Computing Dressians

This file implements Algorithm 1, and gives an input file to gfan.

Start here

Input:
n = size of ground set
r = rank
the nonbases (here we use the Desargues configuration as an example)
what you want the file tame to be.

```
hesse = {{0, 1, 2}, {3, 4, 5}, {6, 7, 8}, {0, 3, 6},
    {1, 4, 7}, {2, 5, 8}, {1, 3, 8}, {0, 5, 7}, {1, 5, 6}, {2, 3, 7}};

r = 3;
n = 9;
nonbases = hesse;
name = "hesse";
```

Once you have checked that the Plücker relations for your (n,r) are pre-loaded below, run the entire notebook.

Computation

Pre-loaded Plücker relations

Plucker relations for (2,4)

```
pluckers[{2, 4}] = {p[1, 2] * p[0, 3] - p[0, 2] * p[1, 3] + p[0, 1] * p[2, 3]};
```

Plucker relations for (2,5)

```
pluckers[{2, 5}] = {p[2, 3] * p[1, 4] - p[1, 3] * p[2, 4] + p[1, 2] * p[3, 4],
   p[2, 3] * p[0, 4] - p[0, 3] * p[2, 4] + p[0, 2] * p[3, 4],
   p[1, 3] * p[0, 4] - p[0, 3] * p[1, 4] + p[0, 1] * p[3, 4],
   p[1, 2] * p[0, 4] - p[0, 2] * p[1, 4] + p[0, 1] * p[2, 4],
   p[1, 2] * p[0, 3] - p[0, 2] * p[1, 3] + p[0, 1] * p[2, 3]};
```

Plucker relations for (3,5)

Plucker relations for (4,8)

Plucker relations for (3,6)

Plucker relations for Grassmannian (3,7)

Plucker Relations for Grassmannian (3,8)

Plucker Relations for Grassmannian (3,9)

Plucker Relations for Grassmannian (3,10)

Computation

Commands needed for the computation

```
threeTerm[L_] := Select[L, Length[CoefficientRules[#]] == 3 &]
```

```
redpoly[f_] :=
 If[Length[MonomialList[f]] > 1, Simplify[f / PolynomialGCD @@ MonomialList[f]], f]
```

```
ExponentList[f_] :=
 Table[Exponent[f, Variables[f][[i]]], {i, 1, Length[Variables[f]]}]
```

```
ShrinkIdeal[gens_] := Module[{newgens, badvar,
    twoterms, oldgens, rhs, k, eqn, equations, illegalvars, elimvars},
  newgens = gens;
  equations = {};
  illegalvars = {};
  twoterms = DeleteCases[newgens, x_ /; Length[CoefficientRules[x]] ≠ 2];
  elimvars = Variables[twoterms];
  While[Length[elimvars] > 0,
   badvar = elimvars[[1]];
   eqn = Select[twoterms, Exponent[#, badvar] == 1 &][[1]];
   rhs = Solve[eqn == 0, badvar][[1, 1, 2]];
   oldgens = newgens;
   newgens = {};
   If[MemberQ[Table[Length[CoefficientRules[
           redpoly[Expand[Simplify[ReplaceAll[oldgens[[k]], badvar → rhs] *
               Denominator[Together[ReplaceAll[oldgens[[k]], badvar → rhs]]]]]]]] ≠
         Length[CoefficientRules[oldgens[[k]]]] && Length[CoefficientRules[
           oldgens[[k]]]] == 3, {k, 1, Length[oldgens]}], True],
    AppendTo[illegalvars, badvar]; newgens = oldgens,
    For[k = 1, k ≤ Length[oldgens], k = k + 1,
     If[Exponent[oldgens[[k]], badvar] == 0,
       AppendTo[newgens, oldgens[[k]]],
       AppendTo[newgens,
        redpoly[Expand[Simplify[ReplaceAll[oldgens[[k]], badvar → rhs] *
            Denominator[Together[ReplaceAll[oldgens[[k]], badvar → rhs]]]]]]
       ]
      ];
    ];
    newgens = DeleteCases[newgens, x_ /; Length[CoefficientRules[x]] < 2];
    twoterms =
     DeleteCases[DeleteCases[newgens, x_ /; Length[CoefficientRules[x]] ≠ 2],
      f_ /; Not[MemberQ[ExponentList[f], 1]]];
   ];
   elimvars = Complement[Variables[twoterms], illegalvars];
   AppendTo[equations, {badvar, rhs}]
  ];
  Return[{newgens, equations}]
 ]

oneSide = (Head[#][Subtract @@ #, 0] &);

tozero = Table[
   p[## & @@ Table[nonbases[[i]][[j]], {j, 1, r}]] == 0, {i, 1, Length[nonbases]}];

Dress = Simplify[threeTerm[pluckers[{r, n}]], tozero];
```

```
DeleteCases[Dress, x_ /; Not[MemberQ[Variables[x], p[6, 7, 8]]]];

MakeEquations[n_, nonbases_] :=
  Module[{Dress, tozero, i, smallDress, Eqns, Assmps, Eqnswassmps, theguys},
   tozero = Table[p[## & @@ Table[nonbases[[i]][[j]], {j, 1, r}]] == 0,
     {i, 1, Length[nonbases]}];
   Dress = Simplify[threeTerm[pluckers[{r, n}]], tozero];
   smallDress = ShrinkIdeal[Dress];
   Eqns = Table[smallDress[[1, i]] == 0, {i, 1, Length[smallDress[[1]]]}];
   Assmps =
    Table[Variables[Dress[[1]]][[i]] ≠ 0, {i, 1, Length[Variables[Dress[[1]]]]}];
   Eqnswassmps = DeleteDuplicates[Simplify[Eqns, Assumptions → Assmps]];
   theguys = Table[(Eqnswassmps[[i]] // oneSide)[[1]], {i, 1, Length[Eqnswassmps]}];
   Return[{Expand[theguys], smallDress[[2]]}];
  ];

Length[Dress]
630

Length[Variables[Dress]]
74
```

