5th MATHMOD VIENNA

Proceedings

Volume 1: Abstract Volume
Volume 2: Full Papers CD

5th Vienna Symposium on Mathematical Modelling

February 8-10, 2006
Vienna University of Technology,
Austria

ARGESIM Report no. 30

ARGESIM REPORT

ISBN 3-901608-30-3

ARGESIM REPORT

ARGESIM Report no. 30

I. Troch, F. Breitenecker (Eds).

Proceedings
5th MATHMOD Vienna

Volume 1: Abstract Volume
Volume 2: Full Papers CD

5th Vienna Symposium on Mathematical Modelling

February 8-10, 2006
Vienna University of Technology, Austria

ARGESIM - Verlag, Vienna, 2006
ISBN 3-901608-30-3

ARGESIM Reports

Published by **ARGESIM** and **ASIM**, Arbeitsgemeinschaft Simulation,
Fachausschuss GI im Bereich ITTN – Informationstechnik und Technische Nutzung der Informatik

Series Editor:
Felix Breitenecker (ARGESIM / ASIM)
Div. Mathematical Modelling and Simulation, Vienna University of Technology
Wiedner Hauptstrasse 8 - 10, 1040 Vienna, Austria
Tel: +43-1-58801-10115, Fax: +43-1-58801-10199
Email: Felix.Breitenecker@tuwien.ac.at

ARGESIM Report no. 30

Titel: Proceedings 5th MATHMOD Vienna –
5th Vienna Symposium on Mathematical Modelling
Volume 1: Abstract Volume
Volume 2: Full Papers CD

Editors: Inge Troch, Felix Breitenecker
Div. Mathematical Modelling and Simulation,
Vienna University of Technology
Wiedner Hauptstrasse 8 - 10, 1040 Vienna, Austria
Email: Inge.Troch@tuwien.ac.at

ISBN 3-901608-30-3

ARGE Simualtion News (ARGESIM)
c/o F. Breitenecker, Div. Mathematical Modelling and Simulation, Vienna Univ. of Technology
Wiedner Hauptstrasse 8-10, A-1040 Vienna, Austria
Tel.: +43-1-58801-10115, Fax: +43-1-58801-42098
Email: info@argesim.org; WWW: http://www.argesim.org

OBJECT-ORIENTED DYNAMICS MODELING OF WALKING ROBOTS FOR MODEL-BASED TRAJECTORY OPTIMIZATION AND CONTROL

R. Höpler[1], M. Stelzer[2], and O. von Stryk[2]

[1] TESIS DYNAware GmbH
Baierbrunnerstrasse 15, 81379 Munich, Germany
email: robert.hoepler@tesis.de

[2] Technische Universität Darmstadt
Hochschulstrasse 10, 64289 Darmstadt, Germany
Phone: + 49 6151 16-4722, Fax: + 49 6151 16-6648
email: stelzer@sim.tu-darmstadt.de

Modeling of walking robots is an intricate subject when dealing with optimization and model based control. The resulting multibody systems with a free floating base lead to high dimensional equation systems and a frequently changing kinematic structure due to time-varying contact situations. However, specialized algorithms that treat the contacts separately and thus may make use of the tree structure of the system can be applied. This paper discusses the development and application of object-oriented modeling and implementation techniques to achieve a representation of the mechanical model amenable to the various requirements by legged robot applications. This leads to a uniform, modular, and flexible code generation while reaping the performance of efficient domain-specific articulated body algorithms.

Trajectory optimization problems of bipedal and quadrupedal robots are investigated as applications. The optimal control problems involve different optimality criteria such as energy or time and involve the complete dynamics of the system as side conditions as well as several nonlinear inequality and boundary constraints that ensure stable, symmetric gaits. It is shown how a high-level specification of multibody dynamics models using component libraries serves as a basis for generation of a number of modules forming an “overall” computational dynamics model tailored to the needs of the optimal control techniques involved. Exact sensitivities may be calculated directly from the dynamics algorithm and thus in future might help for better convergence of the optimal control problems.

The selected examples illustrate how this approach tackles the emerging complexity by integrating various algorithm modules for, e.g., equations of motion, non-linear boundary conditions, symmetry and transition conditions, for multiple phases that differ in the number of legs in contact with ground during motion. Variations of the mechanical structure, e.g., contact conditions can be treated either by multiple models or by reconfiguration of one model.
Parts of the model may be used, e.g. one single leg for considerations of the leg swing height. Numerical results are presented for time- and energy-optimal walking trajectories of full three-dimensional models of a humanoid robot and a Sony four-legged robot.

References

M. Buss, M. Hardt, J. Kiener, J. Sobotka, M. Stelzer, O. von Stryk, and D. Wollherr. Towards an autonomous, humanoid, and dynamically walking robot: Modeling, optimal trajectory planning, hardware architecture, and experiments. In Proc. IEEE/RAS Humanoids 2003, page erscheint. Springer-Verlag, 2003.

M. Hardt and O. von Stryk. Dynamic modeling in the simulation, optimization, and control of bipedal and quadrupedal robots. *Z. Angew. Math. Mech.*, 83(10):648–662, 2003.

