Units

- The new Simple environment for units makes computations with units easier than ever. It allows you to work with expressions such as $5\,\mathrm{mm}+x$, where x is unassigned, as long as the dimensional analysis of the expression determines there is a valid quantity with units for the variable x.

- Since the introduction of the Units package for Maple 7 (in 2001), there have been three ways to compute with unit expressions:
  - The default units environment, in which you explicitly invoke commands such as combine,units or convert,dimensions whenever you want to use unit functionality. This environment is suitable for expert users.
  - The Standard units environment, in which unit functionality is enabled for most basic operations. This environment requires that units are explicitly marked as such.
  - The Natural units environment, in which unit functionality is also enabled for most basic operations. This environment automatically interprets any variable name that corresponds to a unit name, as that unit.

- Maple 2017 introduces a fourth way to compute with unit expressions: the Simple units environment. This differs from the Standard and Natural environments in two important ways, both of which make it easier to work with this environment in many cases:
  - The Standard and Natural environments require that whenever you use an unassigned variable (in the case of the Natural environment, an unassigned variable that is not the name of a unit), it represents a unitless quantity. This rule applies for verifying dimensional correctness - the process that makes sure you cannot add a distance to a velocity, for example. Concretely, you cannot just use a variable called $distance$ to represent a distance to be determined later: instead, you might need to define $distance$ by $distance \coloneqq distance\_km\ \mathrm{km}$, so that $distance$ is now an assigned variable and $distance\_km$ is a unitless quantity representing the number of kilometers that $distance$ is long. This restriction is relaxed for the Simple units environment: unassigned variables are not assumed to have any particular dimension during testing of dimensional correctness.
  - The Simple units environment disregards so-called unit annotations when verifying dimensional correctness. These annotations are a mechanism to indicate that the

physical quantity described by the unit is of a particular type. For example, the unit $mile\_per\_gallon$ of fuel economy is defined in Maple as $\frac{mile}{gallon(petroleum)}$; this allows Maple to distinguish it from a unit of the dimension $\frac{1}{length^2} = \frac{1}{area}$ (which would normally happen when simplifying an expression of the dimension $\frac{length}{length^3} = \frac{length}{volume}$). Another situation in which these annotations occur is in units of angle: a radian is defined as the angle that describes a circle sector in which the radius and the arc length are the same. This is defined in Maple as the unit $\frac{m}{m(radius)}$.

In many cases, however, users *want* to combine units while disregarding the unit annotations. This is particularly frequent whenever units of angle crop up. For that reason, the Simple units environment allows the addition of, for example, quantities with the dimension of angle to unitless quantities. The Standard and Natural environments do not.

- Before Maple 2017, the default units environment always followed the same approach as the Standard and Natural units environments, with respect to these two issues. In Maple 2017, you can use the new $Units[UseMode]$ command to switch between these two approaches in the default units environment.

Examples

```
> with(Units:-Simple) :
```

Consider a situation where you know an object will undergo uniform acceleration, with an initial velocity of 2.4 meters per second. We can derive the generic formula for the position of the object at an arbitrary time as follows. We start with the general formula for velocity under uniform acceleration.

```
> v := v0 + a t
```

$$v := a\,t + v0 \tag{1}$$

Now we substitute the given starting velocity.

```
> eval(v, v0 = 2.4 m/s)
```

$$a\,t + 2.4\ \frac{\mathrm{m}}{\mathrm{s}} \tag{2}$$

```
> vs := (2)
```

$$vs := a\,t + 2.4\ \frac{\mathrm{m}}{\mathrm{s}} \tag{3}$$

Integrating this over time gives the requested answer.

> $x := x0 + \int_0^{t0} vs \, dt$

$$x := x0 + 0.5000000000\, a\, t0^2 + 2.400000000\, t0\, \frac{\mathrm{m}}{\mathrm{s}} \tag{4}$$

Now let's additionally specify that the object starts at the origin, and the acceleration is gravity. We orient our axis of displacement so that the initial velocity was upward, so in the opposite direction to gravity.

> $gravity := evalf(ScientificConstants{:-}Constant(g, units))$

$$gravity := 9.80665\, \frac{\mathrm{m}}{\mathrm{s}^2} \tag{5}$$

> $eval(x, [x0 = 0\ \mathrm{m}, a = -gravity])$

$$-4.903325000\, t0^2\, \frac{\mathrm{m}}{\mathrm{s}^2} + 2.400000000\, t0\, \frac{\mathrm{m}}{\mathrm{s}} \tag{6}$$

If we want to know the position after 0.4 seconds, we can obtain it like this.

