The LUA-PHYSICAL library

Version 1.0.4

Thomas Jenni

September 15, 2020

Abstract

`lua-physical` is a pure Lua library, which provides functions and objects for the computation of physical quantities. The package has been written, to simplify the creation physics problem sets. The package provides units of the SI and the imperial system. In order to display the numbers with measurement uncertainties, the package is able to perform gaussian error propagation.

Contents

1 Introduction

The author of this package is a physics teacher at the high school *Kantonsschule Zug*, Switzerland. The main use of this package is to write physics problem sets. It is possible to integrate physical calculations directly into LuaLaTeX. The package has been in use since 2016. Many bugs have been found and fixed. Nevertheless it still is possible, that some were not found yet. Therefore the author recommends not to use this package in industry or science. If one does so, it's the responsability of the user to check results for plausability. If the user finds some bugs, they can be reported at github.com.

1.1 Dependencies

This is a standalone library. However, it is compatible with the `siunitx` package. The results of calculations can be printed to LuaLaTeX by calling the `physical.Quantity.tosiunitx()` method. It is recommended to use a macro for this purpose. The preamble in the next section, simplifies the printing of quantities by the macros `\q{}`, `\qs{}` and `\qu{}`.

2 Loading

By calling `require("physical")` the `lua-physical` library is loaded. The following LuaLaTeX preamble loads the `lua-physical` package, does some configuration of the `siunitx` package and defines the macros `\q{}`, `\qs{}` and `\qu{}` for printing physical quantities.

Listing 1: basic preamble

```
\usepackage{luacode}
\usepackage{siunitx}

% initialize the lua-physical package
\begin{luacode*}
  physical = require("physical")
  N = physical.Number
\end{luacode*}

% configure the siunitx package
\sisetup{
  output-decimal-marker = {.},
  per-mode = symbol,
  separate-uncertainty = true,
  add-decimal-zero = true,
  exponent-product = \cdot,
  round-mode = off
}

% declare the unitless unit (siunitx package)
\DeclareSIUnit\number{}
```

```
% print a quantity using the \SI{}{} macro.
\newcommand{\q}[1]{%
  \directlua{
    tex.print(
      physical.Quantity.tosiunitx(
        #1,
        "scientific-notation=fixed,exponent-to-prefix=false"
      )
    )
  }%
}

% print a quantity in scientific notation using \SI{}{} macro.
\newcommand{\qs}[1]{%
  \directlua{
    tex.print(
      physical.Quantity.tosiunitx(
        #1,
        "scientific-notation=true,exponent-to-prefix=false,
          round-integer-to-decimal=true"
      )
    )
  }%
}

% print the unit of a quantity using the \si{} macro
\newcommand{\qu}[1]{%
  \directlua{
    tex.print(
      physical.Quantity.tosiunitx(
        #1,
        nil,
        physical.Quantity.SIUNITX_si
      )
    )
  }%
}
```

2.1 License

This code is freely distributable under the terms of the MIT license.

3 Usage

Given the basic preamble 1, units can be used in lua code directly. By convention, all units have an underscore in front of them, i.e. meter is `_m`, second is `_s`. For a complete list of all available units, see section 4. The following example illustrates the use of this library.

Listing 2: Velocity of a car.

```
\begin{luacode}
  d = 10 * _m
  t = 2 * _s
  v = d / t
\end{luacode}

A car travels $\q{d}$ in $\q{t}$. Calculate its velocity.
%
\begin{equation*}
  v=\frac{d}{t} = \frac{\q{d}}{\q{t}} = \underline{\q{v}}
\end{equation*}
```

A car travels 10 m in 2 s. Calculate its velocity.

$$v = \frac{s}{t} = \frac{10\,\mathrm{m}}{2\,\mathrm{s}} = \underline{5.0\,\mathrm{m/s}}$$

In the above listing 2, the variable `s` stands for displacement and has the unit meter `_m`, the variable `t` stands for time and is given in seconds `_s`. If physical quantities are divided or multiplied, derived quantities are created. In the example problem above, the velocity `v` has the unit `_m/_s`. By using the macro `\q{}` all quantities can be printed to the LuaLaTeX code directly.

3.1 Unit conversion

It is often the case, that the result of a calculation has to be converted to other units. Lets assume, that in the problem of listing 2, the velocity should be determined in `_km/_h`. This can be done using the `:to()` method, which is available on all quantity objects, see 3.

Listing 3: Velocity of a car in kilometers per hour.

```
\begin{luacode}
  d = 10 * _m
  t = 2 * _s
  v = (s / t):to(_km/_h)
\end{luacode}
```

```
A car travels $\q{d}$ in $\q{t}$. Calculate its velocity in $\qu{_km/_h
      }$.
%
\begin{equation*}
  v = \frac{d}{t} = \frac{\q{d}}{\q{t}} = \underline{\q{v}}
\end{equation*}
```

A car travels 10 m in 2 s. Calculate its velocity in km/h.

$$v = \frac{d}{t} = \frac{10\,\mathrm{m}}{2\,\mathrm{s}} = \underline{18.0\,\mathrm{km/h}}$$

Another example is given in listing 4. The task is to calculate the volume of a cuboid. The length of the edges are given in different units. The result of the multiplication has the unit cm mm m. If the unit cm^3 is preferred, it has to be converted explicitly. At first this looks a bit cumbersome. The reason of this behaviour is, that the software is not able to guess the unit of the result. In many cases, like in the example problem, it's not clear what unit the volume should have. Is it `_m^3`, `_cm^3` or `_L`? The user has to give that convertion explicitly.

Listing 4: Volume of a cuboid.

```
\begin{luacode}
  a = 12 * _cm
  b = 150 * _mm
  c = 1.5 * _m

  V = ( a * b * c ):to(_dm^3)
\end{luacode}

Find the volume of a rectangular cuboid with lengths $\q{a}$,
$\q{b}$ and $\q{c}$.
%
\begin{equation*}
  V= a \cdot b \cdot c
  = \q{a} \cdot \q{b} \cdot \q{c}
  = \q{V}
  = \underline{\q{V}}
\end{equation*}
```

Find the volume of a rectangular cuboid with lengths 12 cm, 150 mm and 1.5 m.

$$V = a \cdot b \cdot c = 12\,\mathrm{cm} \cdot 150\,\mathrm{mm} \cdot 1.5\,\mathrm{m} = 27.0\,\mathrm{dm}^3 = \underline{27.0\,\mathrm{dm}^3}$$

The `siunitx` package has definitions or all SI units plus some non-SI units. If a quantity has a unit, which is not defined by the `siunitx` package, it has to be declared using the `\DeclareSIUnit` macro.

Listing 5: Non-SI units.

```
% add this declaration to the preamble
\DeclareSIUnit\inch{in}


% document
\begin{luacode}
  l = 12 * _in
\end{luacode}

Convert $\q{l}$ to the unit $\qu{_cm}$.
%
\begin{equation*}
  l = \q{l} \cdot \frac{\q{_in:to(_cm)}}{\qu{_in}} = \q{l:to(_cm)}
\end{equation*}
```

Convert 12 in (inches) to the unit cm.

$$l = 12\,\mathrm{in} \cdot \frac{2.54\,\mathrm{cm}}{\mathrm{in}} = \underline{30.48\,\mathrm{cm}}$$

3.1.1 Temperature Conversion

Most physical units transform linearly. Exceptions are temperature units lie degree Celsius `_degC` and degree Fahrenheit `_degF`. These units are ambigous and can be interpreted as temperature differences or as an absolute temperatures. In the latter case, the conversion to base units is not a linear, but an affine transformation. This is because degree Celsius and degree Fahrenheit scales have their zero points at different temperatures compared to the unit Kelvin.

By default `_degC` and `_degF` units are temperature differences. If one wants to have it converted absolutely, it has to be done adding / subtracting `_degC_0 = 273.15*_K` or `_degF_0 = (273.15 - 32*(5/9)) * _K`, the zero point temperatures of the scales.

In the following problem, listing 6, the task is to convert temperatures given in the unit degree Celsius and degree Fahrenheit to Kelvin.

Listing 6: Temperature conversion.

```
\begin{luacode}
  theta_1 = 110 * _degC
  T_1 = ( theta_1 + _degC_0 ):to(_K)

  T_2 = 100 * _K
  theta_2 = ( T_2 - _degC_0 ):to(_degC)

  theta_3 = 212 * _degF
  T_3 = ( theta_3 + _degF_0 ):to(_K)
  T_4 = 100 * _K
  theta_4 = ( T_4 - _degF_0 ):to(_degF)

  theta_5 = 100 * _degC
  theta_6 = ( ( theta_5 + _degC_0 ):to(_K) - _degF_0):to(_degF)
\end{luacode}

\begin{align*}
  \q{theta_1} &\mathrel{\widehat{=}} \q{T_1} \\
  %
  \q{theta_2} &\mathrel{\widehat{=}} \q{T_2} \\
  %
  \q{theta_3} &\mathrel{\widehat{=}} \q{T_3} \\
  %
  \q{theta_4} &\mathrel{\widehat{=}} \q{T_4} \\
  %
  \q{theta_5} &\mathrel{\widehat{=}} \q{theta_6} \\
\end{align*}
```

$$\begin{aligned}
110\,{}^{\circ}\mathrm{C} &\mathrel{\widehat{=}} 383.15\,\mathrm{K} \\
-173.15\,{}^{\circ}\mathrm{C} &\mathrel{\widehat{=}} 100\,\mathrm{K} \\
212\,{}^{\circ}\mathrm{F} &\mathrel{\widehat{=}} 373.15\,\mathrm{K} \\
-279.67\,{}^{\circ}\mathrm{F} &\mathrel{\widehat{=}} 100\,\mathrm{K} \\
100\,{}^{\circ}\mathrm{C} &\mathrel{\widehat{=}} 212.0\,{}^{\circ}\mathrm{F}
\end{aligned}$$

3.2 Uncertainty Propagation

The `lua-physical` library supports uncertainty propagation. To create a number with an uncertainty, an instance of `physical.Number` has to be created. It has to be remembered, that `N` is a alias for `physical.Number`. The first argument of the constructor `N(mean, uncertainty)` is the mean value and the second one the uncertainty of the measurement.