R. Höpler, M. Stelzer, and O. von Stryk. Object-oriented dynamics modelling for legged robot trajectory optimization and control. In *Proc. IEEE Conf. on Mechatronics and Robotics (MechRob)*, pages 972–977, Aachen, Sept. 13-15 2004. Sascha Eysoldt Verlag.

M. Stelzer, M. Hardt, and O. von Stryk. Efficient dynamic modeling, numerical optimal control and experimental results for various gaits of a quadruped robot. In *CLAWAR 2003: International Conference on Climbing and Walking Robots, Catania, Italy, Sept. 17-19*, pages 601–608, 2003.

O. von Stryk. User’s guide for DIRCOL version 2.1: A direct collocation method for the numerical solution of optimal control problems. Technical report, Fachgebiet Simulation und Systemoptimierung, Technische Universität Darmstadt, 2001. http://www.sim.informatik.tu-darmstadt.de/sw/dircol.

OBJECT-ORIENTED DYNAMICS MODELING OF WALKING ROBOTS FOR MODEL-BASED TRAJECTORY OPTIMIZATION AND CONTROL

R. Höpler[1], M. Stelzer[2], and O. von Stryk[2]

[1]TESIS DYNAware GmbH
Baierbrunnerstrasse 15, 81379 Munich, Germany
email: robert.hoepler@tesis.de

[2]Technische Universität Darmstadt
Hochschulstrasse 10, 64289 Darmstadt, Germany
Phone: + 49 6151 16-4722, Fax: + 49 6151 16-6648
email: [stelzer|stryk]@sim.tu-darmstadt.de

Abstract. Modeling walking robot dynamics is an intricate subject when dealing with optimization and model based control. When modeled as multibody systems with a free floating base walking robots lead to high dimensional equation systems. Walking involves frequent changes in the kinematic structure due to varying contact situations. However, specialized algorithms that treat the contacts separately and thus can make use of the tree structure of the system can be used. This paper discusses the dynamics algorithms used and discusses the development and application of objectoriented modeling and implementation techniques to achieve a representation of the mechanical model amenable to the various requirements by legged robot applications. Optimal control techniques are involved to generate optimal walking trajectories of a biped and a quadruped robot. Numerical results are shown.

1 Introduction

Generating flexible dynamic motions of walking robots by optimal control techniques puts high demands on dynamics computations. Walking robots are characterized by a high number of degrees of freedom, where usually one joint is driven by a single motor. The base of the robot is free-floating. Frequent changes in the kinematic structure occur due to varying contact situations during walking. If the contacts are considered separately, the systems show tree structure, i.e. there are generally no kinematic loops. General multibody algorithms can not benefit from this special structure. To cope with the high complexity of the nonlinear dynamics of legged robots model-based methods for real-time actuator control, for trajectory optimization, and for controller design specifically tailored to the one scenario of legged robots must be developed. For rapid and virtual prototyping a most desireable goal is not only to apply the same abstract dynamic model representations and mathematical models but also as many parts as possible of the same code for off- and on-line evaluations of legged robot dynamics during all stages of design, development, implementation and operation of a legged robot. To facilitate the investigation of new concepts of nonlinear model-based optimization and control methods also the sensitivity of the legged robot dynamics model with respect to its state variables and parameters are needed. The formalisms and tools applied at the same time have to cope with (i) the complex underlying mechanical model, characterized by many degrees of freedom, actuator dynamics, and interaction of the robot with its physical environment including time-varying contacts and collisions, (ii) the wide range of required numerical schemes, including kinematics, dynamics, sensitivity information, etc., (iii) efficient code generation which is particularly vital for on-line computations, and must rely on the power of dedicated (recursive) algorithms, and (iv) interaction of the computational model with the system it is running on, e.g., communication with sensors and actuators.

The outline of the paper is as follows: In chapter 2 the characteristcs of the dynamics of walking robots and

the dynamics algorithms used in this research are discussed. Chapter 3 outlines the optimal control problem for generating walking trajectories and the methods used for solving them. The new modeling and implementation techniques are presented in chapter 4. Numerical results are given in chapter 5.

2 Dynamics Modeling of legged Robots

Legged robots are modeled as fully three-dimensional multibody systems with a free floating base and periodically changing contact situations at the feet of the robot depending on the gait pattern of the robot. The contacts which currently are modelled as joints without elasticity and are treated separately to preserve the tree structure of the system.

2.1 Forward Dynamics Model

The joint space equations of motion of a rigid multibody system experiencing ground contact, i.e. including tip contact forces and holonomic tip constraints, are

$$\ddot{\mathbf{q}} = \mathcal{M}^{-1}(\mathbf{q}) \left(B\mathbf{u} - \mathcal{C}(\mathbf{q}, \dot{\mathbf{q}}) - \mathcal{G}(\mathbf{q}) + J_c^T(\mathbf{q})\, \mathbf{f}_c \right) \quad (1)$$

$$0 = \mathbf{g}_c(\mathbf{q}), \quad (2)$$

where $\mathbf{q}, \dot{\mathbf{q}} \in \mathbb{R}^{n_d}$ are the column matrices of joint position and velocity variables, $\mathcal{M}$ is the positive-definite joint space mass matrix, $\mathcal{C}$ and $\mathcal{G}$ are the vectors of gyroscopic and gravitational forces, the vector $\mathbf{u} \in \mathbb{R}^m$ is the vector of actively controlled joints, which is mapped with the constant matrix B. The n_c holonomic ground contact constraints $\mathbf{g}_c \in \mathbb{R}^{n_c}$ result in a constraint Jacobian $J_c = \frac{\partial \mathbf{g}_c}{\partial \mathbf{q}} \in \mathbb{R}^{n_c \times n_d}$ and $\mathbf{f}_c \in \mathbb{R}^{n_c}$ is the vector of ground constraint forces.

