MANCHESTER 1824
The University of Manchester

Particle Catalogue - PHYS30762 C++ Project

Laxman Seelan
Student ID: 10717429
Department of Physics and Astronomy
The University of Manchester

(Dated: May 2024)

Abstract: This report outlines the design and implementation of a particle class hierarchy in C++ for cataloguing sub-atomic particles and their decay modes. The particles are derived from the abstract base class, 'Particle', and a container was created to hold the objects. The classes adhere to the 'rule of 5', which is required to handle objects and manage resources efficiently. Each particle contains a unique pointer to a four-momentum object, validated for physical plausibility using the particle's mass and Einstein's equation.

1 Introduction

The SM is built upon the framework of local relativistic quantum field theory [1], and fundamental particles are required to be invariant under a local gauge transformation of $SU(3) \times SU(2)_L \times U(1)$ [2] symmetry group. The model gives a description of the 'building blocks' of the universe and the interactions between them are mediated by 3 forces: the strong force, the weak force, and the electromagnetic force. The elementary particles are split into bosons (integer spin) and fermions (spin 1/2).

The fermions obey the Pauli Exclusion Principle and can be split into three generations [3]. There are four types of fermions: up-type quarks (u, c, t), down-type quarks (d, s, b), neutral lepton (ν_e, ν_μ, ν_τ), charged lepton (e, μ, τ). Each generation has the same properties, except flavour, and the masses vary significantly [3]. Furthermore, antiparticles, predicted by Dirac's equation for a charged massive fermion, are particles of the same mass and spin but opposite quantum numbers, such as charge and baryon number. The up-type quarks have a charge of $+2/3$ and the down-type have a charge of $-1/3$, and hadronise to form baryons, and mesons while adhering to Quantum Chromodynamics (QCD). The leptons carry quantum number lepton number, while the quarks carry baryon number and colour charge.

The vector bosons (gluons, $W^{\pm}$, Z, γ) are the force carriers. The massless γ mediates the EM interaction, which couples charged particles. The Weak interaction is mediation by the massive $W^{\pm}$ and couples neutral and charged leptons or up-type quarks to down-type quarks and the Z boson which couples particles of the same flavour. The scalar higgs boson is produced by the higgs field. he scalar Higgs boson is generated by the Higgs field, and particles interact with this field to acquire mass through the Higgs Mechanism.

The strong interactions are described by an $SU(3)$ local gauge theory called QCD, which uses three colour charges (red, green, blue). The interaction is mediated by gluons which carry a colour and anticolour charge. The theory includes confinement which predicts that no isolated coloured charge exists, including gluons [1]. This concept is applied to the hadronic decays in this project, as described in 2.1.5.

This project will implement the ideas from the SM to create a particle class hierarchy in `C++` for cataloguing particles and their decays.

2 Code Design and Implementation

This project uses an abstract base class, 'Particle', which contains common properties of the derived classes, such as spin and rest mass. Derived classes are inherited from the base class to create Bosons, Leptons and Quarks, as seen in 1. The Bosons are further split into the force carriers and the Higgs Boson, while the lepton class is split into the three flavours of charged leptons and neutrinos. The neutrinos and quarks are not split into their flavour particles to avoid code duplicity, as the features between the flavours are similar, unlike the Lepton and Boson classes, which interact differently. This section will give a brief description of each class.