For the uncertainty propagation the gaussian formula

$$\Delta f = \sqrt{\left(\frac{\partial f}{x_1} \cdot \Delta x_1\right)^2 + \cdots + \left(\frac{\partial f}{x_n} \cdot \Delta x_2\right)^2}$$

is used. This formula is a good estimation for the uncertainty Δf, if the quantities $x_1, \ldots, x_n$ the function f depends on, have no correlation. Further, the function f has to change linearly, if quantities x_i are changed in the range of their uncertainties.

The example in listing 7 shows the usage of `N()`. At the defintion of the distance and the speed of light, the constants are given with full precision, i.e. The distance `_au` is 149 597 870.7 km and `_c` is 299 792.458 km/s. By multipling these quantities with N(1,0.0001) the precision is reduced. The uncertainty propagation takes care of rounding the resulting time t to the correct precision. For printing, the macro \qs{} for scientific notation is used.

Listing 7: Time of flight.

```
\begin{luacode}
  d = N(1,0.0001) * ( _au ):to(_km)
  v = N(1,0.0001) * ( _c ):to(_km/_s)
  t = ( d/v ):to(_min)
\end{luacode}

Calculate the time, a lightray travels from the surface of the sun to
       the earth.
The mean distance from the sun to the eart is $\qs{d}$. The speed of
       light is $\qs{v}$.
%
\begin{equation*}
  t = \frac{d}{v} = \frac{\q{d}}{\q{v}} = \underline{\q{t}}
\end{equation*}
```

Calculate the time, a lightray travels from the surface of the sun to the earth. The mean distance from the sun to the eart is $1.496 \cdot 10^{8}\,\mathrm{km}$. The speed of light is $2.998 \cdot 10^{5}\,\mathrm{km/s}$.

$$t = \frac{d}{v} = \frac{1.496 \cdot 10^{8}\,\mathrm{km}}{2.998 \cdot 10^{5}\,\mathrm{km/s}} = \underline{8.32\,\mathrm{min}}$$

Another example is given in listing 8, the task is to find the volume of an ideal gas. Given are pressure p in _bar, amount of substance n in _mol and absolute temperature T in degree celsius _degC.

Listing 8: Volume of an ideal gas.

```
\begin{luacode}
  N.omitUncertainty = true

  p = N(1.013,0.0001) * _bar
  n = N(1,0.01) * _mol
  T = N(30,0.1) * _degC

  V = ( n * _R * (T + _degC_0):to(_K) / p ):to(_L)
\end{luacode}

An ideal gas ($\q{n}$) has a pressure of $\q{p}$ and a temperature of $\
       q{T}$. Calculate the volume of the gas.
%
\begin{equation*}
  V=\frac{ \q{n} \cdot \q{_R} \cdot \q{(T + _degC_0):to(_K)} }{ \q{p} }
  = \q{V}
  = \underline{\q{V}}
\end{equation*}
```

An ideal gas (1.0 mol) has a pressure of 1.013 bar and a temperature of 30 °C. Calculate the volume of the gas.

$$V = \frac{1.0\,\mathrm{mol} \cdot 8.31\,\mathrm{J/(mol\,K)} \cdot 303\,\mathrm{K}}{1.013\,\mathrm{bar}} = \underline{25\,\mathrm{L}}$$

This example shows, that the result has only two digits. If more digits are needed, the uncertainties of the given quantities should be made smaller.

3.2.1 Print the Uncertainty explicitly

It is possible to print the uncertainty explicitly. By default the parameter `N.omitUncertainty` is set to `true`. In listing 9 it is set to `false` and the uncertainty is printed.

Listing 9: Uncertainty in area calculation.

```
\begin{luacode}
  N.omitUncertainty = false

  a = N(2,0.1) * _m
  b = N(3,0.1) * _m

  A = ( a * b ):to(_m^2)
\end{luacode}

Calculate the area of a rectangle with lengths $\q{a}$ and $\q{b}$.
%
\begin{equation*}
  A = a \cdot b
  = \q{a} \cdot \q{b}
  = \underline{\q{A}}
\end{equation*}
```

Calculate the area of a rectangle with lengths $(2.00 \pm 0.10)\,\mathrm{m}$ and $(3.00 \pm 0.10)\,\mathrm{m}$.

$$A = a \cdot b = (2.00 \pm 0.10)\,\mathrm{m} \cdot (3.00 \pm 0.10)\,\mathrm{m} = \underline{(6.0 \pm 0.4)\,\mathrm{m}^2}$$

3.3 Mathematical operations

Two physical quantities with identical dimensions can be added or subtracted. The library checks the validity of those operations and throws an error if two addends haven't the same dimensions.

Listing 10: Addition and subtraction

```
l_1 = 1 * _m
l_2 = 2 * _cm
t = 2 * _s

l_1 + t
Error: Cannot add '1* _m' to '2 * _s', because they have different
       dimensions.

l_1 + l_2
102.0 * _cm
```

New physical quantities can be created by division and multiplication. As long as no division by zero is made, no errors should occur.

Listing 11: Multiplication and Division

```
l_1 = 1 * _m
l_2 = 2 * _cm

(l_1 * l_2):to(_m^2)
0.02 * _m^2

(l_1 / l_2):to(_1)
50.0 * _1
```

Physical quantities can be exponentiated. The library doesn't check, if the result has units with non integer exponents.

Listing 12: Exponentiation

```
l = 5 * _m
A = l^2

A:to(_m^2)
25.0 * _m^2

A:sqrt()
5.0 * _m

A^0.5
5.0 * _m
```

Exponential functions an the logarithms should have dimensionless arguments. The library throws an error if that's not the case.

Listing 13: Exponential function and logarithm

```
N_0 = 1000 * _1
lambda = Q.log(2)/(2*_h)
t = 50 * _min

N_0 * Q.exp(-lambda * t)
749.15353843834 * _1
```

4 Supported Units

All supported units are listed in this section. Subsection 4.2 lists the seven base units of the International System of Units (SI). In subsection 4.3 mathematical and physical constants are defined. The subsection 4.4 contains all coherent derived units from the SI system and 4.5 those which are accepted to use with the SI. The subsection 4.6 lists nominal astronomical units, which are proposed by [4]. Subsection 4.7 lists units, which are common but outside of the SI system. The subsections 4.8 and 4.9 are dedicated to imperial and U.S. customary units. The last subsection 4.10 containts international currencies.

4.1 Prefixes

All SI units have prefixed versions, i.e. `_us` microsecond, `_cm` centimeter, `_mN` millinewton, see table 1. Some units of data processing, like `_bit` have prefixes which are powers of 2. They are called binary or IEC prefixes, see table 2 [2, 121].

Prefix	Symbol	Definition	Prefix	Symbol	Definition
yotta	`Y`	`1e24`	deci	`d`	`1e-1`
zetta	`Z`	`1e21`	centi	`c`	`1e-2`
exa	`E`	`1e18`	milli	`m`	`1e-3`
peta	`P`	`1e15`	micro	`u`	`1e-6`
tera	`T`	`1e12`	nano	`n`	`1e-9`
giga	`G`	`1e9`	pico	`p`	`1e-12`
mega	`M`	`1e6`	femto	`f`	`1e-15`
kilo	`k`	`1e3`	atto	`a`	`1e-18`
hecto	`h`	`1e2`	zepto	`z`	`1e-21`
deca	`da`	`1e1`	yocto	`y`	`1e-23`

Table 1: SI prefixes [2, 121]

Prefix	Symbol	Definition
kibi	`Ki`	`1024`
mebi	`Mi`	`1048576`
gibi	`Gi`	`1073741824`
tebi	`Ti`	`1099511627776`
pebi	`Pi`	`1125899906842624`
exbi	`Ei`	`1152921504606846976`
zebi	`Zi`	`1180591620717411303424`
yobi	`Yi`	`1208925819614629174706176`

Table 2: IEC prefixes [2, 121]

4.2 Base Units

The `lua-physical` library has nine base quantities. These are the seven basis units or basis quantities of the SI system [3] and in addition the base quantity of information `_bit` and of currency `_EUR`. All other quantities are derived from these base units.

Quantity	Unit	Symbol	Dim.	Definition
number [1]	–	`_1`	1	The dimensionless number one.
time	second	`_s`	T	The SI unit of time. It is defined by taking the fixed numerical value of the caesium frequency $\Delta\nu_{Cs}$, the unperturbed ground-state hyperfine transition frequency of the caesium 133 atom, to be 9 192 631 770 when expressed in the unit 1/s.
length	meter	`_m`	L	The SI unit of length. It is defined by taking the fixed numercial value of the speed of light in vacuum c to be 299 792 458 when expressed in the unit of m/s.

[1] The number one is a unit with dimension zero. Stricly speaking it is not a base unit.