2.2 Reduced Dynamics Algorithm

Numerical difficulties arise when differential algebraic equations of high index, like those from equations (1) and (2) are constraints for an optimal control problem. Nevertheless, these difficulties may be avoided by transcribing (1) and (2) into a system of ordinary differential equations involving the independent states $\mathbf{q}_I$, which are global orientation and position and states related to legs in contact with the ground, and using inverse kinematics to determine the dependent states $\mathbf{q}_D$ of the other legs ("reduced dynamics method", [Har99, HvS03]):

$$\begin{aligned} \mathbf{q}_I &:= \text{global orientation, position; swing leg(s) states} \\ \mathbf{q}_D &:= \text{contact leg(s) states.} \end{aligned}$$

$\mathbf{q}_I$ may be computed from all states $\mathbf{q}$ using a constant mapping Z, i.e. $\mathbf{q}_I = Z\mathbf{q}$. The solution of the resulting reduced system of ordinary differential equations (ODEs)

$$\ddot{\mathbf{q}}_I = Z\mathcal{M}\left(\tilde{\mathbf{q}}\right)^{-1} \left(B\mathbf{u} - \mathcal{C}\left(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}\right) - \mathcal{G}\left(\tilde{\mathbf{q}}\right) + J_c^T(\mathbf{q})\, \mathbf{f}_c \right),$$

where $\tilde{\mathbf{q}}$ consists of the independent states and of the dependent states determined from inverse kinematics on position and velocity level, then may be proven to be the solution of the initial system of differential algebraic equations [Har99]. Inverse kinematics for each leg of the robot here has a unique solution if agreements concerning the bending of joints are made and if it has a solution at all (which is ensured by some constraints, cf. 3.1). The reduced dynamics algorithm in summary is: first determine dependent states from inverse kinematics, then calculate full dynamics, and finally determine independent state using the Z-mapping. As a result, 24 independet states instead of 36 dependent variables can describe the model and a set of ordinary differential equations only instead of a system of differential algebraic equations may be considered. This not only decreases the problem size but also improves convergence of the optimal control problem as a system of ordinary differential equations instead of differential algebraic equations is included to the problem as side condition.

2.3 Sensitivity Computations

For trajectory optimization the sensitivity of the state space forward dynamics model (1) w.r.t. control and state variables is required:

$$\delta\ddot{\mathbf{q}} = \nabla_{\mathbf{u}}\ddot{\mathbf{q}}\, \delta\mathbf{u} + \nabla_{\mathbf{q}}\ddot{\mathbf{q}}\, \delta\mathbf{q} + \nabla_{\dot{\mathbf{q}}}\ddot{\mathbf{q}}\, \delta\dot{\mathbf{q}}. \quad (3)$$

Quality of these computations is essential for optimization problems, non-linear analysis and linearization. Linearized forward and inverse dynamics models for robots are useful in motion planning and control applications [JR93].

The objectives in trajectory optimization of legged robots to apply sensitivities of the forward dynamics model are (i) more robust and faster convergence in gradient based methods and (ii) more reliable approximation of the Hessian from the exact analytical first derivatives. This technique is more robust than numerical differentiation, it is less likely to prodcue errors and is scalable to large-dimensional systems.

2.4 Dynamics Algorithms

Efficient dynamics algorithms are needed especially for high dimensional systems. The tree structure of legged systems can be exploited for improved performance compared to standard approaches. Hence forward dynamics computations are based on $\mathcal{O}(N)$ Articulated Body Algorithm (ABA) [Fea83], which is an efficient and numerically stable algorithm for moderately constrained tree-structured systems with many degrees of freedom. The contact case and reduced dynamics algorithms are handled by methods described in [Har99] applying the operational space inertia matrix [Kha83], which serves as a basis for the solution of collision dynamics, too. It is used to calculate the amount of impulse when tips of the robot collide with the ground. The method to distribute the impulse over the MBS is described in [AKD94]. The algorithms applied to calculate the linearized forward dynamics model are based on differentation of a recursive symbolic forward model [JR93]. The numerical evaluation of $\delta\ddot{\mathbf{q}}$ is of order $\mathcal{O}(N)$. All algorithms are of recursive nature and well-suited for efficient object-oriented implementation as shown below.

3 Trajectory Optimization

3.1 Formulation of the Optimal Control Problem

High complexity and redundancy of the underlying mechanical structure makes it very hard to find stable gaits for walking robots with four or two legs. Usually full dynamics of the robot cannot be considered by heuristic methods like inverted pendulum method as on-line computing of walking trajectories is computationally too expensive. On the other hand full three-dimensional dynamics models may be handled when stating the problem of finding periodic and statically or dynamically stable gaits as off-line problems e.g. as an optimal control problem. This also decreases the on-line computational costs.

