A new Mathematica–based program for solving overdetermined systems of PDE

Stelios Dimas

Department of Mathematics, University of Patras, Rio, Greece
spawn@math.upatras.gr

Dimitri Tsoubelis

Department of Mathematics, University of Patras, Rio, Greece
tsoubeli@math.upatras.gr

The outline and the main features of a heuristic free algorithm implemented in *Mathematica* [1] for solving overdetermined systems of PDEs is decribed. This algorithm, which we call "Seek&Solve", is the core element of the command `Solve-OverdeterminedEqs` of the Mathematica based symmetry analysis package *SYM*, developed by the authors recently [2]. Characteristic examples are presented, showing the effectiveness of the algorithm and, in particlar, its ability to deal with subsets of the initial system, corresponding to special parameter values.

■ Introduction

Symmetry analysis is a well established tool in the field of differential equations of applied mathematics. Part of its success is due to the algorithmic way of determining the symmetries of an actual differential system, a task which is very tedious and error−prone when undertaken by hand. As a result, many symmetry−finding programs and specific packages based on various combuter algebra systems have already been developed.

One of the main components of every symmetry−finding package is the solver of overdetermined system of differential equations. An example of this kind of systems is given by the "determining equations" which arise when one imposes the linearized symmetry condition on the differential equation(s) under study. Even though the determining equations are linear, the total number of independent and dependent variables is usually so large that solving the corresponding equations is a very difficult task. As a result, the overdetermined system solver used in dealing with this task usually does not manage to obtain the most general solution.

There are two main categories of algorithms that can be employed in the construction of such an overdetermined system solver: the heurestic and the non−heurestic or heurestic free. By heurestic we mean an algorithm which tries to integrate the determining equations starting with certain assumptions regarding the structure of the saught solutions. For example, it is usually assumed that a particular subset of the solution functions are polynomials in the independent variables involved. By heurestic free, we mean an algorithm that attempts to integrate the determining system without making use of any ansätze. Usually, such an algorithm will first bring the system into a simpler form, called *canonical* or *echelon* form. Then, it will either seek the general solution of the system, including all those cases that correspond to special values of the parameters involved, [6, 7] or, use the canonical form in order to obtain the main features of the solution space and the structure constants of the underlining algebra [3, 4, 5]. The advantage of the heurestic free algorithms is the fact that they will always give the full set of solutions of the system under examination. However, the calculation of the canonical form can be a very tedious and time−consuming process depending exponentially on the complexity of the system. Moreover, because the methods used in obtaining the canonical form are based on the theory of differential algebra, e.g. differential Gröbner bases, they mostly work when the system consists of PDEs that are in polynomial form with respect to the uknown functions and their derivatives. The heurestic algorithms, on the other hand, are much faster but, in general, they do not give the complete solution space of the determining equations under investigation.

In this paper, we present the outline of the implementation we employed for the integration of overdetermined systems of equations. We call this *Mathematica*−based algorithm "Seek&Solve", in order to stress the method of integration it employes and the Artificial Intelligence (AI) aspects of its structure. The capabilities of this implemention in SYM, under the command `SolveOverdeterminedEqs` [2], are exhibited in the last section of the paper by comparison against known results. (The symmetry−finding package SYM can be obtained by contacting the authors)

■ The algorithm

As its name denotes, the ''Seek&Solve" algorithm is a heurestic free algorithm which employs a suite of search strategies to ''Seek" the most appropriate equation to solve and then integrate —"Solve"— it using the built−in function `DSolve` of *Mathematica*. Let us stress that, eventhough we use the latter function as the basis of our integrator, we have also further tweaked and augmented it so as to render it more effective in the problem at hand. The use of *Mathematica's* internal solver offers the advandage of harnessing the capabilities of this ever expanding function. Furthermore, by adding to it extra integration rules we "overload" its purpose and overcome its current limitations. As a consequence, we do not have to make any a priori assumptions on the kind and form of the differential equations involved.

In addition to starting the integration process from the most appropriate equation, which it locates thanks to its pattern recognition capabilities, our algorithm simplifies the differential equations at hand and turns integro−differential equations to differential ones. Furthmore, it splits differential equations involving uknown functions with different arguments into simpler ones compatible with the originals. For example, the equation $f'(x) + x^2 g'(y) = 0$ splits into the equations $\frac{f'(x)}{x^2} = c$, $g'(y) = -c$. Moreover, when multiple possible solutions or special values of the parameters, that might lead to alternative solutions, are found, the algorithm has the ability to auto−branch itself and examine each case separately in a sequence. Using parallel programming, the above auto−branching process can be readily implemented in a multiprocessor or grid enviroment. With each case assigned to a different unit, the full analysis of the original system of equations can be completed much faster than usual.

□ Outline of the algorithm

Below, we give the algorithm, divided in 6 steps, which we have incorporate it into our package. These 6 steps form a closed loop that terminates only if no equation that can be solved remains. The implementation in *Mathematica* is realized with the use of the internal function `FixedPoint`.

Step 1: Search for integro−differential equations and turn them into differential. Moreover, split the equations to simpler ones if possible, as described above.

Step 2: Simplify the system, with cross−differentiation when possible, and search for any special values of the parameters, if any, that satisfy any of the equations. For example, the equation $(\alpha - 3)\,\beta\,(u_{,xy} - 1) = 0$ is satisfied when $u_{,xy} = 1$ or if the parameters take the values $\alpha = 3$ or $\beta = 0$. In that case, continue with the first of them and put on queue the rest (branching).