Quantity	Unit	Symbol	Dim.	Definition
mass	kilogram	`_kg`	M	The SI unit of mass. It is defined by taking the fixed numerical value of the Planck constant h to be $6.626\,070\,15 \cdot 10^{-34}$ when expressed in $\mathrm{m}^2\,\mathrm{kg/s}$.
electric current	ampere	`_A`	I	The SI unit of electric current. It is defined by taking the fixed numerical value of the elementary charge e to be $1.602\,176\,634 \cdot 10^{-19}$ when expressed in A s.
thermodynamic temperature	kelvin	`_K`	K [1]	The SI unit of the thermodynamic temperature. It is defineed by taking the fixed numerical value of the Boltzmann constant k_B to be $1.380\,649 \cdot 10^{-23}$ when expressed in $\mathrm{kg\,m}^2/(\mathrm{s}^2\,\mathrm{K})$
amount of substance	mole	`_mol`	N	The SI unit of amount of substance. One mole contains exactly $6.022\,140\,76 \cdot 10^{23}$ elementary entities. This number is the fixed numerical value of the Avogadro constant N_A when expressed in 1/mol.
luminous intensity	candela	`_cd`	J	The SI unit of luminous intensity in a given direction. It is defined by taking the fixed numerical value of the luminous efficacy of monochromatic radiation of frequency $5.4 \cdot 10^{14}\,\mathrm{Hz}$, K_{cd}, to be 683 when expressed in the unit $\mathrm{cd\,sr\,s}^3/(\mathrm{kg\,m}^2)$.
information	bit	`_bit`	B	The smallest amount of information.
currency	euro	`_EUR`	C	The value of the currency Euro.

Table 3: Base units

[1] The SI symbol for the dimension of temperature is Θ, but all symbols of this library consist of roman letters, numbers and underscores only. Therefore the symbol for the dimension of the thermodynamic temperature is the letter K.

4.3 Constants

All physical constants are taken from the NIST webpage [1].

Name	Symbol	Definition
pi	`_Pi`	`3.1415926535897932384626433832795028841971 * _1`
eulersnumber	`_E`	`2.7182818284590452353602874713526624977572 * _1`
speedoflight	`_c`	`299792458 * _m/_s`
gravitationalconstant	`_Gc`	`N(6.67408e-11,3.1e-15) * _m^3/(_kg*_s^2)`
planckconstant	`_h_P`	`6.62607015e-34 * _J*_s`
reducedplanckconstant	`_h_Pbar`	`_h_P/(2*_Pi)`
elementarycharge	`_e`	`1.602176634e-19 * _C`
vacuumpermeability	`_u_0`	`4e-7*Pi * _N/_A^2`
vacuumpermitivity	`_e_0`	`1/(_u_0*_c^2)`
atomicmassunit	`_u`	`N(1.66053904e-27, 2e-35) * _kg`
electronmass	`_m_e`	`N(9.10938356e-31, 1.1e-38) * _kg`
protonmass	`_m_p`	`N(1.672621898e-27, 2.1e-35) * _kg`
neutronmass	`_m_n`	`N(1.674927471e-27, 2.1e-35) * _kg`
bohrmagneton	`_u_B`	`_e*_h_Pbar/(2*_m_e)`
nuclearmagneton	`_u_N`	`_e*_h_Pbar/(2*_m_p)`
electronmagneticmoment	`_u_e`	`N(-928.4764620e-26,5.7e-32) * _J/_T`
protonmagneticmoment	`_u_p`	`N(1.4106067873e-26,9.7e-35) * _J/_T`
neutronmagneticmoment	`_u_n`	`N(-0.96623650e-26,2.3e-26) * _J/_T`
finestructureconstant	`_alpha`	`_u_0*_e^2*_c/(2*_h_P)`
rydbergconstant	`_Ry`	`_alpha^2*_m_e*_c/(2*_h_P)`
avogadronumber	`_N_A`	`6.02214076e23/_mol`
boltzmannconstant	`_k_B`	`1.380649e-23 * _J/_K`
molargasconstant	`_R`	`N(8.3144598, 4.8e-6) * _J/(_K*_mol)`
stefanboltzmannconstant	`_sigma`	`_Pi^2*_k_B^4/(60*_h_Pbar^3*_c^2)`
standardgravity	`_g_0`	`9.80665 * _m/_s^2`

Table 4: Physical and mathematical constants

4.4 Coherent derived units in the SI

All units in this section are coherent derived units from the SI base units with special names, [2, 118].

Quantity	Unit	Symbol	Definition
Plane Angle[1]	radian	`_rad`	`_1`
Solid Angle[2]	steradian	`_sr`	`_rad^2`
Frequency	hertz	`_Hz`	`1/_s`
Force	newton	`_N`	`_kg*_m/_s^2`
Pressure	pascal	`_Pa`	`_N/_m^2`
Energy	joule	`_J`	`_N*_m`
Power	watt	`_W`	`_J/_s`
Electric Charge	coulomb	`_C`	`_A*_s`
Electric Potential	volt	`_V`	`_J/_C`
Electric Capacitance	farad	`_F`	`_C/_V`
Electric Resistance	ohm	`_Ohm`	`_V/_A`
Electric Conductance[3]	siemens	`_S`	`_A/_V`
Magnetic Flux	weber	`_Wb`	`_V*_s`
Magnetic Flux Density	tesla	`_T`	`_Wb/_m^2`
Inductance	henry	`_H`	`_Wb/_A`
Temperature[4]	celsius	`_degC`	`_K`
Luminous Flux	lumen	`_lm`	`_cd*_sr`
Illuminance	lux	`_lx`	`_lm/_m^2`
Activity	becquerel	`_Bq`	`1/_s`
Absorbed Dose	gray	`_Gy`	`_J/_kg`
Dose Equivalent	sievert	`_Sv`	`_J/_kg`
Catalytic Activity	katal	`_kat`	`_mol/_s`

[1] In the SI system, the quantity Plane Angle has the dimension of a number.

[2] In the SI system, the quantity Solid Angle has the dimension of a number.

[3] The unit `_PS` stands for peta siemens and is in conflict with the metric version of the unit horsepower (german Pferdestärke). Since the latter is more common than peta siemens, `_PS` is defined to be the metric version of horsepower.

[4] The unit `_degC` is by default interpreted as a temperature difference.

4.5 Non-SI units accepted for use with the SI

There are a few units with dimension 1. [2, 124].

Quantity	Unit	Symbol	Definition
Time	minute	`_min`	`60 * _s`
	hour	`_h`	`60 * _min`
	day	`_d`	`24 * _h`
Plane Angle	degree	`_deg`	`(_Pi/180) * _rad`
	arcminute	`_arcmin`	`_deg/60`
	arcsecond	`_arcsec`	`_arcmin/60`
Area	hectare	`_hectare`	`1e4 * _m^2`
Volume	liter	`_L`	`1e-3 * _m^3`
Mass	tonne	`_t`	`1e3 * _kg`

4.6 Nominal Astronomical Units

The nominal values of solar, terrestrial and jovial quantities are taken from IAU Resolution B3 [4].

Quantity	Unit	Symbol	Definition
Length	nomsolradius	`_R_S_nom`	`6.957e8 * _m`
Irradiance	nomsolirradiance	`_S_S_nom`	`1361 * _W/_m^2`
Radiant Flux	nomsolluminosity	`_L_S_nom`	`3.828e26 * _W`
Temperature	nomsolefftemperature	`_T_S_nom`	`5772 * _K`
Mass Parameter	nomsolmassparameter	`_GM_S_nom`	`1.3271244e20 * _m^3*_s^-2`
Length	nomterreqradius	`_Re_E_nom`	`6.3781e6 * _m`
Length	nomterrpolradius	`_Rp_E_nom`	`6.3568e6 * _m`
Mass Parameter	nomterrmassparameter	`_GM_E_nom`	`3.986004e14 * _m^3*_s^-2`
Length	nomjoveqradius	`_Re_J_nom`	`7.1492e7 * _m`
Length	nomjovpolradius	`_Rp_J_nom`	`6.6854e7 * _m`
Mass Parameter	nomjovmassparameter	`_GM_J_nom`	`1.2668653e17 * _m^3*_s^-2`

4.7 Other Non-SI units

The unit Bel is only available with prefix decibel, because `_B` is the unit byte.

Quantity	Unit	Symbol	Definition
Length	angstrom	`_angstrom`	`1e-10 * _m`
	fermi	`_fermi`	`1e-15 * _m`
Time	svedberg	`_svedberg`	`1e-13 * _s`
	week	`_wk`	`7 * _d`
	year	`_a`	`365.25 * _d`
	astronomicalunit	`_au`	`149597870700 * _m`
	lightsecond	`_ls`	`_c*_s`
	lightyear	`_ly`	`_c*_a`
	parsec	`_pc`	`(648000/_Pi) * _au`
Area	barn	`_barn`	`1e-28 * _m^2`
	are	`_are`	`1e2 * _m^2`
Volume	metricteaspoon	`_tsp`	`5e-3 * _L`
	metrictablespoon	`_Tbsp`	`3 * _tsp`
Plane Angle	gradian	`_gon`	`(Pi/200) * _rad`
	turn	`_tr`	`2*Pi * _rad`
Solid Angle	spat	`_sp`	`4*Pi * _sr`
Force	kilopond	`_kp`	`_kg*_g_0`
Pressure	bar	`_bar`	`1e5 * _Pa`
	standardatmosphere	`_atm`	`101325 * _Pa`
	technicalatmosphere	`_at`	`_kp/_cm^2`
	millimeterofmercury	`_mmHg`	`133.322387415 * _Pa`
	torr	`_Torr`	`(101325/760) * _Pa`
Energy	thermochemicalcalorie	`_cal`	`4.184 * _J`
	internationalcalorie	`_cal_IT`	`4.1868 * _J`
	gramoftnt	`_g_TNT`	`1e3 * _cal`
	tonoftnt	`_t_TNT`	`1e9 * _cal`

