PHYS Simulations

0.1.0

Generated by Doxygen 1.8.17

Chapter 1

Todo List

Namespace PHYS::Elements

Finish all element definitions

Namespace PHYS::Particles

Finish all particle definitions

Member PHYS::Particles::Bc

Correct properties

Member PHYS::Particles::phi_1020

Correct properties

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

Namespace Documentation

5.1 PHYS Namespace Reference

Namespaces

- Constants
- Elements
- Particles
- Units

Classes

- class Cartesian

  Cartesian Class.
- class Decay

  Decay Channel Class.
- class DecayTable

  Decay Table Class.
- class Isotope

  Isotope Subclass.
- class LorentzVector

  Lorentz Vector Class.
- class Nucleus

  Nucleus Class.
- class Object

  Object Class.
- class Particle

  Particle Class.
- class SimpleBody

  Simple Body Class.
- class Spherical

  Spherical Class.
- class Spring

  Spring Class.
- class Vector

  Vector Class.

Typedefs

- typedef Cartesian Force

  Force on a classical body.
- typedef Cartesian Coordinate

  Cartesian Co-ordinates.

Functions

- std::ostream & **operator**<< (std::ostream &os, Decay &d)
- const double getBindingEnergyPerNucleon (const Nucleus *n)
- const double fracIsotopesAt (const Isotope *i, const double t)

Variables

- const DecayTable BcDecays = gen_BcDecays()

  PHYS::DecayTable of all decays of the B_c meson. The `gen_BcDecays` function generates all the relevant channels.
- const DecayTable KplusDecays = gen_KplusDecays()

  DecayTable of all decays of the $K^{\pm}$ meson. The `gen_KplusDecay` function generates all the relevant channels.

5.1.1 Detailed Description

The PHYS namespace contains all currently accepted particles within the standard model and beyond. The masses and data are taken from the Particle Data Group listings [1].

Warning

It is recommend you do not use the namespace within the script as there is a high probability of accidental overwriting. Instead use the particles as PHYS::Particles::particle

Author

K. Zarebski

Copyright

MIT License

Date

last modified 2019-08-26

5.1.2 Function Documentation

5.1.2.1 fracIsotopesAt()

```
const double PHYS::fracIsotopesAt (
            const Isotope * i,
            const double t )
```

Nuclei decay - Get the fraction of isotopes at time t
$\frac{N(t)}{N_0} = e^{-t/\lambda}$

Parameters

i	Isotope to decay in time t
t	Time $t$in seconds

Definition at line 25 of file NuclearPhysics.cxx.

5.1.2.2 getBindingEnergyPerNucleon()

```
const double PHYS::getBindingEnergyPerNucleon (
             const Nucleus * n )
```

Returns the binding energy per nucleon for the given nucleus calculated using the semi-empirical mass formula: $\frac{\mathrm{BE}}{A \cdot MeV} = a_v - \frac{a_s}{A^{1/3}} - a_c \frac{Z^2}{A^{4/3}} - a_a \frac{(N-Z)^2}{A^2} \pm \frac{a_p}{A^{3/2}}$ [2]

Parameters

n	Nucleus on which to perform calculation

Definition at line 11 of file NuclearPhysics.cxx.

5.2 PHYS::Constants Namespace Reference

Variables

- const double pi = 3.141592653589793
  Constant of a circle π.
- const double c = 2.99792458E6∗PHYS::Units::m/pow(PHYS::Units::sec, 2)
  Speed of light c.
- const double N_A = 6.02214076E23
  Avogadro's constant N_A.
- const double h = 6.626070040E-34∗PHYS::Units::J∗PHYS::Units::sec
  Planck's constant h.
- const double hbar = h/(2∗PHYS::Constants::pi)
  Reduced Planck's constant $\hbar$.
- const double e = 1.6021766208E-19∗PHYS::Units::C
  Electron charge.
- const double m_e = 0.5109989461∗PHYS::Units::MeV
  Electron mass m_e.
- const double m_p = 938.2720813∗PHYS::Units::MeV
  Proton mass m_p.
- const double epsilon_0 = 8.854187817E-12∗PHYS::Units::F∗pow(PHYS::Units::m, -1)
  Permitttivity of free space ϵ_0.
- const double mu_0 = 4∗PHYS::Constants::pi∗1E-7∗PHYS::Units::N∗pow(PHYS::Units::A, -2)
  Permeability of free space μ_0.

