# 3

# Robot Simulation for Control Design

Leon Žlajpah
*Jožef Stefan Institute*
*Slovenia*

## Abstract

Research in the field of robotics is tightly connected to simulation tools for many reasons. On one side, simulation supports the development of new advanced control algorithms and on the other side, it is always not feasible to build a whole robot system to test some algorithms or it is not safe to perform tests on a real system (at least in the first design stages). The simulation has also a very important role for off-line programming, to design mechanical structure of robots, to design robotic cells and production lines, etc.
In the paper, an overview of the simulation in robotics is given and some topics like: how simulation makes things easier, advantages and backdraws of the simulation in robotics, virtual and real world, are pointed out. The scope of the paper is the role of the simulation in different fields of robotics, especially the dynamic simulation of robot manipulators. We present an integrated environment for the design and testing of advanced robot control schemes. The main capabilities of such environment are: the simulation of the kinematics and dynamics of manipulators, the integration of different sensor systems like vision and force sensors, scenarios for complex robot tasks, the visualization of robots and their environment and the integration of real robots in the simulation loop. We give an overview of simulation and visualization tools suitable for the simulation of robot systems using general dynamic engines and graphic languages. Finally, we present some typical simulation examples in different fields of robotics from offline programming, mobile robots to space robotics.

## 1. Introduction

Simulation has been recognized as an important research tool since the beginning of the 20th century. In the beginning, simulation was first of all an academic research tool. The "good times" for simulation started with the development of computers. First, the analog computers and later the digital computers have boosted simulation to new levels. So, the simulation is now a powerful tool supporting the design, planning, analysis, and decisions in different areas of research and development. Simulation has become a strategic tool in many fields, used by many researchers, developers and by many manufacturers. Of course, robotics as a modern technological branch is no exception. Actually, in robotics simulation plays a very important role, perhaps more important than in many other fields and we like to present in the following some insight in the robotics from the simulation point of view.

### 1.1 The role of simulation

Being able to simulate opens a wide range of options for solving many problems creatively. You can investigate, design, visualize, and test an object or even if it does not exists. You can

see the results of a system yet to be built. It is possible that your solutions may fail or even blow up, but only in simulation. So, using the simulation tools one can avoid injuries and damages, unnecessary changes in design after the production of parts has already started, to long cycle times in manufacturing process, and even unnecessary paper work. Simulation enables us to work even in four dimensions. For example, one can observe within a few minutes how a planned production will be realized in next month, or a fast process can be slowed down to observe all details in "slow motion". All these make things easier and cheaper. One of the problems in classical design and planning are "what-if" questions. Due to the system complexity many of them are often unasked or not answered. With up-to-date simulation tools one can deal with exact geometry, consider the dynamic characteristics of a system, include the man-machine interfaces, and visualize the object in 3D in detail. Having all these in mind there is no reason for avoiding any "what-if" question. The boundaries for what is possible or not are pushed far away especially in advanced virtual reality tools. Using simulator researchers may build experimental environments according to their own imagination. Complexity, reality, specificity can be gradually increased to a level where virtual systems can head to real challenges of the physical world and even beyond.
Simulation is a highly interdisciplinary field since it is widely used in all fields of research from engineering and computer science to economics and social science, and at different levels from academic research to manufactures. Of course, simulation has been also recognized as an important tool in robotics: in designing new products, investigating its performances and in designing applications of these products. Simulation allows us to study the structure, characteristics and the function of a robot system at different levels of details each posing different requirements for the simulation tools. As the complexity of the system under investigation increases the role of the simulation becomes more and more important.

## 2. Simulation of robot manipulators

The ways and methods in robotics research and development have always been influenced by the tools used. This is especially true when one considers the profound impact of recent technologies on robotics, especially the development of computers which have become indispensable when designing the complex systems like robots. Not many years ago, computing cost was still a significant factor to consider when deriving algorithms and new modeling techniques (Fenton & Xi, 1994; Latombe, 1995; Zhang & Paul, 1988). Nowadays, distributed computing, network technology and the computing power developed by commercial equipment open new possibilities for doing systems design and implementation. However, in spite of all that, the creativity of a human designer can not be left out in the design process. The best solution seems to be to provide the designer with proper tools which significantly increase his efficiency. Among them, the simulation has been recognized as an important tool in designing the new products, investigating their performances and also in designing applications of these products. For complex systems as robots, the simulation tools can certainly enhance the design, development, and even the operation of the robotic systems. Augmenting the simulation with visualization tools and interfaces, one can simulate the operation of the robotic systems in a very realistic way.
A large amount of simulation software is available for robot systems, and it is already being used extensively. The majority of the robot simulation tools focus on the motion of the robotic manipulator in different environments. As the motion simulation has a central role in all simulation systems they all include the kinematic or dynamic models of robot manipulators. Which type of models will be used depends on the objective of the simulation system. For

