**Calculating (deriving) Dimensionless Products**

Let us imagine we derived an expression given below.

Replacing each term with the variables *p*, *q*, *r* and *s* we get,

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* **"mass"**}  {*:symbol* **"y"**, *:dimension* **"length"**}  {*:symbol* **"t"**, *:dimension* **"time"**}]) |  |

**Expressions and equation setup.**

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

|  |  |  |
| --- | --- | --- |
|  | (**def** p\_equation {*:lhs* **"p^(1)"**,  *:rhs* {*:term1* **"x^(2)\*y^(-1)\*t^(1)"**}})  (**def** q\_equation {*:lhs* **"q^(1)"**,  *:rhs* {*:term1* **"x^(1)\*y^(6)\*t^(20)"**}})  (**def** r\_equation {*:lhs* **"r^(1)"**,  *:rhs* {*:term1* **"x^(3)\*y^(-3)\*t^(-3)"**}})  (**def** s\_equation {*:lhs* **"s^(1)"**,  *:rhs* {*:term1* **"x^(4)\*t^(8)"**}}) |  |

Then

|  |
| --- |
| (**require** '[diman.formula *:refer* [formula-eqn-side formula-eqn-side-manifold]])  (**require** '[diman.dimensions *:refer* [standard\_formula update-sformula]]) |

|  |
| --- |
| => (formula-eqn-side varpars (:rhs p\_equation))  "[M^(2)\*T^(1)\*L^(-1)]" |

|  |
| --- |
| => (formula-eqn-side varpars (:rhs q\_equation))  "[M^(1)\*T^(20)\*L^(6)]" |

|  |
| --- |
| => (formula-eqn-side varpars (:rhs r\_equation))  "[M^(3)\*T^(-3)\*L^(-3)]" |

|  |
| --- |
| => (formula-eqn-side varpars (:rhs s\_equation))  "[M^(4)\*T^(8)]" |

These equations van be evaluated in one step using the formula-eqn-side-manifold function.

Let us prepare the argument

|  |  |  |
| --- | --- | --- |
|  | (**def** manifold\_eqn  [{*:name* **"name-p"**, *:eqn* (*:rhs* p\_equation)}  {*:name* **"name-q"**, *:eqn* (*:rhs* q\_equation)}  {*:name* **"name-r"**, *:eqn* (*:rhs* r\_equation)}  {*:name* **"name-s"**, *:eqn* (*:rhs* s\_equation)}]) |  |

Then

|  |
| --- |
| => (pprint (formula-eqn-side-manifold varpars manifold\_eqn))  [{:quantity "name-p", :formula "[L^(-1)\*M^(2)\*T^(1)]"}  {:quantity "name-q", :formula "[L^(6)\*M^(1)\*T^(20)]"}  {:quantity "name-r", :formula "[T^(-3)\*L^(-3)\*M^(3)]"}  {:quantity "name-s", :formula "[M^(4)\*T^(8)]"}] |

Next, to have these formulae as part of the standard\_formula definition we invoke the update-sformula function. To do this just pass the result returned from calling the formula-eqn-side-manifold function.

|  |
| --- |
| => (update-sformula (formula-eqn-side-manifold varpars manifold\_eqn))  [{:quantity "volume", :sformula "[L^(3)]"}  {:quantity "frequency", :sformula "[T^(-1)]"}  {:quantity "velocity", :sformula "[L^(1)\*T^(-1)]"}  {:quantity "acceleration", :sformula "[L^(1)\*T^(-2)]"}  {:quantity "force", :sformula "[M^(1)\*L^(1)\*T^(-2)]"}  …  {:quantity "name-s", :sformula "[M^(4)\*T^(8)]"}  {:quantity "name-r", :sformula "[T^(-3)\*L^(-3)\*M^(3)]"}  {:quantity "name-q", :sformula "[L^(6)\*M^(1)\*T^(20)]"}  {:quantity "name-p", :sformula "[L^(-1)\*M^(2)\*T^(1)]"}] |

We are now ready for defining the equation we are interested in

|  |  |  |
| --- | --- | --- |
|  | (**def** leftside **"u^(1)"**) (**def** rightside {*:term1* **"p^(1)",** *:term2* **"q^(1)"**  *:term3* **"q^(1)",** *:term4* **"r^(1)"**}) (**def** equation {*:lhs* leftside, *:rhs* rightside}) |  |

Thus,

|  |
| --- |
| => (formula-eqn-side varpars (:rhs equation))  "[L^(-1)\*M^(2)\*T^(1)] + [L^(6)\*M^(1)\*T^(20)] + [L^(6)\*M^(1)\*T^(20)] + [T^(-3)\*L^(-3)\*M^(3)]" |

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

|  |  |
| --- | --- |
| (**require** '[diman.buckingham  [dimensional-matrix *:refer* [generate-dimmat]]  [homogeneous-equation :refer  [get-augmented-matrix solve get-solved-matrix]]  [dimensionless-product :refer  [get-dimensionless-products get-pi-expression]]]) |  |

|  |  |  |
| --- | --- | --- |
|  | (**def** varpars2 [{*:symbol* **"p"**, *:dimension* **"name-p"**}  {*:symbol* **"q"**, *:dimension* **"name-q"**}  {*:symbol* **"r"**, *:dimension* **"name-r"**}  {*:symbol* **"s"**, *:dimension* **"name-s"**}]) |  |

Since the definition of the symbols *p*, *q*, *r* and *s* are now part of the standard\_formula, passing their definition into the generate-dimmat function will return the dimensional matrix for the above equation.

|  |
| --- |
| => (generate-dimmat varpars2)  [[1N 20N -3N 8N]  [2N 1N 3N 4N]  [-1N 6N -3N 0]] |

This one is a 3 × 4 dimensional matrix.