Quantity	Unit	Symbol	Definition
	electronvolt	_eV	_e*_V
	wattsecond	_Ws	_W*_s
	watthour	_Wh	_W*_h
Power	voltampere	_VA	_V*_A
Electric Charge	amperesecond	_As	_A*_s
	amperehour	_Ah	_A*_h
Information	nibble	_nibble	4 * _bit
	byte	_B	8 * _bit
Information Transfer Rate	bitpersecond	_bps	_bit/_s
Number	percent	_percent	1e-2 * _1
	permille	_permille	1e-3 * _1
	partspermillion	_ppm	1e-6 * _1
	partsperbillion	_ppb	1e-9 * _1
	partspertrillion	_ppt	1e-12 * _1
	partsperquadrillion	_ppq	1e-15 * _1
	decibel	_dB	_1
Power	metrichorsepower	_PS	75 * _g_0*_kg*_m/_s
Activity	curie	_Ci	3.7e10 * _Bq
Absorbed Dose	rad	_Rad	1e-2 * _Gy
Dose Equivalent	rem	_rem	1e-2 * _Sv
Viscosity	poiseuille	_Pl	_Pa*_s

4.8 Imperial Units

Quantity	Unit	Symbol	Definition
Length	inch	_in	2.54e-2 * _m
	thou	_th	1e-3 * _in
DTP Point[1]	point	_pt	_in/72
	pica	_pica	12 * _pt
	hand	_hh	4 * _in
	foot	_ft	12 * _in
	yard	_yd	3 * _ft
	rod	_rd	5.5 * _yd
	chain	_ch	4 * _rd
	furlong	_fur	10 * _ch
	mile	_mi	8 * _fur
	league	_lea	3*_mi
	nauticalmile	_nmi	1852 * _m
	nauticalleague	_nlea	3 * _nmi
	cable	_cbl	0.1 * _nmi
	fathom	_ftm	6 * _ft
Velocity	knot	_kn	_nmi/_h
Area	acre	_ac	10 * _ch^2
Volume	gallon	_gal	4.54609*_L
	quart	_qt	_gal/4
	pint	_pint	_qt/2
	cup	_cup	_pint/2
	gill	_gi	_pint/4
	fluidounce	_fl_oz	_gi/5
	fluiddram	_fl_dr	_fl_oz/8

[1] The desktop publishing point or PostScript point is 1/72 of an international inch.

Quantity	Unit	Symbol	Definition
Mass	grain	_gr	64.79891*_mg
	pound	_lb	7000*_gr
	ounce	_oz	_lb/16
	dram	_dr	_lb/256
	stone	_st	14*_lb
	quarter	_qtr	2*_st
	hundredweight	_cwt	4*_qtr
	longton	_ton	20*_cwt
	troypound	_lb_t	5760*_gr
	troyounce	_oz_t	_lb_t/12
	pennyweight	_dwt	24*_gr
	firkin	_fir	56*_lb
Time	sennight	_sen	7*_d
	fortnight	_ftn	14*_d
Temperature[1]	fahrenheit	_degF	(5/9)*_K
Force	poundforce	_lbf	_lb*_g_0
	poundal	_pdl	_lb*_ft/_s^2
Mass	slug	_slug	_lbf*_s^2/_ft
Pressure	poundforcepersquareinch	_psi	_lbf/_in^2
Torque,Energy	thchembritishthermalunit	_BTU	(1897.83047608/1.8)*_J
Torque,Energy	intbritishthermalunit	_BTU_it	1055.05585262 * _J
Power	horsepower	_hp	33000*_ft*_lbf/_min

[1] The unit _degF is by default interpreted as a temperature difference.

4.9 U.S. customary units

In the U.S., the length units are bound to the meter differently than in the imperial system. The followin definitions are taken from `https://en.wikipedia.org/wiki/United_States_customary_units`.

Quantity	Unit	Symbol	Definition
Length	ussurveyinch	`_in_US`	`_m/39.37`
	ussurveyhand	`_hh_US`	`4 * _in_US`
	ussurveyfoot	`_ft_US`	`3 * _hh_US`
	ussurveylink	`_li_US`	`0.66 * _ft_US`
	ussurveyyard	`_yd_US`	`3 * _ft_US`
	ussurveyrod	`_rd_US`	`5.5 * _yd_US`
	ussurveychain	`_ch_US`	`4 * _rd_US`
	ussurveyfurlong	`_fur_US`	`10 * _ch_US`
	ussurveymile	`_mi_US`	`8 * _fur_US`
	ussurveyleague	`_lea_US`	`3 * _mi_US`
	ussurveyfathom	`_ftm_US`	`72 * _in_US`
	ussurveycable	`_cbl_US`	`120 * _ftm_US`
Area	ussurveyacre	`_ac_US`	`_ch_US * _fur_US`
Volume	usgallon	`_gal_US`	`231 * _in^3`
	usquart	`_qt_US`	`_gal_US/4`
	uspint	`_pint_US`	`_qt_US/2`
	uscup	`_cup_US`	`_pint_US/2`
	usgill	`_gi_US`	`_pint_US/4`
	usfluidounce	`_fl_oz_US`	`_gi_US/4`
	ustablespoon	`_Tbsp_US`	`_fl_oz_US/2`
	usteaspoon	`_tsp_US`	`_Tbsp_US/3`
	usfluiddram	`_fl_dr_US`	`_fl_oz_US/8`
Mass	usquarter	`_qtr_US`	`25 * _lb`
	ushundredweight	`_cwt_US`	`4 * _qtr_US`
	uston	`_ton_US`	`20 * _cwt_US`

4.10 International Currencies

International currency units based on exchange rates from 7.3.2019, 21:00 UTC.

Name	Symbol	Definition
AfghanAfghani	_AFN	0.012 * _EUR
AfghanPul	_cAFN	0.01 * _AFN
AlbanianLek	_ALL	0.008 * _EUR
ArmenianDram	_AMD	0.0018 * _EUR
ArmenianLuma	_cAMD	0.01 * _AMD
AngolanKwanza	_AOA	0.0028 * _EUR
AngolanCentimo	_cAOA	0.01 * _AOA
ArgentinePeso	_ARS	0.021 * _EUR
ArgentineCentavo	_cARS	0.01 * _ARS
AustralianDollar	_AUD	0.63 * _EUR
AustralianCent	_cAUD	0.01 * _AUD
AzerbaijaniManat	_AZN	0.63 * _EUR
AzerbaijaniQepik	_cAZN	0.01 * _AZN
BosnianMark	_BAM	0.51 * _EUR
BosnianFenings	_cBAM	0.01 * _BAM
BangladeshiTaka	_BDT	0.011 * _EUR
BangladeshiPoisha	_cBDT	0.01 * _BDT
BurundianFranc	_BIF	0.00049 * _EUR
BurundianCentime	_cBIF	0.01 * _BIF
BolivianBoliviano	_BOB	0.13 * _EUR
BolivianCentavo	_cBOB	0.01 * _BOB
BrazilianReal	_BRL	0.23 * _EUR
BrazilianCentavo	_cBRL	0.01 * _BRL
BotswanaPula	_BWP	0.083 * _EUR
BotswanaThebe	_cBWP	0.01 * _BWP
BelarusianRuble	_BYN	0.42 * _EUR
BelarusianKapiejka	_cBYN	0.01 * _BYN

Name	Symbol	Definition
CanadianDollar	_CAD	0.66 * _EUR
CanadianCent	_cCAD	0.01 * _CAD
CongoleseFranc	_CDF	0.00055 * _EUR
CongoleseCentime	_cCDF	0.01 * _CDF
SwissFranc	_CHF	0.88 * _EUR
SwissRappen	_cCHF	0.01 * _CHF
ChileanPeso	_CLP	0.0013 * _EUR
ChileanCentavo	_cCLP	0.01 * _CLP
ChineseRenminbiYuan	_CNY	0.13 * _EUR
ChineseRenminbiFen	_cCNY	0.01 * _CNY
ColombianPeso	_COP	0.00028 * _EUR
ColombianCentavo	_cCOP	0.01 * _COP
CostaRicanColon	_CRC	0.0015 * _EUR
CostaRicanCentimos	_cCRC	0.01 * _CRC
CzechKoruna	_CZK	0.039 * _EUR
CzechHaler	_cCZK	0.01 * _CZK
DanishKrone	_DKK	0.13 * _EUR
DanishOre	_cDKK	0.01 * _DKK
DominicanPeso	_DOP	0.018 * _EUR
DominicanCentavo	_cDOP	0.01 * _DOP
AlgerianDinar	_DZD	0.0074 * _EUR
AlgerianSanteem	_cDZD	0.01 * _DZD
EgyptianPound	_EGP	0.051 * _EUR
EgyptianPiastre	_cEGP	0.01 * _EGP
EthiopianBirr	_ETB	0.031 * _EUR
EthiopianSantim	_cETB	0.01 * _ETB
FijianDollar	_FJD	0.42 * _EUR
FijianCent	_cFJD	0.01 * _FJD

Name	Symbol	Definition
PoundSterling	_GBP	1.16 * _EUR
PennySterling	_cGBP	0.01 * _GBP
GeorgianLari	_GEL	0.33 * _EUR
GeorgianTetri	_cGEL	0.01 * _GEL
GhanaianCedi	_GHS	0.16 * _EUR
GhanaianPesewa	_cGHS	0.01 * _GHS
GambianDalasi	_GMD	0.018 * _EUR
GambianButut	_cGMD	0.01 * _GMD
GuineanFranc	_GNF	9.6e-05 * _EUR
GuineanCentime	_cGNF	0.01 * _GNF
GuatemalanQuetzal	_GTQ	0.12 * _EUR
GuatemalanCentavo	_cGTQ	0.01 * _GTQ
GuyaneseDollar	_GYD	0.0043 * _EUR
GuyaneseCent	_cGYD	0.01 * _GYD
HongKongDollar	_HKD	0.11 * _EUR
HongKongCent	_cHKD	0.01 * _HKD
HonduranLempira	_HNL	0.036 * _EUR
HonduranCentavo	_cHNL	0.01 * _HNL
CroatianKuna	_HRK	0.13 * _EUR
CroatianLipa	_cHRK	0.01 * _HRK
HaitianGourde	_HTG	0.011 * _EUR
HaitianCentime	_cHTG	0.01 * _HTG
HungarianForint	_HUF	0.0032 * _EUR
HungarianFiller	_cHUF	0.01 * _HUF
IndonesianRupiah	_IDR	6.2e-05 * _EUR
IndonesianSen	_cIDR	0.01 * _IDR
IsraeliNewShekel	_ILS	0.25 * _EUR
IsraeliNewAgora	_cILS	0.01 * _ILS