example, trajectory planning algorithms rely on kinematic models. Similarly, the construction of a robotized cell can be simulated efficiently by using only kinematic models of robot manipulators, without considering the dynamics or drives. On the other hand, dynamic models are needed to design the actuators. For example, modern control systems of robotic manipulators use internally different robot kinematic and dynamic models to improve the performance.
To model and simulate a robot manipulator different approaches are possible. They can differ in the way the user builds the model. Block diagram oriented simulation software requires that the user describes the system by combining the blocks, and there are other packages requiring the manual coding. To overcome the problems which arise when the system is very complex (and the robots usually are) several approaches exist to automatically generate the kinematic and/or dynamic models of robots.
The simulation tools for robotic systems can be divided into two major groups: the tools based on general simulation systems and special tools for robot systems. The tools based on general simulation systems are usually special modules, libraries or user interfaces which simplify the building of robot systems and environments within these general simulation systems. One of the advantages of such integrated toolboxes is that they enable you to use other tools available in the simulation system to perform different tasks. For example, to design control system, to analyse simulation results, to visualize results, etc. There exist several general simulation tools which are used for simulation of robot systems like MATLAB/Simulink, Dymola/Modelica, 20-sim, Mathematica, etc. Special simulation tools for robots cover one or more tasks in robotics like off-line programming, design of robot work cells, kinematic and dynamic analysis, mechanical design. They can be specialized for special types of robots like mobile robots, underwater robots, parallel mechanisms, or they are assigned to predefined robot family.
Simulation tools for robotic systems differ from each other regarding the aspect of the robot research they support, how open they are or on which platforms they work. However, many tools are not always fulfilling all the requirements of the research activities in robotic laboratories like reconfigurability, openness and ease of use, etc.
Reconfigurability and openness are features already recognized by many as essential in the development of advanced robot control algorithms (Alotto et al., 2004; Lambert et al., 2001; Lippiello et al., 2007). Not only is it important to have easy access to the system at all levels (e.g. from high-level supervisory control all the way down to fast servo loops at the lowest level), but it is a necessity to have open control architectures where software modules can be modified and exteroceptive sensors like force/torque sensors and vision systems can be easily integrated. Reconfigurability should also be reflected when more fundamental changes to the controller architecture are required, in the necessity of quickly being able to make modifications in the original design and verify the effect of these modifications on the system. In other words, the user should be able to quickly modify the structure of the control without having to alter the simulation system itself.
In the last decade the software has become more and more easy to use. This is still one of the main major issues when selecting a software tool. First of all, the tools are used by many users in a laboratory and not all of them have the same expertise. To boost the knowledge exchange, it is of benefit that they work with the same tools. Next, testing of different control algorithms on real robotic systems is in general not very user friendly: the algorithms usually have to be rewritten for the real-time execution and the different implementation details have to be considered (Lambert et al., 2001; Žlajpah, 2001). This forces the user to devote a large part of the design time to topics not connected with the main issues of the control de-

sign, especially when he is not interested in software implementation issues. The ease of use becomes even more important when students are working with robots. In most cases they work in a laboratory for a shorter period, they are focused on their projects and they could become frustrated if they have to learn a lot of things not directly connected to their tasks. Finally, in research laboratories different robot systems are used equipped with more or less open proprietary hardware and software architecture. Therefore, it is much desired that the control design environment is unified, i.e. the same tools can be used for all robot systems.
The simulation tools for robotic systems can be divided into two major groups: tools based on general simulation systems and special tools for robot systems. Tools based on general simulation systems are usually represented as special modules, libraries or user interfaces which simplify the building of robot systems and environments within these general simulation systems (e.g. SolidWorks (RobotWorks, 2008)). On the other hand, special simulation tools for robots cover one or more tasks in robotics like off-line programming and design of robot work cells (e.g. Robcad (RobCAD, 1988)) or kinematic and dynamic analysis (Corke, 1996; SimMechanics, 2005). They can be specialized for special types of robots like mobile robots, underwater robots, parallel mechanisms, or they are assigned to predefined robot family. Depending on the particular application different structural attributes and functional parameters have to be modelled.
For the use in research laboratories, robot simulation tools focused on the motion of the robotic manipulator in different environments are important, especially those for the design of robot control systems (Corke, 1996; MSRS, 2008; SimMechanics, 2005; Webots, 2005). Recently, Microsoft Robotics Studio (MSRS, 2008) has been launched with a general aim to unify robot programming for hobbyist, academic and commercial developers and to create robot applications for a variety of hardware platforms. The system enables both remotely connected and robot-based scenarios using .NET and XML protocols. The simulation engine enables real-time physics simulation and interaction between simulated entities. Each part of the control loop can be substituted with the real or simulated hardware. Although the system is still under development, it is not easy to add new entity, for example a new robot or a new sensor. One of the major drawbacks seems to be the low data throughput rate, which does not allow the realization of complex control laws at high sampling frequency. Therefore, it is not clear yet if MSRS is appropriate for research robotics, especially for complex systems. Real time requirements are better solved in another programming/simulation framework, MCA2 (MCA2, 2008). MCA is a modular, network transparent and realtime capable C/C++ framework for controlling robots and other hardware. The main platform is Linux/RTLinux, but the support for Win32 and MCA OS/X also exists. However, it is still a complex system and therefore less appropriate for education and students projects.

### 2.1 MATLAB based tools

MATLAB is definitely one of the most used platforms for the modelling and simulation of various kind of systems and it is not surprising that it has been used intensively for the simulation of robotics systems. Among others the main reasons for that are its capabilities of solving problems with matrix formulations and easy extensibility. As an extension to MATLAB, SIMULINK adds many features for easier simulation of dynamic systems, e.q. graphical model and the possibility to simulate in real-time. Among special toolboxes that have been developed for MATLAB we have selected four: (a)Planar Manipulators Toolbox (Žlajpah, 1997), (b)Planar Manipulators Toolbox with SD/FAST (SD/FAST, 1994), (c)"A Robotic

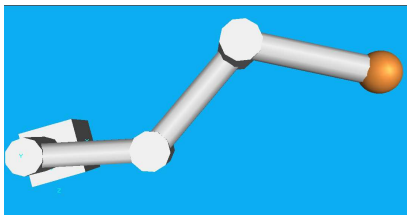


Fig. 1. Simple 3-R planar manipulator

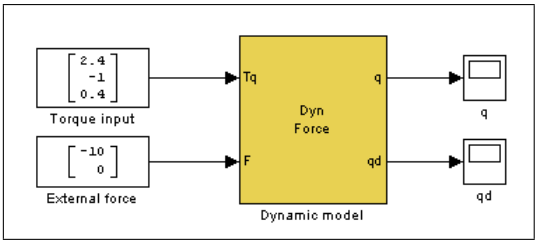



Fig. 2. Top level block scheme

Toolbox" (Corke, 1996), (d) "SimMechanics Toolbox" (SimMechanics, 2005) and (e) "20-sim" (Kleijn, 2009).

