Proceeding of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space:
i-SAIRAS 2003, NARA, Japan, May 19-23, 2003

Software Environment for Assembly, Modeling and Online-Simulation of Redundant Light-Weight Manipulators

Rainer Krenn, Bernd Schäfer, Gerd Hirzinger
German Aerospace Center (DLR)
Oberpfaffenhofen, D-82234 Wessling, Germany
rainer.krenn@dlr.de, bernd.schaefer@dlr.de, gerd.hirzinger@dlr.de

Keywords Virtual Assembly, Modeling Language, Dynamics Simulation, Space Robotics, Redundant Kinematics, Inverse Kinematics

Abstract

Currently, DLR is developing its 3rd generation of light-weight robots. It is a modular system with new optimized drives, carbon fiber links and novel wrist units. The robots are dedicated for applications on ground as well as for future space missions.

The information of the new robotic components is stored in a database that can be accessed via a user interface inside a simulation environment. The software tool enables a system engineer to rapidly design a fully operating concept of a robotic system for testing and verification.

One of the design tasks is the definition of the robot kinematics, respectively the number of joints and the sequence of joint axis orientations. For space applications redundant kinematics with at least 7 joints are preferred due to their increased skill performance and flexibility. Thus, a generic method for solving the particular inverse kinematics problem must be part of the assembly environment. Currently a Lagrange constraint optimization and a method using a differential equation system are provided.

In this paper the novel software environment for rapid assembly, modeling and online simulation will be described in detail.

1 Introduction

After a developing phase of about two years, the 3rd generation of DLR's light-weight robots (internally called LBR 3, see Fig. 1) is now ready for operation [1]. A major design goal of the robotic system was its modularity using well defined interfaces of each robot part. This method keeps the system open for future kinematical adaptations and changes, that will not affect the overall robotic system. The general idea behind is to provide a kind of construction kit for robotic systems.

Fig. 1: DLR's 3rd Generation of Light-Weight Robots

The robotics systems that are based on LBR 3 technology are dedicated for applications on ground (e.g. servicing and medical technology) as well as for future space missions. Currently the components of LBR 3 are used in different space robotics scenarios like

- in the manipulator inspection system MISSISS (Fig. 2) as a contribution to the International Space Station [2],

- in satellite servicing missions like the ROSAT capturing and de-orbiting (Fig. 3), discussed in [3],
- in our national robotic component verification experiment on ISS (Fig. 4), called ROKVISS [4].

