Diagnosinghigherorder singularities(HOS)

Anirudh Chandrasekaran

Loughborough University, United Kingdom

Last Updated: 29 July 2022.

In this notebook, we implement an algorithm for calculating the corank, codimension and determinacy of a multivariate polynomial. There is no restriction on the degree of the polynomial, number of variables or even the choice of symbols for the variables. The function is written to automatically detect the variables in the polynomial and the degree, so that it can be directly fed any polynomial by the user, without the requirement of any pre-processing. In particular, we can feed it Taylor expansion of functions about arbitrary points, to try and diagnose a singularity (if one is present).

```
In[•]:= DiagnoseHOS[Poly_, MaxDeg_ : -1, Resln_ : 10^-6] := Module[
   {PolyDeg, SpaceDim, KVec, Corank, HessMat,
    HessEigs, JacobMat, PolyDerivs, Monoms, AllMonoms, Partns,
    PostvDistPartns, PartnsWithZeros, BasisPolys, CoeffMatrix,
    ZeroList, OneList, BMat, SolMat, Solved, Determinacy, Codimension},

   KVec = Variables[Poly];
   (*We detect the symbolic variables used in the polynomial.*)
   (*Print[KVec];*)

   SpaceDim = Length[KVec];

   (*To compute the corank, we need to compute the Hessian matrix first:*)
   HessMat = First@Last@Reap[
       Do[
        Sow[
         First@Last@Reap[
            Do[
             Sow[ReplaceAll[D[Poly, K1, K2], Table[Ki → 0, {Ki, KVec}]]];
             , {K2, KVec}]
           ]
        ]
        , {K1, KVec}]
      ];

   HessEigs = Chop[Eigenvalues[HessMat], Resln] // Rationalize;

   (*The corank is the number of zero eigenvalues of the Hessian:*)
```

```
Corank = Count[HessEigs, 0];

(*The Jacobian is necessary to determine
 if we have a critical point in the first place:*)
JacobMat = Chop[First@Last@Reap[
      Do[
       Sow[ReplaceAll[D[Poly, K1], Table[Ki → 0, {Ki, KVec}]]]
       , {K1, KVec}]
     ], Resln] // Rationalize;

(*Print[JacobMat];*)
Solved = False;

If[Count[JacobMat, 0] == SpaceDim, (*If all the
  elements of the Jacobian are zero, we have a critical point.*)
 If[Corank > 0, (*If at least one or more eigenvalues
   of the Hessian are zero, indicated by a non-zero corank,
  then we have a higher order critical point.*)

  If[IntegerQ[MaxDeg] && MaxDeg > 0,
   (*Sanity check to ensure that we have a legitimate polunomial.*)
   PolyDeg = MaxDeg;
   ,
   PolyDeg = ResourceFunction["PolynomialDegree"][Poly, KVec] + 1
   (*We add one to the actual degree of the polynomial since the algorithm
     can narrow down the determinacy to either the actual det or det+1,
   so that the check for determinacy has to be performed up to at least up
    one degree higher than the suspected determinacy of the polynomial.*)
  ];

  (*Given polynomial p and variables x₁,x₂,...,xₙ, we need to compute the
   first derivatives of the polynomial with respect to the variables:*)
  PolyDerivs = First@Last@Reap[
      Do[
       Sow[D[Poly, Ki]]
       , {Ki, KVec}]
     ];

  (*As we attempt to check for determinacy at increasing orders,
  at each order k, we first accumulate all the monomials of
   degree k into the list AllMonoms. We then accumulate the
   basis polynomials obtained by multiplying these monomials by
   the derivative polynomials in PolyDerivs, into BasisPolys.*)
  AllMonoms = {};
  BasisPolys = {};
  ZeroList = {};
```

```
On[LinearSolve::nosol];
(*Turning on the notification
 in case it has been turned off by the user.*)

Do[(*A loop to check for determinacy order by
  order up to one degree higher than the actual degree of
  the polynomial (or up to a level specified by the user).*)

 (*Our strategy for constructing all the monomials of
   degree k in n variables is to first compute the distinct
   partitions of k into n positve integers. Mathematica returns
   this as a collection of sets of length n. We then repeat the
   procedure for partitions of k into n-1 positive integers,
 then n-2 positive integers and so on until partition of k into
   one positive integer. For those partitions of k into less than n
   integers (where Mathematica returns sets of length less than n),
 we append zeros into the set to make up n integers in total. After this,
 we generate all the permutations of this set,
 which then gives us all possible ways of writing k =
  i1 + i2 + ... + in for integer ij ≥ 0. This then allows us to
    compute all the monomials of degree k in n variables.*)
 PostvDistPartns = IntegerPartitions[CurrDeg, {SpaceDim}];
 (*Partitions of k into n positive integers.*)

 PartnsWithZeros = {}; (*Partitions of k into less than n
  positive integers. Rest of elements are taken to be zeros.*)

 Do[
  PartnsWithZeros = Join[PartnsWithZeros, (Join[ConstantArray[0, i], #]) &/@
     IntegerPartitions[CurrDeg, {SpaceDim - i}]]
  , {i, 1, SpaceDim - 1}];

 Partns =
  Flatten[Permutations /@ Join[PostvDistPartns, PartnsWithZeros], 1];