Different gait patterns (such as walk, trot, rack, canter and rotary or transverse gallop for four-legged robots or walk and run for biped robots) differ in the duty factor of each foot, i.e. the fraction of a total stride cycle during which the foot is in contact with ground, and relative phases of feet, i.e. the order and time displacement of feet reaching ground [Ale84].

The optimal control problem is stated as follows [HvS03]:

$$
\begin{aligned}
\min \mathcal{J}[\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}, t_f] \text{ s. t.} &\quad \text{minimize the merit} \\
&\quad \text{function } \mathcal{J} \text{ subject to} \\
\mathcal{M}\ddot{\mathbf{q}} = B\mathbf{u} - \mathcal{C} - \mathcal{G} + J_c^T \mathbf{f}_c &\quad \text{system of MBS ODEs,} \\
\mathbf{g}_c(\mathbf{q}) = 0 &\quad \text{contact algebraic conditions,} \\
\mathbf{b}(\mathbf{q}_0, \mathbf{q}_f, t_0, t_f) = 0 &\quad \text{boundary conditions,} \\
\mathbf{n}(\mathbf{q}, \mathbf{u}) \geq 0 &\quad \text{nonlinear inequality constr.,} \\
\mathbf{q}_{\min} \leq \mathbf{q} \leq \mathbf{q}_{\max}, &\quad \text{box constraints on state,} \\
\mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max} &\quad \text{and control variables.}
\end{aligned}
$$

The original DAE system (1) and (2) can be replaced for the numerical solution by the reduced dynamics equations of 2.2. Merit functions $\mathcal{J}$ may, e.g., be time t_f, energy or a weighted sum of both [HvS03]. Boundary conditions at initial, intermediate and/or final time of a gait cycle contain conditions for

- symmetry resp. anti-symmetry of states,
- foot placement, i.e. conditions that force the feet to be placed on desired positions (which may depend on parameters and therefore may also be subject to the optimization),

- contact forces at the end of a stance phase, that allow the foot to lift off

(see [HvS03] for more details). Nonlinear inequality constraints are:

- Hips of legs in contact with the ground must stay within a maximum radius of the leg, so that the inverse kinematics solution required for reduced dynamics has a well-defined solution.
- The swing feet are kept above the ground according to a given contour, for example a proper sine curve. This property increases stability by avoiding contact with the ground resulting from non-modeled deflexions of bodies and joints, which could lead to stumbling of the robot.
- Slipping is avoided by limiting the horizontal contact forces relative to the vertical contact forces.
- Vertical contact forces must be positive, i.e. the robot may only push to ground but may not pull from ground.
- Further constraints to be considered are detailed motor characteristics. By now the box constraints for minimal and maximal values of angular velocities and torques only give a rough estimate of the real actuator data.

Stability may be enforced explicitly by inserting any common criteria into the optimal control problem. A more detailed discussion of the constraints can be found in [BHK+03] for a humanoid model and in [SHvS03] for a four-legged robot.

3.2 Optimization Method

DIRCOL [vS01], a direct collocation method is used for solving the optimal control problem. States and controls both are discretized by piecewise cubic resp. piecewise linear polynomials on a discrete time grid that may be successively refined automatically. The resulting nonlinear programming problem, whose variables are the coefficients of the piecewise polynomials, may be solved by a sequential quadratic programming method [GMS02]. This problem is sparse because of the special structure of the variables. For more details we refer to [vS01, HvS00].

4 Object-Oriented Modeling Architecture

Practical realization of models for non-linear dynamics computations of legged robots must consider (i) the complexity of the mechanical structure, (ii) a great variety of components and actuation methods, (iii) demanding environmental conditions, (iv) types, efficiency, and interaction of available kinematics and dynamics computer algorithms suitable for such systems, and (iv) application scenarios reaching from off-line trajectory optimizations to real-time closed-loop control including integration and communication to external soft- and hardware with tight timing constraints. Based on a detailed analysis this section proposes an object-oriented architecture satisfying the requirements from the legged robot trajectory optimization application leading to (i) modular, (ii) efficient, (iii) consistent, and (iv) reconfigurable characteristics.

4.1 Requirements Analysis

The requirements for a software system can be captured by a system model which describes what to realize and how. This model forms an abstraction in two ways [HP87, SGW94]. First, it abstracts from real world details which are not relevant for the system. Second, it also abstracts from the implementation details and hence precedes an actual implementation in a programming language.

The ultimate goal of the architecture is to implement multibody kinematics and dynamics algorithms efficiently. The primary task is to investigate which type of computations will be required. For brevity we restrict here to the trajectory optimization problem discussed in 3. The trajectory optimization algorithm requires the set of equations of motion of the robotic system over one *complete* gait cycle, i.e., including time-varying support phases reflected by a changing, phase dependent dynamics model. The first part required is a forward dynamics model which can efficiently be implemented by dedicated composite rigid body and articulated body algorithms [WO82, Fea83]. Depending on the current support phase or contact state a contact dynamics model is applied to consider the additional contact constraints and to reduce to state space form [AKD94]. The reduced dynamics requires forward- and inverse kinematics on position- and velocity-level. The treatment of a complete gait cycle as a multi-phase

dynamics problem requires additional collision computations to model the discontinuous state transitions between various support phases. Depending on the applied optimization method derivatives of the equations of motion w.r.t. state variables and design parameters might be required. When optimization is subjected to additional stability criteria new types of computations must be introduced to determine, e.g., the center of mass or the zero moment point of the legged robot depending on the current state $\mathbf{q}$.

