Computation of Minimal Units Monomials

Daniel Lichtblau
Wolfram Research, Inc.
100 Trade Center Dr.
Champaign IL USA 61820
danl@wolfram.com

ACA 2007
Algebraic and Numerical Computation for Engineering and Optimization Problems Session
Oakland University
Rochester MI
July 19−22, 2007

Abstract

In this talk I will consider the following problem. We are given a "units monomial", that is, a product of (possibly negative) integer powers of physical units, e.g. $\frac{\text{meters}^2\,\text{volts}}{\text{farads}\,\text{seconds}^2}$. We might try to make sense of this by finding all equivalent monomials subject to a minimality condition. Good candidates for such a condition involve minimizing exponents. For example one might minimize the sum of absolute values of exponents, or minimize the larger of the sum of numerator and sum of denominator exponents. Given a set of algebraic relations between pairs of such monomials, we will readily set these up as problems in integer linear programming, and discuss various ways in which it might be solved via algebraic or numeric programming.

This arose in−house several months ago in the context of a web site currently under development at Wolfram Research. Hence the ability to tackle it with reasonable computational efficiency (i.e. in real time) is paramount.

Afterword acknowledgment: I thank session organizer Dmitry Chibisov and several attendees, in particular Jaime Villate and Georg Regensberger, whose questions following the talk made me rethink various aspects of this problem.

The Setup

We begin with a set of relations between units.

```
units = {Farads, Ohms, Pascals, Watts, Webers, Teslas, Henries,
   Newtons, Joules, Volts, Amperes, Coulombs, Seconds, Kilograms, Meters};
{f, o, p, wa, we, t, h, n, j, v, a, c, s, k, m} = units;
```

relations = $\left\{f - \frac{c^2 s^2}{k\, m^2},\ o - \frac{k\, m^2}{c^2 s},\ p - \frac{k}{m\, s^2},\ -\frac{k\, m^2}{s^3} + wa,\ -\frac{k\, m^2}{c\, s} + we,\ -\frac{k}{c\, s} + t,\ h - \frac{k\, m^2}{c^2},\ n - \frac{k\, m}{s^2},\ j - \frac{k\, m^2}{s^2},\ -\frac{k\, m^2}{c\, s^2} + v,\ a - \frac{c}{s}\right\}$;

```
polyrelations = Numerator[Together[relations]]
```

$\{$Farads Kilograms Meters^2 − Coulombs^2 Seconds^2,
−Kilograms Meters^2 + Coulombs^2 Ohms Seconds, −Kilograms + Meters Pascals Seconds^2,
−Kilograms Meters^2 + Seconds^3 Watts, −Kilograms Meters^2 + Coulombs Seconds Webers,
−Kilograms + Coulombs Seconds Teslas, Coulombs^2 Henries − Kilograms Meters^2,
−Kilograms Meters + Newtons Seconds^2, −Kilograms Meters^2 + Joules Seconds^2,
−Kilograms Meters^2 + Coulombs Seconds^2 Volts, −Coulombs + Amperes Seconds$\}$

We now have a set of binomial relations.

The Problem

Suppose we are now given some units monomial. In order to make sense of it one might first try to find all equivalent forms (equivalences as per the above relations), subject to a minimality condition. One natural such condition (we will discuss others) is to minimize the sum of numerator and denominator total degrees.

Recall: A method for integer programming [Conti and Traverso 1991] is to conver a stardardinteger linear program (ILP) to binomial toric ideal, then use Gröbner bases to do an optimization.

We will do the opposite. Our set of relations can be regarded as defining a toric ideal. We will convert to an ILP, then use some optimization methods from that area [Aardal, Hurkins, and Lenstra 2000; Aardal, Weismantel, and Wolsey 2002; Lichtblau 2006].

Reformulation as an ILP

We begin by converting monomial power products to exponent vectors, in effect taking logarithms. We moreover will encapsulate units, using *x* in effect as a generic variable "head". For example the binomial relation