> $eval(\mathbf{(6)}, t0 = 0.4\ \mathrm{s})$

$$0.1754680000\ \mathrm{m} \tag{7}$$

This example comes from the example worksheets for the various units environments; specifically, for the version for the Simple units environment. You can compare the default units example worksheet, the Standard units example worksheet, and the Natural units example worksheet.

The first example shows how you can use unassigned variables to represent quantities that are not unitless. The following example will show a case where it comes in handy that annotations are ignored when verifying dimensional correctness.

Consider tightening a nut using a wrench. You apply a constant torque of 30 newton meters over 6 full rotations. The friction increases the temperature of the nut by 5 degrees Celsius; the nut, which is sizeable, has a mass of 80 grams and is made of steel. Steel has a specific heat capacity of 452 joules per kilogram per Kelvin. How much energy did you expend, how much of it was converted to heat, and how much leaked away into the environment?

> $torque := 30.\ \mathrm{N\,m}$

$$torque := 30.\ \mathrm{N\,m} \tag{8}$$

> $angle := 6.\ \mathrm{revolution}$

$$angle := 6.\ \mathrm{rev} \tag{9}$$

> $nut\_mass := 80.\ \mathrm{g}$

$$nut\_mass := 80.\ \mathrm{g} \tag{10}$$

> $heat\_capacity := 452.\ \frac{\frac{\mathrm{J}}{\mathrm{kg}}}{\mathrm{K}}$

$$heat\_capacity := 452.\ \frac{\mathrm{J}}{\mathrm{kg\ K}} \quad \mathbf{(11)}$$

> $temperature\_rise := 5\ \mathrm{degC}$

$$temperature\_rise := 5\ {}^{\circ}\mathrm{C} \quad \mathbf{(12)}$$

> $expended\_energy := torque\ angle$

$$expended\_energy := 1130.973355\ \frac{\mathrm{m}^3\ \mathrm{kg}}{\mathrm{s}^2\ \mathrm{m}(radius)} \quad \mathbf{(13)}$$

> $heat := nut\_mass\ heat\_capacity\ temperature\_rise$

$$heat := 180.8000000\ \mathrm{J} \quad \mathbf{(14)}$$

> $leaked := expended\_energy - heat$

$$leaked := 950.1733550\ \mathrm{J} \quad \mathbf{(15)}$$

TestDimensions

- At the heart of the Simple units environment is the new command Units [TestDimensions]. This command verifies dimensional correctness of one or more expressions with units. By default, it returns **`true`** if there is a valid assignment of dimensions to all subexpressions of **`expr`**, and **`false`** otherwise. Validity is tested for all subexpressions occurring in **`expr`**.
- This command can be useful to verify the dimensional validity of, for example, a set of equations involving some quantities with units.

> $with(Units):$

> $eq1 := F = m\ a$

$$eq1 := F = m\ a \quad \mathbf{(16)}$$

> $eq2 := F = m^2\ a$

$$eq2 := F = m^2\ a \quad \mathbf{(17)}$$

> $TestDimensions\left(eq1, \left\{F::\mathrm{N}, a::\left(\frac{\mathrm{m}}{\mathrm{s}^2}\right), m::\mathrm{kg}\right\}\right)$

$$true \quad \mathbf{(18)}$$

> $TestDimensions\left(eq2, \left\{F::\mathrm{N}, a::\left(\frac{\mathrm{m}}{\mathrm{s}^2}\right), m::\mathrm{kg}\right\}\right)$

$$false \quad \mathbf{(19)}$$

- If you know the dimension of only some of the quantities occurring in a set of equations, you can derive some things about the dimensions of the other symbols and of composite subexpressions. Sometimes you can fully derive the dimension of some quantities, other times there is not enough information.

> $TestDimensions(eq1,\ \{F::\mathrm{N},\ m::\mathrm{kg}\},\ output=units)$

$$\left\{F::\mathrm{N},\ a::\left(\frac{\mathrm{m}}{\mathrm{s}^2}\right),\ m::\mathrm{kg},\ (m\ a)::\mathrm{N}\right\} \tag{20}$$

- If there are multiple valid assignments of dimensions to the symbols in an expression (in other words, if there is not enough information to fully determine the dimensions of some quantities), then Maple will make an arbitrary choice.

> $TestDimensions(eq1,\ \{F::\mathrm{N}\},\ output=units)$

$$\{F::\mathrm{N},\ a::1,\ m::\mathrm{N},\ (m\ a)::\mathrm{N}\} \tag{21}$$

- Instead of expressing the dimension by using a unit of the given expression, Maple can also return the name of the dimension (as it would be returned by convert/dimensions with the **base** option).