Formatting and Sending the output to Gfan

```
everything = MakeEquations[n, nonbases];

eqns = everything[[1]];

substitutions = everything[[2]];

Length[eqns]
43

Length[Variables[eqns]]
12

(*sometimes the list of eqns can be shortened by asking
 for GroebnerBasis[eqns,vars] because there could be duplicate
 equations in the list / ones that are in the ideal already*)

Format[p[a__Integer], InputForm] := SequenceForm[p, a]

InputToGfan[eqns_] :=
 {StringReplace[ToString[StringForm["Q``", ToString[InputForm[Variables[eqns]]]]],
   {"{" → "[", "}" → "]"}],
  {StringReplace[ToString[InputForm[eqns]], {"(" → "", ")" → ""}]}}
```

```
SetDirectory["/Applications/gfan"]
```

```
/Applications/gfan
```

```
Export[StringJoin[name, ".txt"], InputToGfan[eqns], "Table"]
```

```
hesse.txt
```

Next

Open terminal and type:
cd /Applications/gfan
./gfan_tropicalintersection <YOURNAME.txt

Polymake things

Input your rays here

```
rays = {{-11, 11, 3, -3, 3, 0, 5, -1, 1, -5, 0, -3},
{1, -1, -3, 3, 2, 0, 0, 1, -1, 0, 0, -2}, {-10, 10, 0, 0, 5, 0, 5, 0, 0, -5, 0, -5}}
```

```
{{-11, 11, 3, -3, 3, 0, 5, -1, 1, -5, 0, -3},
 {1, -1, -3, 3, 2, 0, 0, 1, -1, 0, 0, -2}, {-10, 10, 0, 0, 5, 0, 5, 0, 0, -5, 0, -5}}
```

Hidden computations for making the polymake file

```
initialvars = Variables[eqns];
```

```
Length[initialvars] == Length[rays[[1]]]
```

```
True
```

```
rules =
  Table[Union[Table[initialvars[[i]] → rays[[k, i]], {i, 1, Length[initialvars]}],
    Table[substitutions[[i, 1]] →
      Total[Table[Exponent[Numerator[substitutions[[i, 2]]],
          Variables[Numerator[substitutions[[i, 2]]]][[j]]] *
        Variables[Numerator[substitutions[[i, 2]]]][[j]],
        {j, 1, Length[Variables[Numerator[substitutions[[i, 2]]]]]}]] -
       Total[Table[Exponent[Denominator[substitutions[[i, 2]]],
          Variables[Denominator[substitutions[[i, 2]]]][[j]]] *
        Variables[Denominator[substitutions[[i, 2]]]][[j]],
        {j, 1, Length[Variables[Denominator[substitutions[[i, 2]]]]]}]]
     , {i, 1, Length[substitutions]}]
   ], {k, 1, Length[rays]}];

nonzerovars = Complement[Subsets[Table[i - 1, {i, 1, n}], {r}], nonbases];
```