$$\text{Farads Kilograms Meters}^2 = \text{Coulombs}^2\ \text{Seconds}^2$$

will be transformed to the linear relation

```
x[Farads] + x[Kilograms] + 2 x[Meters] =
 2 x[Coulombs] + 2 x[Seconds]
```

```
tmp1 = polyrelations /. {Plus → pplus, Times → llist};
tmp2 = tmp1 /. a_Symbol ^e1_. /; MemberQ[units, a] :→ e1 * x[a];
tmp3 = tmp2 /. llist[-1, a__] :→ -Map[-1 * # &, llist[a]];
linearrelations = tmp3 /. {llist → Plus, pplus → Subtract}
```

```
{-2 x[Coulombs] + x[Farads] + x[Kilograms] + 2 x[Meters] - 2 x[Seconds],
 -2 x[Coulombs] + x[Kilograms] + 2 x[Meters] - x[Ohms] - x[Seconds],
 x[Kilograms] - x[Meters] - x[Pascals] - 2 x[Seconds],
 x[Kilograms] + 2 x[Meters] - 3 x[Seconds] - x[Watts],
 -x[Coulombs] + x[Kilograms] + 2 x[Meters] - x[Seconds] - x[Webers],
 -x[Coulombs] + x[Kilograms] - x[Seconds] - x[Teslas],
 2 x[Coulombs] + x[Henries] - x[Kilograms] - 2 x[Meters],
 x[Kilograms] + x[Meters] - x[Newtons] - 2 x[Seconds],
 -x[Joules] + x[Kilograms] + 2 x[Meters] - 2 x[Seconds],
 -x[Coulombs] + x[Kilograms] + 2 x[Meters] - 2 x[Seconds] - x[Volts],
 -x[Amperes] + x[Coulombs] - x[Seconds]}
```

Reformulation...

For each variable we will create its "reciprocal" variable. In the multiplicative setting this amounts to making variables invertible. We use the dedicated head *xr* for the reciprocals. We will be making use of these in the additive setting, as a way of creating new relations. This in turn is needed in order to readily formulate an objective function as a sum of nonnegatives.

```
newvars = Join[Map[x[#] &, units], Map[xr[#] &, units]];
reciprelations = Map[x[#] + xr[#] &, units];
allrelations = Join[linearrelations , reciprelations]
```

```
{-2 x[Coulombs] + x[Farads] + x[Kilograms] + 2 x[Meters] - 2 x[Seconds],
 -2 x[Coulombs] + x[Kilograms] + 2 x[Meters] - x[Ohms] - x[Seconds],
 x[Kilograms] - x[Meters] - x[Pascals] - 2 x[Seconds],
 x[Kilograms] + 2 x[Meters] - 3 x[Seconds] - x[Watts],
 -x[Coulombs] + x[Kilograms] + 2 x[Meters] - x[Seconds] - x[Webers],
 -x[Coulombs] + x[Kilograms] - x[Seconds] - x[Teslas],
 2 x[Coulombs] + x[Henries] - x[Kilograms] - 2 x[Meters],
 x[Kilograms] + x[Meters] - x[Newtons] - 2 x[Seconds],
 -x[Joules] + x[Kilograms] + 2 x[Meters] - 2 x[Seconds],
 -x[Coulombs] + x[Kilograms] + 2 x[Meters] - 2 x[Seconds] - x[Volts],
 -x[Amperes] + x[Coulombs] - x[Seconds], x[Farads] + xr[Farads],
 x[Ohms] + xr[Ohms], x[Pascals] + xr[Pascals], x[Watts] + xr[Watts],
 x[Webers] + xr[Webers], x[Teslas] + xr[Teslas], x[Henries] + xr[Henries],
 x[Newtons] + xr[Newtons], x[Joules] + xr[Joules], x[Volts] + xr[Volts],
 x[Amperes] + xr[Amperes], x[Coulombs] + xr[Coulombs],
 x[Seconds] + xr[Seconds], x[Kilograms] + xr[Kilograms], x[Meters] + xr[Meters]}
```

Recast as a lattice

We take the coefficients of these relations as generators of a lattice.

```
lat = Normal[CoefficientArrays[allrelations, newvars][[2]]];
```

Now create a "nice" basis for purposes of subsequent use; for this we simply use lattice reduction.

(This is an efficiency in cases where coefficients are large; it is not helpful for the specific problem at hand.)

```
nulls = LatticeReduce[lat];
```

We show the first few of these.

```
Take[nulls, 3]
```

```
{{-1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, -1, -1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}
```

For each null vector we will need an auxilliary variable.

```
nullvars = Array[mm, Length[nulls]];
```

Finding Equivalences of Given Units Monomials

Given a units monomial, we wish to find equivalent forms subject to the objective that total degree sums of numerator and denominator be minimized. We will

(1) Rewrite the monomial as an exponent vector.

(2) Add indeterminate sums of each null vector (that is, each member from our basis of equivalences).

(3) Add constraints that all indeterminates take on integer values.

(4) Add constraints that all entries in our vector take on nonnegative values (remember, we have "reciprocal" variables, so denominator negative exponents in effect become reciprocal variable positive exponents).

(5) Find a resulting equivalent form that minimizes the objective.

We do not use the result immediately, but rather require the minimal sum of degrees of numerator and denominator for the next step.

