of particles (substrate first)
3. Create 1 decay channel $\pi^+ \longrightarrow \mu^+ \nu$, represented by object of `PChannel` class.
4. Combine all the channels in the array, in which we place the adresses of `PChannel` objects
   (For us: 1 channel only ⟶ only 1 element in this array)
5. Define the reaction: create 1 object of `PReaction` class, giving the channel array and output file.
6. Execute the loop.

Let's encode the algorithm in the macro:

Substrate

Products

Array of particles

Decay channel

Array of all channels – participating In reactions

Define the reaction

Execute loop of 10000 events

```
R__LOAD_LIBRARY ($PLUTOLIBDIR/libPluto.so)

int pip_mupnu_long ()
{
    PParticle pip ("pi+", 1.0 );
    PParticle mup ("mu+") , nu ("nu");

    PParticle* parttable[] = { &pip , &mup , &nu };

    PChannel* pip_decay = new PChannel ( parttable , 2 );

                                // "2" = no. of decay particles

    PChannel* tab_of_chan[] = { pip_decay };

    PReaction* my_reaction = new PReaction
                ( tab_of_chan , "pip_mupnu_long",

                  1 ,           /* No of channels in table */
                  1 , 0 , 0 , 0 );

    my_reaction->Print ();
    my_reaction->Loop (10000, 1);

    return 0;
}
```

Example 4 Emission of particles from the 2-particle collision.

Let's consider a reaction: p (E_{kin} = 0.2797 GeV) + p ⟶ p + π^0

- It's qualitatively new (two substrates). We'll use the dedicated constructor of `PReaction` class.

- From where is this "strange" value of 0.2797 GeV? It's a minimal E_{kin} that the beam nucleon should have to produce π^0 in NN collision (NN survives). *"Threshold energy"*
  For substrates of masses m_1, m_2 and a product of mass m_X,

$$E_K^{Min} = \frac{m_X^2 + 2m_X(m_1 + m_2)}{2m_2}$$

  Verify that if $E_k(p^+)$ is lowered by 0.1 MeV, Pluto refuses to create π^0.

- However, the process does not end here: π^0 is unstable and after $\tau \approx 10^{-16}$ s it decays.
  Dominant decay channel: $\pi^0 \longrightarrow \gamma\,\gamma$ (BR ≈ 99%)
  Let's activate it in our simulation.

Within "" : E_{kin} of beam particle
Without "" : *p* of beam particle

Substrates (beam and target)

Reaction scheme (string descriptor)

Output file name

We run the loop

```
R__LOAD_LIBRARY ($PLUTOLIBDIR/libPluto.so)

int pp_pppi0_gg ()
{
    Preaction* my_reaction = new PReaction
          ("0.2797",
           "p", "p",
           "p p pi0 [g g]",
           "pp_pppi0_gg",
            1, 0, 0, 0 );

    my_reaction->Print ();
    my_reaction->Loop (10000, 1);
    return 0;
}
```

```
$ data->Scan ("Name():pid:GetSiblingIndex()")

*         0 *         0 *             *      14014 *        -1 *
*         0 *         1 *             p *       14 *         2 *
*         0 *         2 *             p *       14 *         3 *
*         0 *         3 *           pi0 *        7 *         1 *
*         0 *         4 *             g *        1 *         5 *
*         0 *         5 *             g *        1 *         4 *
```

Unknown PID...?

- **Composite particle**

Substrates and products are bound by *Available energy* ($\sqrt{s} = \sqrt{(\sum E_i)^2 - (\sum \vec{p}_i)^2}$) .
It is Lorentz-invariant.
In particular, in the CM frame, from its definition: $\sum \vec{p}_i = \vec{0} \quad \Rightarrow \quad \sqrt{s} = \sum E_i$
Therefore, it is the amount *of energy* available in CM.
From this amount particles-products are generated. They can also move.