 Monoms = First@Last@Reap[
     Do[
      Sow[Product[KVec[[i]]^CurrPartn[[i]], {i, 1, SpaceDim}]]
      , {CurrPartn, Partns}]
    ];

 AllMonoms = Join[AllMonoms, Monoms];
 (*Appending the newly computed monomials into the list AllMonoms.*)

 (*We will be solving the linear system C.B = S,
where C is the coefficient matrix of the base polynomials in
 the basis of monomials(the base polynomials are ontaining by
   multiplying the first derivatives of the original polynomial
   by all the monomials. S is a set containing 1's at the
   position of the monomials of degree k, and zero elsewhere
   (since the aim is to check whether a linear combination of the
     base polynomials can yield all the monomials of degree k))*)
OneList = ConstantArray[1, Length[Monoms]];
BMat = Join[ZeroList, OneList];
ZeroList = Join[ZeroList, ConstantArray[0, Length[Monoms]]];

BasisPolys = Join[BasisPolys,
   First@Last@Reap[
      Do[
       Do[
        Sow[polyderiv monom // Expand]
        , {polyderiv, PolyDerivs}]
       , {monom, Monoms}]
     ]
 ];

CoeffMatrix = First@Last@Reap[
     Do[
      CoeffRow = First@Last@Reap[
          Do[
           Sow[ReplaceAll[
               Coefficient[poly, monom], Table[Ki → 0, {Ki, KVec}]]];
           , {monom, AllMonoms}]
         ];

      Sow[CoeffRow]
      , {poly, BasisPolys}]
    ];

SolMat = Check[LinearSolve[Transpose[CoeffMatrix], BMat], 1];
(*Solving the linear system. If it has a
   solution then the polynomial is k-determinate.*)


If[NumberQ[SolMat] && SolMat == 1, ,
 Solved = True;
 Determinacy = CurrDeg;

 CoeffMatrix = Join[CoeffMatrix,
   First@Last@Reap[
      Do[
       CoeffRow = First@Last@Reap[
              Do[
                Sow[ReplaceAll[Coefficient[
                    polyderiv, monom], Table[Ki → 0, {Ki, KVec}]]];
                , {monom, AllMonoms}]
              ];

          Sow[CoeffRow]

          , {polyderiv, PolyDerivs}]
        ]
      ];

    (*Codimension=(Determinacy^2+3Determinacy)/2-MatrixRank[CoeffMatrix];
    for the two dimensional case.*)

    Codimension = Length[AllMonoms] - MatrixRank[CoeffMatrix];


    Print["Polynomial has corank ", Corank, ". It is ",
     CurrDeg, "-determinate. Its determinacy is either ", CurrDeg,
     " or ", CurrDeg - 1, " and its codimension is ", Codimension,
     ". Comparing with the known list of singularities..."];
    Break[]
   ];

   , {CurrDeg, 1, PolyDeg}];
  ,

  (*If all elements of the Jacobian are zero and Corank = 0,
  we have a good old non-singular quadratic form that could be a maximum,
  minimum or a saddle.*)
  Determinacy = 2;
  Codimension = 1;

  Solved = True;

 ];

Switch[{Corank, Codimension, Determinacy},
  {0, 1, 2},
  Print["Polynomial is equivalent to an ordinary quadratic form with
      a non-singular Hessian."];,
  {1, 1, 2}, Print["Polynomial has a fold (A2) catastrophe."];,
  {1, 1, 3}, Print["Polynomial has a fold (A2) catastrophe."];,
  {1, 2, 3}, Print["Polynomial has a cusp (A3) catastrophe."];,
  {1, 2, 4}, Print["Polynomial has a cusp (A3) catastrophe."];,
  {1, 3, 4}, Print["Polynomial has a swallowtail (A4) catastrophe."];,
  {1, 3, 5}, Print["Polynomial has a swallowtail (A4) catastrophe."];,
  {1, 4, 5}, Print["Polynomial has a butterfly (A5) catastrophe."];,
  {1, 4, 6}, Print["Polynomial has a butterfly (A5) catastrophe."];,
  {1, 5, 6}, Print["Polynomial has a wigwam (A6) catastrophe."];,
  {1, 5, 7}, Print["Polynomial has a wigwam (A6) catastrophe."];,
  {1, 6, 7}, Print["Polynomial has a star (A7) catastrophe."];,
  {1, 6, 8}, Print["Polynomial has a cusp (A7) catastrophe."];,
  {2, 3, 2},
  Print["Polynomial has either an elliptic umbilic (D4-) (monkey saddle)
      or a hyperbolic umbilic (D4+) catastrophe."];,
  {2, 3, 3},
  Print["Polynomial has either an elliptic umbilic (D4-) (monkey saddle)
      or a hyperbolic umbilic (D4+) catastrophe."];,
  {2, 4, 3}, Print["Polynomial has a parabolic umbilic (D5) catastrophe."];,
  {2, 4, 4}, Print["Polynomial has a parabolic umbilic (D5) catastrophe."];,
  {2, 5, 3}, Print["Polynomial has a symbolic umbilic (E6) catastrophe."];,
  {2, 5, 4}, Print["Polynomial has a symbolic umbilic (E6) catastrophe."];,
  {2, 5, 4},
  Print["Polynomial has either a second elliptic umbilic (D6-) or a second
      hyperbolic umbilic (D6+) catastrophe."];,
  {2, 5, 5},
  Print["Polynomial has either a second elliptic umbilic (D6-) or a second
      hyperbolic umbilic (D6+) catastrophe."];,
  _,
  If[Solved == True, Print["Point singularity not identified by name!"],
   Print["The polynomial appears to host a singularity
      that does not have a finite determinacy."];
   Determinacy = ∞;
   Codimension = ∞;
  ]
 ]
 ,

 Solved = True;
 Determinacy = 1;
 Codimension = 0;
 Print[
  "The given polynomial does not have a critical point at the origin. It is
    smoothly equivalent to a linear form, and has determinacy 1."];
];

{Corank, Codimension, Determinacy}

]
```