Name	Symbol	Definition
IndianRupee	`_INR`	`0.013 * _EUR`
IndianPaisa	`_cINR`	`0.01 * _INR`
IraqiDinar	`_IQD`	`0.00074 * _EUR`
IraqiFils	`_cIQD`	`0.001 * _IQD`
IranianRial	`_IRR`	`2.7e-05 * _EUR`
IranianToman	`_cIRR`	`10.0 * _IRR`
IcelandicKrona	`_ISK`	`0.0073 * _EUR`
JamaicanDollar	`_JMD`	`0.007 * _EUR`
JamaicanCent	`_cJMD`	`0.01 * _JMD`
JapaneseYen	`_JPY`	`0.008 * _EUR`
KenyanShilling	`_KES`	`0.0089 * _EUR`
KenyanCent	`_cKES`	`0.01 * _KES`
KyrgyzstaniSom	`_KGS`	`0.013 * _EUR`
KyrgyzstaniTyiyn	`_cKGS`	`0.01 * _KGS`
CambodianRiel	`_KHR`	`0.00022 * _EUR`
NorthKoreanWon	`_KPW`	`0.00099 * _EUR`
NorthKoreanChon	`_cKPW`	`0.01 * _KPW`
SouthKoreanWon	`_KRW`	`0.00078 * _EUR`
SouthKoreanJeon	`_cKRW`	`0.01 * _KRW`
KuwaitiDinar	`_KWD`	`2.93 * _EUR`
KuwaitiFils	`_cKWD`	`0.001 * _KWD`
KazakhstaniTenge	`_KZT`	`0.0023 * _EUR`
KazakhstaniTiyn	`_cKZT`	`0.01 * _KZT`
LaoKip	`_LAK`	`0.0001 * _EUR`
LaoAtt	`_cLAK`	`0.01 * _LAK`
SriLankanRupee	`_LKR`	`0.005 * _EUR`
SriLankanCent	`_cLKR`	`0.01 * _LKR`
LiberianDollar	`_LRD`	`0.0055 * _EUR`
LiberianCent	`_cLRD`	`0.01 * _LRD`

Name	Symbol	Definition
LibyanDinar	_LYD	0.64 * _EUR
LibyanDirham	_cLYD	0.001 * _LYD
MoroccanDirham	_MAD	0.092 * _EUR
MoroccanSantim	_cMAD	0.01 * _MAD
MoldovanLeu	_MDL	0.052 * _EUR
MoldovanBan	_cMDL	0.01 * _MDL
MalagasyAriary	_MGA	0.00025 * _EUR
MalagasyIraimbilanja	_cMGA	0.2 * _MGA
MacedonianDenar	_MKD	0.016 * _EUR
MacedonianDeni	_cMKD	0.01 * _MKD
BurmeseKyat	_MMK	0.00059 * _EUR
BurmesePya	_cMMK	0.01 * _MMK
MongolianTogrog	_MNT	0.00034 * _EUR
MongolianMongo	_cMNT	0.01 * _MNT
MauritanianOuguiya	_MRU	0.025 * _EUR
MauritanianKhoums	_cMRU	0.2 * _MRU
MauritianRupee	_MUR	0.025 * _EUR
MauritianCent	_cMUR	0.01 * _MUR
MaldivianRufiyaa	_MVR	0.058 * _EUR
MaldivianLaari	_cMVR	0.01 * _MVR
MalawianKwacha	_MWK	0.0012 * _EUR
MalawianTambala	_cMWK	0.01 * _MWK
MexicanPeso	_MXN	0.046 * _EUR
MexicanCentavo	_cMXN	0.01 * _MXN
MalaysianRinggit	_MYR	0.22 * _EUR
MalaysianSen	_cMYR	0.01 * _MYR
MozambicanMetical	_MZN	0.014 * _EUR
MozambicanCentavo	_cMZN	0.01 * _MZN

Name	Symbol	Definition
NigerianNaira	_NGN	0.0025 * _EUR
NigerianKobo	_cNGN	0.01 * _NGN
NicaraguanCordoba	_NIO	0.027 * _EUR
NicaraguanCentavo	_cNIO	0.01 * _NIO
NorwegianKrone	_NOK	0.1 * _EUR
NorwegianOre	_cNOK	0.01 * _NOK
NewZealandDollar	_NZD	0.61 * _EUR
NewZealandCent	_cNZD	0.01 * _NZD
PeruvianSol	_PEN	0.27 * _EUR
PeruvianCentimo	_cPEN	0.01 * _PEN
PapuaNewGuineanKina	_PGK	0.26 * _EUR
PapuaNewGuineanToea	_cPGK	0.01 * _PGK
PhilippinePeso	_PHP	0.017 * _EUR
PhilippineSentimo	_cPHP	0.01 * _PHP
PakistaniRupee	_PKR	0.0064 * _EUR
PakistaniPaisa	_cPKR	0.01 * _PKR
PolishZloty	_PLN	0.23 * _EUR
PolishGrosz	_cPLN	0.01 * _PLN
ParaguayanGuarani	_PYG	0.00015 * _EUR
ParaguayanCentimo	_cPYG	0.01 * _PYG
QatariRiyal	_QAR	0.24 * _EUR
QatariDirham	_cQAR	0.01 * _QAR
RomanianLeu	_RON	0.21 * _EUR
RomanianBan	_cRON	0.01 * _RON
SerbianDinar	_RSD	0.0085 * _EUR
SerbianPara	_cRSD	0.01 * _RSD
RussianRuble	_RUB	0.013 * _EUR
RussianKopeyka	_cRUB	0.01 * _RUB

Name	Symbol	Definition
RwandanFranc	_RWF	0.00098 * _EUR
RwandanCentime	_cRWF	0.01 * _RWF
SolomonIslandsDollar	_SBD	0.11 * _EUR
SolomonIslandsCent	_cSBD	0.01 * _SBD
SeychelloisRupee	_SCR	0.065 * _EUR
SeychelloisCent	_cSCR	0.01 * _SCR
SudanesePound	_SDG	0.019 * _EUR
SudaneseQirsh	_cSDG	0.01 * _SDG
SwedishKrona	_SEK	0.094 * _EUR
SwedishOre	_cSEK	0.01 * _SEK
SingaporeDollar	_SGD	0.65 * _EUR
SingaporeCent	_cSGD	0.01 * _SGD
SierraLeoneanLeone	_SLL	0.0001 * _EUR
SierraLeoneanCent	_cSLL	0.01 * _SLL
SomalilandShilling	_SQS	0.00013 * _EUR
SomalilandCent	_cSQS	0.01 * _SQS
SomaliShilling	_SOS	0.0015 * _EUR
SomaliSenti	_cSOS	0.01 * _SOS
SurinameseDollar	_SRD	0.12 * _EUR
SurinameseCent	_cSRD	0.01 * _SRD
SyrianPound	_SYP	0.0017 * _EUR
SyrianPiastre	_cSYP	0.01 * _SYP
ThaiBaht	_THB	0.028 * _EUR
ThaiSatang	_cTHB	0.01 * _THB
TajikistaniSamani	_TJS	0.094 * _EUR
TajikistaniDiram	_cTJS	0.01 * _TJS
Tonganpaanga	_TOP	0.397 * _EUR
TonganSeniti	_cTOP	0.01 * _TOP

Name	Symbol	Definition
TurkishLira	_TRY	0.16 * _EUR
TurkishKurus	_cTRY	0.01 * _TRY
TrinidadAndTobagoDollar	_TTD	0.13 * _EUR
TrinidadAndTobagoCent	_cTTD	0.01 * _TTD
NewTaiwanDollar	_TWD	0.029 * _EUR
NewTaiwanCent	_cTWD	0.01 * _TWD
TanzanianShilling	_TZS	0.00038 * _EUR
TanzanianSenti	_cTZS	0.01 * _TZS
UkrainianHryvnia	_UAH	0.00038 * _EUR
UkrainianKopiyka	_cUAH	0.01 * _UAH
UgandanShilling	_UGX	0.00024 * _EUR
UgandanCent	_cUGX	0.01 * _UGX
USDollar	_USD	0.89 * _EUR
USCent	_cUSD	0.01 * _USD
UruguayanPeso	_UYU	0.027 * _EUR
UruguayanCentesimo	_cUYU	0.01 * _UYU
UzbekistaniSom	_UZS	0.00011 * _EUR
UzbekistaniTiyin	_cUZS	0.01 * _UZS
VenezuelanBolivarSoberano	_VES	0.0003 * _EUR
VenezuelanCentimoSoberano	_cVES	0.01 * _VES
VietnameseDong	_VND	3.8e-05 * _EUR
VietnameseXu	_cVND	0.01 * _VND
SamoanTala	_WST	0.34 * _EUR
SamoanSene	_cWST	0.01 * _WST
YemeniRial	_YER	0.0036 * _EUR
YemeniDinar	_cYER	0.01 * _YER
SouthAfricanRand	_ZAR	0.062 * _EUR
SouthAfricanCent	_cZAR	0.01 * _ZAR
ZambianKwacha	_ZMW	0.074 * _EUR
ZambianNgwee	_cZMW	0.01 * _ZMW

4.10.1 Pegged International Currencies

International currency which are pegged to other currencies.

