A unifying object-oriented methodology to consolidate multibody dynamics computations in robot control

Am Fachbereich Informatik der Technischen Universität Darmstadt eingereichte Dissertation

zur Erlangung des akademischen Grades eines Doktor-Ingenieurs (Dr.-Ing.)

von

Dipl.-Phys. Robert Höpler (geboren in München)

Referenten der Arbeit: Prof. Dr. Oskar von Stryk
Prof. Dr.-Ing. habil. Dieter Bestle

Tag der Einreichung: 23. 07. 2004
Tag der mündlichen Prüfung: 06. 08. 2004
# Contents

Notation VII

Outline X

1 Introduction 1
   1.1 Motivation ................................................. 1
   1.2 Contents and contributions ............................... 4
   1.3 Literature survey ......................................... 6

2 Principles of dynamics computations: modeling robots on mechanical, abstract, and algorithmic level 8
   2.1 Dynamics preliminaries .................................... 9
      2.1.1 Classical mechanics of a single gyrostat ............. 9
      2.1.2 Coordinate representations ............................ 11
   2.2 Paradigm for abstract robot model specification ........ 16
      2.2.1 Component specification model ......................... 17
      2.2.2 Robot specification model ............................ 26
   2.3 Relating recursive dynamics algorithms to dataflow ....... 28

3 Multibody dynamics algorithms for robots 35
   3.1 Fundamental algorithms for rigid manipulators revisited ... 35
      3.1.1 Recursive algorithms—general considerations ........ 35
      3.1.2 Inverse dynamics ...................................... 40
      3.1.3 Forward dynamics ...................................... 43
   3.2 Advanced and new dynamics algorithms .................... 45
      3.2.1 New elastic joint inverse dynamics .................... 46
      3.2.2 Obtaining sensitivity information ..................... 57
      3.2.3 Inverse dynamics for closed chains ................... 59
      3.2.4 Operational space dynamics ........................... 62
# Methodology for operational robot models

4.1 Classes for model specification .................................................. 67
4.2 Mapping robot models .............................................................. 69
   4.2.1 Mapping to a dataflow network ......................................... 70
   4.2.2 Model transformation and optimization ............................... 75
4.3 Reusable components of robot models in control .......................... 77
   4.3.1 Basic solver interfaces .................................................... 77
   4.3.2 Interfaces for algorithm building blocks .............................. 78
4.4 Aspects relevant to realizing a dynamics framework ..................... 81

# Selected applications in research and industry

5.1 DLR light-weight robot inverse dynamics .................................. 85
5.2 Modeling a palletizing robot .................................................... 88
5.3 Calibration models of manufacturing robots ............................... 90
5.4 Trajectory optimization of legged robots .................................. 92

# Useful identities

A Useful identities ................................................................. 100

# Index

Index ......................................................................................... 102

# Bibliography

Bibliography .............................................................................. 103
Outline

The work remedies a class of problems arising from the intermingling of mathematical modeling of robotic mechanisms, numerical schemes, and implementation and integration into a whole robot control software architecture in common practice. The solution proposed in this thesis is a carefully designed object-oriented class hierarchy supporting the robot control engineer in the processes of specifying, implementing, and performing multibody computations required in robot control systems.

The standard approach to multibody computations is to design separate models for different applications of the same robot resulting in stand-alone numerical schemes—usually relying on individual idealizations and formalisms. This has become insufficient with the increasing demand for model-based robot control. The integration of a greater number of schemes proves to be a key enabler. The problem of integration is tackled here by means of an object-oriented methodology consolidating the several conceptual levels of robot modeling, involving abstract, mechanical, numerical, and software representations of the same system under consideration, i.e., the robot mechanism.

The common ground for the numerical schemes as well as their implementation is a new paradigm for the precise description of the mechanical components of a robot and the desired computations. The numerical and mathematical description of the governing equations is captured by a new port-based extension of the mathematical framework of symbolic spatial operators. A series of multibody algorithms, standard ones as well as some novel ones for special purpose are recast in an object-oriented form and classified according to this paradigm. A new dataflow-driven model of computation is proposed for the efficient implementation of recursive algorithms which directly applies to object-oriented software systems. This alleviates the coupling of several models, as shown by some selected example applications, and motivates the effort of applying this methodology to robot dynamics computations.
Chapter 1

Introduction

1.1 Motivation

The description of robotic motion, comprising movements of manufacturing robots as well as legged locomotion of a humanoid robot, requires models describing approximations of particular aspects of reality. Models are indispensable when robots move and when they interact with their environment. Processing and interpretation of perception data or strategies to autonomously execute predefined tasks, all must be specified and performed in terms of abstractions of the real robot and its world. When speaking of a model, however, its final representation will be segments of code running on one or more digital computers that control the energy applied to the actuators.