- const double alpha = pow(PHYS::Constants::e,2)∗pow(4∗PHYS::Constants::pi∗PHYS::Constants::epsilon_0∗PHYS::Constant
  1)

  Fine structure constant α.
- const double r_e = pow(PHYS::Constants::e,2)∗pow(4∗PHYS::Constants::pi∗PHYS::Constants::epsilon_0∗PHYS::Constants::
  -1)

  Classical electron radius r_e.
- const double r_bohr = PHYS::Constants::r_e∗pow(PHYS::Constants::alpha, -2)

  Bohr radius (mass of nucleus is infinite) r_{Bohr}.
- const double sigma_T = 8∗PHYS::Constants::pi∗pow(PHYS::Constants::r_e,2)/3.

  Thomson cross section σ_T.
- const double mu_B = PHYS::Constants::e∗PHYS::Constants::hbar∗pow(2∗PHYS::Constants::m_e,-1)

  Bohr magneton μ_B.
- const double mu_N = PHYS::Constants::e∗PHYS::Constants::hbar∗pow(2∗PHYS::Constants::m_p,-1)

  Nuclear magneton @f\mu_N.
- const double G = 6.67408E-11∗pow(PHYS::Units::m, 3)∗pow(PHYS::Units::kg, -1)∗pow(PHYS::Units::sec,
  -2)

  Gravitational Constant G.
- const double g = 9.80665∗PHYS::Units::m∗pow(PHYS::Units::sec, -2)

  Standard gravitational acceleration g.
- const double k = 1.38064852E-23∗PHYS::Units::J∗pow(PHYS::Units::K, -1)

  Boltzmann constant k.
- const double b = 2.8977729E-3∗PHYS::Units::m∗PHYS::Units::K

  Wien constant b.
- const double sigma = pow(PHYS::Constants::pi, 2)∗pow(PHYS::Constants::k, 4)/(60∗pow(PHYS::Constants::hbar,
  3)∗pow(PHYS::Constants::c, 2))

  Stefan-Boltzmann constant σ.
- const double alpha_s = 0.1181

  Strong coupling constant α_s.
- const double exp = 2.718281828459045235

  Exponential number.
- const double m_planck = pow(PHYS::Constants::hbar∗PHYS::Constants::c/PHYS::Constants::G, 0.5)

  Planck Mass.
- const double l_planck = pow(PHYS::Constants::hbar∗PHYS::Constants::G/pow(PHYS::Constants::c, 3),
  0.5)

  Planck Length.
- const double u = 1.66053906660E-27∗PHYS::Units::kg

  Atomic mass unit.

5.2.1 Detailed Description

Namespace containing all physical constants

Author

K. Zarebski

Copyright

MIT License

Date

last modified 2019-08-27

5.3 PHYS::Elements Namespace Reference

Variables

- const PHYS::Nucleus **Hydrogen** = PHYS::Nucleus("H", 1, 1.0079)
- const PHYS::Nucleus **Helium** = PHYS::Nucleus("He", 2, 4.0026)
- const PHYS::Nucleus **Lithium** = PHYS::Nucleus("Li", 3, 6.941)
- const PHYS::Nucleus **Beryllium** = PHYS::Nucleus("Be", 4, 9.0122)
- const PHYS::Nucleus **Boron** = PHYS::Nucleus("B", 5, 10.811)
- const PHYS::Nucleus **Carbon** = PHYS::Nucleus("C", 6, 12.0107)
- const PHYS::Nucleus **Nitrogen** = PHYS::Nucleus("N", 7, 8)
- const PHYS::Nucleus **Oxygen**
- const PHYS::Nucleus **Fluorine**
- const PHYS::Nucleus **Neon**
- const PHYS::Nucleus **Sodium**
- const PHYS::Nucleus **Magnesium**
- const PHYS::Nucleus **Aluminium**
- const PHYS::Nucleus **Silicon**
- const PHYS::Nucleus **Phosphorus**
- const PHYS::Nucleus **Sulfur**
- const PHYS::Nucleus **Chlorine**
- const PHYS::Nucleus **Potassium**
- const PHYS::Nucleus **Argon**
- const PHYS::Nucleus **Calcium**
- const PHYS::Nucleus **Scandium**
- const PHYS::Nucleus **Titanium**
- const PHYS::Nucleus **Vanadium**
- const PHYS::Nucleus **Chromium**
- const PHYS::Nucleus **Manganese**
- const PHYS::Nucleus **Iron**
- const PHYS::Nucleus **Nickel**
- const PHYS::Nucleus **Cobalt**
- const PHYS::Nucleus **Copper**

