C++ School 11-15 November, DESY

Our Small C++ Project

A simple MC generator to calculate Z production at Born level

Cross section

The Born level cross section is phase space integral of the matrix elements and the observable and it is convoluted to the parton distribution functions (PDFs):

Phase space

$$\begin{aligned}\sigma = \int_0^1 d\eta_a \int_0^1 d\eta_b \int d\Gamma(\eta_a, \eta_b; \{p, f\}_m) \\ \times f_{a/A}(\eta_a, \mu^2) f_{b/B}(\eta_b, \mu^2) \\ \times \left|M(\{p, f\}_m)\right|^2 F(\{p, f\}_m)\end{aligned}$$

PDFs

Observables

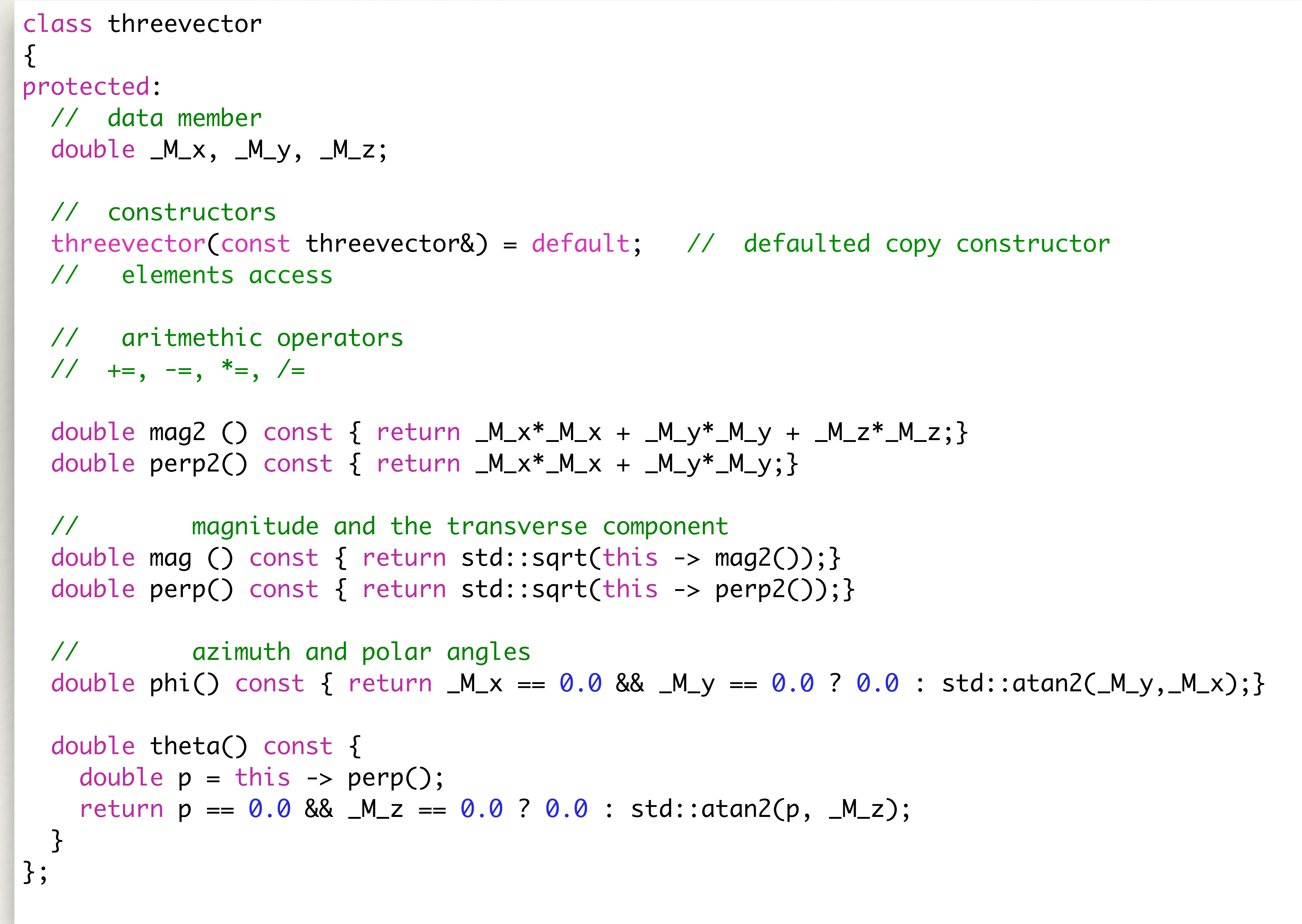
Matrix element

The event is an array of *momenta* and *flavor* of the incoming and outgoing partons.

Need a Lorentz vector

Lorentz vector: Three vector

Lorentz vector has 3 space-like and 1 time-like component. The space-like part is the usual three vector with X, Y, Z component. Thus first we want to define a class that represents three vectors.