Name	Symbol	Definition
UnitedArabEmiratesDirham	_AED	(1/3.6725) * _USD
UnitedArabEmiratesFils	_cAED	0.01 * _AED
NetherlandsAntilleanGuilder	_ANG	(1/1.79) * _USD
NetherlandsAntilleanCent	_cANG	0.01 * _ANG
ArubanFlorin	_AWG	(1/1.79) * _USD
ArubanCent	_cAWG	0.01 * _AWG
BarbadianDollar	_BBD	0.5 * _USD
BarbadianCent	_cBBD	0.01 * _BBD
BulgarianLev	_BGN	0.51129 * _EUR
BulgarianStotinka	_cBGN	0.01 * _BGN
BahrainiDinar	_BHD	(1/0.376) * _USD
BahrainiFils	_cBHD	0.001 * _BHD
BermudianDollar	_BMD	1 * _USD
BermudianCent	_cBMD	0.01 * _BMD
BruneiDollar	_BND	1 * _SGD
BruneiSen	_cBND	0.01 * _BND
BahamianDollar	_BSD	1 * _USD
BahamianCent	_cBSD	0.01 * _BSD
BhutaneseNgultrum	_BTN	1 * _INR
BhutaneseChhertum	_cBTN	0.01 * _BTN
BelizeDollar	_BZD	0.5 * _USD
BelizeCent	_cBZD	0.01 * _BZD
CubanoConvertiblePeso	_CUC	1 * _USD
CubanoConvertibleCentavo	_cCUC	0.01 * _CUC
CubanPeso	_CUP	(1/24) * _CUC
CubanCentavo	_cCUP	0.01 * _CUP

Name	Symbol	Definition
CapeVerdeanEscudo	_CVE	(1/110.265) * _EUR
CapeVerdeanCentavo	_cCVE	0.01 * _CVE
DjiboutianFranc	_DJF	(1/177.721) * _USD
DjiboutianCentime	_cDJF	0.01 * _DJF
EritreanNakfa	_ERN	(1/15) * _USD
EritreanCent	_cERN	0.01 * _ERN
FalklandIslandsPound	_FKP	1 * _GBP
FalklandIslandsPenny	_cFKP	0.01 * _FKP
GuernseyPound	_GGP	1 * _GBP
GuernseyPenny	_cGGP	0.01 * _GGP
GibraltarPound	_GIP	1 * _GBP
GibraltarPenny	_cGIP	0.01 * _GIP
ManxPound	_IMP	1 * _GBP
ManxPenny	_cIMP	0.01 * _IMP
JerseyPound	_JEP	1 * _GBP
JerseyPenny	_cJEP	0.01 * _JEP
JordanianDinar	_JOD	(1/0.708) * _USD
JordanianFils	_cJOD	0.001 * _JOD
KiribatiDollar	_KID	1 * _AUD
KiribatiCent	_cKID	0.01 * _KID
Comorianfranc	_KMF	(1/491.96775) * _EUR
ComorianCentime	_cKMF	0.01 * _KMF
CaymanIslandsDollar	_KYD	1.2 * _USD
CaymanIslandsCent	_cKYD	0.01 * _KYD
LebanesePound	_LBP	(1/1507.5) * _USD
LebaneseQeresh	_cLBP	0.01 * _LBP
MacanesePataca	_MOP	(1/1.03) * _HKD
MacaneseAvo	_cMOP	0.01 * _MOP

Name	Symbol	Definition
NamibianDollar	`_NAD`	`1 * _ZAR`
NamibianCent	`_cNAD`	`0.01 * _NAD`
NepaleseRupee	`_NPR`	`(1/1.6) * _INR`
NepalesePaisa	`_cNPR`	`0.01 * _NPR`
OmaniRial	`_OMR`	`(1/2.6008) * _USD`
OmaniBaisa	`_cOMR`	`0.001 * _OMR`
PanamanianBalboa	`_PAB`	`1 * _USD`
PanamanianCentesimo	`_cPAB`	`0.01 * _PAB`
TransnistrianRuble	`_PRB`	`(1/16.1) * _USD`
TransnistrianKopeck	`_cPRB`	`0.01 * _PRB`
SaudiRiyal	`_SAR`	`(1/3.75) * _USD`
SaudiHalalah	`_cSAR`	`0.01 * _SAR`
SaintHelenaPound	`_SHP`	`1 * _GBP`
SaintHelenaPenny	`_cSHP`	`0.01 * _SHP`
SouthSudanesePound	`_SSP`	`1 * _SDG`
SouthSudanesePiaster	`_cSSP`	`0.01 * _SSP`
SaoTomeAndPrincipeDobra	`_STN`	`(1/24.5) * _EUR`
SaoTomeAndPrincipeCentimo	`_cSTN`	`0.01 * _STN`
SwaziLilangeni	`_SZL`	`1 * _ZAR`
SwaziCent	`_cSZL`	`0.01 * _SZL`
TurkmenistanManat	`_TMT`	`(1/3.5) * _USD`
TurkmenistanTenge	`_cTMT`	`0.01 * _TMT`
TuvaluanDollar	`_TVD`	`1 * _AUD`
TuvaluanCent	`_cTVD`	`0.01 * _TVD`
CentralAfricanCFAFranc	`_XAF`	`(1/655.957) * _EUR`
CentralAfricanCFACentime	`_cXAF`	`0.01 * _XAF`
EasternCaribbeanDollar	`_XCD`	`(1/2.7) * _USD`
EasternCaribbeanCent	`_cXCD`	`0.01 * _XCD`

Name	Symbol	Definition
WestAfricanCFAFranc	`_XOF`	`(1/655.957) * _USD`
WestAfricanCFACentime	`_cXOF`	`0.01 * _XOF`
CFPFranc	`_XPF`	`(1000/8.38) * _EUR`
CFPCentime	`_cXPF`	`0.01 * _XPF`
ZimbabweanBonds	`_ZWL`	`1 * _USD`
ZimbabweanCent	`_cZWL`	`0.01 * _ZWL`

5 Lua Documentation

In this section, the following shortcuts will be used.

```
local D = physical.Dimension
local U = physical.Unit
local N = physical.Number
local Q = physical.Quantity
```

The term `number` refers to a lua integer or a lua float number. By `string` a lua string is meant and by `bool` a lua boolean.

5.1 physical.Quantity

The quantity class is the main part of the library. Each physical Quantity and all units are represented by an instance of this class.

`Q.new(q=nil)`

Copy Constuctor

q : `Q`, `number`, `object`, `nil`

returns : `Q`

As an argument it takes `Q`, `number`, `object` or `nil`. If an instance of `Q` is given, a copy is made and returned. If a `number` or an instance `object` of another class is given, the function creates a dimensionless quantity with the `number` or the instance as a value. In the case `nil` is given, a dimensionless quantity with value 1 is returned.

```
print( Q() )
1

print( Q(42) )
42

print( Q(73*_m) )
73 * _m
```

`Q.defineBase(symbol,name,dimension)`

This function is used to declare base quantities from which all other quantities are derived from.

`symbol` : `string`
    The symbol of the base quantity.

`name` : `string`
: The name of the base quantity.

`dimension` : `D`
: An instance of the `D` class, which represents the dimension of the quantity.

returns : `Q`
: The created `Q` instance.

The function creates a global variable of the created base quantity. The name consist of an underscore concatenated with the `symbol` argument, i.e. the symbol `m` becomes the global variable `_m`.

The `name` is used for example in the siunitx conversion function, e.g `meter` will be converted to `\meter`.

Each quantity has a dimension associated with it. The argument `dimension` allows any dimension to be associated to base quantities.

```
Q.defineBase("m", "meter", L)
Q.defineBase("kg", "kilogram", M)
```

Q.define(symbol, name, q)

Creates a new derived quantity from an expression of other quantities. Affine quantities like the absolute temperature in celsius are not supported.

`symbol` : `string`
: Symbol of the base quantity

`name` : `string`, `nil`
: The Name of the derived quantity.

`q` : `physical.Quantity`
: The definition of the derived quantity.

returns : `Q`
: The created quantity.

The function creates a global variable of the created base quantity. The name consist of an underscore concatenated with the `symbol` argument, i.e. the symbol `N` becomes the global variable `_N`.

The `name` is used for example in the siunitx conversion function, e.g `newton` will be converted to `\newton`.

```
Q.define("L", "liter", _dm^3)
Q.define("Pa", "pascal", _N/_m^2)
Q.define("C", "coulomb", _A*_s)

Q.define("degC", "celsius", _K)
```

Q.definePrefix(symbol,name,factor)

Defines a new prefix.

`symbol` : `string`
Symbol of the base quantity

`name` : `string`
Name of the base quantity

`factor` : `number`
The factor which corresponds to the prefix

```
Q.definePrefix("c", "centi", 1e-2)
Q.definePrefix("a", "atto", 1e-18)
```

Q.addPrefix(prefixes, units)

Create several units with prefixes from a given unit.

`prefixes` : `string`
A list of unit symbols.

`units` : `Q`
A list of quantities.

```
Q.addPrefix({"n","u","m","k","M","G"},{_m,_s,_A})
```

Q.isclose(self,q,r)

Checks if this quantity is close to another one. The argument `r` is the maximum relative deviation. The function returns `true` if the following condition is fullfilled

$$\frac{abs(\texttt{self}-\texttt{q})}{min(\texttt{self},\texttt{q})} \le \texttt{r} \quad .$$

`self` : `Q`, `N`, `number`

`q` : `Q`, `N`, `number`

`r` : `number`
maximum relative deviation of `self` and `q`

returns : `bool`
`true` if q is close to `self`, otherwise `false`

```
s_1 = 1.9 * _m
s_2 = 2.0 * _m
print( s_1:isclose(s_2,0.1) )
true
print( s_1:isclose(s_2,0.01) )
false
```

Q.to(self,q=nil)

Converts the quantity `self` to the unit of the quantity `q`. If no `q` is given, the quantity `self` is converted to base units.