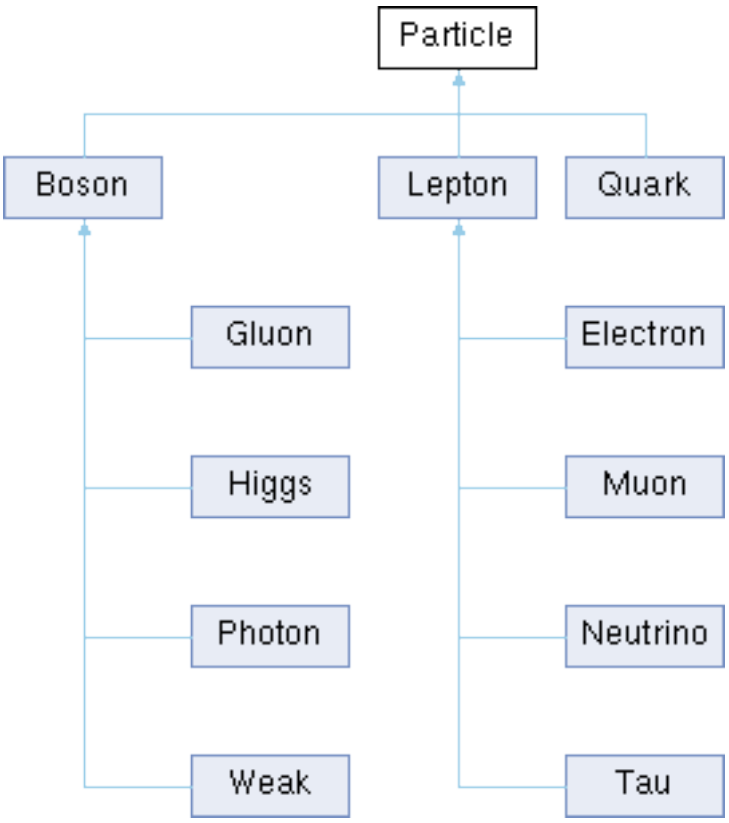

Fig. 1. UML diagram showing the inheritance chain of the SM model.

2.1 Classes in Inheritance Chain

2.1.1 Particle

The `Particle` class is an abstract base class that cannot be instantiated and is only used as an interface. The class contains the protected data members `double spin`, `double rest_mass`, `double charge`, `std::string name`, `bool m_is_antiparticle`, and `std::unique_ptr<FourMomentum> four_momentum_ptr`. These are general properties belonging to the derived classes. The unique pointer binds the particle object with a four-momentum object. The `Particle` class defines pure virtual functions that derived classes, such as a print function, must implement.

2.1.2 Lepton

The `Lepton` class is derived from the `Particle` class and has the protected data member `int lepton_number`. It overrides virtual functions like `print_data()` and `get_type()` from the `Particle` base class.

2.1.3 Electron

The `Electron` class has a private data member, a unique pointer to a `Calorimeter Energy` object. The parameterised constructor splits the electron's energy evenly across the Electromagnetic

Calorimeter (ECAL). The setter for the ECAL from the `Electron` undergoes validation checks to ensure it matches as close as possible to the energy contained in the four-momentum.

2.1.4 Muon

The `Muon` class has a data member to indicate whether or not the muon has interacted and introduces new member setter and getter functions.

2.1.5 Tau

The τ has a much larger mass than the μ and e and decays hadronically and leptonically, with branching fractions 0.649 and 0.351, respectively. The `Tau` class has a member function which decays the tau particle using the branching fractions. However, the hadronic decays are assumed to have a pion decay product for simplicity. The `Tau` class has member functions, unique pointers to the decay products. The decay product's charge and flavour depend on the flavour of the τ. For example, the hadronic decay of τ^- will result in ν_τ and π^- to conserve lepton flavour and charge, while the τ^+ will result in $\bar{\nu_\tau}$ and π^+. The hadronic decay process via W boson requires coupling to an up-type and down-type quark. To implement this, a boolean was added in the `Quark` class so that the Weak decay can use one up-type and one-down type quark in the decay. Furthermore, when initialising the particle of the pion, the colour charge of the two quarks was chosen to be colour singlets (e.g. $r\bar{r}$).

2.1.6 Neutrino

The `Neutrino` class contains a boolean which indicates whether the neutrino has interacted with the detector. Furthermore, the class contains a data member for the neutrino flavour to avoid code duplication.

