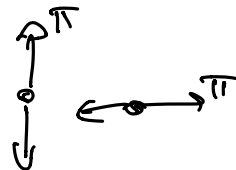
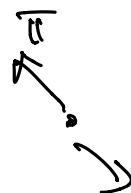
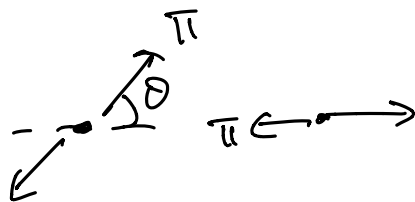


LAB

B rest frame:

$B \rightarrow \pi K$.

\rightarrow
B flight



random gen. of θ and φ uniform $\theta \in [0, \pi]$

$\varphi \in [0, 2\pi]$



pic punti
sui poli
e near all'equatore.

$$d\Omega = r^2 \sin \theta d\theta d\varphi.$$

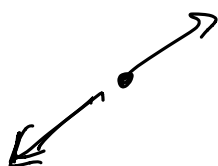
generate flat $\cos \theta$ and φ

double $P_x, P_y, P_z;$

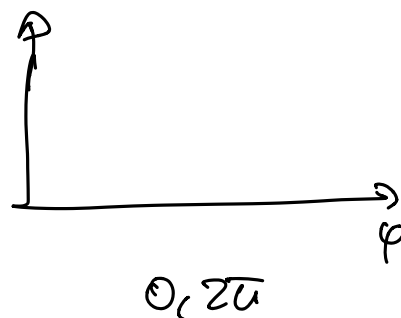
double $P_{ster} = 0.100;$

`TRandom3 * gen = new TRandom3();`

`gen->Sphere(P_x, P_y, P_z, P_{ster});`



Call `Sphere()` N times. and 1D histograms.



$$P_x = P \cdot \sin \theta \cos \phi$$

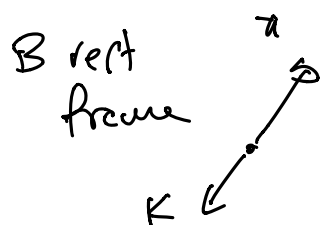
$$P_y = P \cdot \sin \theta \sin \phi$$

$$P_z = P \cdot \cos \theta$$

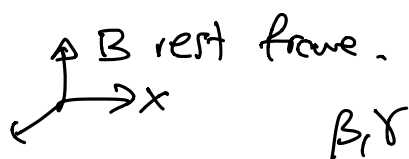
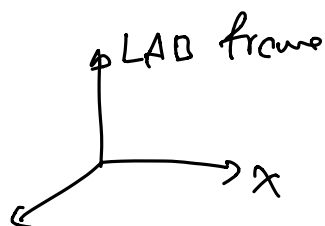
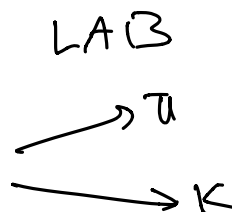
$$\rightarrow \cos \theta = P/P_z$$


$$\phi = \arctan 2(P_y, P_x)$$

$$M_{inv}^2 = (\underbrace{E_T + E_C}_{m_T^2 + p^2})^2 - (\underbrace{\vec{P}_T + \vec{P}_C}_{\vec{P}})^2 = M_B^2$$



boost
→








Join GitHub today

GitHub is home to over 50 million developers working together to host and review code, manage projects, and build software together.

Sign up





 rahatlou example of relativistic boost with root

Latest commit b015479 11 minutes ago 🕒 History

👤 1 contributor

🔗 Example of TLorentzVector in ROOT

This is an example of using the ROOT class `TLorentzVector` to perform kinematic calculations and apply the relativistic boost.

See a previous [lab session](#) for the details.

The complete example is in [boost.cc](#).

First we create the 4-momentum for a B meson in the laboratory frame.

```
TLorentzVector p4_B;
double m_B = 5.279; // GeV
double p_B = 0.3; // GeV
p4_B.SetPxPyPzE(p_B, 0, 0, sqrt(p_B*p_B+m_B*m_B));
```

Then we create the 4-momentum for the the pion the B rest frame

```
double m_pi = 0.140; //GeV
double p_pi = 0.100; // GeV
TLorentzVector p4_pi;
p4_pi.SetPxPyPzE(0, p_pi,0, sqrt(m_pi*m_pi+p_pi*p_pi));
```

B rest frame.