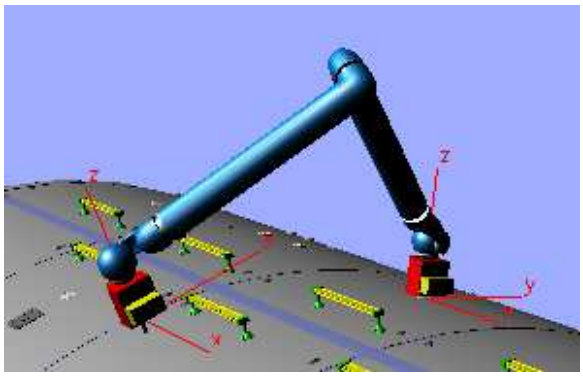

Fig. 2: LBR 3 based MISSISS Manipulator

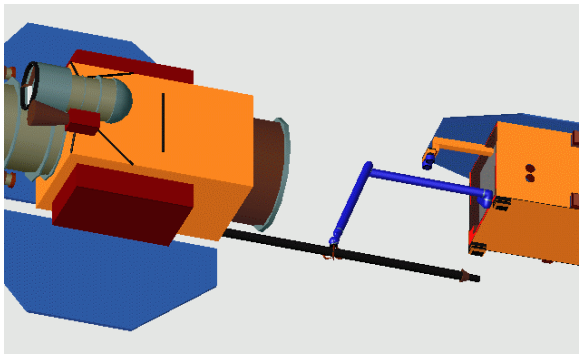

Fig. 3: LBR 3 based ROSAT Capturing Manipulator

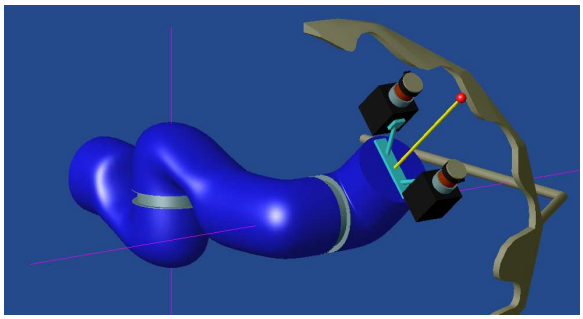

Fig. 4: LBR 3 based ROKVISS 2-Joint Kinematics

For rapid design purposes and for performance assessments of the overall robotic motion behavior, a software environment for virtual assembly, modeling and simulation of robotic systems based on LBR 3 components has been created in parallel to the mechanical setup. Inside this environment a new manipulator configuration can be quickly assembled and tested for intensive support of further hardware developments.

The paper focuses on the architecture and software implementation of the virtual LBR 3 construction kit and the assembly, modeling and simulation capabilities of the according user interface. Special solutions inside the software package like the method to provide the inverse kinematics for general kinematic systems will be pointed out in detail.

2 Components of the LBR 3 Robotic System

The activities of the Institute of Robotics and Mechatronics in the field of light-weight robotics cover the construction of mechanical components like drives, joints and dexterous hands as well as the development of telemanipulation technologies, especially for space application.

Fig. 5: Mechanical Components of LBR 3

The novel LBR 3 drives are compact units of brushless DC multi-pole motors and light-weight Harmonic Drives equipped with an electromagnetic safety brake (Fig. 5, center). By the consequential adaptation of the motor power to the performance (gear ratio and maximum torque) of the attached Harmonic Drive it was possible to reduce the joint weight tremendously. and to minimize the power losses for typical robotic motions.

The arm structures have been realized by use of carbon fiber technology (Fig. 5, left and right) wherein the integrated aluminum retainer at the axial endings of the links build the assembly flange for the joint drives. The weight of such kind of link is negligible compared

with other components of the manipulator. The links are provided in different designs like L-shape, C-shape, S-shape etc. in order to assemble robots with an almost infinite number of kinematical configurations. Moreover, two design families provide the option to create robots with so-called symmetrical (roll joint axis inside the pitch joint plane) and asymmetrical kinematics (Fig. 6).

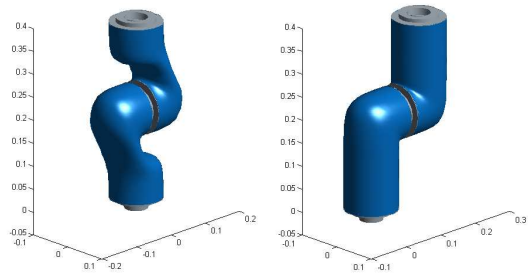

Fig. 6: Symmetric and Asymmetric Configurations

The wrist mechanism of the LBR 3 is a highly integrated roll-pitch-roll (RPR) joint unit. The characteristic component is the ball like housing where the final two joints are assembled. The wrist can be used with a plane end-effector or equipped with an additional bracket in order to provide a kind of gimbaled end-effector kinematics (Fig. 7).

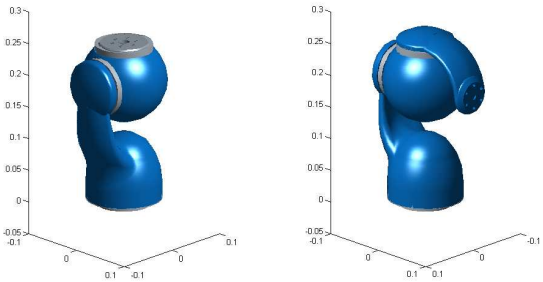

Fig. 7: RPR and Gimbaled Wrist Kinematics

3 Assembly, Modeling and Simulation Environment

In order to enable a system engineer to rapidly assemble a manipulator prototype and to build a fully operating kinematics and dynamics simulation of that prototype, the components of LBR 3 are stored in a database as a virtual construction kit. Our specific assembly, modeling and simulation software provides an interface to access the LBR 3 database and an environment where the robot model can be assembled and tested. The software environment is divided into three layers (Fig. 8): The Construction & Development Layer, the System Engineering Layer and the Simulation Layer.

