

Chapter 8

Using Symbolic Computational Dynamics as an Aid to Design



Matthew P. Cartmell and Niloufar Motazed

Abstract The topic of Symbolic Computational Dynamics, as presented here, has been motivated by the utility of approximate analytical solutions for reduced order models, and the power of computers to cope with the challenges of both problem scale and automation. Application has traditionally been limited by the algebra needed for problems of more than a few coupled coordinates, making such problems excellent candidates for automation through symbolic computation. But there is a lot of useful information that is naturally lost when doing this, due to the on-going processes of algebraic simplification, the different mathematical-physical processes behind the small parameter, and defining relative strengths of physically based terms. We offer a novel symbolic computational process that applies a semi-automated asymptotic method for solution that also retains all information, leading to a first generation approach to the global visualisation of problems in dynamics.

Keywords Nonlinear dynamics · Perturbation methods · Symbolic computation · Design

8.1 Introduction

Research into *Symbolic Computational Dynamics* (SCD) by Cartmell and Khanin began in 1997 under this title and owes something of its heritage to earlier work carried out by others, notably Rand et al., over a period going back to the early nineteen eighties, mainly on the computation of perturbation methods for approximate analytical solutions to reduced order differential equation models. There are several

M. P. Cartmell (✉)

Department of Mechanical and Aerospace Engineering, University of Strathclyde, Glasgow G1 1XJ, UK

e-mail: matthew.cartmell@strath.ac.uk

N. Motazed

Bombardier Transportation UK Ltd., Litchurch Lane, Derby DE24 8AD, UK

e-mail: motazed.n@gmail.com

© Springer Nature Switzerland AG 2020

I. Kovacic and S. Lenci (eds.), *IUTAM Symposium on Exploiting Nonlinear*

Dynamics for Engineering Systems, IUTAM Bookseries 37,

https://doi.org/10.1007/978-3-030-23692-2_8

asymptotic methods that can be very usefully applied to nonlinear dynamics problems when they are represented in differential equation form, including the method of Struble, the Lindstedt–Poincaré method, applications of the Volterra series, Harmonic Balance, the method of multiple scales, and the method of direct separation of motions, amongst others. So, in general, approximate analytical solutions can usually be found for reasonably set up reduced-order differential equation models, given certain conditions and constraints. Such solutions can accurately represent the dynamics of the problem—within the limitations of the approximations made. Ease of application of these potentially powerful methods is often limited by the sheer scale of algebraic manipulation needed for nonlinear problems that involve more than a few coupled coordinates. On this basis they are excellent candidates for semi-automation through symbolic computation. The first published research on the formal use of computers to solve symbolic problems in nonlinear dynamics is attributable to Rand [1] in 1984 who introduced the MACSYMA language for this purpose, and then to Rand and Keith who explored normal forms and centre manifold calculations using MACSYMA in 1985 [2], noting that there are several other significant and relevant publications by Rand et al. from around this time. By 1987 Rand and Armbruster had successfully brought together bifurcation theory, perturbation methods, and computer algebra in a major new book [3]. Rand continued to add to this pioneering body of work with a further book on computer algebra applied to nonlinear dynamics in 1994 [4]. Through these seminal works Rand effectively pioneered, and then firmly established, the formal application of symbolic computation to the principal mathematical topics that are used right across the general area of applied dynamics. Recently Professor Rand has issued a comprehensive treatment of nonlinear vibration in the form of published lecture notes, from which virtually all the theories and treatments required for nonlinear analysis are usefully and practically summarised for the practitioner [5].

The long-standing motivation behind the research presented in this paper is an acknowledgement that certain forms of mathematical information are naturally lost when analytically solving the nonlinear differential equations that realistically describe problems encountered in dynamics. This lost information has potential use because of what it can mean physically. One example of this loss is simply that which occurs due to the on-going processes of algebraic simplification. Another is due to the different mathematical-physical reasoning processes that underpin, say, the use of the small perturbation parameter, as it appears and then re-appears throughout an analysis. A third is the way in which we define the relative strengths of physically based terms when they are first introduced into the equations of motion, and the repercussions of getting this partially, or even completely wrong. So, the objective in this research has been to create a symbolic computational process that efficiently applies a semi-automated asymptotic method in a correct, consistent, and adaptable manner. The process was also intended to provide a facility for the identification and retention of all the mathematical-physical information generated that could finally be represented back to the user in an easy-to-interpret visualisation.