5.3.1 Detailed Description

Namespace containing all stable elements found in CommonNuclei.hxx

Todo Finish all element definitions

Author

K. Zarebski

Copyright

MIT License

Date

last modified 2019-08-27

5.4 PHYS::Particles Namespace Reference

Variables

- const Particle gamma

  Photon, γ.
- const Particle g

  Gluon, g.
- const Particle W

  W Boson, W^+.
- const Particle Z

  Z Boson, Z^0.
- const Particle H

  Higgs Boson, H.
- const Particle e = PHYS::Particle("e", "-", PHYS::Constants::m_e, 3E36∗PHYS::Units::sec)

  Electron, e^-.
- const Particle mu = PHYS::Particle("mu", "-", 105.6583745∗PHYS::Units::MeV, 2.1969811E-6∗PHYS::↩ Units::sec)

  Muon, μ^-.
- const Particle tau = PHYS::Particle("tau", "-", 1776.82∗PHYS::Units::MeV, 2.90610E-13∗PHYS::Units::sec)

  Tau, τ^-.
- const Particle nu_e = PHYS::Particle("nu_e", "", 1E-6∗PHYS::Units::MeV, -1)

  Electron Neutrino, ν_e.
- const Particle nu_mu = PHYS::Particle("nu_mu", "", 1E-6∗PHYS::Units::MeV, -1)

  Muon Neutrino, ν_μ.
- const Particle nu_tau = PHYS::Particle("nu_tau", "", 1E-6∗PHYS::Units::MeV, -1)

  Tau Neutrino, ν_τ.
- const Particle Kplus = PHYS::Particle("K", "+", 493.677∗PHYS::Units::MeV, 1.2380E-8)

  Charged Kaon, K^+.
- const Particle phi_1020 = PHYS::Particle("phi", "", 999, 999)

  Phi 1020, $\phi(1020)$.
- const Particle eta

  Eta meson, η.
- const Particle Piplus = PHYS::Particle("pi", "+", 139.57061∗PHYS::Units::MeV, 2.6033E-8∗PHYS::Units↩ ::sec)

  Charged Pion, π^+.
- const Particle Pi0 = PHYS::Particle("pi", "0", 134.9770∗PHYS::Units::MeV, 8.52E-17∗PHYS::Units::sec)

  Neutral Pion, π^0.
- const Particle Dplus

  Charged D meson, D^+.
- const Particle D0

  Neutral D meson, D^0.
- const Particle Ds

  Neutral D strange meson, D_s.
- const Particle Bplus

  Charge B meson, B^+.
- const Particle B0

  Neutral B meson, B^0.
- const Particle Bs

  Neutral B strange meson, B_s.
- const Particle Bc = PHYS::Particle("Bc", "+", 999, 999)

Charmed B meson, B_c.

- const Particle p

  Proton, p.
- const Particle n

  Neutron, n.
- const Particle Lambda

  Lambda baryon, Λ.
- const Particle Lambdab0

  Lambda b baryon, Λ_b^0.
- const Particle Sigma

  Neutral Sigma baryon, Σ^0.
- const Particle Sigmaplus

  Charged Sigma baryon, Σ^+.
- const Particle Xi0

  Neutral Xi baryon, Ξ^0.
- const Particle Ximinus

  Charged Xi baryon, Ξ^+.
- const Particle Omega

  Omega baryon, Ω.
- const Particle Lambdac

  Lambda c baryon, Λ_c.

5.4.1 Detailed Description

Namespace containing all particles listed within the PDG

Todo Finish all particle definitions

Author

K. Zarebski

Copyright

MIT License

Date

last modified 2019-08-26

5.4.2 Variable Documentation

5.4.2.1 Bc

```
const PHYS::Particle PHYS::Particles::Bc = PHYS::Particle("Bc", "+", 999, 999)
```

Charmed B meson, B_c.

Todo Correct properties

Definition at line 38 of file CommonParticles.cxx.

5.4.2.2 phi_1020

```
const PHYS::Particle PHYS::Particles::phi_1020 = PHYS::Particle("phi", "", 999, 999)
```

Phi 1020, $\phi(1020)$.

Todo Correct properties

Definition at line 28 of file CommonParticles.cxx.