(6) Use the diophantine solving capabilities of *Mathematica*'s `Reduce` function to find ALL minimal equivalent monomials.

The Actual Code...

...is almost shorter in character count than the description above (but only because some of the needed code is also above).

```
minimalUnits[expr_] :=
 Module[{input, llist, sol1, newpolys, constrnt, obj, min, vals, allsols},
  input = ((expr /. Times → llist) /. a_Symbol ^e1_. /; MemberQ[units, a] ⧴ e1 * x[a]) /.
     llist → Plus;
  sol1 = Normal[CoefficientArrays[input, newvars][[2]]];
  newpolys = sol1 + Apply[Plus, nullvars * nulls];
  constrnt = Map[# ≥ 0 &, newpolys];
  obj = Total[newpolys];
  {min, vals} =
   Minimize[{obj, Append[constrnt, Element[nullvars, Integers]]}, nullvars];
  allsols = Reduce[Flatten[{obj == min, constrnt, Element[nullvars, Integers]}],
     nullvars];
  allsols = {ToRules[allsols]};
  Map[Apply[Times, newvars ^(newpolys /. #) /. {x[a_] ⧴ a, xr[a_] ⧴ 1 / a}] &, allsols]]
```

An Example

We start with some wild monomial of units and seek all equivalent such monomials of smallest sum of total degrees.

$$\mathbf{Timing}\left[\mathbf{minimalUnits}\left[\frac{c^2\ m^4\ p\ k^3\ v^2}{h^3\ t^7\ wa^2}\right]\right]$$

$$\Big\{3.315,\ \Big\{\frac{\text{Amperes Farads}^4\ \text{Joules}^2\ \text{Meters}}{\text{Teslas}^4},$$

$$\frac{\text{Amperes Coulombs}^2\ \text{Farads}^3\ \text{Joules Meters}}{\text{Teslas}^4},\ \frac{\text{Amperes Coulombs}^4\ \text{Farads}^2\ \text{Meters}}{\text{Teslas}^4},$$

$$\frac{\text{Coulombs Farads}^4\ \text{Joules Meters Watts}}{\text{Teslas}^4},\ \frac{\text{Coulombs}^3\ \text{Farads}^3\ \text{Meters Watts}}{\text{Teslas}^4},$$

$$\frac{\text{Amperes}^2\ \text{Farads}^4\ \text{Joules}^2}{\text{Pascals Teslas}^3},\ \frac{\text{Amperes}^2\ \text{Coulombs}^2\ \text{Farads}^3\ \text{Joules}}{\text{Pascals Teslas}^3},$$

$$\frac{\text{Amperes}^2\ \text{Coulombs}^4\ \text{Farads}^2}{\text{Pascals Teslas}^3},\ \frac{\text{Amperes Coulombs Farads}^4\ \text{Joules Watts}}{\text{Pascals Teslas}^3},$$

$$\frac{\text{Amperes Coulombs}^3\ \text{Farads}^3\ \text{Watts}}{\text{Pascals Teslas}^3},\ \frac{\text{Farads}^5\ \text{Joules Watts}^2}{\text{Pascals Teslas}^3},\ \frac{\text{Coulombs}^2\ \text{Farads}^4\ \text{Watts}^2}{\text{Pascals Teslas}^3},$$

$$\frac{\text{Coulombs Farads}^3\ \text{Joules}^2\ \text{Meters}}{\text{Ohms Teslas}^4},\ \frac{\text{Coulombs}^3\ \text{Farads}^2\ \text{Joules Meters}}{\text{Ohms Teslas}^4},$$

$$\frac{\text{Coulombs}^5\ \text{Farads Meters}}{\text{Ohms Teslas}^4},\ \frac{\text{Farads}^4\ \text{Joules}^2\ \text{Newtons}}{\text{Teslas}^5},\ \frac{\text{Coulombs}^2\ \text{Farads}^3\ \text{Joules Newtons}}{\text{Teslas}^5},$$

$$\frac{\text{Coulombs}^4\ \text{Farads}^2\ \text{Newtons}}{\text{Teslas}^5},\ \frac{\text{Farads}^4\ \text{Joules}^3}{\text{Henries Pascals Teslas}^3},\ \frac{\text{Coulombs}^2\ \text{Farads}^3\ \text{Joules}^2}{\text{Henries Pascals Teslas}^3},$$

$$\frac{\text{Coulombs}^4\ \text{Farads}^2\ \text{Joules}}{\text{Henries Pascals Teslas}^3},\ \frac{\text{Coulombs}^6\ \text{Farads}}{\text{Henries Pascals Teslas}^3},\ \frac{\text{Amperes Coulombs Farads}^3\ \text{Joules}^2}{\text{Ohms Pascals Teslas}^3},$$