We shall now benchmark this function against a number of known singularities. We shall start with two examples drawn from page 8 of our earlier paper. The two polynomials treated there are deceptively close to each other and even have almost identical looking constant energy contours (level sets). Nevertheless they are distinct singularities leading to different power law divergence for density of states. We begin with the first one:

```
In[∘]:= DiagnoseHOS[-ky^2 + 6 kx^2 ky - 9 kx^4 + kx^6]

••• LinearSolve : Linear equation encountered that has no solution.

••• LinearSolve : Linear equation encountered that has no solution.

••• LinearSolve : Linear equation encountered that has no solution.

••• General : Further output of LinearSolve::nosol will be suppressed during this calculation.

Polynomial has corank 1. It is 6-determinate. Its determinacy is either 6 or 5
  and its codimension is 4. Comparing with the known list of singularities...

Polynomial has a butterfly (A5) catastrophe.

Out[∘]= {1, 4, 6}
```

Now the second one is only slightly different, as mentioned earlier:

```
In[∘]:= DiagnoseHOS[-ky^2 + 6 kx^2 ky - 8 kx^4 + kx^6]

••• LinearSolve : Linear equation encountered that has no solution.

••• LinearSolve : Linear equation encountered that has no solution.

••• LinearSolve : Linear equation encountered that has no solution.

••• General : Further output of LinearSolve::nosol will be suppressed during this calculation.

Polynomial has corank 1. It is 4-determinate. Its determinacy is either 4 or 3
  and its codimension is 2. Comparing with the known list of singularities...

Polynomial has a cusp (A3) catastrophe.

Out[∘]= {1, 2, 4}
```

We can try feeding it another familiar singularity, namely the monkey saddle. To spice things up a bit, let us add a quartic part and a z^2 term to it (that increases the dimension). As mentioned above, the function is smart enough to figure out the variables and the degree, so we don't have to follow any particular naming convention for the variables.

```
In[∘]:= DiagnoseHOS[kx^3 - 3 kx ky^2 + 4 (kx^2 + ky^2)^2 + z^2]

Out[∘]= DiagnoseHOS[kx^3 - 3 kx ky^2 + 4 (kx^2 + ky^2)^2 + z^2]
```