To illustrate different approaches to the dynamic simulation of robot manipulators we have selected as an object a simple planar manipulator which has 3 revolute joints acting in a plane as shown on Fig. 1. The main part of any simulation is the dynamic model. To focus on it, we simulate only the dynamics, without any task controller.

Let the configuration of the manipulator be represented by the vector $\boldsymbol{q}$ of $n$ joint positions, and the end-effector position (and orientation) by $m$-dimensional vector $\boldsymbol{x}$ of task positions. The joint and task coordinates are related by the following expressions

$$\boldsymbol{x} = \mathbf{p}(\boldsymbol{q}), \qquad \dot{\boldsymbol{x}} = \mathbf{J}(\boldsymbol{q})\dot{\boldsymbol{q}}, \qquad \ddot{\boldsymbol{x}} = \mathbf{J}\ddot{\boldsymbol{q}} + \dot{\mathbf{J}}\dot{\boldsymbol{q}} \tag{1}$$

where $\mathbf{J}$ is the Jacobian matrix, and the overall dynamic behaviour of the manipulator is described by the following equation

$$\boldsymbol{\tau} = \mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{h}(\dot{\boldsymbol{q}}, \boldsymbol{q}) + \boldsymbol{g}(\boldsymbol{q}) - \boldsymbol{\tau}_{\mathrm{F}} \tag{2}$$

where $\tau$ is the vector of control torques, $\mathbf{H}$ is the symmetric positive-definite inertia matrix, $\mathbf{h}$ is the vector of Coriolis and centrifugal forces, $\boldsymbol{g}$ is the vector of gravity forces, and vector $\boldsymbol{\tau}_{\mathrm{F}}$ represents the torques due to the external forces acting on the manipulator.

Fig. 2 shows the top level block scheme of the system. This scheme is the same in all cases, only the *Dynamic model* block is changed.

**(a) Planar Manipulators Toolbox**

Planar Manipulators Toolbox is intended for the simulation of planar manipulators with revolute joints and is based on Lagrangian formulation. Planar Manipulators Toolbox can be used to study kinematics and dynamics, to design control algorithms, for trajectory planning. It enables also real time simulation. Due to its concept it is a very good tool for education. To gain the transparency, special blocks have been developed to calculate the kinematic and dynamic models. These blocks are then used to build the desired model. Fig. 3 shows the dynamic model where an external force acts on the end-effector. The block *dymodall* which calculates the system vectors and matrices $\boldsymbol{x}$, $\mathbf{J}$, $\dot{\mathbf{J}}$, $\mathbf{H}$, $\boldsymbol{h}$ and $\boldsymbol{g}$ and then joint accelerations are calculated using Lagrangian equation.

**(b) Planar Manipulators Toolbox with SD/FAST**

In this case we use Planar Manipulators Toolbox but the dynamic model is calculated SD/FAST library. SD/FAST can be used to perform analysis and design studies on any mechanical system which can be modelled as a set of rigid bodies interconnected by joints, influenced by forces, driven by prescribed motions, and restricted by constraints (SD/FAST, 1994). The dynamic model has the same structure as given in Fig. 3 except that the block *dymodall*

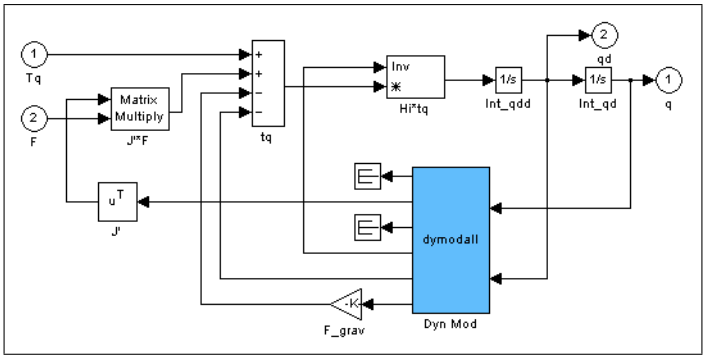



Fig. 3. Dynamic model (Planar Manipulators Toolbox )

is now a special S-function interfacing SD/FAST procedures and Simulink. The robot kinematics (geometry) and link mass properties are passed to SD/FAST in the System Description file (Fig.4). Then using the SD/FAST compiler the dynamic model is generated which is then called in S-function. To calculate the dynamics SD/FAST uses the advanced Kane's formulation and Order($n$) formulation.

```
# model of a planar manipulator with 4dof
language = c
gravity = 0 -9.81 0
#link1
  body = link1  inb = $ground
  joint = pin prescribed = ?
  mass = 1   inertia = 0  0  1
  bodytojoint = 0.5 0 0
  inbtojoint = 0.5 0 0
  pin = 0 0 1
#link2
  body = link2  inb = link1
  joint = pin prescribed = ?
  mass = 1  inertia = 0  0  1
  bodytojoint = 0.5 0 0
  inbtojoint = 0.5 0 0
  pin = 0 0 1
#link3
  body = link3  inb = link2
  joint = pin prescribed = ?
  mass = 1  inertia = 0  0  1
  bodytojoint = 0.5 0 0
  inbtojoint = 0.5 0 0
  pin = 0 0 1
```

Fig. 4. System Description file for 3R planar manipulator (SDFAST)

**(c) Robotics Toolbox**

The Robotics Toolbox provides many functions that are required in robotics and addresses areas such as kinematics, dynamics, and trajectory generation. The Toolbox is useful for the simulation as well as for analysing the results from experiments with real robots, and can be a powerful tool for education. The Toolbox is based on a general method of representing the kinematics and dynamics of serial-link manipulators by description matrices. The inverse dynamics is calculated using the recursive Newton-Euler formulation. Although it was initially meant to be used with MATLAB, it can be also used with Simulink. Fig. 5 shows the definition of the robot model and the block scheme of the dynamic model using Robotics Toolbox.