A very powerful abstraction amenable to further analytic treatment as well as to implementation on digital computers is a mathematical representation of general physical laws and processes governing the physical properties of the robot. We further refer to such a representation as a mathematical model.

The dominant features of reality to be modeled in robotics are the macroscopic mechanical properties of a robot’s mechanism. This comprises its desired motion, its actuators, sensors and interaction with the physical environment a robot exists in. A mathematical model describing all this is referred to as a mechanical model.

A computer program or portions of code that can be executed are called a computational model [82] if they represent certain aspects of a mathematical or mechanical model. The most prominent is the numerical calculation of kinematic and dynamic quantities determining robot motion.

If relevant properties of a model are captured by an abstract representation, this representation by itself is a model, referred to as abstract model or a model specification. The model specification is sufficiently detailed, if one can deduce all required properties of the model from the specification.

The objective of the present work is twofold. First it is shown how computational models emerge from corresponding abstract mechanical models in a seamless and modular way. This problem belongs to the domain of computational mechanics. The resulting computational models provide the physical state of the robot required by hardware and software controlling its motion, cf. Figure 1.1. It should be noted that models can be involved in each part of a robot software down to the level of actuator control. Here, models are discussed that describe aspects of the complete mechanism. Second, design principles are proposed how the involved computational models can be defined and decomposed to integrate well in a robot control system architecture. This process of integration into a large software-based system requires methods from the domains of multibody system dynamics, systems and software design, and applied robotics, each imposing specific constraints on the proposed solution.
CHAPTER 1. INTRODUCTION

The science creating computational models of complex mechanical systems is called multibody system (MBS) dynamics. The main MBS objective is to investigate the evolution of physical quantities with respect to time, briefly denoted by the dynamics, to establish equations of motion and to derive numerical schemes for their efficient evaluation. The astonishingly complex non-linear dynamics of this kind of systems stems predominantly from the coupling of small and simple systems to larger ones, usually bodies connected by joints. The research field of rigid MBS has matured in the last three decades, indicated by several landmark books by Featherstone [40], Murray et al. [98], Roberson and Schwertassek [115], Wittenburg [149], to name a few. An immense variety of formalisms and algorithms to create computational robot models for a vast range of possible applications has been developed by scientists in the field of computational mechanics, e. g., see Schiehlen [123].

At first glance it would be sufficient to establish the equations of motion of the robot just symbolically and, if available, apply a sufficiently powerful symbolic formalism to derive all desired further equations and to generate code. There are some approaches that follow this idea, e. g., [95], but fail in many real-world applications due to some of the following practical constraints listed below. In case of a robot control system, which is often an embedded system with hard real-time constraints, one is faced with robotics domain specific constraints:

(i) **Efficiency**: There are various possibilities to solve a system of equations numerically. Each variant may be efficient for a certain regime. The most prominent example in robot dynamics are so-called $O(N)$-formalisms and composite rigid body algorithms. The former outperforms the latter only for systems with more than five bodies [91]. This valuable domain-knowledge is hard to capture in a completely automated procedure. Furthermore in a complex software system the required reuse of results imposes additional constraints on generated code not implicated in a symbolic approach.

(ii) **Limited resources**: Resources might be limited, so either symbolic code generation requiring a compile-to-code step is not possible or in presence of switching behaviour it is hard to store models for each system state due to the combinatorial explosion.

(iii) **Maintenance**: Completely automatically derived code is cumbersome in several senses, because in most cases the numerical scheme itself is a neglectable amount of code compared to interfacing the
results to the rest of the application. It is often impossible to trace errors and to debug because there is no transparent relationship between the equations and the code.

(iv) **Safety**: Robotics is a safety critical domain because man-machine interaction mandates avoidance of damage and injury to humans. Still it is difficult to formally verify the correct function of completely automatically generated code and often manual coding is preferred in practise.