Composite Particle = "intermediate particle", for which: $\vec{p} = \sum \vec{p}_i \quad E = \sum E_i \quad M = \sqrt{(\sum E_i)^2 - (\sum \vec{p}_i)^2}$
From it, Pluto creates products, if the conservation rules allow for a decay.

$$PID_{CP} = 1000 \times PID_{2\ (Target)} + PID_{1\ (Beam)} .$$

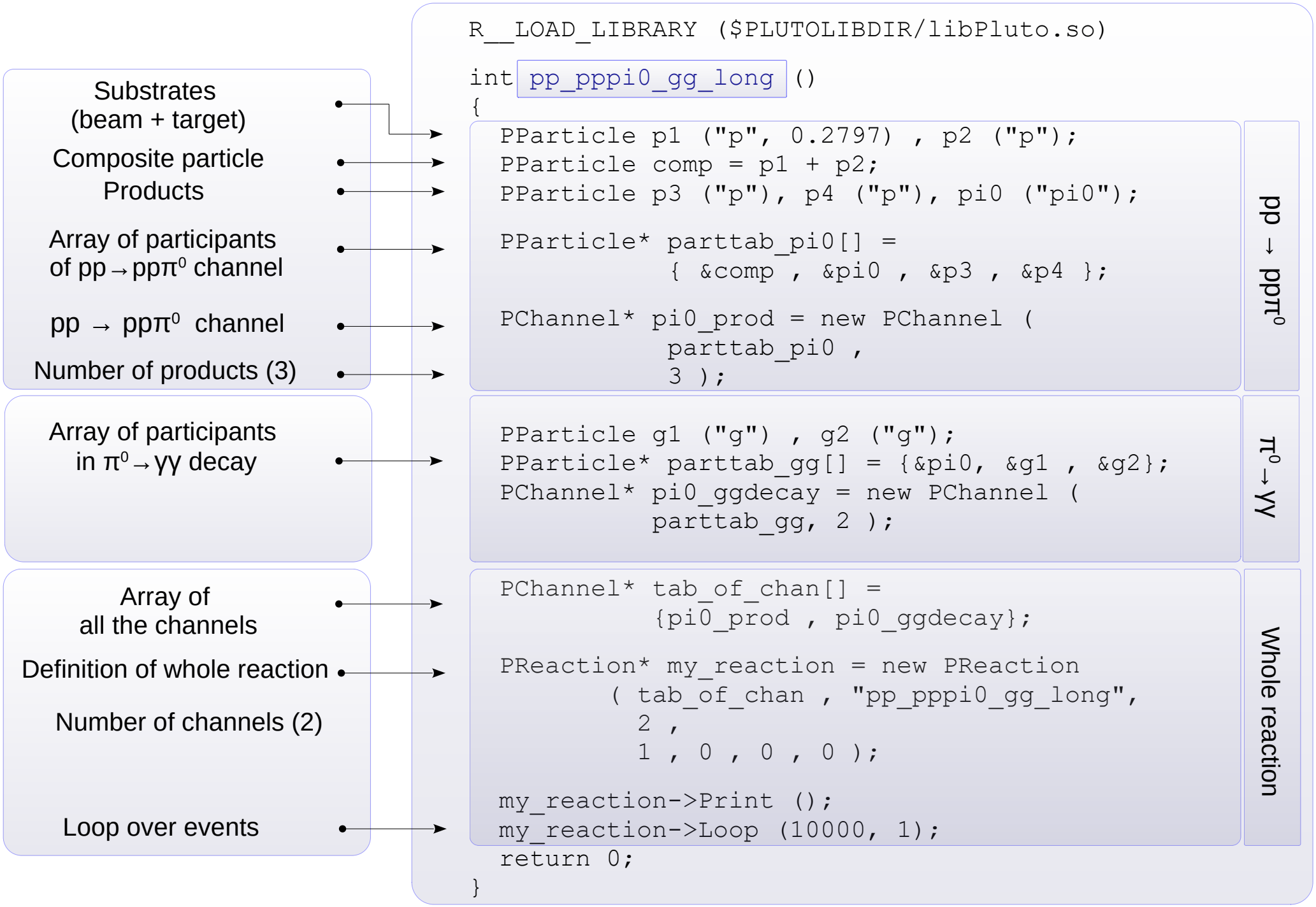
```
$ data->Scan ("E() : P() : M()", " pid==14014 ")
*         0 *         0 * 2.1562446 * 0.7765961 * 2.0115390 *
*         1 *         0 * 2.1562446 * 0.7765961 * 2.0115390 *
```

As we see,
$E^2 - P^2 = M^2$

```
$ data->Draw ("Beta()", " pid==14014 {lub 7} ")
$ data->Draw ("Rapidity()", " pid==7  {lub 1} ")
$ data->Draw ("Pt() : Rapidity()", " pid==7  {lub 1} ")
```

Distribution of β, *y* *for* pions concentrates around β, *y* of composite particle.