$$\frac{\text{Amperes Coulombs}^3\ \text{Farads}^2\ \text{Joules}}{\text{Ohms Pascals Teslas}^3},\ \frac{\text{Amperes Coulombs}^5\ \text{Farads}}{\text{Ohms Pascals Teslas}^3},\ \frac{\text{Farads}^4\ \text{Joules}^2\ \text{Watts}}{\text{Ohms Pascals Teslas}^3},$$

$$\frac{\text{Coulombs}^2\ \text{Farads}^3\ \text{Joules Watts}}{\text{Ohms Pascals Teslas}^3},\ \frac{\text{Coulombs}^4\ \text{Farads}^2\ \text{Watts}}{\text{Ohms Pascals Teslas}^3},\ \frac{\text{Farads}^3\ \text{Joules}^3}{\text{Ohms}^2\ \text{Pascals Teslas}^3},$$

$$\frac{\text{Coulombs}^2\ \text{Farads}^2\ \text{Joules}^2}{\text{Ohms}^2\ \text{Pascals Teslas}^3},\ \frac{\text{Coulombs}^4\ \text{Farads Joules}}{\text{Ohms}^2\ \text{Pascals Teslas}^3},\ \frac{\text{Coulombs}^6}{\text{Ohms}^2\ \text{Pascals Teslas}^3}\Big\}\Big\}$$

$$\mathbf{Timing}\left[\mathbf{minimalUnits}\left[\frac{c^2\ m^4\ p\ k^3\ v^2}{h^3\ t^7\ wa^2}\right]\right]$$

The Result?

$$
\begin{gathered}
\Big\{1.93612,\ \Big\{\frac{\text{Amperes Farads}^4\ \text{Joules}^2\ \text{Meters}}{\text{Teslas}^4},\ \frac{\text{Amperes Coulombs}^2\ \text{Farads}^3\ \text{Joules Meters}}{\text{Teslas}^4},\\
\frac{\text{Amperes Coulombs}^4\ \text{Farads}^2\ \text{Meters}}{\text{Teslas}^4},\ \frac{\text{Coulombs Farads}^4\ \text{Joules Meters Watts}}{\text{Teslas}^4},\\
\frac{\text{Coulombs}^3\ \text{Farads}^3\ \text{Meters Watts}}{\text{Teslas}^4},\ \frac{\text{Amperes}^2\ \text{Farads}^4\ \text{Joules}^2}{\text{Pascals Teslas}^3},\ \frac{\text{Amperes}^2\ \text{Coulombs}^2\ \text{Farads}^3\ \text{Joules}}{\text{Pascals Teslas}^3},\\
\frac{\text{Amperes}^2\ \text{Coulombs}^4\ \text{Farads}^2}{\text{Pascals Teslas}^3},\ \frac{\text{Amperes Coulombs Farads}^4\ \text{Joules Watts}}{\text{Pascals Teslas}^3},\\
\frac{\text{Amperes Coulombs}^3\ \text{Farads}^3\ \text{Watts}}{\text{Pascals Teslas}^3},\ \frac{\text{Farads}^5\ \text{Joules Watts}^2}{\text{Pascals Teslas}^3},\ \frac{\text{Coulombs}^2\ \text{Farads}^4\ \text{Watts}^2}{\text{Pascals Teslas}^3},\\
\frac{\text{Coulombs Farads}^3\ \text{Joules}^2\ \text{Meters}}{\text{Ohms Teslas}^4},\ \frac{\text{Coulombs}^3\ \text{Farads}^2\ \text{Joules Meters}}{\text{Ohms Teslas}^4},\\
\frac{\text{Coulombs}^5\ \text{Farads Meters}}{\text{Ohms Teslas}^4},\ \frac{\text{Farads}^4\ \text{Joules}^2\ \text{Newtons}}{\text{Teslas}^5},\ \frac{\text{Coulombs}^2\ \text{Farads}^3\ \text{Joules Newtons}}{\text{Teslas}^5},\\
\frac{\text{Coulombs}^4\ \text{Farads}^2\ \text{Newtons}}{\text{Teslas}^5},\ \frac{\text{Farads}^4\ \text{Joules}^3}{\text{Henries Pascals Teslas}^3},\ \frac{\text{Coulombs}^2\ \text{Farads}^3\ \text{Joules}^2}{\text{Henries Pascals Teslas}^3},\\
\frac{\text{Coulombs}^4\ \text{Farads}^2\ \text{Joules}}{\text{Henries Pascals Teslas}^3},\ \frac{\text{Coulombs}^6\ \text{Farads}}{\text{Henries Pascals Teslas}^3},\ \frac{\text{Amperes Coulombs Farads}^3\ \text{Joules}^2}{\text{Ohms Pascals Teslas}^3},\\
\frac{\text{Amperes Coulombs}^3\ \text{Farads}^2\ \text{Joules}}{\text{Ohms Pascals Teslas}^3},\ \frac{\text{Amperes Coulombs}^5\ \text{Farads}}{\text{Ohms Pascals Teslas}^3},\ \frac{\text{Farads}^4\ \text{Joules}^2\ \text{Watts}}{\text{Ohms Pascals Teslas}^3},\\
\frac{\text{Coulombs}^2\ \text{Farads}^3\ \text{Joules Watts}}{\text{Ohms Pascals Teslas}^3},\ \frac{\text{Coulombs}^4\ \text{Farads}^2\ \text{Watts}}{\text{Ohms Pascals Teslas}^3},\ \frac{\text{Farads}^3\ \text{Joules}^3}{\text{Ohms}^2\ \text{Pascals Teslas}^3},\\
\frac{\text{Coulombs}^2\ \text{Farads}^2\ \text{Joules}^2}{\text{Ohms}^2\ \text{Pascals Teslas}^3},\ \frac{\text{Coulombs}^4\ \text{Farads Joules}}{\text{Ohms}^2\ \text{Pascals Teslas}^3},\ \frac{\text{Coulombs}^6}{\text{Ohms}^2\ \text{Pascals Teslas}^3}\Big\}\Big\}
\end{gathered}
$$

