B. Lungsi Sharma

TUTORIAL

Generating Dimensional Formulae and Checking for Dimensional Consistency.

Let us imagine we derived an expression given below.

$$x = x_0 + v^2 + t + \frac{1}{2}at^2$$

Before we begin the dimensional analysis using diman© we must first do some preliminary setting up.

Definitions setup.

We set the definitions for the symbols in the expression.

```
(def varpars [{:symbol "x", :dimension "length"}
              {:symbol "v", :dimension "velocity"}
              {:symbol "t", :dimension "time"}
              {:symbol "a", :dimension "acceleration"}])
```

Expressions and equation setup.

We then define the equation whose left- and right-hand sides are based on their defined expressions.

```
(def leftside "x^(1)")
(def rightside {:term1 "x^(1)",
                :term2 "v^(2)",
                :term3 "t^(1)",
                :term4 "0.5*a^(1)*t^(2)"})
(def equation {:lhs leftside, :rhs rightside})
```

Importing functions from diman©

To do our analysis we must import function required for it.

```
(require '[diman.formula :refer [formula-term formula-eqn-side]])
(require '[diman.filter :refer [remove-zero-powers]])
(require '[diman.analyze :refer [dimnames consistent?]])
```

Getting dimensional formula.

Sub-formula of the dimensional formula for one side of the equation.

A sub-formula is practically a dimensional formula for one of the terms on a chosen side (left- or right-hand sides) of the equation. This is because the sub-formula **IS** the dimensional formula for the expression if there is just one term.

Based on our definition we setup we know that the right-hand side of the given equation is

```
=> (:rhs equation)
{:term1 "x^(1)", :term2 "v^(2)", :term3 "t^(1)", :term4 "0.5*a^(1)*t^(2)"}
```

Say, we are interested in viewing the dimensional formula for the `:term4` expression in the `:rhs` of the equation. Then using the `formula-term` function and passing our expression of interest as its argument we get

```
=> (formula-term varpars (:term4 (:rhs equation)))
"[T^(0)*L^(1)]"
```

Notice that this is consistent with the composite unit of the dimensions in the expression.

$$\frac{1}{2}at^2 \to at^2 \to ms^{-2} \cdot s^2 \to ms^0 \to m$$

where ms^{-2} is the unit for acceleration.

The base quantities with zero exponents can be removed by using the `remove-zero-powers` function.

```
=> (remove-zero-powers (formula-term varpars (:term4 (:rhs equation))))
"[L^(1)]"
```

Dimensional formula for one side of the equation.

Similarly, the dimensional formula for a side of the equation can be derived. However, for this we use the `formula-eqn-side` function.

```
=> (formula-eqn-side varpars (:rhs equation))
"[L^(1)] + [T^(-2)*L^(2)] + [T^(1)] + [T^(0)*L^(1)]"
```

As was in the case shown above the base quantities with zero exponents can be removed using the `remove-zero-powers` function.

```
=> (remove-zero-powers (formula-eqn-side varpars (:rhs equation)))
"[L^(1)] + [T^(-2)*L^(2)] + [T^(1)] + [L^(1)]"
```

View dimensional names in the derived formula.

Using the `dimnames` function the notations for the base quantities in the formula can be reflected in terms of their dimensional names.

For the sub-formula of the fourth term in the right-hand side of the equation this is

```
=> (dimnames (formula-term varpars (:term4 (:rhs equation))))
"length^(1)"
```

For the formula of the right-hand side is

```
=> (dimnames (formula-eqn-side varpars (:rhs equation)))
"length^(1) + time^(-2)*length^(2) + time^(1) + length^(1)"
```

Analyze.

Consistency check.

If the correctness of an equations is in doubt checking for dimensional consistency is a useful preliminary step. In diman© this is done using the `consistent?` function.

Thus,

```
=> (consistent? varpars equation)
false
```

Notice that this is consistent when we view the composite unit of the dimensions in the

$$RHS: x_0 + v^2 + t + \frac{1}{2}at^2 \to m + (ms^{-1})^2 + s + ms^{-2} \cdot s^2 \to m + m^2s^{-2} + s + m$$

$$LHS: x \to m$$

That is, the equation $x = x_0 + v^2 + t + {}^1/_2\, at^2$ *fails* the consistency check. Thus,

An equation that is not dimensionally consistent **must be wrong**.

What about the equation $x = x_0 + v_0t + {}^1/_2\, at^2$?

We define this second equation as

```
(def equation2 {:lhs "x^(1)", :rhs {:term1 "x^(1)",
                                    :term2 "v^(1)*t^(1)",
                                    :term3 "0.5*a^(1)*t^(2)"}})
```

And, `consistent?` function on this equation returns

```
=> (consistent? varpars equation2)
true
```

This agrees with

$$RHS: x_0 + v_0t + \frac{1}{2}at^2 \to m + ms^{-1} \cdot s^1 + ms^{-2} \cdot s^2 \to\ m + m + m \to m(1+1+1) \to m$$

$$LHS: x \to m$$

Thus, equation $x = x_0 + v_0t + {}^1/_2\, at^2$ *passes* the consistency check.