5.5 PHYS::Units Namespace Reference

Variables

- const double **cd** = 1.
- const double **m** = 1.
- const double **mm** = 1E-3∗m
- const double **nm** = 1E-9∗m
- const double **km** = 1E3∗m
- const double **cm** = 1E-2∗m
- const double **angstrom** = 1E-10∗m
- const double **mile** = 1.609344E3∗m
- const double **yd** = 0.9144∗m
- const double **AU** = 1.495979E12∗m
- const double **pc** = 3.08567758149E16∗m
- const double **inch** = 0.0254∗m
- const double **kg** = 1.
- const double **t** = 1E3∗kg
- const double **g** = 1E-3∗kg
- const double **sec** = 1.
- const double **ms** = 1E-3∗sec
- const double **ns** = 1E-9∗sec
- const double **ps** = 1E-12∗sec
- const double **J** = 1.
- const double **eV** = PHYS::Constants::e∗J
- const double **keV** = 1E3∗eV
- const double **MeV** = 1E6∗eV

- const double **GeV** = 1E9∗eV
- const double **TeV** = 1E12∗eV
- const double **erg** = 1E-7∗J
- const double **b** = 1E-28∗m∗m
- const double **mb** = 1E-3∗b
- const double **nb** = 1E-9∗b
- const double **pb** = 1E-12∗b
- const double **fb** = 1E-15∗b
- const double **inv_pb** = 1./pb
- const double **inv_fb** = 1./fb
- const double **N** = 1.∗kg∗pow(m,-2)
- const double **dyne** = 1E-5∗N
- const double **kN** = 1E3∗N
- const double **A** = 1.
- const double **V** = 1.∗J∗pow(C,-1)
- const double **C** = 1.∗A∗sec
- const double **F** = 1.∗C∗pow(V,-1)
- const double **W** = J/sec
- const double **Ohms** = V/A
- const double **S** = A/V
- const double **Wb** = V∗sec
- const double **H** = Wb/A
- const double **esu** = pow(2.99792458E-9, -1)∗C
- const double **K** = 1.
- const double **Celsius** = -273.15
- const double **Fahrenheit** = 33.8∗Celsius
- const double **T** = Wb∗pow(m, -2)
- const double **G** = 1E-4∗T
- const double **Pa** = 1.∗N∗pow(m, -2)
- const double **atm** = 1.01325E5∗Pa
- const double **Torr** = pow(760.,-1)∗atm
- const double **rad** = 1.
- const double **deg** = rad∗180/PHYS::Constants::pi
- const double **sr** = 1.
- const double **Hz** = 1./sec
- const double **lm** = cd∗sr
- const double **lx** = lm∗pow(m, -2)
- const double **Bq** = 1.
- const double **Gy** = J∗pow(kg, -1)
- const double **Sv** = J∗pow(kg, -1)
- const double **kat** = 1.

5.5.1 Detailed Description

Namespace containing all units for easy conversion to SI units for calculations

Author

K. Zarebski

Copyright

MIT License

Date

last modified 2019-08-27

Chapter 6

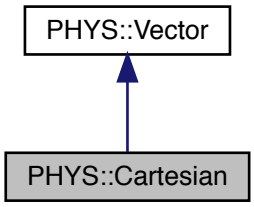
Class Documentation

6.1 PHYS::Cartesian Class Reference

Cartesian Class.

`#include <Vector.hxx>`

Inheritance diagram for PHYS::Cartesian:



Public Member Functions

- **Cartesian** (const double x, const double y, const double z)
- **Cartesian** (const Vector &vec)
- const double **magnitude** () const
- const Cartesian **operator-** (const Cartesian &other) const
- const Cartesian **operator+** (const Cartesian &other) const
- const Spherical inSphericalPolar () const

6.1.1 Detailed Description

Cartesian Class.

Subclass of Vector for Cartesian co-ordinates

Version

0.1.0

Author

Kristian Zarebski

Date

last modified 2019-08-28

Copyright

MIT License

Definition at line 52 of file Vector.hxx.

6.1.2 Member Function Documentation

6.1.2.1 inSphericalPolar()

```
const PHYS::Spherical PHYS::Cartesian::inSphericalPolar ( ) const
```

Converts the given co-ordinates to spherical polar form

Returns

Spherical

Definition at line 75 of file Vector.cxx.

The documentation for this class was generated from the following files:

- include/Vector.hxx
- src/Vector.cxx

6.2 PHYS::Decay Class Reference

Decay Channel Class.

```
#include <DecayTable.hxx>
```

Public Member Functions

- Decay ()
  Blank decay object.
- const std::string getDecStr () const
  Get the decay description as a string.
- Decay (std::vector< Particle > daughters, double probability, Particle mother=Particle())
  Construct a decay with a given mother, daughters and branching ratio.
- bool isValid (double threshold)
  Used to check probability of decay occuring against a threshold.
- const double getBR () const
  Returns the branching ratio.
- const Particle getMother () const
  Returns the mother object.
- void setMother (Particle &Mother)
  Sets the mother particle.
- const std::vector< Particle > getDaughters () const
  Returns a vector of the daughter particles.

