EFFICIENT OBJECT-ORIENTED PROGRAMMING OF MULTIBODY DYNAMICS FORMALISMS

U. Rein
Institut A für Mechanik
Universität Stuttgart
Pfaffenwaldring 9
W-7000 Stuttgart 80
Germany

ABSTRACT. A class structure for the object-oriented, numerical language C++ is presented to replace the built-in scalar type double with a new scalar class se (scalar environment). Matrix- and vector-classes programmed with the use of the scalar class se allow comfortable object-oriented programming and modifying of multibody formalisms. Because an instance of a se stores further information besides the magnitude of a scalar number, operations with se's can be optimized with respect to multibody formalisms. The number of floating point operations remaining after optimization can be counted and an executable program is created, which executes only these minimum number of floating point operations. This serves as a basis for comparing, modifying and optimizing given formalisms with respect to the multibody system under investigation. As an example, an implementation of the recursive formalism is presented in the paper.

1. Introduction

During the last decades a large number of multibody dynamics formalisms for different purposes have been published. Especially the development of the recursive formalism [1], [18], [7], [4] has shown the need for comparing the efficiency of multibody dynamics formalisms. This has been done using different methods, e.g. counting the necessary operations by hand [4], [15], with the aid of a symbolic manipulation program [16], or by measuring the CPU-time for one evaluation of the right-hand side [13]. For a serial chain of rigid bodies connected by revolute joints, Fig. 1 shows the basic difference between an implementation of the recursive and an implementation of the nonrecursive method described in [12]. While the CPU-time of the recursive formalism increases with Order(n), i.e. linearly with the number of joints n, the nonrecursive method increases with Order(n^3). The intersection point is between five and six joints. Surprisingly, this doesn’t hold any more for systems with parallel tree structure, i.e. systems where the number of branches is close to the number of bodies, since the nonrecursive method used here for the comparison [12] also shows an almost linear behaviour. Thus, for many practical applications an optimized nonrecursive method is faster than the recursive method.

M.F.O.S. Pereira and J.A.C. Ambrósio (eds.), Computational Dynamics in Multibody Systems, 37-47.
There were published a number of versions of the recursive formalism [7], [4], [2], [17]. They use different coordinate systems and reference points to evaluate the recursive formalism. These versions of the recursive formalism have been compared in [15] and [16], using again a serial chain of rigid bodies, connected by revolute joints. Fig. 2 shows the differences between the versions of the recursive formalism.

Figure 1: Comparison of the recursive with a nonrecursive formalism [13]

Figure 2: Comparison of variants of the recursive formalism for a serial chain with revolute joints [15]