Each dynamics algorithm has its strengths and advantages but relies on certain modeling assumptions and axioms. In implementation, e.g., in a real-time control system for one certain type of robot, maximum performance is mandatory and the system designer is forced to integrate various algorithms and balance precision, generality, run-time characteristics, and implementation effort of a model or formalism. Efficient solutions often exist but often are problem specific in particular in the robotics domain, see, e.g., [134, 150]. In research and industrial environments this balancing is currently repeated each time a new type of robot or control system is developed depending on the current state-of-the-art, and often software is developed by manual coding. One flexible and efficient solution to this problem is re-use and configuration instead of re-implementation.

The central topic of this work are numerical schemes for multibody computations with emphasis on the class of recursive methods, because they have shown to be most efficient. The idea is to go beyond ‘pure’ implementation of numerical algorithms, where function and efficiency are dominant aspects, and extend to a systems view. A dynamics algorithm inevitably is part of a larger problem architecture. It exists within a software and hardware architecture as well, where not necessarily mathematically described aspects are of concern. The classical simulation environment used by mechanical engineers is just one concrete realization by means of time-integration. Obvious aspects are the interaction with a changing physical environment, communication, flexibility, time-to-market, especially in the field of real-time robot control. To put it in other words: A mathematical formalism and its realization is one way to model a complex technical system. Hatley and Pirbhai stress this important aspect that modeling may not be restricted to a numerical formalism and its implementation. A complex technical system requires to model the requirements and the design:

> [...] components that make up a system—both hardware and software components—are highly interrelated, and, in order to successfully perform their intended function, they must integrate well. The system specification process, therefore, must define the system as a whole, as well as its partitioning into hardware and software components. It must define what problem the system is to solve (its requirements) and how that system is to be structured (its architecture or design structure) [51].

Here we follow a similar idea to focus on design principles to systematically solve the problem of performing multibody computations in robot control. Structured methods can help to approach the vision of a unified MBS model in a robot control system, which does not mean the naive concentration on one outstanding formalism. The focus is not just the model itself, but also its behaviour and its meaning in various contexts. Though the final step is the generation of code solving MBS equations from a simple description of the system elements, the viewpoint is that of a general mapping from multibody-formalisms to a space of abstract (software) entities without sacrificing the power of specialized domain-specific MBS algorithms.

There are several possibilities to structure this kind of problem. This work follows object-oriented (oo) modeling and component-oriented design to simplify model and code generation and software reuse. One motivation to employ object-oriented analysis is that the analogy between the physical model and the
software model is very fruitful [1] because formulating a problem in terms of notions from the problem domain increases comprehensibility. An object-oriented model is a key enabler for another reason: it can be combined with general principles of software engineering, such as separation of concerns, correctness, reliability, and robustness [37].

As the number of robots and other automation subsystems grows, integration becomes increasingly difficult. Software integration costs alone for the United States’ robotics industry are estimated at $1 billion annually [114]. Therefore dynamics computations must integrate in software and hardware architectures with, e.g., hard timing constraints and limited resources. Chen and Yang [25] report the need to integrate several computational models of the robot in one application. Their scope is to use the same algorithms and codes for simulation as well as for control purpose. The problem of covering the whole of multibody algorithms in a robot control system (RCS) is analogous to MBS equations themselves where the high complexity stems from the interconnection of smaller blocks of low or medium complexity. Especially embedded code will increasingly consist of interacting software components [81, 139]. Combining relatively simple blocks containing complicated parametrization and inner dynamics leads to very heterogeneous architectures being non-trivial to design, implement, debug, and maintain. Especially specification and parametrization of the computational dynamics models are often underrated, but are crucial points in code generation and formalism transfer, and thus of true practical relevance.

The goal is to provide a reusable context for components performing multibody computations, i.e., to enable the reuse of algorithm design and code. A lack of reusability is only partially a problem of a lack of documentation. By virtue of so-called frameworks object-oriented systems reach a maximum of reusability [69]. One challenge is that frameworks are among the most complex of all software design approaches [44]. A framework should apply to all imaginable applications in the domain under consideration. This requires profound domain knowledge, i.e., mechanics of robots, multibody formalisms and software architectures, to create a comprehensive design of flexible and extendible characteristics.