Private Attributes

- double _prob
  Probability of decay occuring (i.e. Branching Ratio)
- std::vector< Particle > _daughters
  Daughter particles.
- Particle _mother = Particle()
  Mother particle.

6.2.1 Detailed Description

Decay Channel Class.

Class describing a single particle decay with information on the mother and daughter particles and branching ratio.

Version

0.1.0

Author

Kristian Zarebski

Date

last modified 2019-08-25

Copyright

MIT License

Definition at line 24 of file DecayTable.hxx.

The documentation for this class was generated from the following files:

- include/DecayTable.hxx
- src/DecayTable.cxx

6.3 PHYS::DecayTable Class Reference

Decay Table Class.

`#include <DecayTable.hxx>`

Public Member Functions

- DecayTable (const Particle &mother)
- void addDecay (Decay &decay)

  Add a new decay to the decay table.
- std::vector< Decay > getDecays ()

  Get the decays in decay tablea as a vector.
- const Decay getRandom () const
- void Print ()

  Print the decay table.

Private Attributes

- std::vector< double > _brs
- std::vector< Decay > _decays

  Decays as list.
- std::vector< double > _cumul_brs

  Cumulative branching ratios.
- Particle _mother = Particle()

  Common mother to all decays.

6.3.1 Detailed Description

Decay Table Class.

Class containing all decay channels for a given mother particle this is used to generate a decay at random

Version

0.1.0

Author

Kristian Zarebski

Date

last modified 2019-08-26

Copyright

MIT License

Examples

KplusDecTable.hxx.

Definition at line 63 of file DecayTable.hxx.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 DecayTable()

```
PHYS::DecayTable::DecayTable (
            const Particle & mother )
```

Construct a decay table for a given mother particle.

Parameters

mother	The mother particle

Returns

void

Definition at line 35 of file DecayTable.cxx.

6.3.3 Member Function Documentation

6.3.3.1 getRandom()

```
const PHYS::Decay PHYS::DecayTable::getRandom ( ) const
```

Get a decay at random based on the branching ratios

Returns

std::vector<Decay>

Definition at line 57 of file DecayTable.cxx.

6.3.4 Member Data Documentation

6.3.4.1 _brs

```
std::vector<double> PHYS::DecayTable::_brs  [private]
```

Branching ratios as keys

Definition at line 68 of file DecayTable.hxx.

The documentation for this class was generated from the following files:

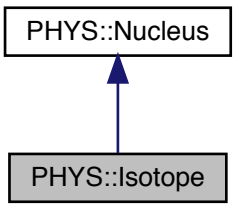
- include/DecayTable.hxx
- src/DecayTable.cxx

6.4 PHYS::Isotope Class Reference

Isotope Subclass.

```
#include <NuclearPhysics.hxx>
```

Inheritance diagram for PHYS::Isotope:



Public Member Functions

- Isotope (const Nucleus &n, const int n_neutrons, const double half_life)
- const double **getHalfLife** () const

Private Attributes

- const double **_half_life** = -1

6.4.1 Detailed Description

Isotope Subclass.

Class representing isotopes which inherits from the Nucleus class

Version

0.1.0

Author

Kristian Zarebski

Date

last modified 2019-08-27

Copyright

MIT License

Definition at line 58 of file NuclearPhysics.hxx.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 Isotope()

```
PHYS::Isotope::Isotope (
            const Nucleus & n,
            const int n_neutrons,
            const double half_life )  [inline]
```

Construct an isotope for a given pre-defined nucleus

Parameters

n	Nucleus for which the isotope shares the same atomic number Z
half_life	The half-life for the given isotope λ

Returns

Isotope of Nucleus n

Definition at line 68 of file NuclearPhysics.hxx.

The documentation for this class was generated from the following file:

- include/NuclearPhysics.hxx

6.5 PHYS::LorentzVector Class Reference

Lorentz Vector Class.

`#include <LorentzVector.hxx>`

Public Member Functions

- LorentzVector ()

  Create a blank LorentzVector with the default values of -9999.
- const double & operator[ ] (size_t i)

  Specify a component of the LorentzVector by index [0, 3] for (x_0, x_1, x_2, x_3).
- double operator[ ] (size_t i) const

  Specify a component of the LorentzVector by index [0, 3] for (x_0, x_1, x_2, x_3).
- LorentzVector (double x0, double x1, double x2, double x3)

  Create a new LorentzVector by giving the four values (x_0, x_1, x_2, x_3).
- const double magnitude () const

  Returns the magnitude of the vector as $r = \sqrt{x_0^2 - x_1^2 - x_2^2 - x_3^2}$.