Example 5 Emission of particle from collision of 2 particles – constructed "from bricks"



Analysis of output TTree within a macro (Example 6)

- Let's analyse the output tree from the reaction $\pi^+ \longrightarrow \mu^+ \ \nu_\mu$.

```
R__LOAD_LIBRARY ($PLUTOLIBDIR/libPluto.so)

int pip_mupnu_readtree ()
{
  TClonesArray* partArray = new TClonesArray ("PParticle", 10);
  PParticle* Part[10];
  TVector3 b;

  Tfile* fileIn = new TFile ("pip_mupnu.root");
  Ttree* tree = (TTree*) fileIn->Get ("data");
  tree->SetBranchAddress ("Particles", &partArray );

  Int_t nR = tree->GetEntries() ;
  cout << "\n * Analysing " << nR << " reactions.\n\n";

  for (Int_t iR = 0 ; iR < nR ; iR++ )
  {
    tree->GetEntry (iR);
    for (Int_t iP = 0; iP < partArray->GetEntries() ; iP++ )
      Part[iP] = (PParticle*) partArray->At (iP);

    Part[1]-> Boost ( -Part[0]->BoostVector() );
    Part[2]-> Boost ( -Part[0]->BoostVector() );

    for (Int_t iP = 1; iP <= 2; iP++)
      cout << Part[iP]->Name() <<'\t'<< Part[iP]->Px() <<'\t'
           << Part[iP]->Py()   <<'\t'<< Part[iP]->Pz() << endl;

    cout << endl;
  }
  return 0;
}
```

Loop over reactions

Reading i-th reaction

Loop over particles

We transform the 4-Momenta of products into the parent's frame

Data printout

Emission of particles from the collision zone of heavy ions

Topic : production of new particles
in collisions of heavy ions
at E_{kin} from 100 MeV/nucleon to LHC.

- It appears that the hadron multiplicities are approximately governed by one quantity : √s .

- It is in-line with the statistical approach to the collision zone (“Fireball”),
built of particles (fermions and bosons)
Having given energies and momenta.
A global thermodynamic equilibrium is assumed.

- A quantum statistics distribution (F-D or B-E) is usually approximated by the *Boltzmann distribution*.

$$\frac{dN}{dE} \propto p\, E\, e^{-E/T}$$

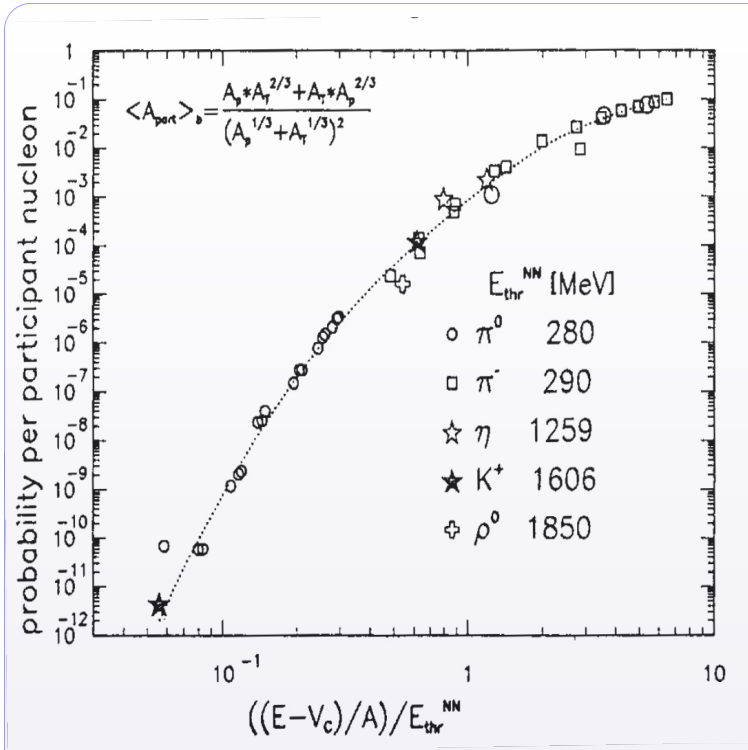
Boltzmann distribution:

(1) is isotropic (equipartition of energy)
(2) $\langle E \rangle = {}^3/_2 kT$ occurs (+ is not a function on particle mass).

