A Mathematica Package for Certifying the Nonexistence of Darboux Polynomials

Maxime Bridoux, Khalil Ghorbal

A Mathematica Package for Certifying the Nonexistence of Darboux Polynomials

Maxime Bridoux Khalil Ghorbal
Inria, France
firstname.lastname@inria.fr

Abstract

We present a Mathematica package to help investigate the existence of Darboux polynomials for (polynomial) ordinary differential equations (ODE). If we denote the derivation encoded by a given ODE by D, the package infers and propagates necessary conditions for a *generic* ansatz p to be a Darboux polynomial for D. By generic, we mean that not only the coefficients of p are unknown, but also the exponent d of its leading monomial. We present some examples for which the package produces a formal certificate proving the nonexistence of nontrivial Darboux polynomials. The package complements well standard algorithms that search for Darboux polynomials up to a fixed bound on the total degree.

1 Introduction

In his seminal work, Gaston Darboux introduced algebraic *particular integrals*, known today as Darboux polynomials [Gor01], as a mean to construct *general integrals* for polynomial ODEs of the standard form $\dot{x}_i = f_i(x_1, \ldots, x_n)$, $i = 1, \ldots, n$, where $f_1, \ldots, f_n$ are multivariate polynomials in $x_1, \ldots, x_n$ over some field, $\dot{x}_i$ denotes the derivative of x_i with respect to an independent variable t. From a differential algebraic perspective, the ODE defines a *derivation* $D = \sum_{i=1}^{n} f_i \partial_i$, acting on the ring of polynomials where ∂_i denotes the partial derivative with respect to x_i. A polynomial p is *Darboux for* D, or simply *Darboux* when D is clear from the context, whenever $D(p) = qp$ for some polynomial q, called the *cofactor* of p. (Polynomials of total degree zero are trivially Darboux.)

Darboux generation algorithms (e.g. [Man93]) are semi-decision procedures enumerating all Darboux polynomials up to a certain fixed bound on the total degree. The bound is eventually incremented until finding a (not necessarily irreducible) Darboux polynomial or reaching memory and/or time limits. The Mathematica package [1] we present is primarily designed to prove the nonexistence of nontrivial Darboux polynomials. The necessary conditions it infers complements however generation algorithms by providing useful constraints on both the degree and support of a generic ansatz p to be Darboux. For the proofs and further details about the implemented algorithms, we refer the reader to [GB24].

[1] Available in `https://gitlab.com/maximebridoux/darbouxcertification`.

2 Case studies

Let $x = (x_1, \ldots, x_n)$ denote a set of variables and $\alpha = (\alpha_1, \ldots, \alpha_n)$ be a vector of natural numbers. We use the shorthand notation x^α to denote the multivariate monomial $\prod_{i=1}^n x_i^{\alpha_i}$. Given a monomial order, the *multidegree* of a multivariate polynomial p is the exponent of its leading monomial, denoted $\mathrm{LM}(p)$. An ansatz $p = \sum_\alpha a_\alpha x^\alpha$ is said to be *generic* when its multidegree d is not fixed. We represent a generic polynomial implicitly by a partial function 'a' which associates to each exponent α the formal symbol a_α. Known expressions $a(\alpha)$ of coefficients a_α are stored in a finite list of *substitution rules* of the form $a_\alpha \to a(\alpha)$. Without loss of generality, we suppose that the generic ansatz p is monic, that is the coefficient of $\mathrm{LM}(p)$ is set to 1. This list of substitution rules is thus initially set to $\{a_d \to 1\}$.

For each example of the following section, our package provides a dedicated Mathematica notebook (version 13.3). All computations (for these benchmarks) take few seconds to run on a standard laptop running Ubuntu 22.04.01 and equipped with an i7-12700H CPU together with 32GB of memory. We refer to [GB24] for further details about the implemented algorithms.