The origins of the research specifically reported here go back to a series of lengthy second order multiple scales analyses undertaken by Cartmell and completed in 1984

[6]. That work stimulated many informal and independent experiments in the symbolic codification of perturbation schemes, in the intervening period up to 1997. This led to the award to Cartmell, then based at the University of Edinburgh, of the first of several research grants from 1997 onwards. New forms of symbolic software in *Mathematica*TM emerged from the work done by Cartmell and Khanin under this funding, enabling both serial and parallel computation [7–10]. The work was transferred to the University of Glasgow in 1998 with further research funding awarded from 2000 and this led to the next generation of programs and the burgeoning idea of ‘term-tracking’ [11–13]. Funded research continued up to 2008. The work was then transferred in 2012 to the University of Sheffield, by which time relatively advanced symbolic solution codes had been developed by Khanin, Forehand, and Cartmell, all written in the *Mathematica*TM language. In addition preliminary research had also been undertaken by Cartmell and Forehand on understanding the challenges inherent in the visualisation of the new information generated by this sort of computation. Motazedi joined the project in 2013 and she subsequently revised and developed the solver software and created an entirely new *term-tracker* package initially based on some of the ideas that had been proposed by Cartmell, Khanin, and Forehand. Motazedi then went on to devise, build, and test new software for visualisation of symbolic data [14, 15], again in *Mathematica*TM. Motazedi also took on the substantial problem of writing semi-automated symbolic code for the treatment of the modulation equations that arise naturally within the perturbation method of multiple scales, and she devised a *generic and adaptable* computational structure for handling these important equations systematically in order to complete the solution procedure [14]. These activities have since led to the identification by the authors of *Symbolic Computational Dynamics* (SCD) as a convenient umbrella term for their approach to this general area of research. The authors make no claim for any wider uptake of this term in this context, as yet.

The choice of *Mathematica*TM as the preferred programming language was strategic and made by Cartmell back in the mid 1990s. The thinking behind this decision was generally influenced by the very powerful high-level nature of this language, the fact that it’s always been aimed fundamentally at symbolic programming, and because of the many specific features of the language that have lent themselves directly to logical implementations for SCD. The current generation of programs operate as digital interactive notebooks within the *Mathematica*TM interface. This provides an essential level of flexibility for the user so that s/he can apply any assumption or simplification when it is required. The current generations of code allow considerable user interaction and customisation of the core solver and term-tracker, to the extent that differential equations can be added at will, and the internal solution procedures themselves can be customised easily if necessary. This offers a great deal of generality and flexibility in use.

8.2 Symbolic Computational Dynamics

8.2.1 *The Process of Symbolic Computational Dynamic Solution*

The principles behind SCD are that the problem should be representable in differential equation form and that the equation(s) should lend themselves to asymptotic analysis, also that the analysis method should be adaptable yet algorithmic enough in structure to enable a symbolic computer code to be written to do all the mathematics reliably and accurately yet also adaptively. In addition SCD, as we define it, requires a facility for identifying, encoding uniquely, and then tracking all the mathematical-physical operations within the solution procedure so that a record of the whole process can be generated from which useful visualisations and information can be extracted. The end result is, for example, a typical perturbation analysis in equation form, which is supplemented by a graphic that shows the interconnectedness of the stages within the analysis—in the context of the system variables, operators, constants, and parameters. A detailed mathematical study has been provided by Motazed [14] and in summarised form in [15], and these two references provide the principal sources for what follows. The utility of the SCD approach depends strongly on the format of the graphic, and this remains an open topic for research with current ideas summarised later. The architecture of an SCD solver is given in Fig. 8.1.

The term-tracker module is based on the authors' *Source and Evolution Encoding Method* (SEEM) and the encoding strategy is summarised in Table 8.1.

An example of how this encoding logic is applied in practice is given in Eqs. (8.1)–(8.5) in which the governing differential equation of motion for a parametrically excited pendulum and the associated multiple scales expansions are all shown in encoded form,