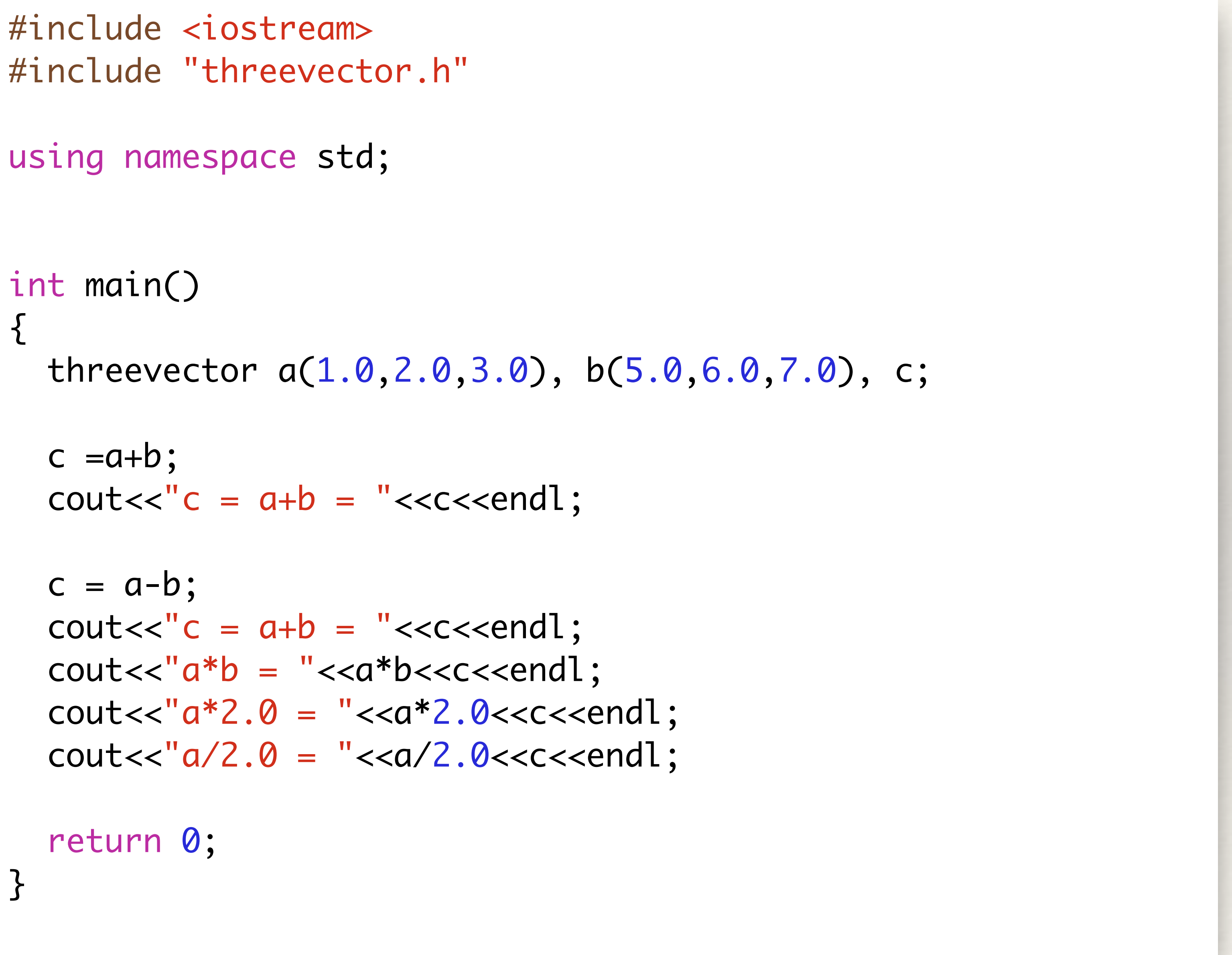


```
class threevector
{
protected:
  //  data member
  double _M_x, _M_y, _M_z;

  //  constructors
  threevector(const threevector&) = default;   //  defaulted copy constructor
  //   elements access

  //   aritmethic operators
  //  +=, -=, *=, /=

  double mag2 () const { return _M_x*_M_x + _M_y*_M_y + _M_z*_M_z;}
  double perp2() const { return _M_x*_M_x + _M_y*_M_y;}

  //        magnitude and the transverse component
  double mag () const { return std::sqrt(this -> mag2());}
  double perp() const { return std::sqrt(this -> perp2());}

  //        azimuth and polar angles
  double phi() const { return _M_x == 0.0 && _M_y == 0.0 ? 0.0 : std::atan2(_M_y,_M_x);}

  double theta() const {
    double p = this -> perp();
    return p == 0.0 && _M_z == 0.0 ? 0.0 : std::atan2(p, _M_z);
  }
};
```

- Write the header file `threevector.h`
- We *don't need* `.cc` file since every functions are simple and they can be inline.
- Play with, try the arithmetic operators with simple examples.

