Symbolic multibody modeling based on recursive multibody operators and expression atomization

Aitor Plaza[†], **Javier Ros**[†], **Xabier Iriarte**[†]

[†]Department of Mechanical, Energetic and Materials Engineering
Public University of Navarra
Campus Arrosadia s/n, 31006 Pamplona, Navarra, Spain
[aitor.plaza, jros, xabier.iriarte, jokin.aginaga]@unavarra.es

Abstract

It has been shown [1, 2, 3, 4] that symbolic modeling of multibody systems can produce highly efficient code for the evaluation of the functions typically appearing in multibody formulations (Jacobians, mass matrix,...). Most frequently, the efficiency of this code greatly surpasses that of non-symbolic approaches. This is particularly true when compiler technology methods are used to avoid the evaluation of sub-expressions that repeatedly appear in different expressions. We refer to the procedures leading to these optimizations as "atomization" [2].

Nevertheless symbolic generation of multibody equations can be a task with high complexity that can, in practice, limit or difficult the work with complex and even not-so-complex multibody systems. This is, probably, the mayor bottleneck in the symbolic generation of multibody equations: On the one hand, it is the well known problem of symbolic "explosion" in the size of symbolic expressions. On the other, it is the one of generating optimal code by the way of atomization.

Nowadays there is a handful of packages that symbolically generate the code for the simulation of multibody systems [5, 6, 7, 8]. Each of them deals with the previous problems adopting different compromises and with different degrees of success. For example, some program might decide to use a specific recursive formulation and relative coordinates with an on-the-way atomization procedure as this will almost remove the repetition of sub-expressions, thus enormously lowering the computational demands that the atomization would otherwise require (this is one of strengths in [5]). These compromises limit, to some extent, the general purpose applicability of the packages in the sense that they limit the implementation of a particular multibody procedure that can exploit the high level of code optimization that is available for a specific formalism. Think of using non-relative coordinates and a non recursive multibody formalism. This has lead to the use of general purpose packages like Matlab Symbolic Toolbox or Maple to help in the implementation of special purpose multibody procedures. This, for example, motivated the work in [4]. This approach can be cumbersome, and it is frequently non optimal in the generated code and/or in the time and memory expended to do it. These problems are to an important extent related to the lack of control and knowledge on the internal workings of the corresponding symbolic kernel, as these kernels are not open. This in turn motivated our work [2].

This paper describes some algorithms and techniques that can be used to bring the described benefits to the general multibody practitioner. As a proof of concept for performance evaluation, as well as a way to make these accessible to the researchers in the field of multibody, the described algorithms and techniques have been implemented in a new version of the in-house former LIB3D_MEC_GiNaC multibody symbolic library [2].

Even if focused to the specific case symbolic multibody systems, the proposed algorithms and techniques are formulation and coordinate-type agnostic. Nevertheless they take advantage of any possible underlying structure/topology of the multibody system, implicitly contained in its parametrization, to allow for the optimal generation of code while attaining a reasonable symbolic computational cost. This algorithms rely on the same point and base tree-structures described in [2], but the procedures presented in that paper have been redesigned in order to maximize the performance of the atomization obtainable from the structure/topology implicit on the parametrization. Basically primitive multibody operators, such as the symbolic computation of position vector of a point, or a given angular velocity vector or velocity vector of a point are computed maximicing the advantage implicit in the structure/topology present in the parameterization. The internal representation of the symbolic expresions is always atomized, each time a new operation is performed a new round of atomization is performed on the new expresions, this

minimizes the memory footprint and the complexity of exporting optimized expresions, operators are implemented so that superfluous atom generation is minimized, duplicated atoms are removed on the fly by the way of hash tables tables that allow for constant time lookups enormously lowering the computational burden of this search.
The standard, symbolic differentiation and substitution operations, have been optimized so that deatomization is not required when performing such operations. These algorithms recursively travel the underlying recursive atom structure (not to be confused with the multibody structure) that internally represent a given expresion (atoms are defined in terms of other atoms). This enormously limits the operation count and memory requirements of the algorithms. As with multibody operators, duplicated atoms are generated seach for duplicates, as before, each time a new algebraic operation is performed.
Basically we translate the typical recursivity found in $O(n^3)$ recursive algorithms, to the operator level. The recursivity usually defined at the frame level, is splited down at the base and point structure level (two trees instead of one). This allows to exploit more in deep any fine grained recursivity that may be implicit in the parameterization despite of the parameterization used. This in turn allows to get superior performance in any context. For example we can get equal or better performance than $O(n^3)$ recursive formulations whenever a similar parameterization is used (relative coordinates), just by applying a the virtual power principle with no other care than that of using the provided multibody operators when writting down the equations.
The "better" comes from the fact that some operation, for example operations that compute to zero or one, do not need to be performed, as they are removed for the final expressions.
This paper will provide a technical overview of those techniques and algorithms providing case studies and numerical examples with the aim of checking the benefits of the algorithms.

References

[1] J-C. Samin, P. Fisette. Symbolic Modeling of Multibody Systems.
Kluwer Academic Publishers, Dordrecht, 2003.

[2] J. Ros, L. Arrondo, J. Gil, X. Iriarte. LIB3D_MEC_GiNaC, a library for symbolic multibody dynamics. In: ECCOMAS Multibody Dynamics, Milano (2007)

[3] P. Gossens, Ch. Schmitke. Symbolic Computation Techniques for Multibody Model Development and Dynamic Analysis In: ECCOMAS Multibody Dynamics, Brussels (2011)

[4] J. Gil. PhD Thesis. Symbolic algebra based preprocessor for multibody dynamics simulation (in Spanish).

[5] Robotran web site: http://www.robotran.be/

[6] Dymola web site: http://www.3ds.com/products-services/catia/capabilities/systems-engineering/modelica-systems-simulation/dymola

[7] MapleSim web site: http://www.maplesoft.com/products/maplesim/index.aspx

[8] Wolfram SystemModeler web site: http://www.wolfram.com/system-modeler