\vec{p}_{π}

— I —> B flight

We can print some of the information

```
cout << "--> LAB p4 B: " << endl;
p4_B.Print();

cout << "--> CoM p4 pi*: " << endl;
p4_pi.Print();
```

B momemtum:

```
--> LAB p4 B:
(x,y,z,t)=(0.300000,0.000000,0.000000,5.287517) (P,eta,phi,E)=(0.300000,-0.000000,0.000000,5.287517)
```

pion momentum in B rest frame:

```
--> CoM p4 pi*:
(x,y,z,t)=(0.000000,0.100000,0.000000,0.172047) (P,eta,phi,E)=(0.100000,-0.000000,1.570796,0.172047)
```

The class also provide the `Beta()` and `Gamma()` methods to compute the boost parameters, which you can compare with their definition:

```
cout << "--> boost parameters of B reference frame" << endl;
cout << "\t beta: " << p4_B.Beta() << "\t"
<< "\t p/E: " << p4_B.P()/p4_B.E() << "\n"
<< "\t gamma: " << p4_B.Gamma() << "\t"
<< "\t E/m: " << p4_B.E()/m_B << "\n"
<< "\t beta*gamma: " << p4_B.Beta()*p4_B.Gamma() << "\t"
<< "\t p/m: " << p_B/m_B
<< endl;
```

boost parameters of the B frame in Laboratory

```
--> boost parameters of B reference frame
      beta: 0.0567374          p/E:0.0567374
      gamma: 1.00161          E/m: 1.00161
      beta*gamma: 0.0568289    p/m: 0.0568289
```

It also provides the `BoostVector()` method

```
cout << "--> boost vector of the B meson" << endl;
p4_B.BoostVector().Print();
```

```
--> boost vector of the B meson
TVector3 A 3D physics vector (x,y,z)=(0.056737,0.000000,0.000000) (rho,theta,phi)=(0.056737,90.000000,0.000000)
```

that we can use to boost the pion to the LAB frame

```
p4_pi.Boost( p4_B.BoostVector() );
cout << "--> LAB p4 pi: " << endl;
p4_pi.Print();
```

\vec{p}_{π}^x

Boost

B rest frame

LAB

NB: the boost method modifies the pion 4-momentum:

```
--> LAB p4 pi:  $p_x$   $p_y$   $p_z$  Boost //  $\hat{x}$ 
(x,y,z,t)=(0.009777,0.100000,0.000000,0.172324) (P,eta,phi,E)=(0.100477,-0.000000,1.473334,0.172324)
```

the pion `py` is unchanged in the LAB as expected (perpendicular to the boost) and now there is a `px` component due to the boost of the B frame.

Tree for data storage

Tree

↳ Branch : Nmeas.

↳ Branch : PCumeas]

→ Branch : E [nmeas]

event == 1 B decay

} →
→

daughters

Branch { →
→
→
→

PB

B → $\pi^+ K^-$

$D^+ D^-$

$K^+ D^-$

$D^0 \pi^0$

nDau

Px [nDau]

Py [nDau]

Pz [nDau]

m [nDau]

charge [nDau]

B →

$B^0 \rightarrow \pi^+ K^-$

not possible

$B^0 \rightarrow D^+ \pi^- \pi^0$

nDau = 3

Simulation: produce and fill Tree.

measurement: some detector collects data.
and fills Tree.

Analysis class: has variables of branches as data members.

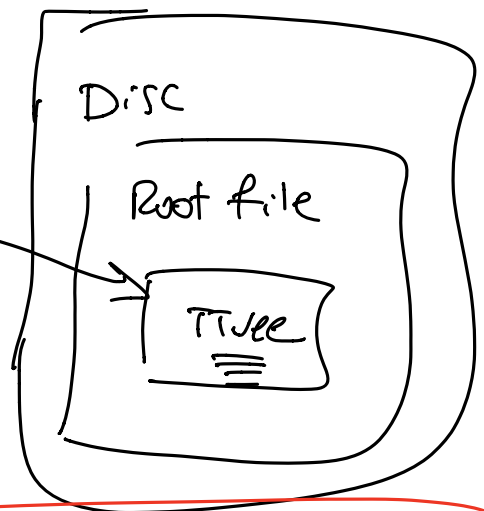
```
class Resistor {  
    private:  
        double R;  
}
```