### 1.2 Contents and contributions

#### Main thesis

The preceding motivation leads to the main thesis which is the attempt to reduce the gap between the heterogeneous worlds of

(i) modeling (model specification) of robotic mechanisms, comprising abstraction and meaning of the abstract representations,

(ii) existing and upcoming multibody formalisms, on the one hand algorithms and their efficient implementation and coupling, on the other hand,

(iii) numerical requirements from, e.g., robot control schemes, simulation, and trajectory optimization methods,

(iv) real-world robotic software applications, demanding modular and extensible, interoperable, and lightweight code running off-line or on real-time systems.

The vision drawn in this work is a general purpose robot dynamics methodology to support a robot control design engineer in (i) robot model specification, (ii) automatic code generation and manual implementation from an optimally chosen mechanical model and multibody formalism, and (iii) integrating the
computational models and components in an evolving software architecture for control, optimization, and simulation. The focus is on a sound mathematical formalism, using object-oriented design principles and reaping maximum performance. The desired result from this modeling of robot models, however, is not a general purpose multibody program demanding the least domain knowledge possible from the user and covering as many problem classes as possible.

**Modeling robotic mechanisms and algorithms**

Without multibody domain expertise it is hard to define practical requirements for software architectures such as frameworks. Ideally the developed approach has to cover all existing and upcoming cases of application. Chapter 2 formulates the relevant aspects of multibody computations in robot control ranging from mechanics to formalisms and algorithms. The physics (classical mechanics) of one body and several bodies are discussed in Section 2.1 to show the importance of coordinate representations, dynamics formalisms and their realizations.

Section 2.2 takes a closer look from an abstract, high-level viewpoint with emphasis on topological issues and structured methods [51]. The complexity arising in MBS model specification is investigated. A general paradigm for the specification based on entity-relationship-attribute (ERA) paradigm [26] and associated semantics are developed. Identifying common implicit assumptions in abstract models of robots improve formalism transfer and consistency. Entities relevant for code generation and implementation are identified which form the basis of the methodology.

To deal with the vast number of numerical multibody algorithms, a rough classification is proposed in Section 3.1.1 to embed the schemes in a ’component space’ spanned by chosen formalism, coordinate representation and desired output values. A number of representative existing multibody algorithms covering a large range of applications are introduced to this classification scheme accompanied by some new specialized algorithms. Several new useful mathematical MBS expressions, e.g., for control applications are derived.

The crucial symbolical description of multibody equations is based on the well-known spatial operator algebra (SOA) by Rodriguez et al. [117]. In order to overcome the restrictions of this powerful mathematical formalism, the Port-Based Spatial Operator Algebra is introduced in Section 2.3 which reaps the advantages of intuitive object-oriented modeling and implementation techniques and the power and expressiveness of the symbolic operator formulation. This allows for a uniform specification and presentation of SOA operator expressions of general multibody-systems including more general components and topologies. The ability to establish spatial operators from topological and component properties pays off in sections 3.1 and 3.2 either for symbolic manipulation or object-implementation while preserving the valuable algebraic properties of the SOA. A dataflow interpretation of recursive algorithms in Section 2.3, which is given for the first time, prepares the ground for efficient code-generation and implementation of this class of algorithms.

**Operational architecture and applications**

The representation of robot model specifications in an object-oriented system is investigated in Section 4.1 and . This section introduces an ontology of interrelated classes describing the mechanism, the desired numerical algorithm and modeling assumptions. This work proposes a clear separation between model specification and transformation issues such as code generation to reach a maximum of generality. This enables the software designer including characteristics that are concern of the user and avoids placing
constraints on design level which is an abstraction level being too high for, e. g., choosing coordinates or
the time and place of code generation [51].

The specification model is subjected to several kinds of mapping in Section 4.2. The discussed mappings are code generation for dataflow networks, model optimization and textual representation. A new object-oriented architecture is proposed in Section 4.2.1 to create executable algorithm objects from a given specification model to decouple the various components. This alleviates extension of applications but still allows interoperability between new and old components. At every level of abstraction the user of the methodology is able to interact with the model specification if required by the concrete application, e. g., to optimize topological properties without the need to go down on lowest level, the equation level. In Chapter 4.3 some fundamental algorithmic robot dynamics building blocks required by control applications, multibody scenarios and boundary conditions are formulated in terms of tools developed in Section 2.2. A framework becomes concrete by choosing an object-oriented programming language [69, 70]. Requirements for a C++ realization are discussed in Chapter 4.4. The architecture presented is applied to several problems arising from industrial and scientific problems in Chapter 5 substantiating the applicability of this work especially in heterogenous robot control applications.