The next step is to get its homogeneous equation which in other words is its augmented matrix. To get this pass the dimensional matrix to the get-augmented-matrix function.

|  |
| --- |
| => (get-augmented-matrix (generate-dimmat varpars2))  [[-3N 8N -1N -20N]  [3N 4N -2N -1N]  [-3N 0 1N -6N]] |

To solve the augmented matrix pass it into the solve function.

|  |
| --- |
| => (solve (get-augmented-matrix (generate-dimmat varpars2)))  [[1N 0N -1/3 2N]  [0N 1N -1/4 -7/4]  [0N 0N 0N 0N]] |

However, the matrix returned by the solve function is not the solution matrix for deriving the dimensionless products.

To get our desired solution matrix pass the matrix returned by the solve function into the get-solved-matrix function.

|  |
| --- |
| => (get-solved-matrix (solve (get-augmented-matrix (generate-dimmat varpars))))  [[1 0 -1/3 -1/4]  [0 1 2N -7/4]] |

Notice that this solution matrix is 2 × 4. Therefore, there will be two dimensionless products.

|  |  |  |
| --- | --- | --- |
|  | (**def** solution\_matrix  (**get-solved-matrix**  (**solve**  (**get-augmented-matrix**  (**generate-dimmat** varpars2))))) |  |

To get the dimensionless products pass the solution matrix and the definition to get-dimensionless-products function.

|  |
| --- |
| => (get-dimensionless-products solution\_matrix varpars2)  [{:symbol "pi0", :expression "p^(1)\*r^(-1/3)\*s^(-1/4)"}  {:symbol "pi1", :expression "q^(1)\*r^(2)\*s^(-7/4)"}] |

Thus the expression for the dimensionless product is

|  |
| --- |
| => (get-pi-expression (get-dimensionless-products solution\_matrix varpars2) "pi0")  "p^(1)\*r^(-1/3)\*s^(-1/4)" |

Let us now consider another case where we derived an expression given below.

We set the definitions for the symbols in the expression.

|  |  |  |
| --- | --- | --- |
|  | (**def** varpars [{*:symbol* **"x"**, *:dimension* **"mass"**}  {*:symbol* **"y"**, *:dimension* **"length"**}  {*:symbol* **"t"**, *:dimension* **"time"**}]) |  |

and their defined expressions.

|  |  |  |
| --- | --- | --- |
|  | (**def** p\_equation {*:lhs* **"p^(1)"**,  *:rhs* {*:term1* **"x^(2)\*y^(1)"**}})  (**def** q\_equation {*:lhs* **"q^(1)"**,  *:rhs* {*:term1* **"x^(-1)\*t^(1)"**}})  (**def** r\_equation {*:lhs* **"r^(1)"**,  *:rhs* {*:term1* **"x^(3)\*y^(-1)"**}})  (**def** s\_equation {*:lhs* **"s^(1)"**,  *:rhs* {*:term1* **"t^(3)"**}})  (**def** t\_equation {*:lhs* **"t^(1)"**,  *:rhs* {*:term1* **"y^(2)\*t^(1)"**}})  (**def** u\_equation {*:lhs* **"u^(1)"**,  *:rhs* {*:term1* **"x^(-2)\*y^(1)\*t^(-1)"**}})  (**def** v\_equation {*:lhs* **"v^(1)"**,  *:rhs* {*:term1* **"x^(1)\*y^(2)\*t^(2)"**}}) |  |

Putting all these expression in one definition we get

|  |  |  |
| --- | --- | --- |
|  | (**def** manifold\_eqn  [{*:name* **"name-p"**, *:eqn* (*:rhs* p\_equation)}  {*:name* **"name-q"**, *:eqn* (*:rhs* q\_equation)}  {*:name* **"name-r"**, *:eqn* (*:rhs* r\_equation)}  {*:name* **"name-s"**, *:eqn* (*:rhs* s\_equation)}  {*:name* **"name-t"**, *:eqn* (*:rhs* t\_equation)}  {*:name* **"name-u"**, *:eqn* (*:rhs* u\_equation)}  {*:name* **"name-v"**, *:eqn* (*:rhs* v\_equation)}]) |  |