Step 3: Search for any algebraic equations and solve them. In case there are multiple solutions, choose the first one and put on queue the rest.

Step 4: Sort the differential equations with respect to their degree of complexity (the number of unkown functions involved and the length).

Step 5: Find the first one that can be solved for any of the uknown functions. Again, in case there are multiple solutions, pick the first one and put on queue the rest.

Step 6: Substitute the solution found to the rest of the equations.

□ Remarks

As mentioned previously, the goal of a heurestic free algorithm is to transform the system of determining equations into one that it is easier to classify and solve. The advantage of "Seek&Solve" is that it realizes the above two step process automatically, without any external intervention. In other words, it keeps simplifying the system of equations while integrates it. In particular, "Seek&Solve" detects the special values of the parameters that must be treated seperately at different stages of the solution process, a fact that makes this algorithm much more time−efficient than existing ones.

■ Examples

In the examples that follow we give some characteristic applications of the implementation of the "Seek&Solve" algorithm in the SYM symmetry−finding package.

• The determining equations that stem from the non classical analysis of the Huxley equation $u_t = u_{,xx} + 2\,u^2\,(1-u)$ are [8]:

$$
\begin{aligned}
\xi_{,uu} &= 0, \\
\eta_{,uu} - 2\,\xi_{,xu} + 2\,\xi\,\xi_{,u} &= 0, \\
2\,\eta_{,xu} - \xi_{,xx} - (2\,\eta + 6\,u^3 - 6\,u^2)\,\xi_{,u} + 2\,\xi\,\xi_{,x} + \xi_{,t} &= 0, \\
\eta_{,xx} - 2\,\eta\,\xi_{,x} + 2\,u^2\,(u-1)\,(\eta_u - 2\,\xi_{,x}) - \eta_{,t} + (4\,u - 6\,u^2)\,\eta &= 0
\end{aligned}
\tag{1}
$$

The complete set of solutions, as found automatically by SYM, are
1. $\xi = 1,\ \eta = 0$
2. $\xi = \pm(3\,u - 1),\ \eta = 3\,u^2\,(1-u)$

• The determining equations from the analysis of a generalization of the Chazy equation $u''' = 2\,u\,u'' - \beta(u')^2$ are [6]:

$$
\begin{aligned}
\xi_{,u} &= 0, \\
\phi_{,u} &= 0, \\
\phi_{,u} + \xi_{,x} &= 0, \\
-2\,u\,\phi_{,xx} + \phi_{,xxx} &= 0, \\
-2\,\phi - 2\,u\,\xi_{,x} + 3\,\phi_{,xu} - 3\,\xi_{,xx} &= 0, \\
2\,\beta\,\phi_{,x} - 4\,u\,\phi_{,xu} + 2\,u\,\xi_{,xx} + 3\,\phi_{,xxu} - \xi_{,xxx} &= 0
\end{aligned}
\tag{2}
$$

In this example we have two different cases: one for $\beta \neq 3$ and one for $\beta = 3$ (Chazy equation). Its worth noting that in the above reference onother case was found, namely $\beta = -\frac{1}{7}$. As remarked by the authors, the latter case is a mere artifact of the calculations and is covered by the general case for $\beta \neq 3$. The use of SYM leads to the following results automatically whithout ever encountering the artificial $\beta = -\frac{1}{7}$ case.
1. $\beta \neq 3$: $\xi = -c_1\,x + c_2,\ \phi = c_1\,u$
2. $\beta = 3$: $\xi = -\frac{1}{2}\,x(c_2\,x + 2\,c_1) + c_3,\ \phi = c_1\,u + (3 + x\,u)\,c_2$

■ Acknowledgements

[1] Wolfram Reasearch, Inc., Mathematica, Version 5.x, Wolfram Reasearch, Inc., Illinois, 2003.

[2] Dimas S., Tsoubelis D., SYM: A new symmetry−−finding package for Mathematica,in Proceedings of 10th International Conference in Modern Group ANalysis (Octomber,24−−30,2004, Larnaca,Cypus), Editors N. H. Ibragimov, C. Sophocleous, P. A. Damianou, *Proceedings of MOGRAN X*, Cyprus, 2005 , 64-70.

[3] Reid G. J., A triangularization algorithm which determines the Lie symmetry algebra of any system of PDEs, *J. Phys. A: Math. Gen*, 1990, V. 23, L853-L859.

[4] Reid G. J., Algorithms for reducing a system of PDEs to standard form, determining the dimension of its solution space and calculating its Taylor series solution, *Eur. J. Appl. Math.*, 1991, V. 2, 293-318.

[5] Reid G. J., Finding symmetries of differential equations without integrating determining equations, *Eur. J. Appl. Math.*, 1991, V. 2, 319-340.

[6] E. L. Mansfield, P. A. Clarkson, Applications of the Differential Algebra Package `diffgrob2` to Classical Symmetries of Differential Equations, *J. Symb. Comput.*, 1997, V. 23, 517-533.

[7] E. L. Mansfield, Algorithms for Symmetric Differential Systems, *Found. Comput. Math.*, 2001, V. 1, 335-383.

[8] P. E. Hydon, Symmetry Methods for Differential Equations - A Beginner's Guide, Cambridge University Press, 2000.