Makes the weight vector

```
w = Table[-1 * Table[p[## & @@ Table[nonzerovars[[i, j]], {j, 1, r}]],
      {i, 1, Length[nonzerovars]}] //. rules[[k]], {k, 1, Length[rays]}]
```

```
{{-22, -22, -18, -17, -14, -19, -18, -18, -14, -13, -10, -15, -22, -18, -14,
  -19, -18, -17, -14, 11, -13, -15, -9, -14, -11, -8, -8, -4, -3, 0, -5, -12,
  -8, -7, -4, -8, -7, -9, 0, -5, 1, -4, -1, -8, -4, -3, -5, -4, -3, 0, -5, 1,
  4, 5, 0, 3, -7, -4, -9, -3, 0, -5, 1, -4, -1, -3, 0, -5, 1, -4, -1, 5, 0, 3},
 {-7, -5, -4, -5, -3, -3, -9, -7, -6, -7, -5, -5, -5, -4, -3, -3, -2, -3,
  -1, -1, -2, 0, -1, -1, 1, -11, -9, -8, -9, -7, -7, -7, -6, -7, -5, -4,
  -5, -3, -2, -2, -3, -3, -1, -9, -8, -9, -7, -6, 3, -5, -5, -6, -4, -5,
  -5, -3, -5, -3, -3, -4, -2, -2, -3, -3, -1, -2, 0, 0, -1, -1, 1, 0, 0, 2},
 {-27, -26, -21, -21, -16, -21, -26, -25, -20, -20, -15, -20, -26, -21, -16, -21,
  -20, -20, -15, 10, -15, -15, -10, -15, -10, -17, -16, -11, -11, -6, -11, -17, -12,
  -12, -7, -11, -11, -11, -1, -6, -1, -6, -1, -16, -11, -11, -11, -10, 0, -5, -10,
  -5, 0, 0, -5, 0, -11, -6, -11, -6, -1, -6, -1, -6, -1, -5, 0, -5, 0, -5, 0, 5, 0, 5}}
```

Makes the vertices of the polytope, ready for polymake

```
P = Prepend[#, 1] & /@ Table[If[MemberQ[nonzerovars[[i]], j - 1], 1, 0],
    {i, 1, Length[nonzerovars]}, {j, 1, n}];
```

```
InputToPolymake[P_, w_] := Flatten[{StringJoin["$M=new Matrix<Rational>(",
    StringReplace[ToString[P], {"{" → "[", "}" → "]"}], ");"],
   "$p = new Polytope<Rational>(POINTS=>$M);", "sub matroidal {
  my ($edges)=@_;
  foreach my $vector (@$edges) {
     if (1 != grep { $_ == 1 } @$vector or
         1 != grep { $_ == -1 } @$vector or
         2 != grep { $_ != 0 } @$vector) {
         return 0;  # false
     }
  }
  return 1; # true
};", "print matroidal($p->GRAPH->EDGE_DIRECTIONS);",
   Table[
    {StringJoin["$w", ToString[k - 1], " = new Vector<Rational>",
      StringReplace[ToString[w[[k]]], {"{" → "(", "}" → ")"}], ";"],
     StringJoin["$S", ToString[k - 1],
      " = new fan::SubdivisionOfPoints(POINTS=>$M,WEIGHTS=>$w",
      ToString[k - 1], ");"],
StringJoin["$PC", ToString[k - 1], " = $S", ToString[k - 1],
      "->POLYHEDRAL_COMPLEX;"],
StringJoin["print matroidal(edge_directions($PC", ToString[k - 1],
      "->GRAPH, $p->VERTICES));"],
     StringJoin["print $S", ToString[k - 1], "->MAXIMAL_CELLS;"]}
    , {k, 1, Length[rays]}
   ]
  }]
```

Run this to make the polymake code

```
SetDirectory["/Users/maddiebrandt/Desktop/"]
```

```
/Users/maddiebrandt/Desktop
```

```
Export[StringJoin[name, "_polymake.txt"], InputToPolymake[P, w], "Table"]
```

```
hesse_polymake.txt
```