`self` : `Q`

`q` : `Q`, `nil`

```
s = 1.9 * _km
print( s:to(_m) )
1900.0 * _m

T = 10 * _degC
print( T:to(_K) )
10.0 * _K

print( T:to() )
10 * _K
```

Q.tosiunitx(self,param,mode=Q.siunitx_SI)

Converts the quantity into a `siunitx` string.

`self` : `Q`

`param` : `string`

`mode` : `number`

If `mode` is equal `Q.SIUNITX_SI`, which is the default, the quantity is converted to an \SI{}{} macro. If mode is `Q.SIUNITX_num`, the quantity is converted to \num{} and if it is `Q.SIUNITX_si` the macro \si{} is printed.

```
s = 1.9 * _km

print( s:tosiunitx() )
\SI{1.9}{\kilo\meter}

print( s:tosiunitx(nil,Q.SIUNITX_num) )
\num{1.9}

print( s:tosiunitx(nil,Q.SIUNITX_si) )
\si{\kilo\meter}
```

`Q.min(q1, q2, ...)`

Returns the smallest quantity of the given ones. The function returns `q1` if the Quantities are equal.

`q1` : `Q`, `N`, `number`

`q2` : `Q`, `N`, `number`

...

`qN` : `Q`, `N`, `number`

returns : `Q`
the smallest quantity of `q1`, ... , `qN`

```
s_1 = 15 * _m
s_2 = 5 * _m
print(s_1:min(s_2))
5 * _m
```

`Q.max(q1, q2, ...)`

Returns the biggest quantity of several given ones. The function returns `q1` if the Quantities are equal.

`q1` : `Q`, `N`, `number`

`q2` : `Q`, `N`, `number`

...

`qN` : `Q`, `N`, `number`

returns : `Q`
the biggest quantity of `q1`, ... , `qN`

```
s_1 = 15 * _m
s_2 = 5 * _m
print(s_1:max(s_2))
15 * _m
```

`Q.abs(q)`

Returns the absolute value of the given quantity `q`.

`q` : `Q`, `N`, `number`

returns : `Q`
the absolute value of `q`

```
U = -5 * _V
print(U)
-5 * _V
print(U:abs())
5 * _V
```

`Q.sqrt(q)`

Returns the square root of the given quantity.

`q` : `Q`, `N`, `number`
dimensionless argument

returns : `Q`
the square root of `q`

```
A = 25 * _m^2
s = A:sqrt()
print(s)
5.0 * _m
```

`Q.log(q, base=nil)`

Returns the logarithm of a given quantitiy to the given base. If no base is given, the natural logarithm is returned.

`q` : `Q`, `N`, `number`
dimensionless argument

`base` : `Q`, `N`, `number`, `nil`
dimensionless argument

returns : `Q`
logarithm of `q` to the `base`

```
I = 1 * _W/_m^2
I_0 = 1e-12 * _W/_m^2
print(10 * (I/I_0):log(10) * _dB )
120.0 * _dB
```

`Q.exp(q)`

Returns the value of the natural exponential function of the given quantitiy.

`q` : `Q`, `N`, `number`
dimensionless argument

returns : Q
    natural exponential of q

```
x = 2 * _1
print( x:exp() )
7.3890560989307
```

Q.sin(q)

Returns the value of the sinus function of the given quantitiy.

q : Q, N, number
    dimensionless argument

returns : Q
    sine of q

```
alpha = 30 * _deg
print( alpha:sin() )
0.5
```

Q.cos(q)

Returns the value of the cosinus function of the given quantity. The quantity has to be dimensionless.

q : Q, N, number
    dimensionless argument

returns : Q
    cosine of q

```
alpha = 60 * _deg
print( alpha:cos() )
0.5
```

Q.tan(q)

Returns the value of the tangent function of the given quantity. The quantity has to be dimensionless.

q : Q, N, number
    dimensionless argument

returns : Q
    tangent of q

```
alpha = 45 * _deg
print( alpha:tan() )
1.0
```

`Q.asin(q)`

Returns the value of the arcus sinus function of the given quantity. The quantity has to be dimensionless.

`q` : `Q`, `N`, `number`
dimensionless argument

returns : `Q`
inverse sine of `q`

```
x = 0.5 * _1
print( x:asin():to(_deg) )
30.0 * _deg
```

`Q.acos(q)`

Returns the value of the arcus cosinus function of the given quantity. The quantity has to be dimensionless.

`q` : `Q`, `N`, `number`
dimensionless argument

returns : `Q`
inverse cosine of `q`

```
x = 0.5 * _1
print( x:acos():to(_deg) )
60.0 * _deg
```

`Q.atan(q)`

Returns the value of the arcus tangent function of the given quantity. The quantity has to be dimensionless.

`q` : `Q`, `N`, `number`
dimensionless argument

returns : `Q`
inverse tangent of `q`

```
x = 1 * _1
print( x:atan():to(_deg) )
45.0 * _deg
```

Q.sinh(q)

Returns the value of the hyperbolic sine function of the given quantity. The quantity has to be dimensionless. Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\sinh(x) = 0.5 \cdot e^x - 0.5/e^x \quad .$$

q : Q, N, number
  dimensionless argument

returns : Q
  hyperbolic sine of q

```
x = 1 * _1
print( x:sinh() )
1.1752011936438
```

Q.cosh(q)

Returns the value of the hyperbolic cosine function of the given quantity. The quantity has to be dimensionless. Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\cosh(x) = 0.5 \cdot e^x + 0.5/e^x \quad .$$

q : Q, N, number
  dimensionless argument

returns : Q
  hyperbolic cosine of q

```
x = 1 * _1
print( x:cosh() )
1.5430806348152
```

Q.tanh(q)

Returns the value of the hyperbolic tangent function of the given quantity. The quantity has to be dimensionless. Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad .$$

q : Q, N, number
  dimensionless argument

returns : Q
  hyperbolic tangent of q

```
x = 1 * _1
print( x:tanh() )
0.76159415595576
```

`Q.asinh(q)`

Returns the value of the inverse hyperbolic sine function of the given quantity. The quantity has to be dimensionless. Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\operatorname{asinh}(x) = \ln\left(x + \sqrt{x^2 + 1}\right) \quad .$$

`q` : `Q`, `N`, `number`
dimensionless argument

returns : `Q`
inverse hyperbolic sine of `q`

```
x = 1 * _1
print( x:asinh() )
0.88137358701954
```

`Q.acosh(q)`

Returns the value of the inverse hyperbolic cosine function of the given quantity. The quantity has to be dimensionless. Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\operatorname{acosh}(x) = \ln\left(x + \sqrt{x^2 - 1}\right) \quad , x > 1 \quad .$$

`q` : `Q`, `N`, `number`
dimensionless argument bigger or equal to one

returns : `Q`
inverse hyperbolic cosine of `q`

```
x = 2 * _1
print( x:acosh() )
1.3169578969248
```

`Q.atanh(q)`

Returns the value of the inverse hyperbolic tangent function of the given quantity. The quantity has to be dimensionless. Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\operatorname{atanh}(x) = \ln\left(\frac{1+x}{1-x}\right) \quad , -1 < x < 1 \quad .$$

`q` : `Q`, `N`, `number`
  dimensionless argument with magnitude smaller than one

returns : `Q`
  inverse hyperbolic tangent of `q`

```
x = 0.5 * _1
print( x:atanh() )
0.54930614433405
```

5.2 physical.Dimension

All physical quantities do have a physical dimension. For example the quantity *Area* has the dimension L^2 (lenght to the power of two). In the SI-System there are seven base dimensions, from which all other dimensions are derived. Each dimension is represented by an n-tuple, where n is the number of base dimensions. Each physical quantity has an associated dimension object. It is used to check equality and if addition or substraction is allowed.

`D.new(d)`

Constructor of the `Dimension` class.

`d` : `Dimension` or `string`, `nil`
The name or symbol of the dimension.

returns : `D`
The created `D` instance

If `d` is a string, a copy of the perviously defined dimension is made. If `d` is a dimension, a copy of it is made. If no argument ist given, a dimension *zero* is created.

Example

```
V_1 = D("Velocity")
L = D("L")
V_2 = D(L/T)
```

`D.defineBase(symbol, name)`

Defines a base dimension.

`symbol` : `string`

`name` : `string`

returns : `D`
The created `D` instance

Example

```
V_1 = D("Velocity")
L = D("L")
V_2 = D(L/T)
```

5.3 physical.Unit

The task of this class is keeping track of the unit term. The unit term is a fraction of units. The units in the enumerator and denominator can have an exponent.

`Unit.new(u=nil)`

Copy Constructor. It copies a given unit object. If nothing is given, an empty unit is created.

`u` : `Unit`
The unit object which will be copied.

returns : `Unit`
The created `Unit` object

`Unit.new(symbol, name, prefixsymbol=nil, prefixname=nil)`

Constructor. A new `Unit` object with symbol is created. The prefixsymbol and prefixname are optional.

`symbol` : `String`
The symbol of the unit.

`name` : `String`
The name of the unit.

`prefixsymbol` : `String`
The optional symbol of the prefix.

`prefixname` : `String`
The optional name of the prefix.

returns : `Unit`
The created `Unit` object

`Unit.tosiunitx(self)`

The unit term will be compiled into a string, which the LaTeX package siunitx can understand.

returns : `String`
The siunitx representation of the unit term.