All computations reflect certain physical aspects of the *same* legged machine in certain states leading to a strong coupling between all parts. Consequently there must exist a common representation of the mechanical model and the problem setup. An abstract high-level description could serve as a basis for all types of required computations. There is the need to support many different types of legged machines, such as humanoids, four-legged walking machines, and a large variety of components forming the mechanical structure, different actuators, and contact models.

Computational efficiency is of high importance in trajectory optimization and most control applications of realistic legged robots. The complexity of the governing dynamics equations for full three dimensions, the large number of degrees of freedom, and the possibly large number of dynamics evaluations during iterations of trajectory optimization methods makes this a challenging problem. As a consequence, the most efficient and robust MBS algorithms must be chosen which often are problem specific solutions. The most prominent example is the analytical solution of the inverse kinematics problem, which gives fast and reliable solutions when compared to a general purpose approach. To integrate in one formalism general purpose and problem specific algorithms, which have been generated either automatically or manually coded, a flexible software architecture is indispensible.

4.2 Design of the Architecture

We propose to reconcile these concurrent requirements by a carefully designed object-oriented operational architecture. The goal is to enable easy implementation and integration of dynamics computations but not to create a general purpose multibody program. This comprises a

- high-level and component-oriented description of the robot model and problem setup, called *specification model*,
- objects serving as domain-specific code generators mapping the descriptions creating and
- objects serving as encapsulated algorithms with well-defined interfaces and semantics.

4.2.1 Specification Model

Legged systems are rather complex mechanisms so using a high-level model description being modular and hierarchical is beneficial. Components belonging to the MBS domain, such as links, bodies, joints, drives, are classified in an abstract way as *MBS entities.* In object-oriented terms these are descendants of the class *MBSEntSpec*. The domain component library contains a carefully selected, finite set of components representing concrete mechanical parts, such as drive-trains, or mathematical concepts such as contact constraints.

The topology of the mechanical model is formed by defining relations, called *MBS connections,* between certain ports being part of each MBS entity. These interaction ports belong to the class *MBSPortSpec*, the MBS connection is represented by class *MBSConnectionSpec*.

The set of components and relations between the ports forms an a-cyclic graph where the edges are the connections and the vertices are the MBS entities. This is the specification of the mechanical model, which is represented by class *MBSModelSpec*. There is an important semantic issue to note. Algorithms often are not valid for the complete model, but only for parts, e.g., inverse kinematics or the Jacobian for one single leg. In order to keep all computations consistent the model specification aggregates *references* to MBS entities and MBS connections ensuring that all algorithms consistently operate on the same data and topology. That is crucial in walking robot scenarios where changing structural conditions require reconfiguration of the models or after a calibration cycle.

The desired function a also is considered part of the problem description, i.e., the type of computation, is determined by *solver specification*s. These are objects of class *SolverSpec*. These are parametrized by (i) the expected behaviour, (ii) the type of algorithm applied, and (iii) further tags to denote for instance the coordinate representation used in the computations. The SolverSpec shown in the last line in Fig. 2 represents an articulated forward dynamics algorithm using body-fixed coordinates, e.g., see [RJKD91].

The specification model of the legged robot, however, does not provide any executable code. The purpose is exclusively to decouple dynamics model specification from its implementation and to form an object-oriented basis

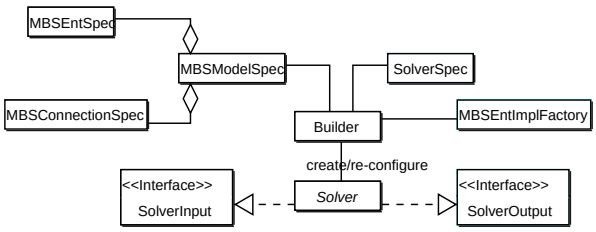

Figure 1: UML class diagram showing the relations between the main classes of the robot modeling framework. The central parts are the Builder and Solver objects. The former converts a given MBS model specification into an executable solver instance able to perform the desired dynamics calculations.

for other abstract representations such as textual or graphical representations. This additional level of abstraction is amenable to formal analysis, e.g., for verification or optimization of the specification w.r.t. to certain criteria, or when making the state of the system persistent.

4.2.2 Operational Model

This section introduces two additional types of objects, the *Builder* and the *Solver*, reaping the benefits from the abstract model specification. The main purpose of a builder is to automatically transform, to *map*, the specification model to one or more products, the solvers. Solvers discussed in this work are executable units of code, instances performing the desired computations. This behaviour of translating a high-level information into low-level implementation makes a builder object a domain-specific *application generator* [Cle88]. The main difference is that a builder is able to immediately create or reconfigure executable solver instances and does not need to invoke a time- and resource-consuming compile-to-code step. The classes representing the specification and the operational model and their relations are shown in Fig. 1.