Then

|  |
| --- |
| => (pprint (formula-eqn-side-manifold varpars manifold\_eqn))  [{:quantity "name-p", :formula "[M^(2)\*L^(1)]"}  {:quantity "name-q", :formula "[M^(-1)\*T^(1)]"}  {:quantity "name-r", :formula "[L^(-1)\*M^(3)]"}  {:quantity "name-s", :formula "[T^(3)]"}  {:quantity "name-t", :formula "[L^(2)\*T^(1)]"}  {:quantity "name-u", :formula "[T^(-1)\*M^(-2)\*L^(1)]"}  {:quantity "name-v", :formula "[M^(1)\*L^(2)\*T^(2)]"}] |

Adding these formulae as part of the standard\_formula definition we get

|  |
| --- |
| => (update-sformula (formula-eqn-side-manifold varpars manifold\_eqn))  [{:quantity "volume", :sformula "[L^(3)]"}  {:quantity "frequency", :sformula "[T^(-1)]"}  {:quantity "velocity", :sformula "[L^(1)\*T^(-1)]"}  {:quantity "acceleration", :sformula "[L^(1)\*T^(-2)]"}  {:quantity "force", :sformula "[M^(1)\*L^(1)\*T^(-2)]"}  {:quantity "mass density", :sformula "[M^(1)\*L^(-3)]"}  {:quantity "energy", :sformula "[M^(1)\*L^(2)\*T^(-2)]"}  …  {:quantity "name-v", :sformula "[M^(1)\*L^(2)\*T^(2)]"}  {:quantity "name-u", :sformula "[T^(-1)\*M^(-2)\*L^(1)]"}  {:quantity "name-t", :sformula "[L^(2)\*T^(1)]"}  {:quantity "name-s", :sformula "[T^(3)]"}  {:quantity "name-r", :sformula "[L^(-1)\*M^(3)]"}  {:quantity "name-q", :sformula "[M^(-1)\*T^(1)]"}  {:quantity "name-p", :sformula "[M^(2)\*L^(1)]"}] |

Since our equation of interest is

|  |  |  |
| --- | --- | --- |
|  | (**def** varpars2 [{*:symbol* **"p"**, *:dimension* **"name-p"**}  {*:symbol* **"q"**, *:dimension* **"name-q"**}  {*:symbol* **"r"**, *:dimension* **"name-r"**}  {*:symbol* **"s"**, *:dimension* **"name-s"**}  {*:symbol* **"t"**, *:dimension* **"name-t"**}  {*:symbol* **"u"**, *:dimension* **"name-u"**}  {*:symbol* **"v"**, *:dimension* **"name-v"**}]) |  |

We solve its dimensional matrix.

|  |  |  |
| --- | --- | --- |
|  | (**def** solution\_matrix  (**get-solved-matrix**  (**solve**  (**get-augmented-matrix**  (**generate-dimmat** varpars2))))) |  |

Thus,

|  |
| --- |
| => solution\_matrix  [[1 0 0 0 -11N 5N 8N]  [0 1 0 0 9N -4N -7N]  [0 0 1 0 -9N 5N 7N]  [0 0 0 1 15N -6N -12N]] |

Notice that this solution matrix is 4 × 7. Therefore, there will be four dimensionless products which are

|  |
| --- |
| => (get-dimensionless-products solution\_matrix varpars2)  [{:symbol "pi0", :expression "p^(1)\*t^(-11)\*u^(5)\*v^(8)"}  {:symbol "pi1", :expression "q^(1)\*t^(9)\*u^(-4)\*v^(-7)"}  {:symbol "pi2", :expression "r^(1)\*t^(-9)\*u^(5)\*v^(7)"}  {:symbol "pi3", :expression "s^(1)\*t^(15)\*u^(-6)\*v^(-12)"}] |

Thus the expression for the dimensionless product is

|  |
| --- |
| => (get-pi-expression (get-dimensionless-products solution\_matrix varpars2) "pi2")  "r^(1)\*t^(-9)\*u^(5)\*v^(7)" |

Let us now consider

To get the dimensionless

Since the definition of the symbols *p*, *q*, *r* and *s* are now part of the standard\_formula, passing their definition into the generate-dimmat function will return the dimensional matrix for the above equation.

|  |  |  |
| --- | --- | --- |
|  | (**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) |  |

nn

|  |  |  |
| --- | --- | --- |
|  | (**def** leftside **"u^(1)"**) (**def** rightside {*:term1* **"p^(1)"**,  *:term2* **"q^(1)"**,  *:term3* **"r^(1)"**,  *:term4* **"s^(1)"**}) (**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.

where 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

That is, the equation *fails* the consistency check. Thus,

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

What about the equation ?

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

Thus, equation *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** |
|  | (**def** eqn1 {*:lhs* **"e^(1)"**,  *:rhs* **"m^(2)\*v^(2)"**}) |
|  | (**def** eqn2 {*:lhs* **"e^(1)"**,  *:rhs* **"0.5\*m^(1)\*v^(2)"**}) |
|  | (**def** eqn3 {*:lhs* **"e^(1)"**,  *:rhs* **"m^(1)\*a^(1)"**}) |
|  | (**def** eqn4 {*:lhs* **"e^(1)"**,  *:rhs* **"0.1875\*m^(1)\*v^(2)"**}) |
|  | (**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, and 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 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 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"}] |