2.1 Van der Pol Oscillator

The dynamic of the Van der Pol oscillator is defined by

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = \mu(1 - x_1^2)x_2 - x_1$$

which we implement as a key-value association. Monomial orders are encoded via their associated weight matrix [Rob85].

```
In[1]:= vdp = <| x1 <— x2, x2 <— μ (1 — x1^2) x2 — x1 |>;
        lex = {{1,0}, {0,1}}; dlex21 = {{1,1}, {0,1}};
```

The routine `ComputeQuotient(ode, d, a, morder)` returns the simplified quotient q of the division of $D(p)$ by p with respect to the monomial order `morder`. The symbol `a`, together with the multidegree `d`, provide an implicit encoding of the generic ansatz p. Hence `a[{1 + d[1], −1 + d[2]}]` represents the coefficient a_{d_1+1,d_2-1} of p. The polynomial $D(p)$ is computed only partially on the fly: only monomials of $D(p)$ that are divisible by $\mathrm{LM}(p)$ are relevant. We can effectively retrieve the result of [GB24, Proposition 5.6] (stated for $\mu = 1$) using the following command:

```
In[2]:= q = ComputeQuotient[vdp, {d[1], d[2]}, a, dlex21]

Out[2]= a[{1 + d[1], −1 + d[2]}] (1 + d[1]) + μ d[2] — x1^2 μ d[2]
```

Once a quotient q is computed, the next step is to compute the coefficients of the remainder $r = D(p) - qp$. This is performed by the routine `Coeff(ode, d, a, q, alpha)`, which outputs the general expression of the coefficient of x^α in r. For p to be Darboux, all the coefficients of r have to vanish. By scanning the coefficients of r, one infers necessary conditions on a_α and d. The routine `LinearPropagation(ode, d, a, q, morder, iterationLimit, Substd*, Substp*, Conditions*)` implements the constants' propagation described in [GB24, Section 6]. It outputs a tuple of three lists of substitution rules for p to be Darboux: (1) a list of necessary conditions on the multidegree d, (2) a list of necessary conditions on the coefficients a_α, and (3) a formal *certificate* briefly described in the sequel. We run below the procedure for the Van der Pol dynamic with 6 as an iteration limit.

```
In[3]:= {substd, substp, certif} = LinearPropagation[vdp, {d[1], d[2]}, a, q, dlex21, 6]

Out[3]= {{}, {a[{d[1], d[2]}] -> 1}, {}}
```

This first run didn't infer any extra condition and simply returned the default values. In such cases, one can either increase the iteration limit, or add extra initial assumptions on d or a_α by supplying a list of substitution rules via the optional arguments `"Substd"` and `"Substp"` respectively. A closer look at the Van der Pol example reveals that the propagation requires some extra knowledge on the parameter μ of the dynamic. Indeed, depending on μ, D might have a Darboux polynomial. For instance, when $\mu = 0$, the oscillator reduces to the standard pendulum for which the polynomial $x_1^2 + x_2^2 + c$ is Darboux for any constant c. The optional argument `"Conditions"` serves precisely this purpose and allows the user to specify extra conditions on the parameters of the ODE. A second run assuming $\mu \neq 0$ is shown below.

```
In[4]:= {substd, substp, certif} = LinearPropagation[vdp, {d[1], d[2]}, a, q, dlex21, 6,
         "Conditions" -> {μ != 0}]

Out[4]= {{d[1] -> 0, d[2] -> 0}, {a[{0, 0}] -> 1}, ... }
```

The procedure outputs that for p to be Darboux, the multidegree d has to be $(0, 0)$, reducing p to a constant. This concludes the proof of nonexistence of nontrivial Darboux polynomials. Note that 6 is the minimum iteration limit for which a complete proof is found. The corresponding certificate (which is truncated with dots in the previous output cell) is given as an ordered list of pairs $(d+\eta, \text{substitution rule})$, where η is a vector of integers. The pair means that the substitution rule was deduced from computing the coefficient $c_{d+\eta}$ of $x^{d+\eta}$ in r and solving the equation $c_{d+\eta} = 0$. For instance, the first element of the certificate is:

```
In[5]:= certif[[1]]

Out[5]= {{-1 + d[1], 1 + d[2]}, a[{-3 + d[1], 1 + d[2]}] -> d[1]/μ}
```