The transformation process performed by a builder without compile-to-code step is as follows. A builder first checks if the specification and context information meets its requirements to create the output solver. This includes a check for correctness of the model description. In the next step a builder may create a intermediate specifications, e.g., to improve model properties largely to enhance run-time characteristics. Finally it assembles executable objects containing code solving parts or the complete set of the required mathematical equations. These objects of class *MBSEntImpl* represent the multibody domain knowledge of the robotics engineer, coded and precompiled, implementing a well-defined interface. Objects of class *MBSEntImplFactory* supply the builder with MBSEntImpl objects in the demanded form. This can be viewed as the data base for pre-coded component equations and is realized using a factory pattern [GHJV94].

From a specification perspective [FS99] a solver represents an algorithm interface which is implemented by one single MBSEntImpl or a complete dataflow network of inter-connected components. In this case a solver acts like the Multi-Graph-Kernel known in the Model Integrated Computing approach [SK97, HM01, Höp04] conducting computations by controlling the dataflow in a dataflow process network. This approach of distributing the computations has two disctinct advantages: It enables either possible reuse of equations and code at compile-time or reuse of numerical results during run-time through solver coupling.

5 Results and Applications

5.1 Architecture for Gait Trajectory Optimization

Application to trajectory optimization leads to the package structure depicted in Fig. 3. The DIRCOL optimizer interface comprises three main blocks depending on the system under investigation, the differential equation, the linear- and non-linear boundary conditions, and behaviour when switching between various leg support phases. In our realization each is satisfied by a set of coupled solver objects.

The model parts of the first package 'Differential equations' are implemented by one solver containing an $\mathcal{O}(N)$ articulated body algorithm for tree-structured systems [Fea83], a collection of hand-tailored inverse kinematics solvers for each single leg and Jacobian or operational space inertia solvers for the velocity inverse kinematics. The contact state is represented by a solver depending on the time-varying contact model specification. The second package 'Constraints' contains solvers wich compute the center of gravity of the complete mechanical system, and

```
// Problem specification block
RevoluteJoint joint1;
RigidLink link1;
MBSConnectionSpec connection1(joint1.port2,link1.port1);
...
MBSModelSpec leg1,leg2,torso,humanoid;
leg1.add(joint1);
leg1.add(link1);
leg1.add(connection1);
...
humanoid.add(leg1,leg2,torso);
// Builder block
SolverSpec<FwDyn,ABA,BF> DynSpec;
Builder<DynSpec,...> builder;
Solver<DynSpec,...> dynamics=builder.create(humanoid);
...
```

Figure 2: Example C++ pseudo-code exemplifying problem specification and solver creation. The first block sketches the specification of components and topology of a humanoid and its constituents. The second block shows how a builder creates a solver doing forward dynamics (FwDyn), using an articulated body algorithm (ABA) and body-fixed coordinates (BF).

one solver to detect foot collisions with ground and between legs. The task of the third package 'Phase switching' is to react on requests from DIRCOL to change the support phase by invoking a collision solver. The contact state is changed and the system of solvers reconfigured accordingly to keep all components consistent. DIRCOL is able to process user-defined gradient information. In order to enhance convergence properties an optional set of solvers provides exact sensitivities of the differential equations. The actual implementation supports sensitivities for robots with fixed base.

Creation of this dynamics model involves three steps: First, creating objects representing the mechanical components applied for model specification are rigid links, revolute and six-dof joints, contact points, and drives. Second, creating descriptions comprising single legs, needed for inverse kinematics, the complete free-flying legged mechanism, needed for the reduced dynamics algorithm, and the complete mechanism including contact and collision interaction with the environment, for reduced dynamics and phase switching. Third, all solvers mentioned are instantiated and reconfigured by a collection of dedicated builder objects from this unique high-level specification using solver specifications to denote the desired computations.

5.2 Trajectory Optimization Experiments

In this section numerical results of optimized gait trajectories for walking robots are shortly discussed. Full three-dimensional models have been involved in the optimization of walking trajectories for a four-legged robot [SHvS03] and a two legged humanoid robot [BHK$^+$03] (cf. Fig. 4). In [BHK$^+$03] and [SHvS03] originally SOAFOR, a set of Fortran subroutines implementing articulated body algorithm, spatial operator algebra and reduced dynamics for legged robots has been used, which has evolved from [Har99] but does not provide sensitivities. Here, we demonstrate that these solutions also can be obtained with almost the same efficiency using the much more flexible object-oriented modeling approch presented here.

The data needed to build up the dynamics model of Sony's four-legged robot ERS-210A have been provided by the manufacturer. Each of the robot's legs consists of three actuated degrees of freedom. The model has successively been refined using observations made from real experiments on the robot with the calculated trajectories. Thus box constraints of maximum applied torques and maximum angular velocities of the joints have been added to improve agreement between calculated and measured trajectories. A basic friction model has been added to the robot's feet to avoid slipping which lead to a falling of the robot in experiments. Energy optimal walking trajectories have been calculated. Symmetry of the gaits considered like trot and walk has been exploited to reduce the optimization to only one halfstride. While for the trot contact situation does not change during one halfstride and consequently only one phase must be considered, contact situation changes for the walk, so that two phases are introduced.