Private Attributes

- double **x0** =-9999
- double **x1** =-9999
- double **x2** =-9999
- double **x3** =-9999

Friends

- std::ostream & operator<< (std::ostream &os, const LorentzVector &lv)

  Output the components of the LorentzVector to the ostream when printing.

6.5.1 Detailed Description

Lorentz Vector Class.

Class to represent Lorentz vectors of the form e.g. four momenta

Version

0.1.0

Author

Kristian Zarebski

Date

last modified 2019-08-26

Copyright

Definition at line 16 of file LorentzVector.hxx.

The documentation for this class was generated from the following files:

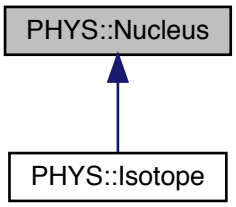
- include/LorentzVector.hxx
- src/LorentzVector.cxx

6.6 PHYS::Nucleus Class Reference

Nucleus Class.

`#include <NuclearPhysics.hxx>`

Inheritance diagram for PHYS::Nucleus:



Public Member Functions

- Nucleus (const std::string name, const int atomic_number, const double atomic_mass)
- const std::string getName () const

  Get name of nucleus.
- const int N () const

  Get number of neutrons N.
- const int A () const

  Get atomic mass number A.
- const int Z () const

  Get number of protons/atomic number Z.
- const double M () const

  Get atomic mass.
- const double mass_defect () const

  Get the mass defect for the nucleus.

Private Attributes

- const std::string **_name**
- const int **_n_protons**
- const int **_n_neutrons**
- const double **_atomic_mass**

6.6.1 Detailed Description

Nucleus Class.

Class representing a nucleus which can be used to construct the elements

Version

0.1.0

Author

Kristian Zarebski

Date

last modified 2019-08-27

Copyright

MIT License

Definition at line 18 of file NuclearPhysics.hxx.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Nucleus()

```
PHYS::Nucleus::Nucleus (
            const std::string name,
            const int atomic_number,
            const double atomic_mass )  [inline]
```

Create a new nucleus object

Parameters

name	Symbol/Name of nucleus
atomic_number	Number of protons/Atomic number Z
atomic_mass	Mass of the nucleus in atomic mass units

Definition at line 31 of file NuclearPhysics.hxx.

The documentation for this class was generated from the following files:

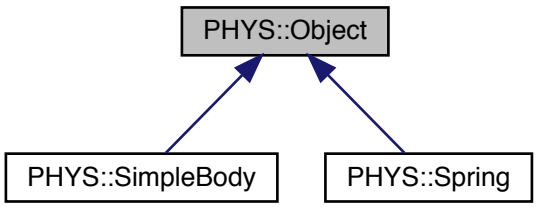
- include/NuclearPhysics.hxx
- src/NuclearPhysics.cxx

6.7 PHYS::Object Class Reference

Object Class.

`#include <ClassicalMechanics.hxx>`

Inheritance diagram for PHYS::Object:



Public Member Functions

- **Object** (const std::string label="", const Coordinate position={-999,-999,-999})
- void **Place** (const Coordinate &c)
- const std::string **getName** () const
- const Coordinate **getPosition** () const
- void applyForce (Force &f)
- const Force resolve () const

Private Attributes

- std::string **_name**
- Coordinate **_position**
- std::vector< Force > **_forces**

6.7.1 Detailed Description

Object Class.

Class to describe all basic objects used in the classical mechanics library

Version

0.1.0

Author

Kristian Zarebski

Date

last modified 2019-08-27

Copyright

MIT License

Definition at line 25 of file ClassicalMechanics.hxx.

6.7.2 Member Function Documentation

6.7.2.1 applyForce()

```
void PHYS::Object::applyForce (
            PHYS::Force & f )
```

Apply a force to the body

Parameters

f	Force in the form of a Vector

Returns

void

Definition at line 3 of file ClassicalMechanics.cxx.

6.7.2.2 resolve()

```
const PHYS::Force PHYS::Object::resolve ( ) const
```

Resolve all applied forces to give the resultant force

Returns

Force (Vector)

Definition at line 8 of file ClassicalMechanics.cxx.