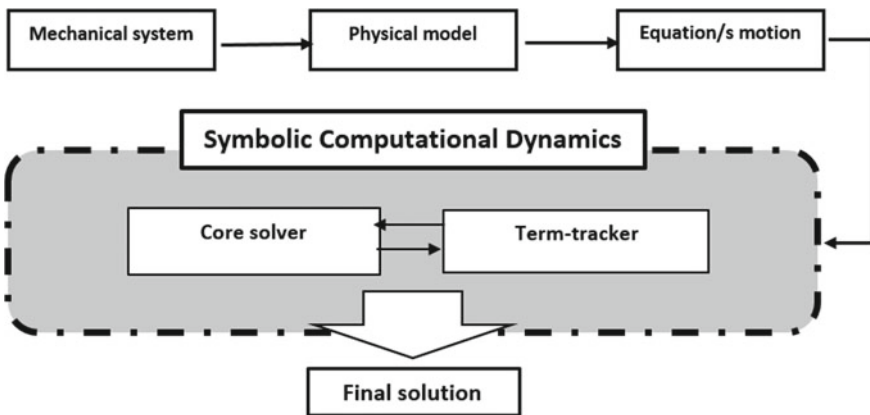


Fig. 8.1 SCD solver architecture [14]

Table 8.1 SEEM logic summary [14]

Encoding level	Description	Visualisation
First level	Origin of the equation and the perturbation order	(i, j)
Second level	First level + equation number in which it becomes explicit	(i, j, k)
Third level	Second level + equation number in which the explicit form is modified	(i, j, k, l)
Compound-I	When two quantities such as e and i subsume, two or more individual e and i terms	$(*, i)$
Compound-II	$(i, j, k, l)_1 \text{ expadd}^*(i, j, k, l)_2$	$(*, k_{latest})$
	$(i, j, k, l)_3 \text{ expadd}(i, j, k, l)_4$	$(\#, k_{latest})$
	$(i, j, k, l)_5 \text{ expadd}(i, j, k, l)_6$	$(\$, k_{latest})$
	$(i, j, k, l)_7 \text{ expadd}(i, j, k, l)_8$	(\pounds, k_{latest})
Encoding of signs	Signs are not normally encoded unless they are created as the result of an algebraic operation. The divisor is not encoded as well	(i, j, k, l)

$$\ddot{\theta} + \underbrace{2\varepsilon\beta\dot{\theta}}_{(1,1)} + \left(\underbrace{1}_{(1,0)} - \underbrace{\varepsilon q \omega^2}_{(1,1)} \cos\left(\underbrace{\omega}_{(1,0)} T_0\right) \right) \theta + \underbrace{\varepsilon\gamma}_{(1,1)} \theta^3 = 0 \quad (8.1)$$

$$\theta(\tau, \varepsilon) = \underbrace{\theta_0}_{(2,0)} + \underbrace{\varepsilon\theta_1}_{(2,1)} + O(\varepsilon^2) \quad (8.2)$$

$$\frac{d}{dt} = \underbrace{D_0}_{(3,0)} + \underbrace{\varepsilon D_1}_{(3,1)} + O(\varepsilon^2) \quad (8.3)$$

$$\frac{d^2}{dt^2} = \underbrace{D_0^2}_{(4,0)} + \underbrace{2\varepsilon D_0 D_1}_{(4,1)} + O(\varepsilon^2) \quad (8.4)$$

$$\theta^3 = \underbrace{\theta_0^3}_{(2,0,5)} + O(\varepsilon). \quad (8.5)$$

The time domain solution is given by Eq. (8.6), with full SEEM encoding shown,

$$\theta(\tau, \varepsilon) = \underbrace{a[T_1]}_{(20,0,31)} \cos\left(\underbrace{1}_{(9,0,31)} - \underbrace{\frac{2}{2}}_{(21,0)} \underbrace{\frac{2}{2}}_{(13,0,34)} \tau + \frac{1}{2} \left[\underbrace{\frac{2}{2}}_{(13,0)} - \underbrace{\frac{2}{2}}_{(9,0)} + \underbrace{\omega}_{(14,0,34)} \right] \tau + \underbrace{\psi}_{(24,0)} \right) \\ - \frac{\underbrace{\varepsilon}_{(2,1)} \underbrace{\varepsilon}_{(1,1,19)}}{\underbrace{\varepsilon}_{(2,1,19)}} \frac{1}{\underbrace{2^3}_{(20,0)}} \frac{1}{\underbrace{8}_{(19,0)}} \underbrace{\gamma}_{(1,1,19)} \underbrace{a[T_1]^3}_{(20,0)} \cos\left(\underbrace{3}_{(1,0,10,19)} \tau \right)$$