2.1.7 Quark

The `Quark` class is derived from the base class and is distinct from the `Lepton` class. The class has data members `double baryon_number`, `std::string colour_charge`, `bool m_is_up_type`, and `int flavour`. The boolean is used in the Weak decays of τ and $W^\pm$, while the flavour data member is used to simplify the code as each generation can be accessed using the integer.

2.1.8 Boson

The `Boson` class has a data member that indicates what type of force or mechanism the boson is associated with.

2.1.9 Gluon

The `Gluon` class has a data member associated with its two-colour charge. The colour charge pairs were chosen to be not colourless to adhere to QCD.

2.1.10 Higgs

The `Higgs` class has a `std::vector` data member, which contains the decay products of the Higgs. The decay products were chosen to be charge conjugate and have the same flavour. A member function was also created to print the decay products of the Higgs boson.

2.1.11 Photon

The photon contains no unique data members, but it is required as the Higgs Boson can decay to two photons.

2.1.12 Weak

The `Weak` class is similar to the `Higgs` class and has a `std::vector` data member that contains the decay products of the Higgs. A member function was also implemented to print the decay products of the $W^{\pm}$ or $Z^{\pm}$ boson. The decays were chosen so that colour-singlet decay products were produced in hadronic decays.

2.2 Other Classes

2.2.1 FourMomentum

The `FourMomentum` can be initialised without the `Particle` class and contains setter and getter functions. The energy of the four-momentum is cross-checked with the momentum and particle mass to create a physical four-momentum. This is achieved by using Einstein's equation.

2.2.2 CalorimeterEnergies

The `CalorimeterEnergies` class is used by the `Electron`, and its functionality is discussed in Section 2.1.3.

2.2.3 ParticleCatalogue

The `ParticleCatalogue` class is a template class that can contain any object. Furthermore, the class uses `std::map` to assign a key to each particle. The key is made to be unique. Furthermore,

the container has the data member `std::map<std::string, std::pair<SafeSharedPtr<ParticleType>, std::string>> particle_container`. The second element of the pair is a string, which is the type of particle contained. The `Particle` hierarchy uses five particle types: Charged Lepton, Neutrino, Boson, Up-type Quarks, and Down-type Quarks. The container also has member functions that print all the particles it contains, the number of particles it contains, summing four-momentum of all parties and getting a sub-container of a certain type.

2.3 Advanced Features

This subsection will summarise some of the advanced features used in this project.

The `std::map` was used instead of `std::vector` as a key that could be used to obtain the particle. Furthermore, the container is automatically sorted and prevents reallocation.

A template `SafeSharedPtr` was created to overload the $->$ operator so that when attempting to dereference a null ptr, it can safely exit without having to write if statements to check if the pointer was a null ptr every time. This is shown in Figure 2.

```
int main()
{
    SafeSharedPtr<Particle> pointer = nullptr;
    if(pointer)
    {
        pointer->print_data();
    }
    else
    {
        std::cerr<<"Error: Safe pointer is null."<<std::endl;
    }

    pointer->print_data();


    catalogue["Electron_21"]->print_data();
}
```

Fig. 2. Example C++ code for the SafeSharedPtr.

3 Results

The code is made to currently safely quit the program if there was an attempt to dereference a null `SafeSharedPtr`, as seen in Figure 3. This can be improved by overloading the $->$ operator to check the pointer before attempting to dereference.

```
----------------------------------------
Key not found: Electron_21
Error: Attempted to dereference a null Safe Shared Pointer. Terminating Program.
```

Fig. 3. Exiting after trying to derefence a SafeSharedPtr.