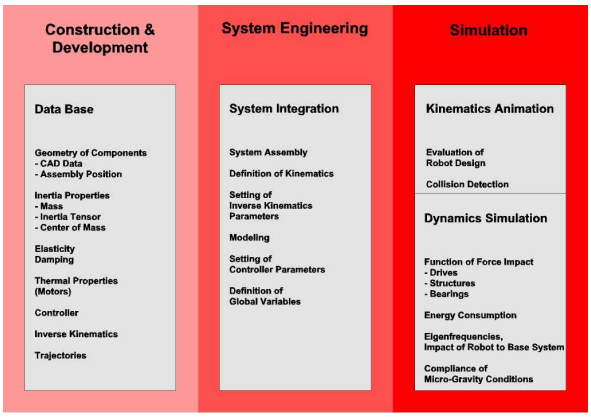

Fig. 8: Layers of the LBR 3 Assembly, Modeling and Simulation Environment

Fig. 9 provides an overview how the information and tasks of the particular layers are mapped into the software environment.

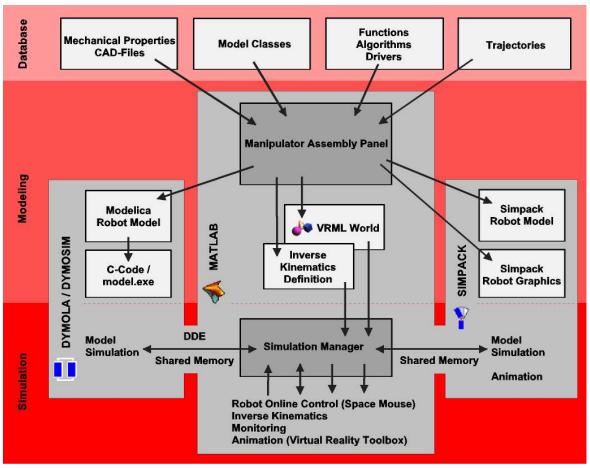

Fig. 9: Implementation of Assembly, Modeling and Simulation Tasks in Software

3.1 Construction & Development Layer

The first layer, the Construction & Development Layer, is implemented in software by different kind of tables, routines and libraries:

- In the first group information of single robot components like links, motors and gears is stored. The

collection contains graphical information of the component surface shape taken from CAD tools and tables of physical properties of the components like mass, inertia tensor, center of mass, physical limits, gear ratio, stiffness, damping, thermal properties etc.

- The second group is a collection of model classes of the robot component types (links, motors, gearboxes, joint controller, etc.). The classes describe the dynamics behavior of the component types. In the modeling phase, instances of the classes will be taken for each single component of the assembled robot and will be parameterized accordingly. Since the simulation software packages used inside our environment are DYMOLA and SIMPACK, the model classes are stored in the respective languages MODELICA and SIMPACK code.
- A third group of libraries are functions that will be linked to the robot model at simulation time. In our environment these functions are drivers for user input devices like a SpaceMouse 6D controller or an haptic interface, and inverse kinematics algorithms for global robot control. Currently two different solutions of the inverse kinematics problem are provided inside the presented environment. The first one is a Lagrange constraint optimization, which provides the possibility to minimize a function, mostly a quadratic expressions in joint position, speed, acceleration or torque subject to the end-effector misalignment. A second and more uncommon way of inverse kinematics implementation is to make use of the differential equations system of the robot motion whereby the robot is modeled as a passive chain of joints and limbs. A detailed description of inverse kinematics implementation is given in chapter 4.
- In the last group static tables are stored that contain predefined trajectories and test routines for system testing and verification.

3.2 System Engineering Layer

The implementation of the System Engineering Layer is a MATLAB based graphical user interface, we call Manipulator Assembly Panel. It provides access to the database objects and the capability to assemble the robot model.

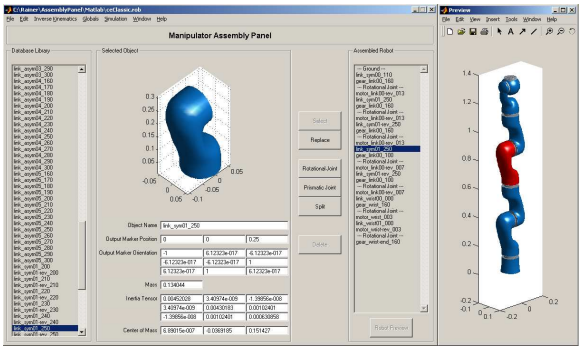

Fig. 10: Graphical User Interface for Robot Assembly

Database Access

The first phase of system integration is the virtual robot assembly. Herein, single robot parts or pre-assembled units taken from the LBR 3 database can be connected to a complete robotic system just by mouse clicks. The user will be supported by graphical and textual information about the currently chosen component and by a preview of the assembled system.