Consistency checking in dimensional analysis although useful it is still a preliminary step in the analysis because

A dimensionally consistent equation **does not guarantee** correct equation.

Let us illustrate this with multiple equations.

Equation	Definition setup
$e = m^2v^2$	`(def eqn1 {:lhs "e^(1)",` `:rhs "m^(2)*v^(2)"})`
$e = \frac{1}{2}mv^2$	`(def eqn2 {:lhs "e^(1)",` `:rhs "0.5*m^(1)*v^(2)"})`
$e = ma$	`(def eqn3 {:lhs "e^(1)",` `:rhs "m^(1)*a^(1)"})`
$e = \frac{3}{16}mv^2$	`(def eqn4 {:lhs "e^(1)",` `:rhs "0.1875*m^(1)*v^(2)"})`
$e = \frac{1}{2}mv^2 + ma$	`(def eqn5 {:lhs "e^(1)",` `:rhs {:term1 "0.5*m^(1)*v^(2)",` `:term2 "m^(1)*a^(1)"}})`

The defined variables/parameters used in the equations are

```
(def varpars [{:symbol "e", :dimension "energy"}
              {:symbol "m", :dimension "mass"}
              {:symbol "v", :dimension " velocity "}
              {:symbol "a", :dimension "acceleration"}])
```

Then, running the consistency check using the `consistent?` function we get

```
=> (consistent? varpars eqn1)
false

=> (consistent? varpars eqn2)
true

=> (consistent? varpars eqn3)
false

=> (consistent? varpars eqn4)
true

=> (consistent? varpars eqn5)
false
```

Thus, $e = {}^{1}/_{2}\,mv^2$ and $e = {}^{3}/_{16}\,mv^2$ are dimensionally correct equations. But which of these two is the actual “correct” equation? Consistency checking cannot answer this. For this particular example, referring to the definition of kinetic energy we know that $e = {}^{1}/_{2}\,mv^2$ is the correct equation.

Standardizing a formula.

What if the user has determined the correct equation and would like to perform dimensional analysis on another equation such that it has expressions that incorporate the correct equation? Can the user avoid again deriving the dimensional formula for the equation he/she knows is correct? diman© provides the flexibility to insert the previously determined correct equation into `standard_formula`. This is a collection of predefined formulae available in diman©.

After importing this predefined collection with

```
(require '[diman.dimensions :refer [standard_formula]])
```

the predefined formulae in `standard_formula` can be viewed with

```
 => (pprint standard_formula)
 [{:quantity "volume", :sformula "[M^(0)*L^(3)*T^(0)]"}
  {:quantity "velocity", :sformula "[M^(0)*L^(1)*T^(-1)]"}
  {:quantity "acceleration", :sformula "[M^(0)*L^(1)*T^(-2)]"}
  {:quantity "force", :sformula "[M^(1)*L^(1)*T^(-2)]"}
  {:quantity "mass density", :sformula "[M^(1)*L^(-3)*T^(0)]"}]
```

Let us illustrate how the user can extend this collection of predefined formulae using $e = {}^{1}/_{2}\,mv^2$ as our correct equation. Since we know that the correct dimensional formula for kinetic energy is

```
 => (formula-eqn-side varpars (:rhs eqn2))
 "[M^(1)*T^(-2)*L^(2)]"
```

our objective is to insert `[M^(1)*T^(-2)*L^(2)]` into the `standard_formula`.

To do this we first create a new collection.

```
(def updated_sform
  (conj standard_formula
        {:quantity "energy",
         :sformula "[M^(1)*T(-2)*L^(2)]"}))
```

The newly defined collection however needs to replace the `standard_formula`. This can be done with

```
(intern 'diman.dimensions 'standard_formula updated_sform)
```

One can check the updated `standard_formula` with

```
=> (pprint standard_formula)
[{:quantity "volume", :sformula "[M^(0)*L^(3)*T^(0)]"}
 {:quantity "velocity", :sformula "[M^(0)*L^(1)*T^(-1)]"}
 {:quantity "acceleration", :sformula "[M^(0)*L^(1)*T^(-2)]"}
 {:quantity "force", :sformula "[M^(1)*L^(1)*T^(-2)]"}
 {:quantity "mass density", :sformula "[M^(1)*L^(-3)*T^(0)]"}
 {:quantity "energy", :sformula "[M^(1) *T(-2)*L^(2)]"}]
```

Furthermore, let say the variables/parameters in the equation were already defined prior to the update. Then this definition can be updated with

```
(def varpars (conj varpars {:symbol "e", :dimension "energy"}))
```

Thus,

```
=> (pprint varpars)
[{:symbol "m", :dimension "mass"}
 {:symbol "v", :dimension "velocity"}
 {:symbol "a", :dimension "acceleration"}
 {:symbol "e", :dimension "energy"}]
```