**(d) SimMechanics Toolbox**

SimMechanics extends Simulink with the tools for modelling and simulating mechanical systems. With SimMechanics, you can model and simulate mechanical systems with a suite of tools to specify bodies and their mass properties, their possible motions, kinematic constraints,

```
%% Definition of the R3 planar robot
for i=1:nj
 LR{i}=link([0 L(i) 0 0 0],'standard');
 LR{i}.m=m(i);
 LR{i}.r=[-Lc(i),0,0];
 LR{i}.I=[1 1 1 0 0 0]*II(i);
 LR{i}.Jm=0;
 LR{i}.G=1;
 LR{i}.B=Bv(i);
 LR{i}.Tc=[0 0];
end R3=robot(LR);
R3.name='R3';
R3.gravity=[0 9.81 0];
R3.q=q0';
```

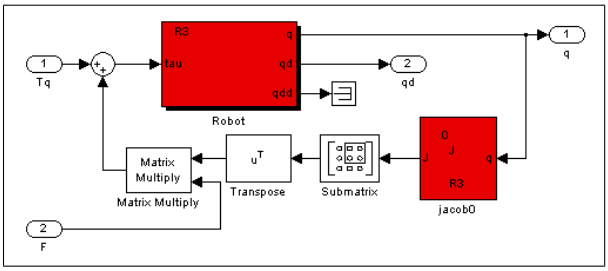



Fig. 5. Dynamic model (Robotics toolbox)

and coordinate systems, and to initiate and measure body motions (SimMechanics, 2005). To get a dynamic model of a robot manipulator we have first to build the link model, i.e. to connect link masses with joints as it is shown on Fig. 7. All link models are then connected together to the complete model (Fig. 6).

**(e) 20-sim**

Although 20-sim is a stand-alone simulation system (described later), it has a possibility to export the model to Simulink blocks as C-mex function. For comparison, we have modelled our robot manipulator using the 3D Mechanic Editor where you can model mechanical systems by specifying bodies, joints, sensors and actuators (Kleijn, 2009). To get a dynamic model of a robot manipulator we have first defined the links and then we have connected links with joints as it is shown on Fig. 8. Adding the trajectories generator, controllers and power amplifiers with gears a complete model of the system can be built (Fig. 9). Using the C code generator in 20-sim we have generated a Simulink block of the manipulator subsystem (R3). This block is then used in Simulink simulation scheme as shown in Fig. 2.

In all five cases it has been very easy to build the robot system. One of the differences between these tools is that special toolboxes for robot modelling have predefined more specific functions and blocks as the general toolboxes. The other difference is the execution time. In Fig. 10 we give the calculation time for the dynamic model for all five approaches. First we can see that SD/FAST is significantly faster than other and is increasing more slowly versus the

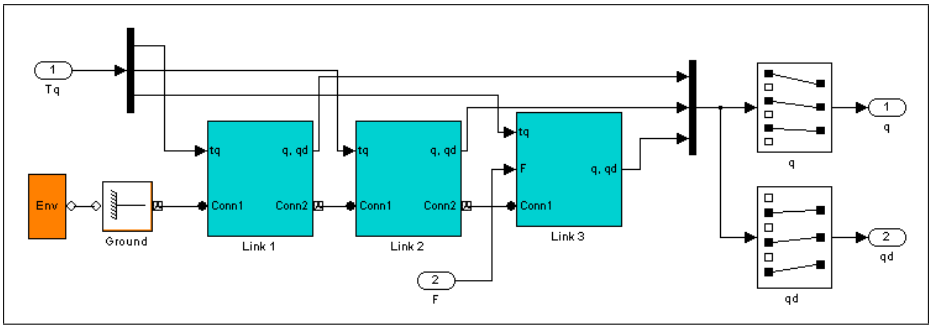



Fig. 6. Dynamic model of 3R manipulator (SimMechanics toolbox)

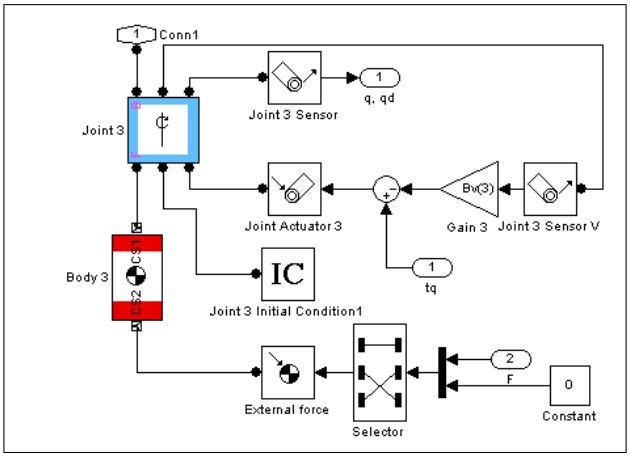



Fig. 7. Model of one link (SimMechanics toolbox)