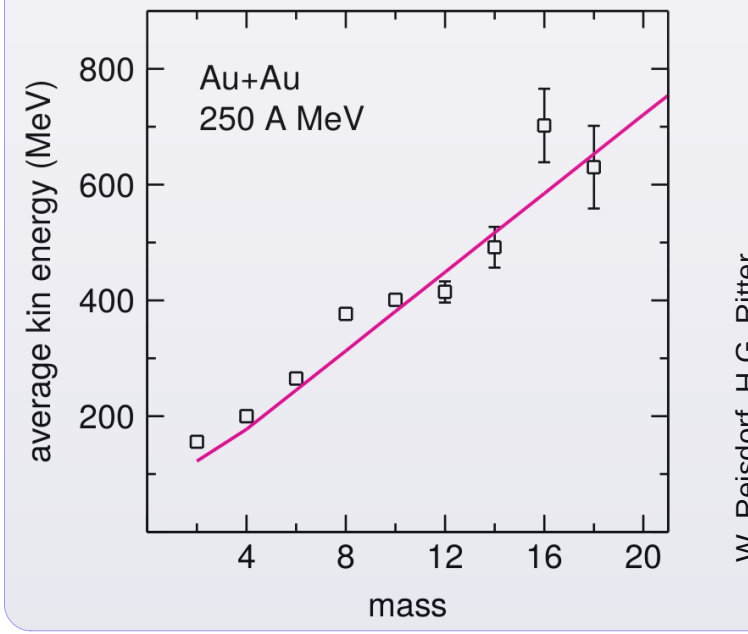
- Quite often these features are not observed experimentally.

Momentum distributions of some particles can be described only as superposition of 2 distributions (2 thermal sources)

Dependence <*E*> = f(*m*) → prompts to postulate:
except the thermal motion there also exists
a common radial expansion (collective motion).

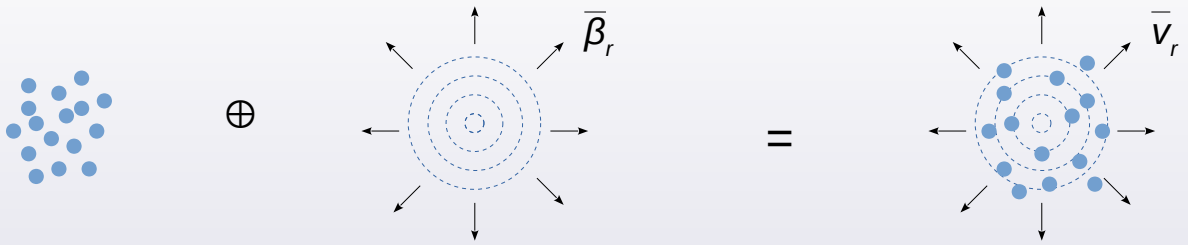


V.Metag, Prog. Part. Nucl. Phys. 30, 75 (1993)



W. Reisdorf, H.G. Ritter,
Annu. Rev. Nucl. Part. Sci. 47, 663 (1997)

Siemens-Rasmussen model: overlay of thermal motion (temp. *T*) on the radial expansion (velocity β).



Thermal motion (Boltzmann) ⊕ Collective motion (radial expansion) = Combined together

$$\left.\frac{d^3n}{dp^3}\right|_{CM} = \frac{N_i}{Z(T)} \cdot e^{-\frac{\gamma_r E}{T}} \cdot \left[\left(\gamma_r + \frac{T}{E}\right)\frac{sh\,\alpha}{\alpha} - \frac{T}{E}ch\,\alpha\right]$$

gdzie: $\alpha = \gamma_r \beta_r p/T$

N_i, Z : normalization const.