$$\begin{aligned}
 & + \frac{\overbrace{3}^{(20,0,36)}}{\underbrace{2}_{(21,0)}} \left[\left(\underbrace{2}_{(13,0)} \underbrace{-2}_{(9,0)} - \underbrace{2}_{(13,0,34)} \underbrace{\omega}_{(14,0,34)} \right) \tau - \underbrace{\psi}_{(24,0)} \right] \\
 & + \frac{\overbrace{\varepsilon}^{(2,1)} \overbrace{\varepsilon}^{(1,1,19)}}{\underbrace{\varepsilon}_{(2,1,19)}} q \omega^2 \underbrace{\frac{1}{2}}_{(1,1,19)} \underbrace{\frac{1}{8}}_{(1,0,11,19)} \underbrace{\frac{1}{2}}_{(19,0)} \underbrace{\frac{1}{2}}_{(20,0)} \cos \\
 & \left[\left(\underbrace{2}_{(13,0,19)} + \underbrace{1}_{(9,0,19)} + \frac{\underbrace{2}_{(13,0)}}{\underbrace{2}_{(21,0)}} - \frac{\underbrace{2}_{(9,0)}}{\underbrace{2}_{(21,0)}} - \frac{\underbrace{2}_{(13,0,34)}}{\underbrace{2}_{(21,0)}} - \underbrace{2}_{(13,0,34)} \right) \tau \right. \\
 & \left. + \left(\underbrace{\omega}_{(14,0,34)} + \frac{\underbrace{\omega}_{(14,0,34)}}{\underbrace{2}_{(21,0)}} \right) \tau + \frac{\underbrace{\psi}_{(24,0)}}{\underbrace{2}_{(21,0)}} \right] + O(\varepsilon^2) \tag{8.6}
 \end{aligned}$$

In order to generate Eq. (8.6) an appropriate form of the method of multiple scales for this problem is run automatically by the solver so that every symbolic and numerical computation can be picked up and recorded by the term-tracker, running in parallel, to generate a completely encoded approximate solution in the time domain, supplemented by the modulation equations from which amplitudes and phases can be obtained. The time domain solution for a first order multiple scales expansion with encoding is given in Eq. (8.6). The as-generated encoding data is clearly quite unwieldy in form but it can still be used to identify the sources of each term, and to make a connection back to the physical conceptualisation of the problem. The specific meaning of the encoding information is directly dependent on the solution procedure and the way that has been introduced mathematically. The complexity of the SEEM generated encodings invariably increases significantly as the problem solution develops. The cancellation of parameters with the solution procedure that would naturally take place in an efficient and elegant algebraic process is deliberately avoided in SEEM, unless both the quantity and the first two encoding digits are identical. The small parameters that are introduced at different points early in a multiple scales perturbation analysis do not necessarily have the same numerical value, and the encoding reflects this. However the reverse may be true where different instances of the small parameter end up being encoded differently (because of their historical treatment) but where in fact these are the same small parameter numerically. SEEM accommodates all such possibilities in all the necessary contexts. The programming strategy for a generic SEEM analysis is given in full in [14] and summarised in [15], and is based on 28 different algorithms which have been created and then coded in *Mathematica*TM in order to implement the solver/tracker.

8.2.2 Visualisation

There are two requirements for an SCD visualiser, these being the need to represent the complete form of information and also the practical requirement for this information to be immediately readable, understandable and assimilable. As there is a fundamental conflict between these requirements a series of pragmatic decisions ultimately led to the *visualisation graph* format of Fig. 8.2. This relates directly to Eqs. (8.1)–(8.5) and shows the user the equations in a layered sequential form together with the *link* between Eqs. (8.2) and (8.5), via encoding (2,0,5). The software has since been configured to allow selective viewing of the analysis with zoom and converge features providing concentrated visualisation of the finest detail, together with controllable zoom-out for the bigger picture. Automated colour and line thickness features have been included to differentiate between links based on information flows based on different perturbation order and systematic definitions of term complexity. A statistical package has also been devised to provide a *strength factor* metric visualised in the form of shaded circles of different diameter. This allows comparisons to be made between the relative importance of terms within the same equation and also with terms as they relate to other terms in other equations, travelling forwards or backwards through the analysis. This is shown in Fig. 8.3.

The visualisations of Figs. 8.2 and 8.3 are based on a sequential hierarchy, in which the analysis proceeds graphically from top right to bottom left, however Fig. 8.4 shows an example from one highly promising avenue of research in which the sequential constraint is dropped in favour of an alternative algorithm which presents the problem three dimensionally and emphasises the *interconnectedness* as a dominant feature. This visualisation could be more useful than the sequential approach, when combined with the strength factor metric and full three dimensional positioning and automated rotation about a user specifiable axis. The left hand image in Fig. 8.4 is a plan view of the interconnectedness visualisation showing two distinct regions, which relate to the two parts of the analysis of a 2 DoF autoparametric oscillator (physically representing a pair of coupled beams).