class DataTree

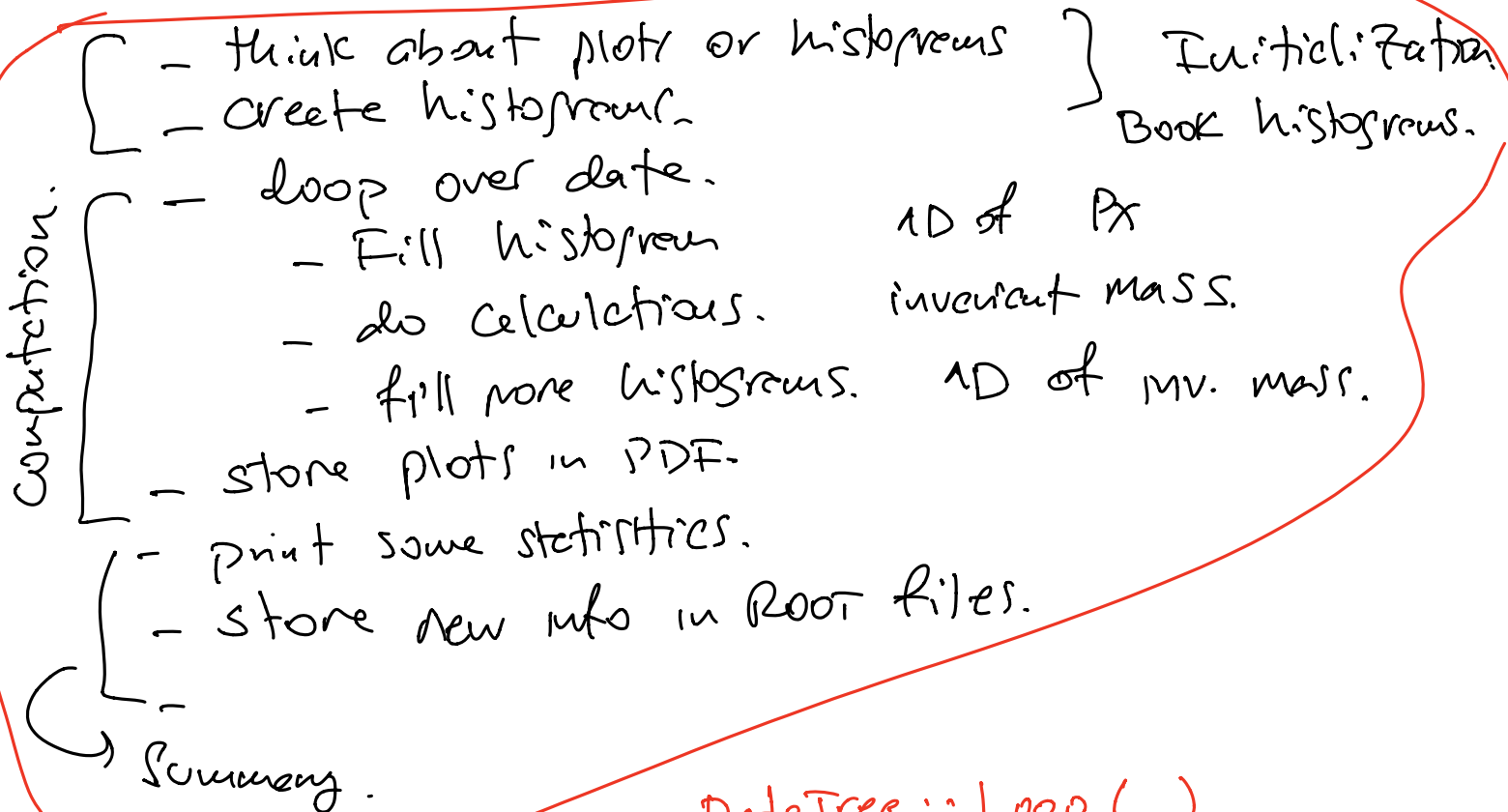
DataTree tree(---)

tree.pb.

tree.nmeas



Analysis process:



DataTree::Loop()

datenumbers.

GetEntry(j)