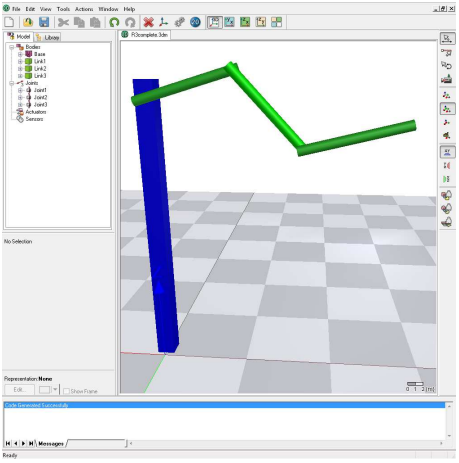


Fig. 8. Modelling robot manipulator using 20-sim 3D Mechanic Editor

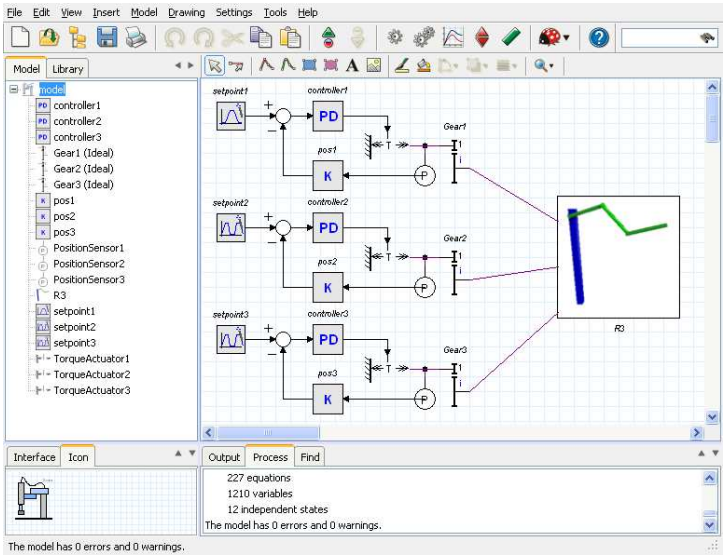



Fig. 9. Complete model of 3R manipulator (20-sim)

degrees-of freedom than other. Next, Planar Manipulators Toolbox is fast for small number of degrees-of-freedom and the execution time increases fast with the number of degrees-of-freedom. The Robotics Toolbox is relatively fast as long as we use only the inverse dynamics (Note that in Fig. 10 only the calculation time for the inverse dynamic model is shown). Otherwise, e.g. for the calculation of the Jacobian matrix, it is significantly slower, because the calculation is based on M-functions. Also, the model generated in 20-sim is fast (simulation within 20-sim environment is even faster). A little slower is the SimMechanics Toolbox. In both cases the execution time versus the number of degrees-of-freedom increases similarly. However, if the models the of robot manipulators should be used in the controller (e.g. the Jacobian matrix), then SimMechanics Toolbox and 20-sim are not appropriate.

### 2.2 Other general simulation systems

Similarly as in MATLAB the robot system can be simulated in Dymola and Modelica, or 20-sim. Here, the MultiBody library provides 3-dimensional mechanical components to model rigid multibody systems, such as robots. The robot system is built by connecting the blocks representing parts of the robot like link bodies, joints, actuators, etc. Fig. 11 shows the block scheme of a complete model of the KUKA robot including actuators, gears and the controller (Kazi & Merk, 2002). Fig. 12 shows the simulation of a parallel robot manipulator with 20-sim (3D Mechanics Toolbox) (Kleijn, 2009).

Robotica is a computer aided design package for robotic manipulators based on Mathematica (Nethery & Spong, 1994). It encapsulates many functions into a Mathematica package allowing efficient symbolic and numeric calculation of kinematic and dynamic equations for multi-degree-of-freedom manipulators. Robotica is intended, first of all, for model generation and analysis of robotic systems and for simulation.

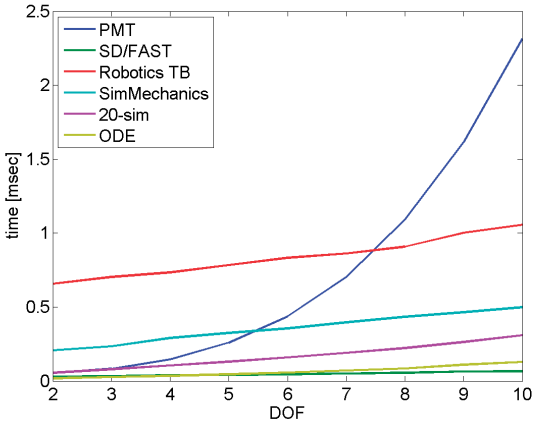



Fig. 10. Comparison of the calculation time versus number of DOF for the dynamic model of *n*-R planar robot manipulator