Let us look at the example treated in the notebook 'Band_Series_Expansion.nb', where we obtained the monkey saddle (elliptic umbilic catastrophe) in the Haldane model:

```
In[∘]:= DiagnoseHOS[
  0.5773502691896257` + 3.28819020499404` p1^4 + 3.8971143170299682` p1^2 p2 +
   6.576380409988073` p1^2 p2^2 - 1.299038105676661` p2^3 + 3.28819020499404` p2^4]

••• LinearSolve : Linear equation encountered that has no solution.

••• LinearSolve : Linear equation encountered that has no solution.
```

```
Polynomial has corank 2. It is 3-determinate. Its determinacy is either 3 or 2
  and its codimension is 3. Comparing with the known list of singularities...
```

Polynomial has either an elliptic umbilic (D_4^-) (monkey saddle) or a hyperbolic umbilic (D_4^+) catastrophe.

```
Out[ ]= {2, 3, 3}
```

The unnamed singularities can also be detected:

In[ ]:= **DiagnoseHOS**$[u^2 v - v^7]$

```
LinearSolve : Linear equation encountered that has no solution.
LinearSolve : Linear equation encountered that has no solution.
LinearSolve : Linear equation encountered that has no solution.
General : Further output of  LinearSolve::nosol  will be suppressed during this calculation.
```

```
Polynomial has corank 2. It is 7-determinate. Its determinacy is either 7 or 6
  and its codimension is 7. Comparing with the known list of singularities...

Point singularity not identified by name!

Out[ ]= {2, 7, 7}
```

Now we look at one of the so called unimodal singularities which have a continuous real parameter giving rise to distinct singularities. Perhaps the main one of interest in band theoretic systems is the X_9 singularity class, which is a collection of distinct, non-equivalent singularities of the form $k_x^4 + k_y^4 + 2\,c\,k_x^2\,k_y^2$ for real c. They all have corank 2. For the case $c = -3$ (which has particle-hole symmetry), it has codimension 8 and determinacy 4. Let us check it with the algorithm, feeding a maximum determinacy check value of 4 (equal to the degree of the polynomial). The reason for this extra input will become clear below.

In[ ]:= **DiagnoseHOS**$[u^4 - 6\,u^2\,v^2 + v^4, 4]$

```
LinearSolve : Linear equation encountered that has no solution.
LinearSolve : Linear equation encountered that has no solution.
LinearSolve : Linear equation encountered that has no solution.
General : Further output of  LinearSolve::nosol  will be suppressed during this calculation.
```

```
The polynomial appears to host a singularity that does not have a finite determinacy.

Out[ ]= {2, ∞, ∞}
```

So the algorithm appears to have misdiagnosed this unimodal singularity. The reason is that we need to always check for determinacy up to one degree higher than the degree of the polynomial, which is what the function does by default. We just fed it a maximum degree of 4 to check for determinacy, on purpose, in order to illustrate this point. If we instead simply just feed the polynomial or the polynomial along with maximum determinacy check value of 5:

```
In[ ]:= DiagnoseHOS[u^4 - 6 u^2 v^2 + v^4]

    LinearSolve : Linear equation encountered that has no solution.

    LinearSolve : Linear equation encountered that has no solution.

    LinearSolve : Linear equation encountered that has no solution.

    General : Further output of LinearSolve::nosol will be suppressed during this calculation.

    Polynomial has corank 2. It is 5-determinate. Its determinacy is either 5 or 4
      and its codimension is 8. Comparing with the known list of singularities...

    Point singularity not identified by name!

Out[ ]= {2, 8, 5}
```

Of course, it also recognizes ordinary critical points:

```
In[ ]:= DiagnoseHOS[x^2 - v^2 + v x^3 + x^5]

    Polynomial is equivalent to an ordinary quadratic form with a non-singular Hessian.

Out[ ]= {0, 1, 2}
```

It can also detect situations wherein there is no critical point:

```
In[ ]:= DiagnoseHOS[kx + ky^2 - kx^3 - 3 kx ky^2 + 4 (kx^2 + ky^2)^2 + kz^5]

    The given polynomial does not have a critical point at the origin.
      It is smoothly equivalent to a linear form, and has determinacy 1.

Out[ ]= {2, 0, 1}
```

Lastly a situation when the singularity does not have finite determinacy:

```
In[ ]:= DiagnoseHOS[x^2 - x y^4]

    LinearSolve : Linear equation encountered that has no solution.

    LinearSolve : Linear equation encountered that has no solution.

    LinearSolve : Linear equation encountered that has no solution.

    General : Further output of LinearSolve::nosol will be suppressed during this calculation.

    The polynomial appears to host a singularity that does not have a finite determinacy.

Out[ ]= {1, ∞, ∞}
```