Model Export

In the model export phase the robot will be provided by different descriptions that are required for subsequent simulations. Thereby, an important item is the consistency of all exported code. This will be guaranteed by the central modeling environment, the Manipulator Assembly Panel, that generates the code.

a) Robot description table as input for the inverse kinematics algorithm:

We assume that for control purposes the robot can be represented by a chain of rigid bodies that are connected by joints with one degree of freedom each. Therefore, the required data to parameterize the inverse kinematics function (e.g. Lagrangian optimization algorithm) are

- the number of joints,
- the joint type, linear or rotational,
- the distance vectors between the assembly points of a link, called input marker and output marker,
- the rotation matrix between input marker and output marker,

- the weighting factors of the optimization criteria for the desired robot joint motion.

b) Robot description file adapted to the dynamics simulation tool:

For simulation purposes the robot model including its joint controller will be exported in MODELICA code or SIMPACK code. In both options the robot model code consists of objects (parameterized instances) of classes taken from the database and information about connections between the objects. Additionally, the models will be equipped with interfaces for simulation data exchange like global end-effector command input and system state variables output.

For a simulation run, the SIMPACK model can be taken as it is. It will be processed inside the SIMPACK simulation environment. On the other hand, the MODELICA code must be translated into C-code using DYMOLA and compiled and linked with specific libraries afterwards. The compilation result is a stand-alone simulation application of the robot, called DYMOSIM, that runs independently from specific simulation software packages.

c) Robot description readable from graphics animation software:

For graphics animation of the robot motion, again two different options are supported by the simulation environment: The first option is the SIMPACK 3D graphics animation. Choosing this option, the Assembly Panel exports a robot shape description in SIMPACK code as well as SLP render files of the robot components to the SIMPACK CAD-geometry database. Using these files, the SIMPACK simulation environment can provide a robot animation in parallel to the running robot simulation.

The second option is the export of a VRML robot shape description. Due to its standardization VRML offers a more independent way for graphics animation. This option is preferred for simulations using DYMOLA/DYMOSIM environment, because its graphics capabilities are only rudimentary.

3.3 Simulation Layer

The Simulation Layer covers the robot motion command generation, the dynamics simulation and the kinematics animation of the robot model. A MATLAB based Simulation Manager provides an interface to the simulation setup and the data monitoring. Inside this layer the robotic system will be verified and validated. The simulations can run in non-real-time as well as in real-time. This option is essential, if online control is provided by the simulation environment.

a) Command Generation:

The command generation is part of the Simulation Manager. Hereby, command generation means online-control using the Space Mouse 6D input device and the calculation of joint motion commands by the inverse kinematics algorithm. The data exchange with the simulation modules is implemented by the Dynamic Data Exchange (DDE) mechanism with DYMOSIM only or via shared memory with DYMOSIM and SIMPACK. Basically, the data exchange via DDE is preferred due to its increased flexibility.

b) Dynamics Simulation:

The time integration of the robot model will be performed inside DYMOSIM or SIMPACK environment under control of the central Simulation Manager. Analogously to the command input, the state variables output takes place via DDE or shared memory. For simulation properties settings (integration method, time interval etc.) the Simulation Manager exploits the capabilities of the used simulation tools.

c) Kinematics Animation:

The 3D animation of the robot motion runs in two different setups. The first one uses the animation capabilities of the SIMPACK environment. However, this is only possible, if the simulation itself is running in the SIMPACK environment, too. The second setup uses VRML capabilities together with a JAVA based server that controls the animation in a VRML compatible viewer (e.g. HTML browser). The implementation of this setup is done using MATLAB's Virtual Reality Toolbox functions.

d) Data Monitoring:

The monitoring of state variables and parameters of the running simulation is implemented using MATLAB's graphics capabilities. The variables to be shown can be selected in a Variable Browser and mapped to a Scope Window.