Finally, appendices with a small glossary and an index and tables of useful mathematical identities are intended to prevent the reader from getting lost in notation and connotations.

1.3 Literature survey

Software for multibody computations in robotics

The main focus is on symbolic recursive formalisms to compute the highly non-linear dynamics of robots for they are known to be numerically efficient, which is indispensable in real-time control of robots. The first recursive technique reported was developed by Vereshchagin [144] in 1974. For a review and classification of recursive schemes see Jain [60].

The existing packages can be subdivided according to the offered computations and the type of application, either an executable generating code, or a programming library usually realized as collections of source code. The main objectives of nearly all commercial and non-commercial tools are the generation of equations of motion from simple input model description and time integration (simulation) of the differential equations.

For a survey of dynamics formalisms developed until 1988 refer to the book of Roberson and Schwertassek [115]. A number of packages mainly for simulation purpose are reviewed in [123]. The disadvantages of large commercial general purpose tools are lack of computational efficiency and flexibility, consumption of resources and high effort to integrate own optimized and specialized components, all paramount when migrating to embedded systems like robot controls. This work does not intend to provide yet another simulation package, but to support the robot domain specialist in implementing various methods and formalisms.

This basic idea of providing components for multibody computations is taken up in the package AUTOLEV [122]. This collection of functions helps the multibody dynamics domain specialist in formulating the equations of motion, but is restricted to Kane’s equations and generation of a complete simulator code. The main advantage is full control over the equation formulation process, inevitable when exploiting maximum performance.

The package MOBILE [74] for simulation of various types of mechanical systems is based on object-oriented principles and implemented in C++. A component oriented design helps the domain engineer in
1.3. LITERATURE SURVEY

describing the multibody system properties and to generate code as source code. This approach focuses
on time integration, specialized for medium to heavily constrained mechanical systems. However, such
aspects are not prevailing in robot control.

DYNA-MECHS is a multi-purpose collection of robot dynamics algorithms implemented in C++ by McMil-
lan et al. [92]. It is designed for simulation of under-water vehicles and legged robots and comprises some
popular dynamics algorithms for certain types of multibody systems. It is merely driven by implementa-
tion of multibody formalisms but does not emphasize a high-level design.

A C++ package intended for use in robotics is ROBOOP [46]. It comprises several classes for kinematics
and dynamics computations, but is restricted to certain types of robots and choices of coordinates. A
similar library is by Corke [28] implemented in Matlab scripting language. DARTS [61] is a collection of
functions written in C forming an engine used for robot simulations especially for space applications [16].
The idea to use the same algorithms and codes for simulation as well as for control purposes has been
reported by Chen and Yang [25]. Their approach is restricted to the class of tree-structured robots and
one special dynamics formalism.

There is a great number of commercial tools mostly dedicated to simulation, i.e., time integration, of
general multibody systems, often providing export of symbolic code, too. To name only a few prominent
examples: SIMPACK [120] is based on an $O(N)$-formalism, ADAMS [103] is based on Kahn’s equations.
SIMMECHANICS [89] are multibody blocksets which enables control system design for mechanical sys-
tems within the SIMULINK environment.

Software architectures for robot systems

From the perspective of a high-level software design for robot control software the project open source
robot control software (ORocos) [22, 88] is closest to the ideas of modeling and software design de-
veloped in this work. The long-term objective is to provide generic and public-domain C++ software
components for all concerns of robot control. The block of kinematics and dynamics computations was
not developed during the period this work was performed.

SMART [11] is a component-oriented control architecture for tasks on a higher level than model com-
putations such as collision avoidance, trajectory generation, integration of sensor and haptic devices,
especially in the field of teleoperated systems. It is possible to adapt the software system by reconfig-
uring various modules representing operational modes. Code is generated, compiled, downloaded and
initialized including a re-synchronization of the robotic system even on a system with several CPUs.

RIPE by Miller and Lennox [93] is a set of C++ base classes intended to represent a robotic system by base
classes ‘WorkPiece’, ‘Station’ and ‘Device’. Those generic classes can be derived by a user to implement
specific features of a real system.