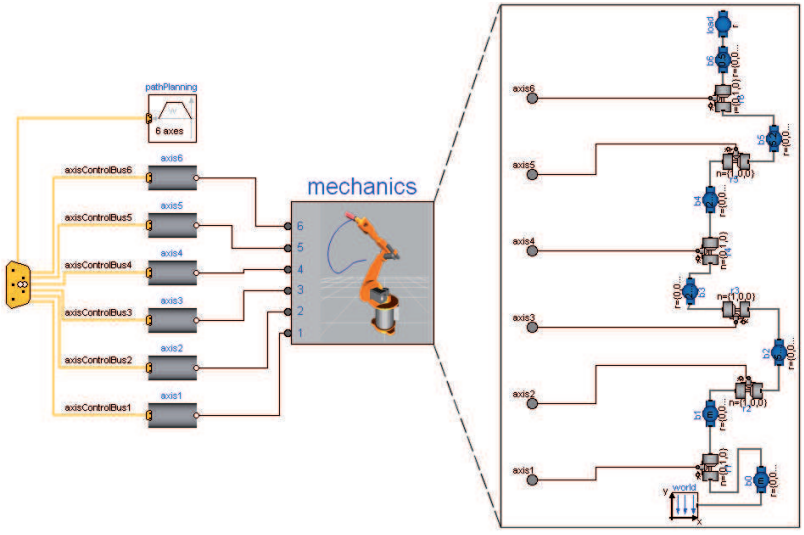



Fig. 11. Simulation of a robot with Modelica

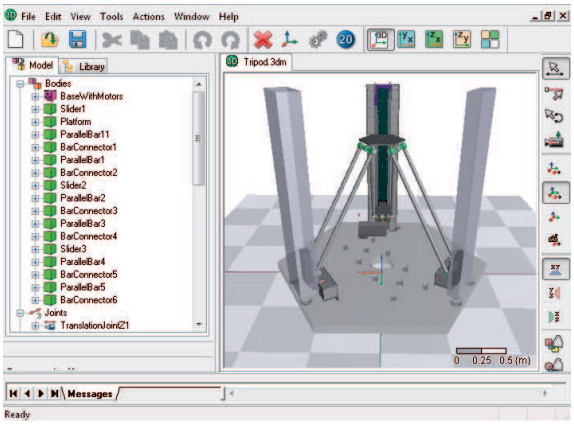



Fig. 12. Simulation of a Tripod with 20-sim 3D Mechanics Toolbox

### 2.3 Multibody dynamic engines

In the last years new simulation tools have been available based on the general engines for the simulation of physics environments (NGD, 2008; ODE, 1994; SD/FAST, 1994). These engines provide libraries for simulating the multi-body dynamics, i.e. the physics of motion of an assembly of constrained or restrained bodies. As such they encompass the behaviour of nearly every object and among them are, of course, also robot manipulators. These dynamic engines have beside the dynamics simulation engine also a collision detection engine. The collision engine is given information about the shape of each body and then it figures out which bodies touch each other and passes the resulting contact point information to the user. The user can then take the proper actions.

As an example we have selected the Open Dynamic Engine (ODE, 1994). Building the model of a robot is straightforward. First you have to create all bodies and connect them if desired with proper joints. For example, the 3DOF model as shown before can be defined as shown in Fig. 13. For comparison with MATLAB based tools the computational time for $n$R planar manipulators (n=2,. . . ,10) is shown in Fig. 10. It can be seen that the computational efficiency of ODE is comparable to the SD/FAST library.

Unfortunately, most of dynamics engines do not support functionality necessary to include robot models in the control algorithms. Advanced control algorithms including robot models include Jacobian matrices, inertia matrices, gravity forces, etc., and they are not explicitly defined. The user can use some implicit algorithms or other tools to get these parameters.

The dynamic simulation of multibody systems becomes very important when introducing robotics into human environments (Go et al., 2004; Khatib et al., 2002; Miller & Christensen, 2003) where the success will not depend only on the capabilities of the real robots but also on the simulation of such systems. For example, in applications like virtual prototyping, teleoperation, training, collaborative work, and games, physical models are simulated and interacted with both human users and robots.

For example, the dynamics engine within a robotic grasping simulator known as GraspIt! (Miller & Allen, 2004) computes the motions of a group of connected robot elements, such as an arm and a hand, under the influence of controlled motor forces, joint constraint forces, contact forces and external forces. This allows a user to dynamically simulate an entire grasping

```
// create world
contactgroup.create (0);
world.setGravity (9.81,0,0);
dWorldSetCFM (world.id(),1e-5);
dPlane plane(space,0,0,1,0);

// fixed robot base
xbody[0].create(world);
xbody[0].setPosition(0,0,SIDE/2);
box[0].create(space,SIDE,SIDE,SIDE);
box[0].setBody(xbody[0]);
bjoint = dJointCreateFixed (world,0);
dJointAttach (bjoint,xbody[0],0);
dJointSetFixed (bjoint);
// robot links
for (i=1; i<=NUM; i++) {
  xbody[i].create(world);
  xbody[i].setPosition(0,(i-0.5)*LENG,(i
  m.setBox(1,SIDE,LENG,SIDE);
  m.adjust(MASS);
  xbody[i].setMass(&m);
  box[i].create(space,SIDE,LENG,SIDE);
  box[i].setBody(xbody[i]);
}
// robot joints
for (i=0; i<NUM; i++) {
  joint[i].create(world);
  joint[i].attach(xbody[i],xbody[i+1]);
  joint[i].setAnchor(0,(i)*LENG,(i+1)*SIDE);
  joint[i].setAxis(0,0,1);
}
```

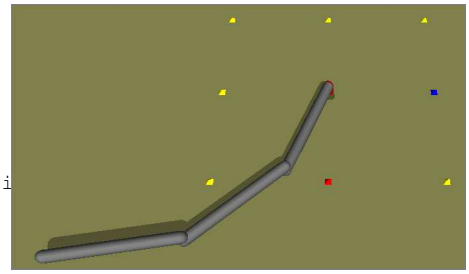


Fig. 13. Definition of 3R planar manipulator in ODE

task, as well as test custom robot control algorithms. Fig. 14 shows how a robot hand can grasp a mug (Miller & Allen, 2004). In this example, all contacts between the fingers and the mug and related forces are analysed.