The documentation for this class was generated from the following files:

- include/ClassicalMechanics.hxx
- src/ClassicalMechanics.cxx

6.8 PHYS::Particle Class Reference

Particle Class.

```
#include <Particle.hxx>
```

Public Member Functions

- Particle ()
  Default constructor with properties set to -1.
- Particle (std::string, std::string, double, double)
  Construct a particle with a given name, sign, mass and lifetime. The energy will be taken to be the rest mass.
- Particle (double, double, double, double)
  Construct a particle giving only the four momentum components.
- const std::string getName () const
  Returns as a string the particle name.
- const Particle anti () const
  Get the antiparticle partner of the current particle.
- void Fire (double)
  Fire the particle at a given energy.
- const double M () const
  Get the mass of the particle.
- const double phi () const
  Get the azimuthal angle ϕ of the particle trajectory.
- const double y () const
  Get the rapidity $y = \frac{1}{2}\log\left(\frac{E+p_z}{E-p_z}\right)$ of the particle.
- const double eta () const
  Get the pseudorapidity $\eta = -\log\left(\tan\frac{\theta}{2}\right)$ of the particle.
- const double theta () const
  Get the angle θ of the particle trajectory in the $x-z$ plane.
- const PHYS::LorentzVector momentum () const
  Get the LorentzVector of the given particle.

- const double PT () const

  Get the transverse momentum p_T of the particle.
- const double P () const

  Get the 3-momentum magnitude p.
- const double beta (const int p_i=0) const

  Calculate the speed $\beta = \frac{p_i}{E}$ in terms of c. Optional argument allows you to specify a component of momentum [1-3] for (x, y, z).
- const double gamma () const

  Calculate the Lorentz factor γ.
- const double ctau () const

  Return decay length in particle rest frame.
- operator bool () const

  Returns true if the particle has been initialised (does not have default blank values)

Private Attributes

- std::string **_name** ="X"
- std::string **_sign** =""
- double **_mass** = -1
- double **_lifetime** = -1
- PHYS::LorentzVector **_momentum**

Friends

- std::ostream & operator<< (std::ostream &, Particle &)

  Output the particle properties to the ostream.

6.8.1 Detailed Description

Particle Class.

Class representing a single particle with information on mass and lifetime

Version

0.1.0

Author

Kristian Zarebski

Date

last modified 2019-08-26

Copyright

MIT License

Definition at line 21 of file Particle.hxx.

The documentation for this class was generated from the following files:

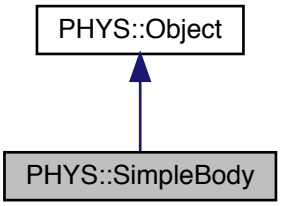
- include/Particle.hxx
- src/Particle.cxx

6.9 PHYS::SimpleBody Class Reference

Simple Body Class.

`#include <ClassicalMechanics.hxx>`

Inheritance diagram for PHYS::SimpleBody:



Public Member Functions

- SimpleBody (const std::string label, const double mass=1.∗PHYS::Units::kg, const double radius=1.∗PHY↩ S::Units::cm)

Private Attributes

- const double **_mass**
- const double **_radius**

6.9.1 Detailed Description

Simple Body Class.

Subclass of Object representing a simple object represented as a sphere in the co-ordinate system

Version

0.1.0

Author

Kristian Zarebski

Date

last modified 2019-08-27

Copyright

MIT License

Definition at line 54 of file ClassicalMechanics.hxx.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 SimpleBody()

```
PHYS::SimpleBody::SimpleBody (
            const std::string label,
            const double mass = 1.*PHYS::Units::kg,
            const double radius = 1.*PHYS::Units::cm )  [inline]
```

Construct a simple body, additional arguments are optional

Parameters

label	Name of simple body
mass	If specified, sets the mass of the body
radius	If specified, sets the radius of the sphere representing the body

Definition at line 65 of file ClassicalMechanics.hxx.

The documentation for this class was generated from the following file:

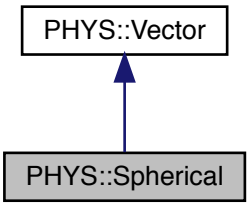
- include/ClassicalMechanics.hxx

6.10 PHYS::Spherical Class Reference

Spherical Class.

```
#include <Vector.hxx>
```

Inheritance diagram for PHYS::Spherical:



Public Member Functions

- **Spherical** (const double r, const double theta, const double phi)
- **Spherical** (const Vector &vec)
- const Cartesian inCartesian () const

6.10.1 Detailed Description

Spherical Class.