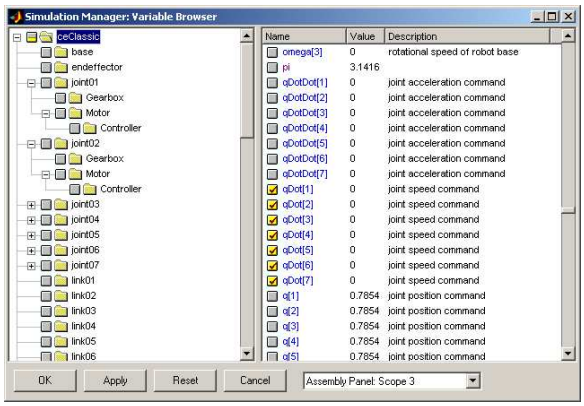

Fig. 11: Robot Model Variable Browser

A special method to visualize the saturation of physical limits (e.g. joint torque limits, link stress limits) during simulation is the integration of displays into the animation window. In our environment two methods are provided: One is the integration of display arrows at the points of interest (Fig. 12 left). The second method maps a color spectrum to the robot components for displaying data (Fig. 12 right).

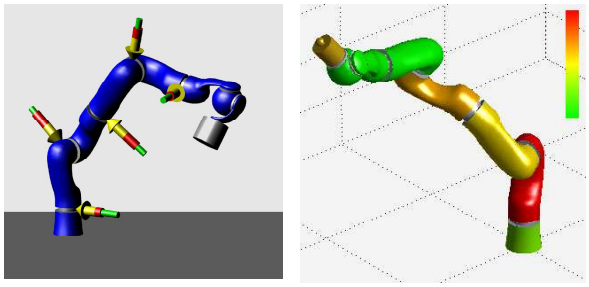

Fig. 12: Methods of Inline Data Displaying

4 Inverse Kinematics for Redundant Manipulators

One of the major robot design tasks is the definition of the robot kinematics, respectively the number of joints and the sequence of joint axis orientations. For space applications mostly redundant kinematics with at least seven joints are preferred due to their increased skill performance and flexibility. However, the problem of solving the inverse kinematics increases accordingly to the number of joints. Additionally, the inverse kinematics has to support online interactive control by user input without any command pre-processing. Currently two different solutions are provided inside the realized environment.

4.1 Lagrangian Optimization

The first solution of the inverse kinematics problem is a Lagrangian constraint optimization (1), which provides the possibility to minimize a function **f** subject to the end-effector misalignment $\boldsymbol{\Delta}$ (2), respectively **h** (3). For a detailed description, we refer to [5].

$$L = f(\mathbf{q}) + \boldsymbol{\lambda}^T \cdot \mathbf{h}(\mathbf{q}) \rightarrow \min \quad (1)$$

$$\boldsymbol{\Delta} = \mathbf{B}_{desired}^{-1} \cdot \mathbf{B}_{robot} \quad (2)$$

$$\mathbf{h} = \begin{pmatrix} \Delta_{14} \\ \Delta_{24} \\ \Delta_{34} \\ \Delta_{12} - \Delta_{21} \\ \Delta_{23} - \Delta_{32} \\ \Delta_{31} - \Delta_{13} \end{pmatrix} = \mathbf{0} \quad (3)$$

The function to be minimized mostly consists of quadratic expressions in joint position (4), speed (5), acceleration (6) or drive power (7). In space robotics often the force/torque impact (8) from the robot base to the support system (satellite) has to be minimized in order to guarantee micro-gravity conditions or attitude control (see Fig. 2 and Fig. 3).

$$f_{\mathbf{q}} = (\mathbf{q} - \mathbf{q}_{ref})^T \mathbf{P}_{\mathbf{q}} (\mathbf{q} - \mathbf{q}_{ref}) \quad (4)$$

$$f_{\dot{\mathbf{q}}} = \dot{\mathbf{q}}^T \mathbf{P}_{\dot{\mathbf{q}}} \dot{\mathbf{q}} \quad (5)$$

$$f_{\ddot{\mathbf{q}}} = \ddot{\mathbf{q}}^T \mathbf{P}_{\ddot{\mathbf{q}}} \ddot{\mathbf{q}} \quad (6)$$

$$f_{\mathbf{p}} = \mathbf{p}^T \mathbf{P}_{\mathbf{p}} \mathbf{p} \quad (7)$$

$$f_{\mathbf{F}_0} = \mathbf{F}_0{}^T \mathbf{P}_{\mathbf{F}} \mathbf{F}_0 \quad (8)$$

4.2 Solving a Differential Equation System

A second and novel way of inverse kinematics implementation is to make use of the differential

equations system of the robot motion whereby the robot is modeled as a passive chain of joints and limbs (Fig. 13).