5.4 physical.Number

This class enhances the Lua number with an uncertainty and gaussian error propagation. A number instance has a mean value called `x` and an uncertainty value called `dx`.

`N.omitUncertainty=true`

This variable controls, wether the uncertainty `dx` is printed or not. The default value is `true`, i.e. the uncertainty is omitted.

```
n = N(45,0.012)

N.omitUncertainty = false
print(n)
(45.000 +/- 0.012)

N.omitUncertainty = true
print(n)
45.0
```

`N.seperateUncertainty=true`

This variable controls, how the uncertainty is printed. If set to `false`, the parenthesis notation is used. If set to `true` the plus minus notation is used. The default value is `true`.

```
n = N(56,0.025)

N.seperateUncertainty = false
print(n)
56.00(3)

N.seperateUncertainty = true
print(n)
(56.00 +/- 0.03)
```

`N.format=N.SCIENTIFIC`

This variable controls, how the number is printed. If set to `N.SCIENTIFIC`, scientific notation is used. If set to `N.DECIMAL` the decimal notation is used. The default value is `N.SCIENTIFIC`.

```
n = N(12000000,0.1)

N.format = N.SCIENTIFIC
print(n)
1.2000000e7
```

```
N.format = N.DECIMAL
print(n)
12000000
```

N.new(n=nil)

This is the copy Constructor for the Number class. It copies a given number object. If n is nil, an instance representing number zero with uncertainty zero is created.

n : Number
The number object to be copied.

returns : Number
The created Number instance.

```
n = N(56,0.012)
m = N(n)
print(m)
56.0
```

N.new(x, dx=nil)

This constructor, creates a new instance of N with mean value x and uncertainty dx. If dx is not given, the uncertainty is zero.

x : number
mean value

dx : number, nil
uncertainty value

returns : N
The created N instance.

```
n = N(56,0.012)
print(n)
56.0
```

N.new(str)

This constructor creates a new instance of N from a string. It can parse strings of the form "3.4", "3.4e-3", "5.4e-3 +/- 2.4e-6" and "5.45(7)e-23".

str : string

returns : `N`

```
n_1 = N("12.3e-3")
print(n_1)
0.0123

n_2 = N("12 +/- 0.1")
print(n_2)
12

n_3 = N("12.0(1)")
print(n_3)
12

n_4 = N("15.0(12)")
print(n_4)
15.0
```

`N.mean(n)`

Returns the mean value of `n`.

Parameters / Return

returns : `number`

```
n = N(1.25,0.0023)
print( n:mean() )
1.25
```

`N.uncertainty(n)`

Returns the uncertainty value of `n`.

`n` : `N`

returns : `number`

```
n = N(1.25,0.0023)
print( n:uncertainty() )
0.0023
```

`N.abs(n)`

Returns the absolute value of `n`.

`n` : `N`

returns : `N`

The uncertainty Δy is calculated by the following expression

$$\Delta y = \Delta x \quad .$$

```
n = N(-10,1)
print( n:abs() )
10.0
```

`N.sqrt(n)`

Returns the square root of `n`.

`n` : `N`

returns : `N`

The uncertainty Δy is calculated by the following expression

$$\Delta y = \frac{1}{2\sqrt{x}} \cdot \Delta x \quad .$$

```
n = N(25,1)
print( n:sqrt() )
5
```

`N.log(n,base=nil)`

Returns the logarithm of a given number `n` to the given base `base`. If no base is given, the natural logarithm of `n` is returned.

`n` : `N`

`base` : `number`, `nil`

returns : `N`

The uncertainty Δy is calculated by the following expression

$$\Delta y = \frac{1}{\mid x \cdot \log(b) \mid} \cdot \Delta x \quad .$$

```
n = N(25,1)
print( n:log() )
3.2
```

N.exp(n)

Returns the value of the natural exponential function of the given number.

q : N

returns : N

The uncertainty Δy is calculated by the following expression

$$\Delta y = e^x \cdot \Delta x \quad .$$

```
n = N(25,1)
print( n:sqrt() )
5
```

N.sin(n)

Returns the value of the sine function of the given number.

n : N

returns : N

The uncertainty Δy is calculated by the following expression

$$\Delta y = \mid \cos(x) \mid \cdot \Delta x \quad .$$

```
n = N(3,0.1)
print( n:sin() )
0.1
```

N.cos(n)

Returns the value of the cosine function of the given number.

n : N

returns : N

The uncertainty Δy is calculated by the following expression

$$\Delta y = \mid \sin(x) \mid \cdot \Delta x \quad .$$

```
n = N(0.5,0.01)
print( n:cos() )
0.88
```

N.tan(n)

Returns the value of the tangent function of the given number.

n : N

returns : N

The uncertainty Δy is calculated by the following expression

$$\Delta y =\mid \frac{1}{\cos^2(x)} \mid \cdot \Delta x \quad .$$

```
n = N(1.5,0.001)
print( n:tan() )
14
```

N.asin(n)

Returns the value of the inverse sine function of the given number.

n : N

returns : N

The uncertainty Δy is calculated by the following expression

$$\Delta y = \frac{1}{\sqrt{1-x^2}} \cdot \Delta x \quad .$$

```
n = N(0.99,0.001)
print( n:asin() )
1.43
```

N.acos(n)

Returns the value of the inverse cosine function of the given number.

n : N

returns : N

The uncertainty Δy is calculated by the following expression

$$\Delta y = \frac{1}{\sqrt{1-x^2}} \cdot \Delta x \quad .$$

```
n = N(0.99,0.001)
print( n:acos() )
0.14
```

N.atan(n)

Returns the value of the inverse tangent function of the given number.

n : N

returns : N

The uncertainty Δy is calculated by the following expression

$$\Delta y = \frac{1}{\sqrt{1+x^2}} \cdot \Delta x \quad .$$

```
n = N(1,0.001)
print( n:atan() )
0.785
```

N.sinh(q)

Returns the value of the hyperbolic sine function of the given number.

n : N

returns : N

Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\sinh(x) = 0.5 \cdot e^x - 0.5/e^x \quad .$$

The uncertainty Δy is calculated by the following expression

$$\Delta y = (0.5 \cdot e^x + 0.5/e^x) \cdot \Delta x \quad .$$

```
n = N(1,0.001)
print( n:sinh() )
1.18
```

N.cosh(q)

Returns the value of the hyperbolic cosine function of the given number.

n : N

returns : N

Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\cosh(x) = 0.5 \cdot e^x + 0.5/e^x \quad .$$

The uncertainty Δy is calculated by the following expression

$$\Delta y = (0.5 \cdot e^x - 0.5/e^x) \cdot \Delta x \quad .$$

```
n = N(1,0.001)
print( n:cosh() )
1.54
```

N.tanh(q)

Returns the value of the hyperbolic tangent function of the given number.

n : N

returns : N

Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad .$$

The uncertainty Δy is calculated by the following expression

$$\Delta y = \frac{1}{(0.5 \cdot e^x + 0.5/e^x)^2} \cdot \Delta x \quad .$$

```
n = N(1,0.001)
print( n:tanh() )
0.762
```

Q.asinh(q)

Returns the value of the inverse hyperbolic sine function of the given number.

n : N

returns : N

Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\operatorname{asinh}(x) = \ln\left(x + \sqrt{x^2+1}\right) \quad .$$

The uncertainty Δy is calculated by the following expression

$$\Delta y = \frac{1}{\sqrt{x^2+1}} \cdot \Delta x \quad .$$

```
n = N(1,0.001)
print( n:asinh() )
0.881
```

`Q.acosh(q)`

Returns the value of the inverse hyperbolic cosine function of the given number.

`n` : `N`

returns : `N`

Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\operatorname{acosh}(x) = \ln\left(x + \sqrt{x^2-1}\right) \quad , x > 1 \quad .$$

The uncertainty Δy is calculated by the following expression

$$\Delta y = \frac{1}{\sqrt{x^2-1}} \cdot \Delta x \quad .$$

```
n = N(2,0.001)
print( n:acosh() )
1.317
```

`Q.atanh(q)`

Returns the value of the inverse hyperbolic tangent function of the given number.

`n` : `N`

returns : `N`

Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\operatorname{atanh}(x) = \ln\left(\frac{1+x}{1-x}\right) \qquad , -1 < x < 1 \quad .$$

The uncertainty Δy is calculated by the following expression

$$\Delta y = \frac{1}{\mid x^2 - 1 \mid} \cdot \Delta x \quad .$$

```
n = N(-0.5,0.0001)
print( n:atanh() )
-0.549
```

6 Change History

V1.0.4 (2020/09/15) **Minor release**
Changed default value of `Number.omitUncertainty` and `Number.seperateUncertainty` to `true`. Documentation added.

V1.0.3 (2020/09/09) **Minor release**
Changed foldername `physical` to `src`. Changed Classvariables `Q.siunitx_SI`, `Q.siunitx_num` and `Q.siunitx_si` to uppercase `Q.SIUNITX_SI`, `Q.SIUNITX_num` and `Q.SIUNITX_si`.

V1.0.2 (2020/09/07) **Minor release**
Path issues resolved. Documentation added.

V1.0.1 (2020/09/05) **Minor release**
Files renamed.

V1.0.1 (2020/09/03) **First official release.**

References

[1] Webpage https://physics.nist.gov/cuu/index.html, August 2019.

[2] Bureau International des Poids et Mesures. The international system of units (si), 2006.

[3] Bureau International des Poids et Mesures. Resolutions of the 26th cgpm, November 2018.

[4] Prša et al. Nominal values for selected solar and planetary quantities: Iau 2015 resolution b3. The Astronomical Journal, 152:41, August 2016.

Index of Units

Index of Currencies

Index of Lua Classes and Methods