Subclass of Vector for Spherical Polar co-ordinates

Version

0.1.0

Author

Kristian Zarebski

Date

last modified 2019-08-28

Copyright

MIT License

Definition at line 75 of file Vector.hxx.

6.10.2 Member Function Documentation

6.10.2.1 inCartesian()

```
const PHYS::Cartesian PHYS::Spherical::inCartesian ( ) const
```

Converts the given co-ordinates to cartesian form

Returns

Cartesian

Definition at line 84 of file Vector.cxx.

The documentation for this class was generated from the following files:

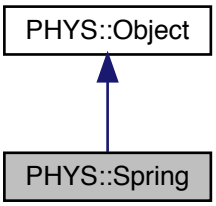
- include/Vector.hxx
- src/Vector.cxx

6.11 PHYS::Spring Class Reference

Spring Class.

`#include <ClassicalMechanics.hxx>`

Inheritance diagram for PHYS::Spring:



Public Member Functions

- Spring (const std::string label, const double spring_constant, const double length)
- const Force **forceSpring** ()
- void Attach (Object *other)
- void Update ()

  Update the Spring co-ordinates based on attached objects.

Private Attributes

- const double **_spring_constant**
- const double **_length_at_rest**
- Coordinate **_points** [2] = {{0,0,0}, {0,1,0}}
- std::vector< Object * > **_attachments**

6.11.1 Detailed Description

Spring Class.

Subclass of Object representing a spring

Version

0.1.0

Author

Kristian Zarebski

Date

last modified 2019-08-27

Copyright

MIT License

Definition at line 79 of file ClassicalMechanics.hxx.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 Spring()

```
PHYS::Spring::Spring (
            const std::string label,
            const double spring_constant,
            const double length )  [inline]
```

Construct a spring within the workspace

Parameters

label	Name of the spring
spring_constant	The spring constant k of the spring
length	The unextended spring length

Returns

Spring

Definition at line 93 of file ClassicalMechanics.hxx.

6.11.3 Member Function Documentation

6.11.3.1 Attach()

```
void PHYS::Spring::Attach (
            PHYS::Object * other )
```

Attach the spring to another object

Parameters

other	Another object of type Object

Definition at line 39 of file ClassicalMechanics.cxx.

The documentation for this class was generated from the following files:

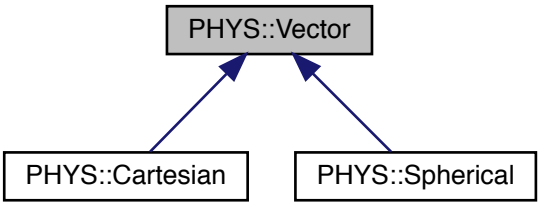
- include/ClassicalMechanics.hxx
- src/ClassicalMechanics.cxx

6.12 PHYS::Vector Class Reference

Vector Class.

`#include <Vector.hxx>`

Inheritance diagram for PHYS::Vector:



Public Member Functions

- **Vector** (const double x1, const double x2, const double x3)
- const double **operator[ ]** (const int index) const
- const Vector **operator+** (const Vector &other) const
- const Vector **operator-** (const Vector &other) const
- const bool **operator==** (const Vector &other) const
- const Vector **operator+=** (const Vector &other) const
- const Vector **operator=** (const Vector &other) const
- const Vector **operator**/ (const double &other) const

Private Attributes

- const double **_x1**
- const double **_x2**
- const double **_x3**

Friends

- const friend Vector **operator**∗ (const double &factor, const Vector &other)
- std::ostream & **operator**<< (std::ostream &out, const Vector &other)

6.12.1 Detailed Description

Vector Class.

Base class representing a vector

Version

0.1.0

Author

Kristian Zarebski

Date

last modified 2019-08-28

Copyright

MIT License

Definition at line 17 of file Vector.hxx.

The documentation for this class was generated from the following files:

- include/Vector.hxx
- src/Vector.cxx

Chapter 7

Example Documentation

7.1 KplusDecTable.hxx

An example of the Decay Table class for K+

```
#ifndef __KPLUSDECTABLE__
#define __KPLUSDECTABLE__
#include "DecayTable.hxx"
#include "CommonParticles.hxx"
const PHYS::DecayTable gen_KplusDecays();
namespace PHYS
{
    extern const DecayTable KplusDecays;
}
#endif
```

Bibliography

[1] K. A. Olive et al. Review of Particle Physics. *Chin. Phys.*, C38:090001, 2014. 10

[2] N. R. Sree Harsha. The tightly bound nuclei in the liquid drop model. *Eur. J. Phys.*, 39(3):035802, 2018. 11

Index