## 3. Control design and integrated environment

A very important part of the robotic system is the control system. In the process of controller design different steps have to be performed. First of all, the system has to be modelled. In the next step, the control algorithm is developed. The first results are then obtained by the simulation. If the results are satisfactory, then in the final stage the control algorithms are tested on a real system. For this, a real-time code should be generated and implemented on the real system. The integration of all these steps, although essential, is very difficult. Namely, the different steps in the development of the controller require the use of different methods for which different tools are needed. Hence, the results from one step to another have to be transferred often by hand. This bottleneck can be overcome if control design and testing are done in an integrated environment.

The importance of simulation tools in the development of robot control systems has been recognized by researchers very early. We have been using different simulation tools for over 20 years and many of them have been developed in our laboratory. In the last decade we have been using for the control design MATLAB/Simulink based integrated environment based

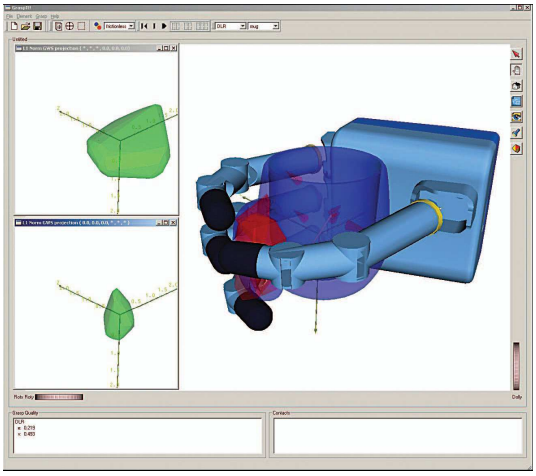


Fig. 14. A force-closure grasp of the mug using the DLR hand, which has a metal palm, inner link surfaces, and rubber fingertips.

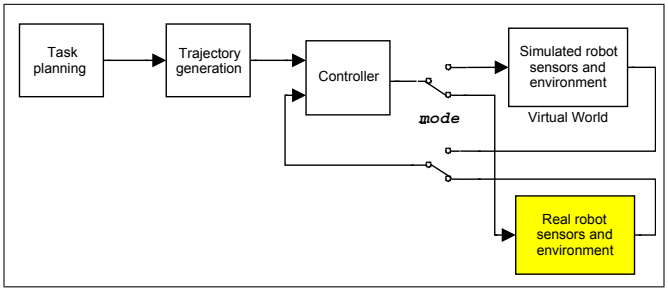



Fig. 15. A block diagram of the integrated environment

on Planar Manipulators Toolbox for dynamic simulation of redundant planar manipulators Žlajpah (2001). It enables the use of different sensors in the control loop and also the real-time implementation of the controller and hardware-in-the-loop simulation. Figure 15 shows the general simulation scheme in this environment. A crucial feature inherited in this scheme is indicated by the mode switches. Namely, the user can easy switch between using model or a real system in the simulation loop. This is one of the main features which we need for development of the robot control systems.

For example, Fig. 16 shows the dynamic model of a manipulator and a sensor detecting the object in the neighbourhood of the manipulator. When the developed controller is tested on a real system we substitute the manipulator with our experimental robot, i.e. the dashed blocks in Fig. 16 are replaced with the interface blocks as shown in Fig. 17a. Fig. 17b shows our laboratory manipulator with four revolute DOF acting in a plane, which has been developed specially for testing the different control algorithms for redundant robotic manipulators, performing an obstacle avoidance task.

The integration of the two modes is the most important feature of the integrated environment. This has been recognized also by many other researchers. For example, one of the goals of the

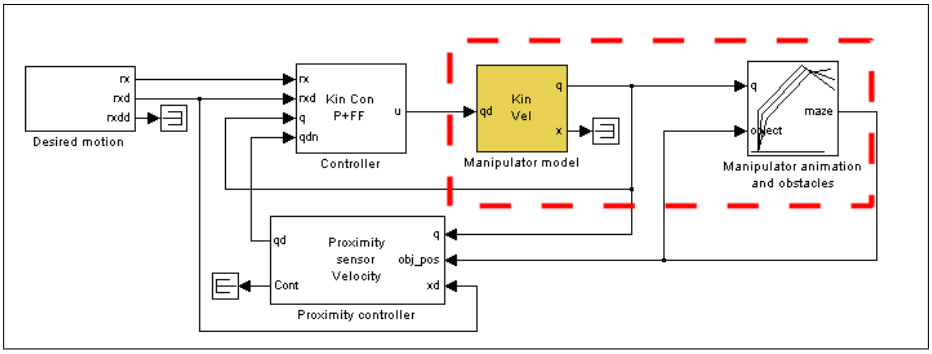
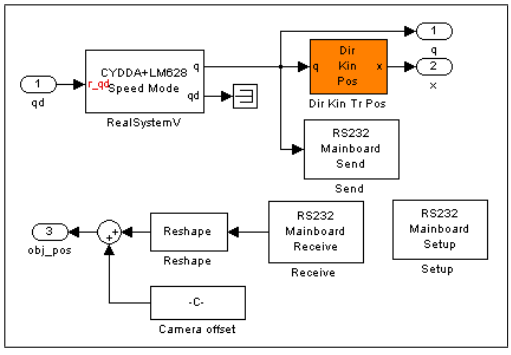
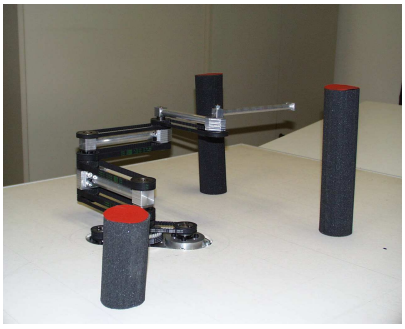



Fig. 16. A block diagram including the dynamic model of a manipulator and a sensor detecting the object in the neighbourhood of the manipulator



a) Robot interface block scheme b) Experimental 4-R manipulator

Fig. 17. Avoiding obstacles - Hardware-in-the loop simulation

IST project "RealSim" was to develop an efficient tool for modelling, simulating and optimising industrial robots (Kazi & Merk, 2002). Fig. 18 shows the structure of the simulation system where a real control unit is connected to the simulator of an industrial robot. Using such system the controller can be tested without a real robot, e.g. before even the robot has been built.