The humanoid robot considered here is a prototype that was developed in a collaboration of our group and a group from TU Berlin (now at TU Munich) [BHK$^+$03]. It consists of one torso and two legs of 6 dof each. A fully three-dimensional model has been used to optimize a half-stride consisting of two phases (single limb support phase and double limb support phase). Nonlinear inequality constraints among other ensure static stability explicitly. One dynamics evaluation for this system according to 5.1 requires about 1.6 ms on a 2 GHz Pentium including exact sensitivities during smooth phases of the trajectory. As current humanoid control cycle times

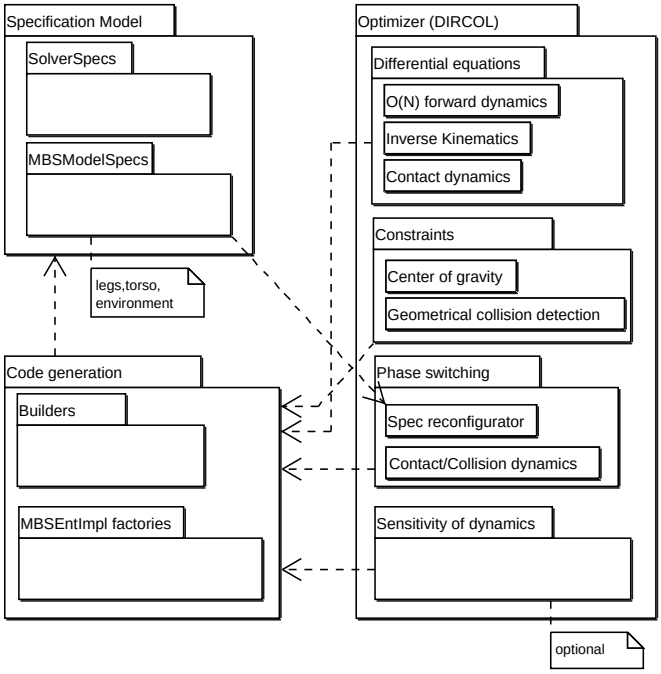

Figure 3: UML package diagram of gait trajectory optimization within DIRCOL using Spec-, Builder-, and Solver objects.

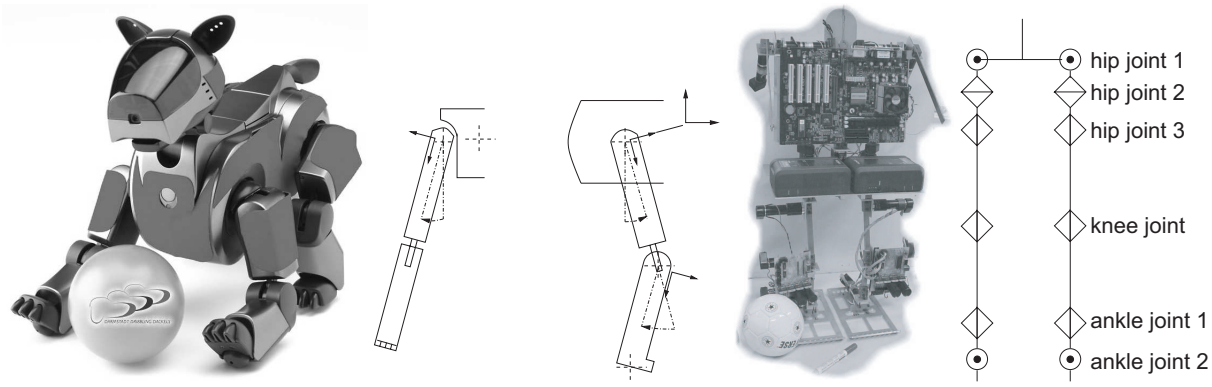

Figure 4: Left: Four legged Sony Aibo robot and kinemtic structure of one leg. Right: prototype of a humanoid robot and kinematic structure of humanoid model. For both robots optimal walking trajectories have been calculated.

are around 1 ms, the approach presented in this paper now enables the investigation of new nonlinear dynamics model-based control methods for humanoid robots by modular and flexible generation and re-using of code.

6 Conclusion

We have discussed techniques to gain trajectories for walking robots, multibody systems dynamics modeling and optimal control methods, and have presented a new object-oriented architecture to provide multibody dynamics model computations. The approach consists of a carefully designed class hierarchy which separates the specification of the mechanical model and the desired calculations, referred to as specification model, from the implementation part. The latter consists of algorithm-specific code-generators (Builder) creating executable instances (Solver) finally performing the desired multibody computations. The two main advantages for the usage of Builder/Solver components are on the one hand the application of dedicated efficient problem-specific algorithms and closed-form solutions to reap maximum efficiency and not to rely on one general purpose formalism and to allow for flexible algorithm interchange. This results in an efficient and light-weight code without a compile-to-code step required. On the other hand the specification model permits more general topologies, components, and algorithms, indispensible for our further research which will be directed to legged robot models of increased complexity, including arm dynamics and advanced actuator concepts using artificial muscles.