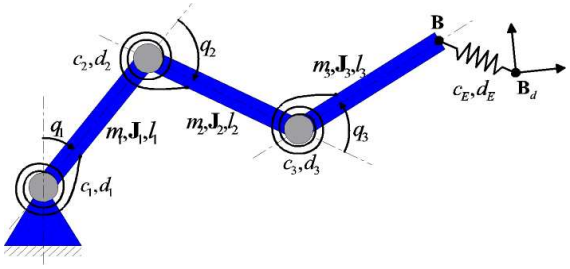

Fig. 13: Robot Chain for Inverse Kinematics Implementation

In this case the motion is initiated by force and torque at the end-effector and the joint motion can be calculated by solving the differential equations. This method is very appealing because it can be easily implemented in available multi body dynamics simulation software.

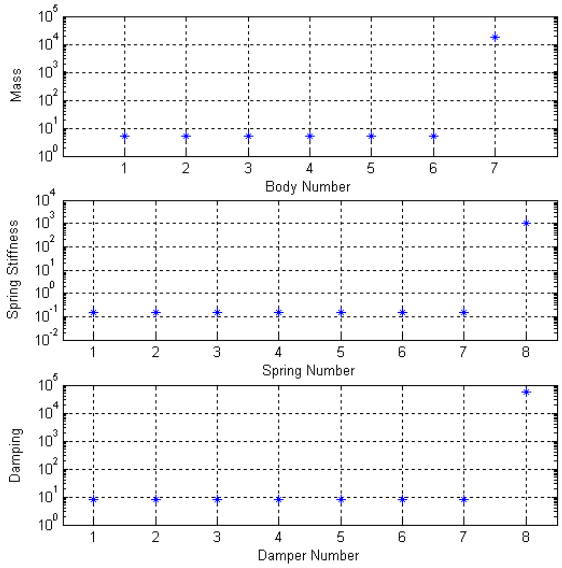

Fig. 14: Property Settings of the Robotic Chain

Since the robot is a kinematically redundant one, variations of the dynamics parameters of the kinematics chain (e.g. joint friction, limb mass) affect the overall robot motion. By suitably adjusting these parameters the motion can be optimized according to a particular goal analogously to the optimization inside the Lagrange algorithm.

In order to use this method online the integration time consumption has to be minimized. This can be achieved by an optimized distribution of limb mass/inertia and joint stiffness/damping inside the robot chain. Assuming the robot chain as a linear system

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{C}\mathbf{q} = \mathbf{F} \tag{9}$$

with springs and dampers in its joints and between the actual end-effector position and the target position, the parameters have to be set in a way that the mass/inertia of the end-effector body, respectively the stiffness/damping of the spring/damper between the end-effector and the target are some orders of magnitude higher than all the rest which have almost identical values each. In Fig. 14 the properties of a basic sample system with seven bodies and one linear direction of motion are presented.

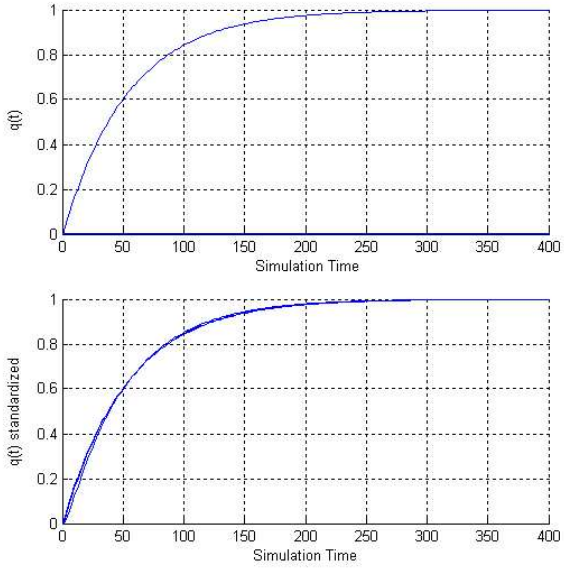

Fig. 15: Eigenmodes of the System

In this configuration the system achieves only one dominant eigenmode (Fig. 15 top) and the time behavior of all eigenmodes almost agree (Fig. 15 bottom). Thus, the eigenmotion reach an acceptable tolerance band around their steady state in a predefined simulation time. From the simulation point of view this system can be integrated very fast using an implicit integration method with wide tolerances and long step sizes in order to obtain the desired result in a very