Three vector

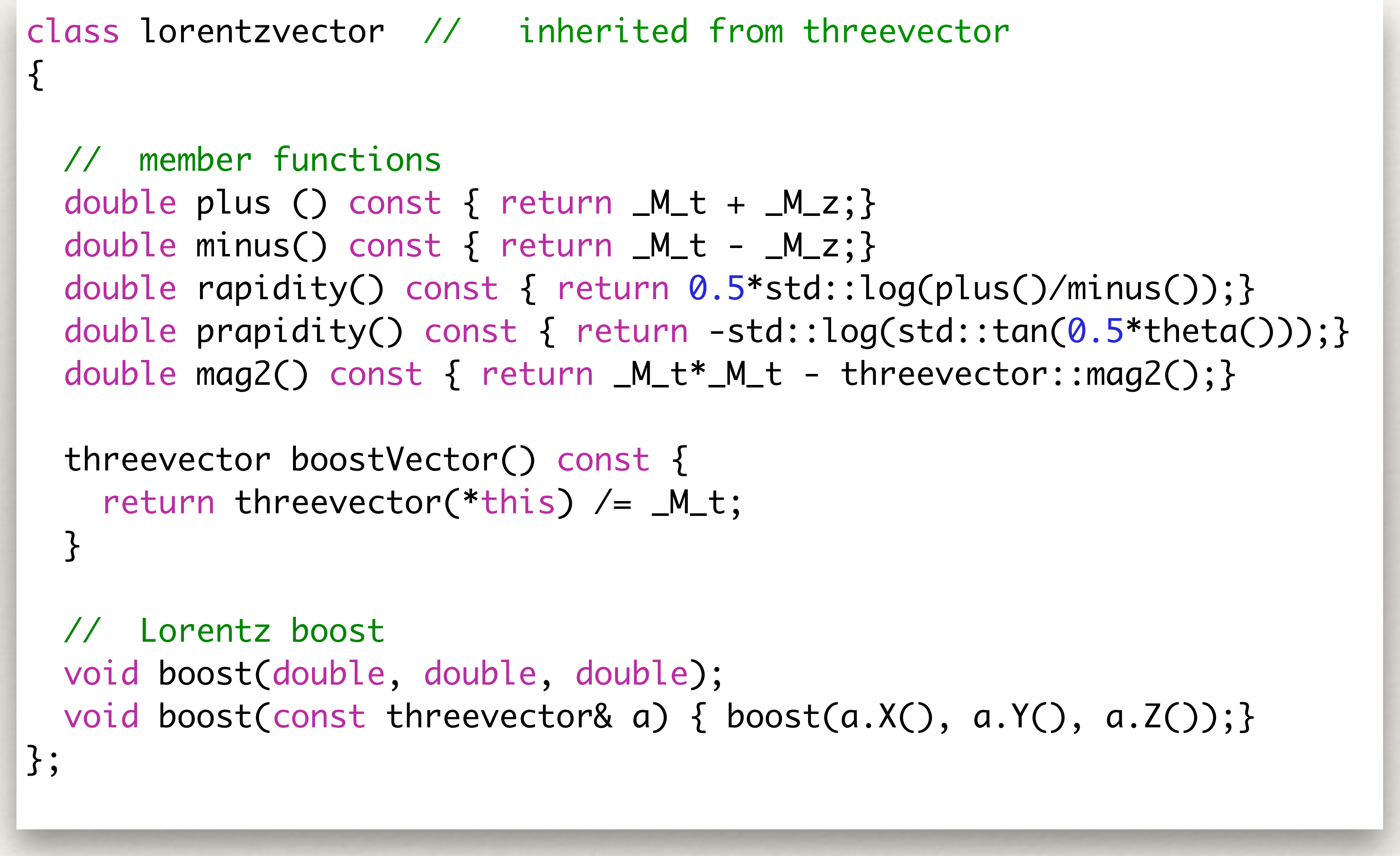
At the end of the day you should be able to do something like this:



```cpp
#include <iostream>
#include "threevector.h"

using namespace std;


int main()
{
  threevector a(1.0,2.0,3.0), b(5.0,6.0,7.0), c;

  c =a+b;
  cout<<"c = a+b = "<<c<<endl;

  c = a-b;
  cout<<"c = a+b = "<<c<<endl;
  cout<<"a*b = "<<a*b<<c<<endl;
  cout<<"a*2.0 = "<<a*2.0<<c<<endl;
  cout<<"a/2.0 = "<<a/2.0<<c<<endl;

  return 0;
}
```

Lorentz vector

Lorentz vector also has time-like component. Define a class inherited from three vector. Define all the arithmetic operators plus some more functions



```
class lorentzvector  //  inherited from threevector
{

  //  member functions
  double plus () const { return _M_t + _M_z;}
  double minus() const { return _M_t - _M_z;}
  double rapidity() const { return 0.5*std::log(plus()/minus());}
  double prapidity() const { return -std::log(std::tan(0.5*theta()));}
  double mag2() const { return _M_t*_M_t - threevector::mag2();}

  threevector boostVector() const {
    return threevector(*this) /= _M_t;
  }

  //  Lorentz boost
  void boost(double, double, double);
  void boost(const threevector& a) { boost(a.X(), a.Y(), a.Z());}
};
```

- Write the header file `lorentzvector.h`
- The `boost(…)` function is implemented in the `lorentzvector.cc` file.
- Play with, try the arithmetic operators with simple examples.