To formally check the certificate a posteriori, one can compute the coefficient $c_{d+\eta}$ of r and verify that the associated substitution rule can indeed be deduced. For instance, for $c_{-1+d_1,1+d_2}$ one gets the (linear) constraint $a_{-3+d_1,1+d_2} = \frac{d_1}{\mu}$.

```
In[6]:= Expand[Coeff[vdp, {d[1], d[2]}, a, q, {-1 + d[1], 1 + d[2]}, dlex21]]

Out[6]= -μ a[{-3 + d[1], 1 + d[2]}] + d[1]
```

2.2 Liénard systems

Our implementation can also infer necessary conditions on the parameters of a given ODE for a Darboux polynomial to exist. Given f, g univariate polynomials, the class of Liénard systems is given by the following ODE (the Van der Pol oscillator is in particular a typical Liénard system):

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -f(x_1)x_2 - g(x_1) \ . \end{aligned}$$

We provide a notebook example for the case where $\deg f = \deg g = 3$. Computing the quotient using the monomial order $\texttt{DLex}_{21}$ and performing the constraints' propagation infer the substitution `{d[1] –> 0}`. When executed with 7 as the iteration limit, the routine prompts a dialog box suggesting that an equation with several possible solutions was found, and asks the user to select one solution to explore. In this example, one can either choose the first option `{d[2] –> 0}`, which completes the ongoing proof (returning $d_1 = d_2 = 0$), or choose to enforce extra conditions on f and g (as we did for μ in the Van der Pol system). Assuming that $d_2 > 0$ and both f_0, g_0 (the constant coefficients of f and g) are distinct from 0, the procedure is unable to find a bound on d_2 but generates the substitutions shown below.

```
In[7]:= {substd, substp, certif} = LinearPropagation[lienard, {d[1], d[2]}, a, q, dlex21, 7,
         "Conditions" –> {f[3] != 0, g[3] != 0}]; substd
```

$$\text{Out[7]= } \{d[1] \to 0,\ g[3] \to \frac{f[3]\ g[0]}{f[0]},\ g[1] \to \frac{f[1]\ g[0]}{f[0]},\ g[2] \to \frac{f[2]\ g[0]}{f[0]}\}$$

Instantiating f and g as suggested, we immediately get $p = x_2 - \frac{g_0}{f_0}$ as a Darboux polynomial for $d_2 = 1$.

2.3 Rössler system

The Rössler system is given by the following ODE featuring three real parameters μ_1, μ_2, and μ_3:

$$\begin{aligned}\dot{x} &= -y - z\\ \dot{y} &= x + \mu_1 y\\ \dot{z} &= xz + \mu_2 - \mu_3 z\end{aligned}$$

In [LZ02, Theorem 1], Llibre and Zhang characterize the conditions on μ_1, μ_2 and μ_3 under which the system has Darboux polynomials. For instance, when $\mu_2 = 0$, z is a Darboux polynomial with cofactor $x - \mu_3$. An important, and particularly tedious, part of their characterization was to prove that, when $\mu_2 \neq 0$, the system has no Darboux polynomial. Our package proves automatically the same result and provides a formal certificate as a chain of 134 substitution rules for the longest proof (cf. the associated notebook).

References

[GB24] Khalil Ghorbal and Maxime Bridoux. On bounding the degree of irreducible darboux polynomials. Submitted to ISSAC 2024., 2024.

[Gor01] Alain Goriely. *Integrability and Nonintegrability of Dynamical Systems*. Advanced series in nonlinear dynamics. World Scientific, 2001.

[LZ02] Jaume Llibre and Xiang Zhang. Darboux integrability for the rössler system. *International Journal of Bifurcation and Chaos*, 12(02):421–428, 2002.

[Man93] Yiu-Kwong Man. Computing closed form solutions of first order odes using the prelle-singer procedure. *J. Symb. Comput.*, 16(5):423–443, 1993.

[Rob85] Lorenzo Robbiano. Term orderings on the polynominal ring. In *EUROCAL*, volume 204, pages 513–517, 04 1985.