P.J. Siemens, J.O. Rasmussen, Phys. Rev. Lett. 42, 880 (1979)

Angular distributions (in ϑ and φ) are often not isotropic.

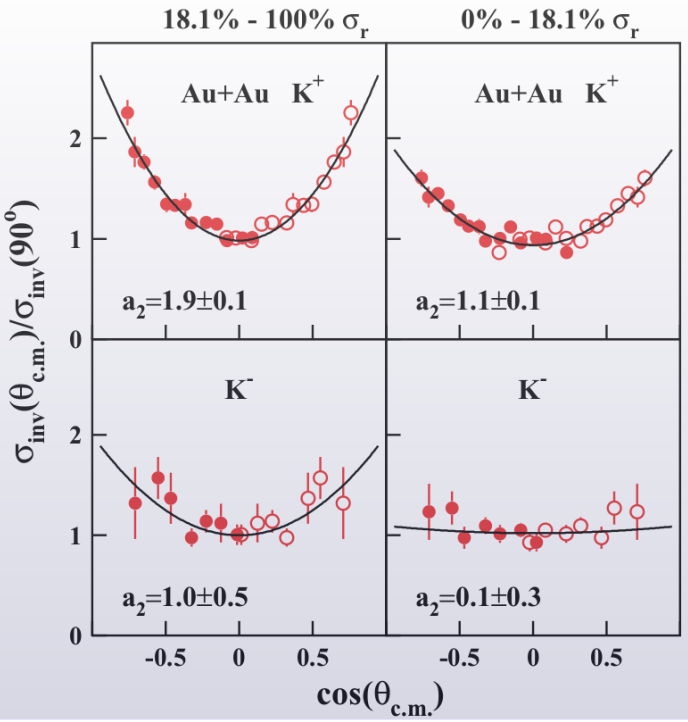
Distribution in ϑ is phenomenologically parameterized by the Legendre polynomial, with weights of subsequent terms.

$$\frac{dN}{d\cos\theta_{NN}} \sim 1 + \sum_n a_n P_n(\cos\theta_{NN})$$

Distribution in φ is phenomenologically parameterized by the Fourier series with weights of subsequent harmonics:

$$\frac{dN}{d\varphi} \sim \frac{1}{2\pi}\left(1 + 2\sum_{n\geq 1}^{\infty} v_n \cos n\varphi\right)$$

In Pluto we can set up weights for a few first terms.



A. Förster et al., Phys. Rev. C 75, 024906 (2007)

Emission of particles from the collision zone of heavy ions

Template for energetic part of distribution: sum of 2 Siemens-Rasmussen functions.

$$\frac{dN}{dE} \propto p\,E \left\{ f e^{-\gamma_r \frac{E}{T_1}} \left[\left(\gamma_r + \frac{T_1}{E}\right) \frac{\sinh\alpha_1}{\alpha_1} - \frac{T_1}{E}\cosh\alpha_1 \right] + (1-f) e^{-\gamma_r \frac{E}{T_2}} \left[\left(\gamma_r + \frac{T_2}{E}\right) \frac{\sinh\alpha_2}{\alpha_2} - \frac{T_2}{E}\cosh\alpha_2 \right] \right\}$$

Caution: (1) for $\beta \to 0$ Siemens-Rasmussen function → Boltzmann function.
(2) for $f = 1$ only the 1st source exists

Templates for θ and φ distrib.: $\frac{dN}{d\cos\theta_{CM}} \sim 1 + a_2\cos^2\theta_{CM} + a_4\cos^4\theta_{CM}$ $\frac{dN}{d\varphi_{RP}} \sim 1 + v_1\cos\varphi_{RP} + v_2\cos^2\varphi_{RP}$

`PFireball` – emission source of "our" particles.