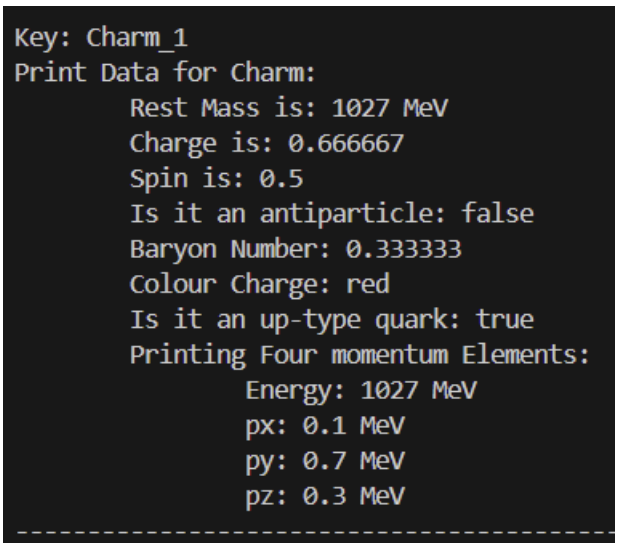

Fig. 4. The output of `print_data()` for the charm quark.

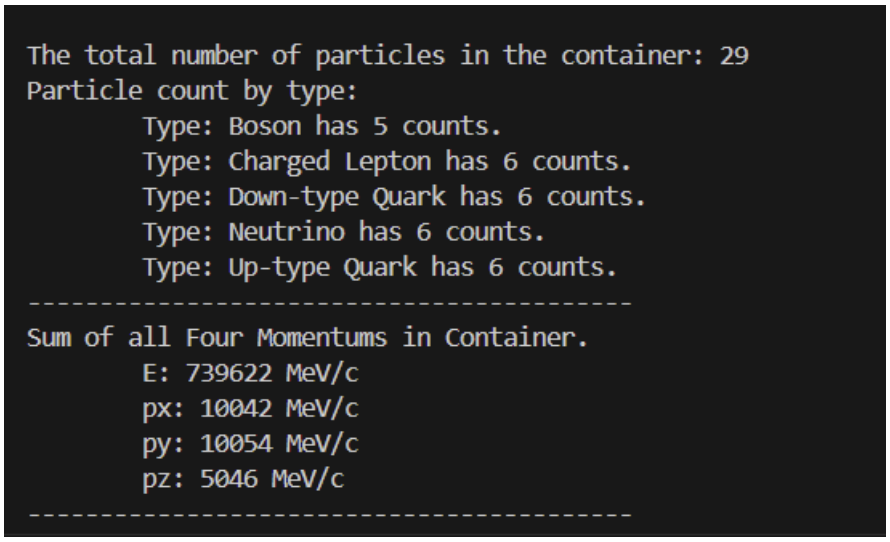

Fig. 5. A screenshot showing the use of the member functions of the `ParticleCatalogue`.

The code successfully prints all the data as well the data of the decay products as seen in Figure 4. Furthermore, the use of the member functions of `ParticleCatalogue` is shown in Figure 5.

4 Conclusion

The inheritance chain has been effectively implemented to carry out the requisite processes, including the management of decay products and unique data members. Furthermore, a template container was created to contain the different objects. However, this project can be improved in multiple ways. Presently, the container is only able to accommodate objects of the same type, necessitating the creation of a separate container to host objects of a different type. This can be improved by allowing the container to have multiple types of objects in the same container. Furthermore, the decay functions do not conserve transverse momentum or energy. Moreover, issues stemming from the abstract base class are evident in the `Weak` and `Higgs` copy constructor and copy assignment operator.

References

[1] G. Altarelli and J. Wells, “Gauge Theories and the Standard Model,” in *Collider Physics within the Standard Model* (Lecture Notes in Physics), Lecture Notes in Physics. Springer, 2017, vol. 937.

[2] S. Stemmle, “Measurement of the time-integrated CP asymmetry in prompt $D^0 \to K^- K^+$ decays with the LHCb experiment,” Master’s thesis, University of Heidelberg, 2015.

[3] E. Kennedy, “The Generation Model of Particle Physics,” in *Particle Physics*. IntechOpen, 2012.