Our original had total degree 24 and the results are of degree 12. While I do not claim one can "make sense" of any of these, I would at least give better odds of doing so than for the original form.

Note this took around 2 seconds. For the purposes of deployment this is acceptable "real time".

Same Example, Different Objective

Monagan and Pearce show a commutative algebra approach to finding an equivalent to a given rational function modulo an ideal, such that the max of total degree of numerator and denominator is minimized. We might instead use that for our objective function. It is easy enough to do this within the framework of ILP. Simply define a new integer variable, *obj* , subject to constraints that *obj* ≥ deg(numerator) and *obj* ≥ deg (denominator).

```
minimalUnits2[expr_] := Module[{input, llist, sol1,
   newpolys, constrnt, min, vals, allsols, numtot, dentot, obj, allvars},
  allvars = Append[nullvars, obj]; input = ((expr /. Times → llist) /.
      a_Symbol ^ e1_. /; MemberQ[units, a] :→ e1 * x[a]) /. llist → Plus;
  sol1 = Normal[CoefficientArrays[input, newvars][[2]]];
  newpolys = sol1 + Apply[Plus, nullvars * nulls];
  constrnt = Map[# ≥ 0 &, newpolys];
  numtot = Total[Take[newpolys, Length[newpolys] / 2]];
  dentot = Total[Drop[newpolys, Length[newpolys] / 2]];
  constrnt = Join[constrnt, {obj ≥ numtot, obj ≥ dentot, Element[allvars, Integers]}];
  {min, vals} = Minimize[{obj, constrnt}, allvars];
  allsols = Reduce[Flatten[{obj == min, constrnt}], allvars];
  allsols = {ToRules[allsols]};
  Map[Apply[Times, newvars ^(newpolys /. #) /. {x[a_] :→ a, xr[a_] :→ 1 / a}] &, allsols]]
```

Result from Second Formulation

```
Timing[minimalUnits2[c^2*m^4*p*k^3*v^2/(h^3*t^7*wa^2)]]
```

$$\left\{1.5441, \left\{\frac{\text{Farads}^3\,\text{Joules}^3}{\text{Ohms}^2\,\text{Pascals}\,\text{Teslas}^3}, \frac{\text{Coulombs}^2\,\text{Farads}^2\,\text{Joules}^2}{\text{Ohms}^2\,\text{Pascals}\,\text{Teslas}^3}, \frac{\text{Coulombs}^4\,\text{Farads}\,\text{Joules}}{\text{Ohms}^2\,\text{Pascals}\,\text{Teslas}^3}, \frac{\text{Coulombs}^6}{\text{Ohms}^2\,\text{Pascals}\,\text{Teslas}^3}\right\}\right\}$$

Notice we have a proper subset of the prior result, that is, those with both numerator and denominator degree equal to 6. Of couse in general this sort of thing need not happen.

A Commutative Algebra Approach

For contrast we might work directly with the binomial equivalences, using methods involving Gröbner bases. To make this work readily we use multiplicative reciprocal relations for each variable (so we can move from numerator to denominator and back). Thus we add a new variable and relation for each existing variable.

```
unitrecips = Map[recip, units];
```

```
polyreciprelations = Map[recip[#] * # - 1 &, units];
```

```
First[polyreciprelations]
```

```
-1 + Farads recip[Farads]
```

```
allrelations = Join[polyrelations , polyreciprelations];
```

```
allvars = Join[units, unitrecips];
```

Compute the Gröbner Basis...