References

[AKD94] D. Agahi and K. Kreutz-Delgado. A star topology dynamic model for efficient simulation of multilimbed robotic systems. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 352–357, 1994.

[Ale84] R. McN. Alexander. The gaits of bipedal and quadrupedal animals. *The International Journal of Robotics Research*, 3(2):49–59, 1984.

[BHK+03] M. Buss, M. Hardt, J. Kiener, M. Sobotka, M. Stelzer, Oskar von Stryk, and D. Wollherr. Towards an autonomous, humanoid, and dynamically walking robot: Modeling, optimal trajectory planning, hardware architecture, and experiments. In *Proc. 3rd IEEE/RAS Intern. Conference on Humanoid Robots, Karlsruhe and München, Germany, Oct. 1-3*, 2003.

[Cle88] J. C. Cleaveland. Building application generators. *IEEE Software*, 4(5):25–33, July 1988.

[Fea83] R. Featherstone. The calculation of robot dynamics using articulated-body inertias. *The International Journal of Robotics Research*, 2(1):13–30, 1983.

[FS99] M. Fowler and K. Scott. *UML Distilled: A brief guide to the standard object modeling language*. Addison Wesley Publ. Co., 2 edition, 1999.

[GHJV94] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison Wesley Publ. Co., 1994.

[GMS02] P.E. Gill, W. Murray, and M.A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12:979–1006, 2002.

[Har99] M. Hardt. *Multibody Dynamical Algorithms, Numerical Optimal Control, with Detailed Studies in the Control of Jet Engine Compressors and Biped Walking*. PhD thesis, Electrical Engineering, University of California San Diego, U.S.A., 1999.

[HM01] R. Höpler and P. J. Mosterman. Model Integrated Computing in robot control to synthesize real-time embedded code. In *Proc. IEEE Conference on Control Applications*, pages 767–772, Mexico City, 2001.

[Höp04] R. Höpler. *A unifying object-oriented methodology to consolidate multibody dynamics computations in robot control*. PhD thesis, Technische Universität Darmstadt, 2004.

[HP87] D. Hatley and I. Pirbhai. *Strategies for Real-Time System Specification*. Dorset House Publishing Company, New York, 1987.

[HSvS04] R. Höpler, M. Stelzer, and O. von Stryk. Object-oriented dynamics modeling for legged robot trajectory optimization and control. In *Proc. IEEE Conf. on Mechatronics and Robotics (MechRob)*, pages 972–977, Aachen, Sept. 13-15 2004. Sascha Eysoldt Verlag.

[HSvS05] R. Höpler, M. Stelzer, and O. von Stryk. Object-oriented dynamics modeling for simulation, optimization and control of walking robots. In *Proc. 18th Symposium on Simulation Technique, ASIM, Erlangen, September 12-15*, pages 588–593, 2005.

[HvS00] M. Hardt and O. von Stryk. Towards optimal hybrid control solutions for gait optimization patterns of a quadruped. In *Proc. 3rd International Conference on Climbing and Walking Robots, CLAWAR 2000*, pages 385–392, Madrid, 2000. Bury St. Edmunds and London, UK: Professional Engineering Publishing.

[HvS03] M. Hardt and O. von Stryk. Dynamic modeling in the simulation, optimization, and control of bipedal and quadrupedal robots. *Z. Angew. Math. Mech.*, 83(10):648–662, 2003.

[JR93] Abhinandan Jain and Guillermo Rodriguez. Linearization of manipulator dynamics using spatial operators. *IEEE Trans. Syst., Man, Cybern.*, 23(1):239–248, 1993.

[Kha83] O. Khatib. Dynamic control of manipulators in operational space. In *Proc. 6th CISM-IFToMM Congress on Theory of Machines and Mechanisms*, New Delhi, India, December 1983.

[RJKD91] G. Rodriguez, A. Jain, and K. Kreutz-Delgado. A spatial operator algebra for manipulator modeling and control. *The International Journal of Robotics Research*, 10(4):371–381, 1991.

[SGW94] B. Selic, G. Gullekson, and P. T. Ward. *Real-time object-oriented modeling*. John Wiley & Sons, 1994.

[SHvS03] M. Stelzer, M. Hardt, and O. von Stryk. Efficient dynamic modeling, numerical optimal control and experimental results for various gaits of a quadruped robot. In *CLAWAR 2003: International Conference on Climbing and Walking Robots, Catania, Italy, Sept. 17-19*, pages 601–608, 2003.

[SK97] J. Sztipanovits and G. Karsai. Model-integrated computing. *IEEE Computer*, 4:110–112, 1997.

[vS01] O. von Stryk. User's guide for DIRCOL version 2.1: A direct collocation method for the numerical solution of optimal control problems. Technical report, Fachgebiet Simulation und Systemoptimierung, Technische Universität Darmstadt, 2001. http://www.sim.informatik.tu-darmstadt.de/sw/dircol.

[WO82] M. W. Walker and D. E. Orin. Efficient dynamic computer simulation of robotic mechanisms. *ASME J. of Dynamic Systems Meas. and Control*, 104:205–211, 1982.