Most of the analysis connected with the primary beam is in and around the upper left area and conversely for the secondary beam in the lower right area. The right hand

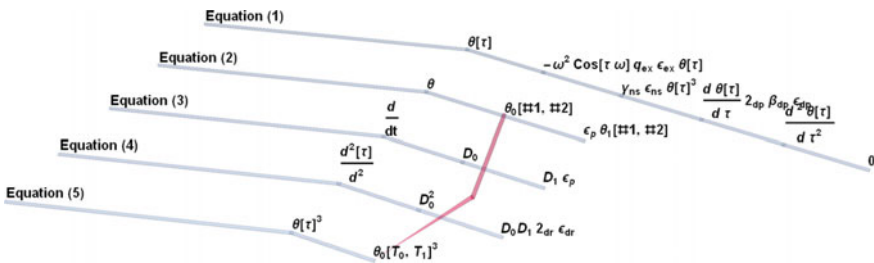


Fig. 8.2 Visualisation graph for the first five equations of the parametrically excited pendulum example (refer to Eqs. (8.1)–(8.5)) [14]

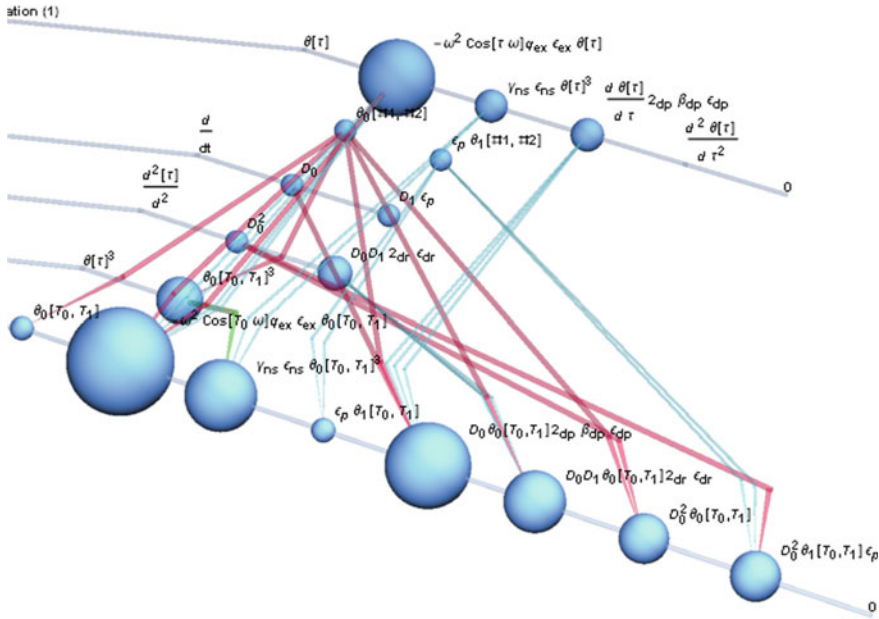


Fig. 8.3 An implementation of the strength factor metric in the early stages of the parametrically excited pendulum example [14]

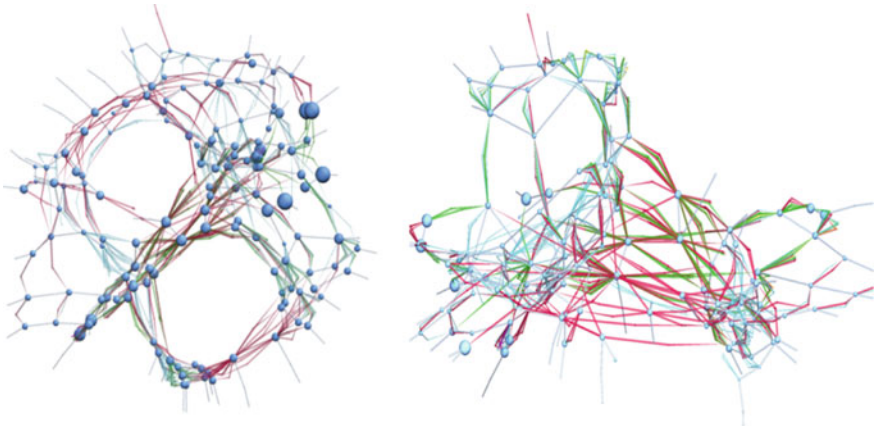


Fig. 8.4 Two images of typical SCD output for a 2 DoF autoparametric oscillator in two different elevations [14]

image shows the same graphic when rotated into a side elevation (noting that these definitions of elevation do not relate in any obvious way to any physical aspects of the problem, they are simply convenient terms for distinguishing between the views shown). Closer investigation of the two images reveals a cylindrical structure emerging for each physical part of the problem.