This next step need only be done once, hence is an "off−line" precomputation. Which is fortunate. As we have an objective function in total degree, a degree−first term ordering suffices for our purposes.

```
Timing[
 gbt = GroebnerBasis[allrelations , allvars, MonomialOrder→ DegreeReverseLexicographic];]
```

```
{124.976, Null}
```

Note that *Mathematica* has some rudimentary improvements for toric ideals, and in this case we can get around a factor of 2 speed improvement. I would imagine more specialized programs could do much better still.

```
Timing[gbt = GroebnerBasis‘ToricGroebnerBasis[
    allrelations , allvars, MonomialOrder→ DegreeReverseLexicographic];]
```

```
{53.3153, Null}
```

Our Example Yet Again

We convert the rational expression by moving denominator terms to numerator (using, of course, the reciprocal relations with the new variables).

```
monom = (c^2 m^4 p k^3 v^2)/(h^3 t^7 wa^2) /. z_^(n_ /; n < 0) :> recip[z]^(-n)
```

```
Coulombs^2 Kilograms^3 Meters^4 Pascals Volts^2 recip[Henries]^3 recip[Teslas]^7 recip[Watts]^2
```

Now reduce with respect to the basis, and change reciprocal variables to ordinary ones in the denominator. Note that this takes but a split second.

```
Timing[Last[PolynomialReduce[monom, gbt, allvars,
    MonomialOrder → DegreeReverseLexicographic]] /. recip[z_] :> 1 / z]
```

```
{0.116007, (Coulombs^6 Farads)/(Henries Pascals Teslas^3)}
```

The drawback of course is that one obtains but a single minimal equivalent monomial from the myriad of possibilities. To get others we'd need Gröbner bases with respect to different degree−based term orders.

Another possibility is to use the total degree of the minimal monomial in the original ILP setting, and thereby skip the minimization step. For this example we would save about 30% of the overall time in that manner, obtaining all equivalent monomials in around 1.3 seconds.

Summary

The problem was to compute one or more minimal equivalent units monomial from a given one, where equivalence is modulo a (toric) ideal of relations among units.

We discussed some reasonable notions of minimality and showed a simple way to go about this by recasting as an integer linear program. This is counter to a recent trend in the literature wherein one does the reverse, taking ILPs to a toric ideal setting.

We also showed a straightforward approach using the original binomial setting and a Gröbner basis computation.

Open questions:

Is the Gröbner basis approach more promising than indicated? Quite possibly my formulation in that setting was suboptimal.

Are the module or similar elimination methods for rational function equivalences, as discussed in [Monagan and Pearce 2006], of use? I was unable to coerce the appropriate basis computations to complete in reasonable time. But again, my casting of the problem in that setting might have been less than optimal.

In the ILP setting, is there any form of off line preprocessing similar to the Gröbner basis precomputation, that might make individual equivalence problems significantly faster?

References

K. Aardal, C. A. J. Hurkens, and A. K. Lenstra. Solving a system of linear diophantine equations with lower and upper bounds on the variables. *Mathematics of Operations Research* **25**:427−442, 2000.

K. Aardal, R. Weismantel, and L. A. Wolsey. Non−standard approaches to integer programming. *Discrete Applied Mathematics* **123**:5−74, 2002.

P. Conti and C. Traverso. Gröbner bases and integer programming. *Proceedings of the 9th International Symposium on Applied Algebra, Algebraic Algorithms and Error−Correcting Codes (AAECC−9).* H. F. Mattson, T. Mora, and T. R. N. Rao, eds. Lecture Notes in Computer Science 539, 130−139. Springer−Verlag, 1991.

D. Cox, J. Little, and D. O'Shea. Ideals, Varieties, and Algorithms: An Introduction to Computational ALgebraic Geometry and Commutative Algebra (third edition). Undergraduate Texts in Mathematics. Springer−Verlag, 2007.

D. Lichtblau. Making change and finding repfigits: Balancing a knapsack. *Proceedings of the Second International Congress on Mathematical Software (ICMS 2006)*, A. Iglesias and N. Takayama, eds. Lecture Notes in Computer Science **4151**:182−193. Springer−Verlag. 2006.

M. Monagan and R. Pearce. Rational simplification modulo a polynomial ideal. *Proceedings of the 2006 Symposium on Symbolic and Algebraic Computation (ISSAC 2006)*, J.−G Dumas, ed, 239−245. ACM Press, 2006.