short computing time compared to that of the robot system simulation. Note, there is no relation between the simulation time of the robotic system itself and the simulation time used to solve the inverse kinematics problem. Here, only the steady state is relevant and will be taken to command the robot in the next sampling step. So, the simulation time that depends on the dynamics of the chain can be arbitrarily long (see Fig. 15).

Important advantages of this method are the robustness close to kinematics singularities and the possibility to command goal positions out of the robot workspace whereby the robot motion results in the most close configuration to that command. Therefore, the method can be applied for systems with less than 6 degrees of freedom as well.

5 Conclusion

The motivation for building our assembly and simulation environment was the rapid design idea supporting the development of robotic-based mission scenarios already in the early design phase.

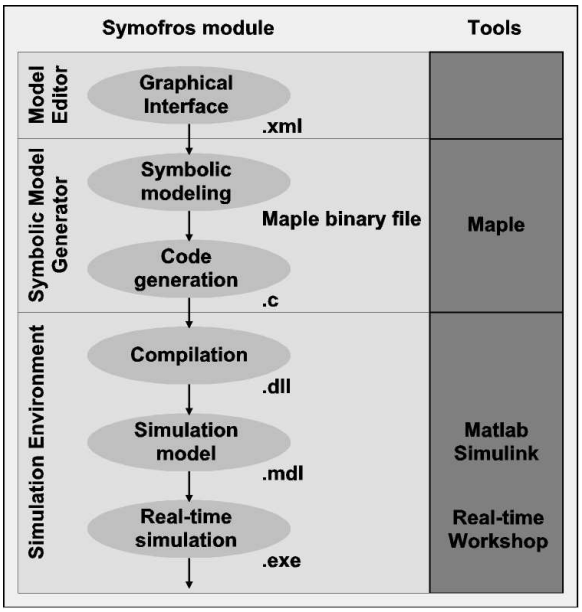

Fig. 16: Symofros Module Overview [6]

The overall performance of the environment has been tested by applying it to space robot mission examples like in ISS-based and satellite capturing scenarios. Of course, the implementation of the environment is not finally finished yet. It will be constantly updated with the latest developments of our institute concerning light-weight robot components.

Beside the completion of the environment database, now, the capability of an additional model description will be implemented. The language is XML (Extensible Markup Language) and should make our system compatible with SYMOFROS (Fig. 16), the modeling and simulation environment of the Canadian Space Agency [6].

Reference

[1] G. Hirzinger, A. Albu-Schäffer, M. Hähnle, R. Krenn, A. Pascucci, M. Schedl, N. Sporer, “DLR’s torque controlled light weight robot III – are we reaching the technological limits now?”, International Conference on Robotics and Automation 2002, Washington D.C., 11-15 May 2002

[2] B. Schäfer, R. Krenn, G. Hirzinger, A.R. da Silva, “Light-Weight Space Robotics: Rapid Design Approach and Efficient Simulation Environment”, Machine Intelligence & Robotic Control, Vol. 3, No. 3, Page 99 – 111 (2001)

[3] OHB System, DLR; “Operationeller Servicing Satellit: OSS Machbarkeitsstudie”, Final Presentation, Bonn, Germany, 27 Sept. 2000

[4] http://www.weblab.dlr.de/onl

[5] B. Schäfer, R. Krenn, B. Rebele, “On Inverse Kinematics and Kinetics of Redundant Space Manipulator Simulation”, Euro Conference on Numerical Methods and Computational Mechanics, University of Miskolc, Miskolc, Hungary, 15-19 July 2002

[6] J.-C. Piedbœuf, J. Kövecses, Ch. Lange, B. Moore, Y. Gonthier, “Advanced Modeling techniques for Robotic Manipulators”, Tutorial Session, International Conference on Robotics and Automation, Washington D.C., 11-15 May 2002