Coding-wise, the `PFireball` object mimics a particle, that decays into "our" `PParticle`.
⇒ one should implement the `PChannel` that links the `PFireball` object and the emitted `PParticle`.

```
PFireball* source_pi0 = new PFireball (
  "pi0",   /* type of particle emitted from source                          */
  0.040,   /* Beam Ekin / nucleon  [GeV/A]                                  */
  0.015,   /* Temperature of thermal source of particles [GeV]              */
  0.,      /* Temperature of 2. thermal source if it exists [GeV]           */
  1.,      /* Weight of 1. source                                           */
  0.,      /* Beta param (v/c) of radial expansion of source if applicable  */
  0.,      /* a2 coefficient of theta angle distribution                    */
  0.,      /* a4 coefficient of theta angle distribution                    */
  0.,      /* v1 coefficient of  phi  angle distribution                    */
  0.       /* v2 coefficient of  phi  angle distribution                    */
);
```

Example 7

π^0 meson emission from the collision zone having temperature of 15 MeV
The beam ion moves in Lab with the E_{kin} of 40 MeV per nucleon.

```
R__LOAD_LIBRARY ($PLUTOLIBDIR/libPluto.so)

int thermal_pi0_gg ()
{
  PFireball* source_pi0 = new PFireball (
      "pi0", 0.040,  0.015,
      0., 1., 0.,
      0., 0.,
      0., 0.
  );
  source_pi0 ->Print ();

  PParticle* pi0 = new PParticle ("pi0");
  PParticle* parttable_fireball_pi0[] = {source_pi0, pi0};
  PChannel*  chan_fireball_pi0 = new PChannel (parttable_fireball_pi0, 1, 1);

  PParticle* g1  = new PParticle ("g");
  PParticle* g2  = new PParticle ("g");
  PParticle* parttable_pi0_gg [] = {pi0, g1, g2};
  PChannel*  chan_pi0_gg = new PChannel (parttable_pi0_gg, 2, 1);

  PChannel*  tab_of_chan[] = { chan_fireball_pi0 , chan_pi0_gg };

  PReaction* reaction_thermal_pi0 = new PReaction (
    tab_of_chan, "thermal_pi0_gg_Eb40MeV_T15MeV",
     2,
     1, 0, 0, 0
    );
   reaction_thermal_pi0 ->Print ();
   reaction_thermal_pi0 ->loop (10000, 1);
   return 0;
}
```

Definition of Fireball

Particle type, beam E_{kin} [GeV], T of source [GeV]
T of 2. source [GeV], weight of 1. source, radial β
a_2 and a_4 coefficients of θ angle distribution
v_1 and v_2 coefficients of φ angle distribution

Linking of Fireball and π^0 into channel

Linking of π^0 and γγ into channel

Table of channels

Definition of reaction

Number of channels in table

Data preview via TNtuple database and THnF histograms

Pluto allows to define the physics variables related to involved particles,
and to **preview them** in the **TNtuple data base**. Notice: 1 reaction = 1 database entry.

① We create the TNtuple base and a dedicated (separate) output file:

```
TFile*   file_TNtuple = new TFile   ("ntufile.root", "RECREATE");
TNtuple* my_ntuple    = new TNtuple ("ntu", "Name", "var1:var2:var3:..." );
```

② We define the physical variables :

```
my_reaction->Do ("thgam = [g]->Theta()* TMath::RadToDeg()" );
my_reaction->Do ("phgam = [g]->Phi()  * TMath::RadToDeg()" );
```

③ We call the reaction to generate the output TNtuple database. We can apply **filters** here :

```
my_reaction->Output (my_ntuple , "if (thgam>20 && thgam < 50) " );
```

④ We can also apply **filters** to the main tree (`TTree data`). Each variable whose name starts from # is a filter. Entry to `TTree` is made only if none of filters is found to be 0.

```
my_reaction->Do ("#myaccept= 1; if (thgam<20 || thgam > 50) ; #myaccept= 0");
```

Preview via control 1/2/3-dimensional histograms (THnF)

① We create a histogram:

```
TH1F *my_histo = new TH1F ("myhisto", "Photon azim. angle", 90, -180., 180.);
```

② In reaction we define special variables **_x (_y, _z)**. We can apply filter here (other than previously) :

```
my_reaction->Do (my_histo, "if thgam>20 && thgam < 50; _x=phi");
```

Notice: Writing of `TNtuple` and `THnF` to file should be usually ordered explicitly :

```
gDirectory->cd ("ntufile.root:/");
my_ntuple ->Write ();                    my_histo ->Write();
```