Addendum

After the talk I was asked the following question. Can one find equivalent forms that minimize the number of units utilized? This is of course a very reasonable sort of thing to want to do. But, as it turns out, it seems not so simple. There is a formulation that recasts as an ILP but it uses a constraint on the maximum degree of any unit used. Moreover it seems computationally much slower.

The idea is to use new variables, one for each entry in our vector, which take on the value 1 if the corresponding variable is used, and 0 if not. To use as few as possible we will minimize thir sum. To enforce these values via inequalities (coupled as ever with integrality), we set each less or equal to one, greater or equal to zero, and less or equal to the corresponding nonnegative integer variable (so when it is not used, this new variable is forced to be 0).

How do we make it be 1 when the corresponding variable is used, that is, is at least 1? This part is a bit less than scientific. We choose some reasonable "maximal unit degree" k. We enforce that k times the new variable be at least as large as the corresponding vector entry. This works fine provided there is no solution with that unit raised to power larger than k. From a theoretical viewpoint the maximal such degree could be quite large (this has me in mind of the notorious Mayr & Meyer example involving lexicographic Gröbner bases, and I suspect the underlying issue here is in fact quite similar). But we are taking a practical approach, and so will artificially cap this degree in ways that seem reasonable, in the examples below.

Addendum...

We show the code below, along with our running example. We use as multiple 1/5, which is to say, we somewhat artificially restrict so that no monomial is used to power larger than five.

```
minimalUnits3[expr_] :=
 Module[{input, llist, sol1, newpolys, constrnt1, constrnt2, constrnt3,
   constrnt4, constrnt, obj, min, vals, allsols, zovec, c, allvars,}k
  input = ((expr /. Times → llist) /. a_Symbol ^ e1_. /; MemberQ[units, a] ⧴ e1 * x[a]) /.
    llist → Plus;
  sol1 = Normal[CoefficientArrays[input, newvars][[2]]];
  k = Total[Abs[sol1]] + 5;
  newpolys = sol1 + Apply[Plus, nullvars * nulls];
  constrnt1 = Map[# ≥ 0 &, newpolys];
  zovec = Array[c, Length[newpolys]];
  obj = Total[zovec];
  constrnt2 = Map[0 ≤ # ≤ 1 &, zovec];
  constrnt3 = Thread[zovec ≤ newpolys];
  constrnt4 = Thread[k * zovec ≥ newpolys];
  constrnt = Join[constrnt1, constrnt2, constrnt3, constrnt4];
  allvars = Join[nullvars, zovec];
  {min, vals} = Minimize[{obj, Append[constrnt, Element[allvars, Integers]]}, allvars];
  allsols = Reduce[Flatten[{obj ⩵ min, constrnt, Element[allvars, Integers]}], allvars];
  allsols = {ToRules[allsols]};
  Map[Apply[Times, newvars ^(newpolys /. #) /. {x[a_] ⧴ a, xr[a_] ⧴ 1 / a}] &, allsols]]
```

Addendum...

```
Timing[minimalUnits3[c^2 * m^4 * p * k^3 * v^2 / (h^3 * t^7 * wa^2)]]
```

$$\left\{680.248, \left\{\frac{\text{Amperes}^3\,\text{Farads}^4\,\text{Meters}^5}{\text{Teslas}^2}, \frac{\text{Amperes}^5\,\text{Farads}^4\,\text{Meters}^3}{\text{Pascals}^2}, \frac{\text{Farads}^5\,\text{Meters}^3\,\text{Watts}^2}{\text{Teslas}^3}, \frac{\text{Amperes}^5\,\text{Farads}^5\,\text{Volts}^2}{\text{Pascals}^3}, \frac{\text{Amperes}^5\,\text{Farads}^4\,\text{Joules}}{\text{Pascals}^3}, \frac{\text{Amperes}^5\,\text{Coulombs}^2\,\text{Farads}^3}{\text{Pascals}^3}, \frac{\text{Amperes}^5\,\text{Coulombs}^5}{\text{Pascals}^3\,\text{Volts}^3}, \frac{\text{Amperes}^3\,\text{Farads}^5\,\text{Watts}^2}{\text{Pascals}^3}, \frac{\text{Farads}^5\,\text{Watts}^5}{\text{Pascals}^3\,\text{Volts}^3}, \frac{\text{Farads}^4\,\text{Meters}^2\,\text{Newtons}^3}{\text{Teslas}^5}, \frac{\text{Farads}^4\,\text{Joules}^2\,\text{Newtons}}{\text{Teslas}^5}, \frac{\text{Farads}^4\,\text{Joules}^3}{\text{Meters}\,\text{Teslas}^5}, \frac{\text{Coulombs}^4\,\text{Farads}^2\,\text{Newtons}}{\text{Teslas}^5}, \frac{\text{Farads}^4\,\text{Newtons}^4}{\text{Pascals}\,\text{Teslas}^5}, \frac{\text{Amperes}^2\,\text{Coulombs}^5}{\text{Ohms}^3\,\text{Pascals}^3}, \frac{\text{Coulombs}^5\,\text{Watts}}{\text{Ohms}^4\,\text{Pascals}^3}, \frac{\text{Joules}^5}{\text{Ohms}^5\,\text{Pascals}^3\,\text{Volts}^3}, \frac{\text{Coulombs}^3\,\text{Joules}^2}{\text{Ohms}^5\,\text{Pascals}^3}, \frac{\text{Coulombs}^5\,\text{Volts}^2}{\text{Ohms}^5\,\text{Pascals}^3}, \frac{\text{Coulombs}^3\,\text{Newtons}^3}{\text{Ohms}^5\,\text{Pascals}^4}\right\}\right\}$$