8.3 Conclusions

This research has taken place over a long period and has led to a series of computational strategies which can generate user graphics for the dynamics. The rules of interpretation are evolving and have not been stated here in detail. The work confirms that the fundamental mathematical model contains a richness of information that is normally not seen, and if this is converted into an explicit form it can be used to provide a supplement to the conventional symbolic and/or numerical solution to the problem. This could offer insights into interrelationships within the mathematical and physical representations of the system, provided the graphics can be interpreted meaningfully and quickly. These remaining challenges of user interpretation are currently under intensive study.

Acknowledgements The authors wish to acknowledge the pioneering research of Professor Richard H. Rand of Cornell University and his colleagues, also the support of Engineering and Physical Sciences Research Council grants EP/C530446/1, GR/N32334/02, GR/N32334/01, GR/N32280/01, GR/L30749/02, and GR/L30749/01, and the support made available to Motazed by the University of Sheffield.

References

1. Rand, R.H.: Computer Algebra in Applied Mathematics: An Introduction to MACSYMA, Research Notes in Mathematics, vol. 94, p. 181. Pitman Publishing Inc. (1984)
2. Rand, R.H., Keith, W.L.: Normal forms and center manifold calculations on MACSYMA. In: Pavele, R. (eds.) Applications of Computer Algebra, pp. 309–328. Kluwer Academic Publishers (1985)
3. Rand, R.H., Armbruster, D.: Perturbation Methods, Bifurcation Theory and Computer Algebra, Applied Math Sciences, vol. 65, p. 243. Springer, Berlin (1987)
4. Rand, R.H.: Topics in Nonlinear Dynamics with Computer Algebra, p. 229. Gordon and Breach (1994)
5. Rand, R.H.: Lecture Notes in Nonlinear Vibrations. Published online via <http://ecommons.library.cornell.edu/handle/1813/28989> (2012)
6. Cartmell, M.P.: Combination instabilities and nonlinear vibratory interactions in beam systems. Ph.D. thesis, University of Edinburgh (1984)
7. Khanin, R., Cartmell, M.P., Gilbert, A.: Applying the perturbation method of multiple scales. Math. Educ. Res. **8**(2), 19–26 (1999)
8. Khanin, R., Cartmell, M.P.: A computerised implementation of the multiple scales perturbation method using *Mathematica*. Comput. Struct. **76**, 565–575 (2000)

9. Khanin, R., Cartmell, M.P.: Parallelisation of perturbation analysis: application to large-scale engineering problems. *J. Symb. Comput.* **31**, 461–473 (2001)
10. Cartmell, M.P., Ziegler, S.W., Khanin, R., Forehand, D.I.M.: Multiple scales analyses of the dynamics of weakly nonlinear mechanical systems. *Trans. ASME Appl. Mech. Rev.* **56**(5), 455–492 (2003)
11. Forehand, D.I.M., Khanin, R., Cartmell, M.P.: A Lagrangian multibody code for deriving the symbolic state-space equations of motion for open-loop systems containing flexible beams. *Math. Comput. Simul.* **67**(1/2), 85–98 (2004)
12. Forehand, D.I.M., Cartmell, M.P., Khanin, R.: Initial development towards an integrated fully symbolic-analytical multibody code. *Int. J. Mech. Eng. Educ.* **33**(2), 149–176 (2005)
13. Forehand, D.I.M., Cartmell, M.P.: The implementation of an automated method for solution term-tracking as a basis for symbolic computational dynamics. *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.* **225**(1), 40–49 (2011)
14. Motazed, N.: The development of solvers for symbolic computational dynamics. Ph.D. thesis, University of Sheffield (2017)
15. Motazed, N., Cartmell, M.P., Rongong, J.A.: Extending the functionality of a symbolic computational dynamics solver by using a novel term-tracking method. *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.* (2017) <https://doi.org/10.1177/0954406217737104>