> $TestDimensions(eq1,\ \{F::\mathrm{N},\ m::\mathrm{kg}\},\ output=dimensions)$

$$\left\{F::\left(\frac{mass\ length}{time^2}\right),\ a::\left(\frac{length}{time^2}\right),\ m::mass,\ (m\ a)::\left(\frac{mass\ length}{time^2}\right)\right\} \tag{22}$$

- In this case, if there is ambiguity in the given set of expressions, Maple will determine an expression representing the fully general dimension for any subexpression. These expressions are mostly useful for programmer access.

> $result := TestDimensions(eq1,\ \{F::\mathrm{N}\},\ output=dimensions)$

$$result := \left\{F::\left(\frac{mass\ length}{time^2}\right),\ a:: \left(length^{\_t58_{1,1}}\ mass^{\_t58_{1,2}}\ time^{\_t58_{1,3}}\ electric\_current^{\_t58_{1,4}}\ thermodynamic\_temperature^{\_t58_{1,5}}\ amount\_of\_substance^{\_t58_{1,6}}\ luminous\_intensity^{\_t58_{1,7}}\ currency^{\_t58_{1,8}}\ amount\_of\_information^{\_t58_{1,9}}\ logarithmic\_gain^{\_t58_{1,10}}\right),\ m:: \left(length^{1-\_t58_{1,1}}\ mass^{1-\_t58_{1,2}}\ time^{-2-\_t58_{1,3}}\ electric\_current^{-\_t58_{1,4}}\ thermodynamic\_temperature^{-\_t58_{1,5}}\ amount\_of\_substance^{-\_t58_{1,6}}\ luminous\_intensity^{-\_t58_{1,7}}\ currency^{-\_t58_{1,8}}\ amount\_of\_information^{-\_t58_{1,9}}\ logarithmic\_gain^{-\_t58_{1,10}}\right),\ (m\ a)::\left(\frac{mass\ length}{time^2}\right)\right\} \tag{23}$$

• By manipulating the result, we can see that the dimension of $m\,a$ is indeed what is claimed.

```
> result_as_equations := map(lhs = rhs, result)
```

$$result\_as\_equations := \left\{F = \frac{mass\ length}{time^2}, a = length^{\_t58_{1,1}}\ mass^{\_t58_{1,2}}\ time^{\_t58_{1,3}}\ electric\_current^{\_t58_{1,4}}\ thermodynamic\_temperature^{\_t58_{1,5}}\ amount\_of\_substance^{\_t58_{1,6}}\ luminous\_intensity^{\_t58_{1,7}}\ currency^{\_t58_{1,8}}\ amount\_of\_information^{\_t58_{1,9}}\ logarithmic\_gain^{\_t58_{1,10}}, m = length^{1-\_t58_{1,1}}\ mass^{1-\_t58_{1,2}}\ time^{-2-\_t58_{1,3}}\ electric\_current^{-\_t58_{1,4}}\ thermodynamic\_temperature^{-\_t58_{1,5}}\ amount\_of\_substance^{-\_t58_{1,6}}\ luminous\_intensity^{-\_t58_{1,7}}\ currency^{-\_t58_{1,8}}\ amount\_of\_information^{-\_t58_{1,9}}\ logarithmic\_gain^{-\_t58_{1,10}}, m\,a = \frac{mass\ length}{time^2}\right\} \quad (24)$$

```
> m_dim := eval(m, result_as_equations)
```

$$m\_dim := length^{1-\_t58_{1,1}}\ mass^{1-\_t58_{1,2}}\ time^{-2-\_t58_{1,3}}\ electric\_current^{-\_t58_{1,4}}\ thermodynamic\_temperature^{-\_t58_{1,5}}\ amount\_of\_substance^{-\_t58_{1,6}}\ luminous\_intensity^{-\_t58_{1,7}}\ currency^{-\_t58_{1,8}}\ amount\_of\_information^{-\_t58_{1,9}}\ logarithmic\_gain^{-\_t58_{1,10}} \quad (25)$$

```
> a_dim := eval(a, result_as_equations)
```

$$a\_dim := length^{\_t58_{1,1}}\ mass^{\_t58_{1,2}}\ time^{\_t58_{1,3}}\ electric\_current^{\_t58_{1,4}}\ thermodynamic\_temperature^{\_t58_{1,5}}\ amount\_of\_substance^{\_t58_{1,6}}\ luminous\_intensity^{\_t58_{1,7}}\ currency^{\_t58_{1,8}}\ amount\_of\_information^{\_t58_{1,9}}\ logarithmic\_gain^{\_t58_{1,10}} \quad (26)$$

```
> expand(m_dim a_dim) = eval(m a, result_as_equations)
```

$$\frac{mass\ length}{time^2} = \frac{mass\ length}{time^2} \tag{27}$$