Example 8 Compton scattering – preview of variables in TNtuple and THnF

```
R__LOAD_LIBRARY ($PLUTOLIBDIR/libPluto.so)

int compton ()
{
  PReaction *R = new PReaction ("0.010",
            "g", "e-", "g e-", "compton_ttree");
  R->Print ();

  TFile *file_TNtuple = new TFile ("compton_tntuple.root", "RECREATE");

  TNtuple *my_ntuple = new TNtuple ("ntu", "Monitor ntuple",
                       "thgam:phgam:thel:phel:opang" );

  R->Do ("RTD = TMath::RadToDeg()");
  R->Do ("thgam = [g]->Theta()    * RTD");
  R->Do ("phgam = [g]->Phi()      * RTD");
  R->Do ("thel  = [e-]->Theta()   * RTD");
  R->Do ("phel  = [e-]->Phi()     * RTD");
  R->Do ("opang = [g]->Angle([e-])* RTD");
  R->Output (my_ntuple, "if (thgam>20 && thgam < 50)" );

  R->Do ("#myAccept=1; if (thgam<20 || thgam > 50); #myAccept=0");

  TH1F *my_histo = new TH1F ("myhisto", "Opening angle", 90, 0., 90.);
  R->Do (my_histo , "if thgam > 20 && thgam < 50 ; _x = opang");

  R->Loop (10000, 1);
  gDirectory->cd ("ntufile.root:/");
  my_ntuple->Write ();
  my_histo ->Write ();

  return 0;
}
```

Definitions of variables

Filter in TNtuple

TTree filter

TNtuple

TH1F

Zapis

Scripting language

Dedicated manual:

https://www.fuw.edu.pl/~kpias/nkfj/pluto/ifroehlich_pluto_script_manual.pdf

Adding new particles to Pluto's data base

To add a new particle and possibly provide the required decays,

① We start from creating the "manager" object:

```
$ PStaticData *sd = makeStaticData() ;
```

② We give PID, name and particle's mass. PID = -1 will assign the first free number. We possibly give Γ width.

```
$ sd->AddParticle ( -1, "New", mass [GeV]);
$ sd->SetParticleTotalWidth ( "New", width [GeV]);
```

③ We assign to this particle all the necessary properties
Caution: missing important data may result in lack of the relevant decay model, and wrong distributions.

```
$ sd->SetParticleBaryon ("New", 1);
                Meson ( )     Charge ( )    Spin ( )     Isopin ( )   ....
```

④ For unstable particles: we define the decay channel(s). PID = -1 will assign the first free No. from the list.

```
$ sd->AddDecay (-1, "New -> b + c", "New", "b,c", 1.);
```

Example 9 Creation of an e^+e^- pair in the vicinity of a nucleus

New particle

```
R__LOAD_LIBRARY ($PLUTOLIBDIR/libPluto.so)

int gamma_pairproduction ()
{
  double mN = 0.938, A = 40, mass_nucl = mN * A;
  PStaticData* sd = makeStaticData() ;
  sd->AddParticle (-1, "A", mass_nucl );

  PReaction* R = new PReaction ( "1.", "g", "A", "A e+ e-", "gamma_ee_ttree");
  R->Print ();

  TFile* file_TNtuple = new TFile ("gamma_pairprod_tntuple.root", "RECREATE");
  TNtuple* my_ntuple = new TNtuple
         ("ntu",  "My Ntuple", "pxep:pyep:pzep:pxem:pyem:pzem:pxA:pyA:pzA" );

  R->Do ("RTD = TMath::RadToDeg()");
  R->Do ("pxep = [e+]->Px() * RTD");
  R->Do ("pyep = [e+]->Py() * RTD");
  R->Do ("pzep = [e+]->Pz() * RTD");
  R->Do ("pxem = [e-]->Px() * RTD");
  R->Do ("pyem = [e-]->Py() * RTD");
  R->Do ("pzem = [e-]->Pz() * RTD");
  R->Do ("pxA  = [A]->Px()  * RTD");
  R->Do ("pyA  = [A]->Py()  * RTD");
  R->Do ("pzA  = [A]->Pz()  * RTD");

  R->Output (my_ntuple);
  R->Loop (10000, 1);

  file_TNtuple->Write ();
  file_TNtuple->Close ();
  return 0;
}
```