We recover equivalent forms with four distinct units. One cannot but help observe that this is far from a "real time" computation.

Addendum...

Here we get serious and limit individual unit exponents to a modest amount larger than the original total degree. We get a small improvement of only three units utilized, but the timing is yet another an order of magnitude worsened.

```
minimalUnits3[expr_] :=
 Module[{input, llist, sol1, newpolys, constrnt1, constrnt2, constrnt3,
   constrnt4, constrnt, obj, min, vals, allsols, zovec, c, allvars, k},
  input = ((expr /. Times → llist) /. a_Symbol ^ e1_. /; MemberQ[units, a] ⧴ e1 * x[a]) /.
    llist → Plus;
  sol1 = Normal[CoefficientArrays[input, newvars][[2]]];
  k = Total[Abs[sol1]] + 5;
  newpolys = sol1 + Apply[Plus, nullvars * nulls];
  constrnt1 = Map[# ≥ 0 &, newpolys];
  zovec = Array[c, Length[newpolys]];
  obj = Total[zovec];
  constrnt2 = Map[0 ≤ # ≤ 1 &, zovec];
  constrnt3 = Thread[zovec ≤ newpolys];
  constrnt4 = Thread[k * zovec ≥ newpolys];
  constrnt = Join[constrnt1, constrnt2, constrnt3, constrnt4];
  allvars = Join[nullvars, zovec];
  {min, vals} = Minimize[{obj, Append[constrnt, Element[allvars, Integers]]}, allvars];
  allsols = Reduce[Flatten[{obj == min, constrnt, Element[allvars, Integers]}], allvars];
  allsols = {ToRules[allsols]};
  Map[Apply[Times, newvars ^(newpolys /. #) /. {x[a_] ⧴ a, xr[a_] ⧴ 1 / a}] &, allsols]]
```

```
Timing[minimalUnits3[c^2 * m^4 * p * k^3 * v^2 / (h^3 * t^7 * wa^2)]]
```

$$\left\{8503.55,\ \left\{\frac{\text{Kilograms}^4\ \text{Pascals}^3}{\text{Teslas}^{13}}\right\}\right\}$$

Addendum 2

We can formulate the original ILP without reciprocal variables. We use a second set of variables, one per vector entry. We constrain so they are in effect the absolute value of the corresponding vector entries (by making them greater−equal to the negative and positive of their respective entries). Our objective is simply the sum of these new variables. Code is provided below. Speed is comparable at least on our running example.

```
minimalUnits[expr_] := Module[
  {input, llist, sol1, newpolys, constrnt, obj, min, vals, allsols, newvars, c, allvars},
  input = ((expr /. Times → llist) /. a_Symbol ^e1_. /; MemberQ[units, a] ⧴ e1 * x[a]) /.
    llist → Plus;
  sol1 = Normal[CoefficientArrays[input, vars][[2]]];
  newvars = Array[c, Length[sol1]];
  newpolys = sol1 + Apply[Plus, nullvars * nulls];
  obj = Total[newvars];
  allvars = Join[nullvars, newvars];
  constrnt = Join[Thread[newvars ≥ -newpolys], Thread[newvars ≥ newpolys]];
  {min, vals} = Minimize[{obj, Join[constrnt, {Element[allvars, Integers]}]}, allvars];
  allsols =
   Reduce[Flatten[{obj ⩵ min, Join[constrnt, {Element[allvars, Integers]}]}], allvars];
  allsols = {ToRules[allsols]};
  Map[Apply[Times, newvars ^(newpolys /. #) /. {c[j_] ⧴ units[[j]]}] &, allsols]
 ]
```