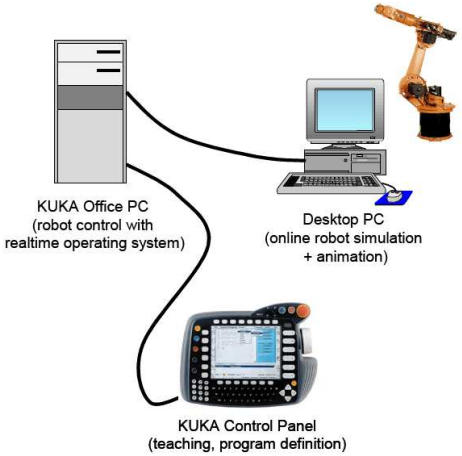



Fig. 18. Real time optimisation system (Project **RealSim** (Kazi & Merk, 2002))

### 3.1 The concept of distributed environment

The Planar Manipulators Toolbox has proved to be a very useful and effective tool for many purposes, but it has been primary designed for the kinematic and dynamic simulation of the planar manipulators and to develop and test control algorithms on the lower control level, especially for redundant manipulators. In the last years, the scope of our research is oriented more in the development of control systems for humanoid and service robots (Gams et al., 2009; Omrčen et al., 2007). These robots have in general a more complex mechanical structure with many degrees-of-freedom. So, complex kinematic and dynamic models are necessary to simulate them. Furthermore, the control methods and algorithms are now usually a part of the higher robot control levels and the low level close-loop control algorithms are assumed to be a solved issue. These high level control algorithms can become very complex and may even require parallel computation distributed over more computers.
Considering all new requirements, which are:

to simulate the kinematics and dynamics of arbitrary chosen kinematic chain describing different manipulators,

to enable integration of different sensor systems like vision and force sensors,

to enable simulation of scenarios for complex robot tasks,

to include the model the robots' environments,

to visualize the robots and their environment and

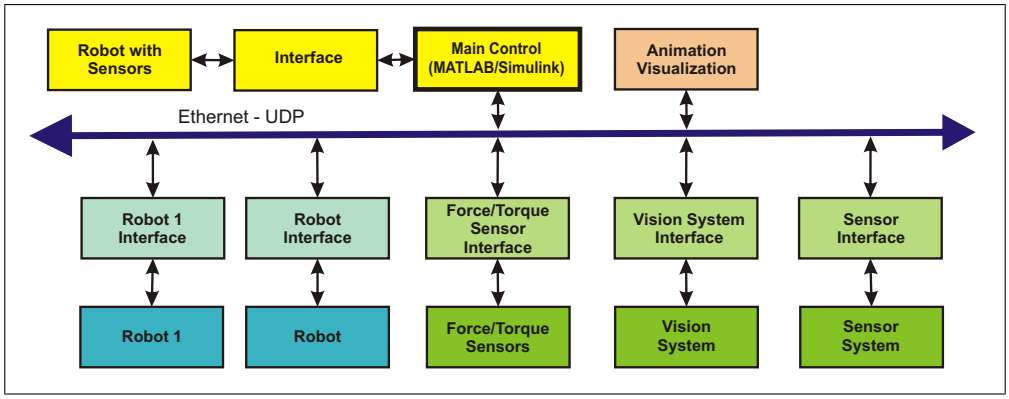



Fig. 19. A functional block diagram of the robot integrated environment in Robotics Laboratory including the robot PA10, mobile platform Nomad XR 4000 and sensor systems

to enable integration of real robots in the simulation loop, we have to reconsider the concept of the control design environment we will use in future. Based on our good experience with MATLAB/Simulink we have decided that this environment will be the kernel of our simulation tools. However, some of the above requirements can be easier fulfilled by using other tools. For example, the visualization of the robot and the environment can be easily done by dedicated graphics tools. Furthermore, advanced robot control strategies rely intensively on feedback sensor information. The most complex sensor system is the vision system, which can have several configurations and can be implemented on a single computer or on a computer cluster composed of many computers running different operating systems. To integrate such a diversity of hardware components in a unique framework we have decided to use the ethernet communication and the UDP protocol. In this way, we have maximal possible "degree-of-openness" of the system. Fig. 19 shows a typical scheme of our robot integrated environment.
In this scheme, each block can represent a real system or a model of that system. Note that because of using ethernet communication between the blocks, different software tools on different platforms can be used to simulate specific parts of the system. Consequently, the simulation environment can consist of several interacting applications, each representing a part of the system (Petrič et al., 2009).

### 3.2 Simulink block library

In Simulink, a system is modelled by combining input-output blocks. To gain the transparency, we try to represent a system by the block structure with several hierarchical levels, i.e. by combining different basic blocks, subsystems are built which become a single block at the higher level. In Figure 15 the typical robot subsystems can be seen: the trajectory generation, the controller, the model of the manipulator and the environment and the animation of manipulator motion. Figure 20 shows the Robot systems block library. The goal of the library is to provide blocks which are needed to simulate robotic systems and can not be modelled with standard blocks. First of all, these are the blocks for robot kinematic and dynamic models, the blocks for sensors systems, the typical transformations present in robot systems and the special interface blocks for robots, sensors and all other communications. Additionally, the

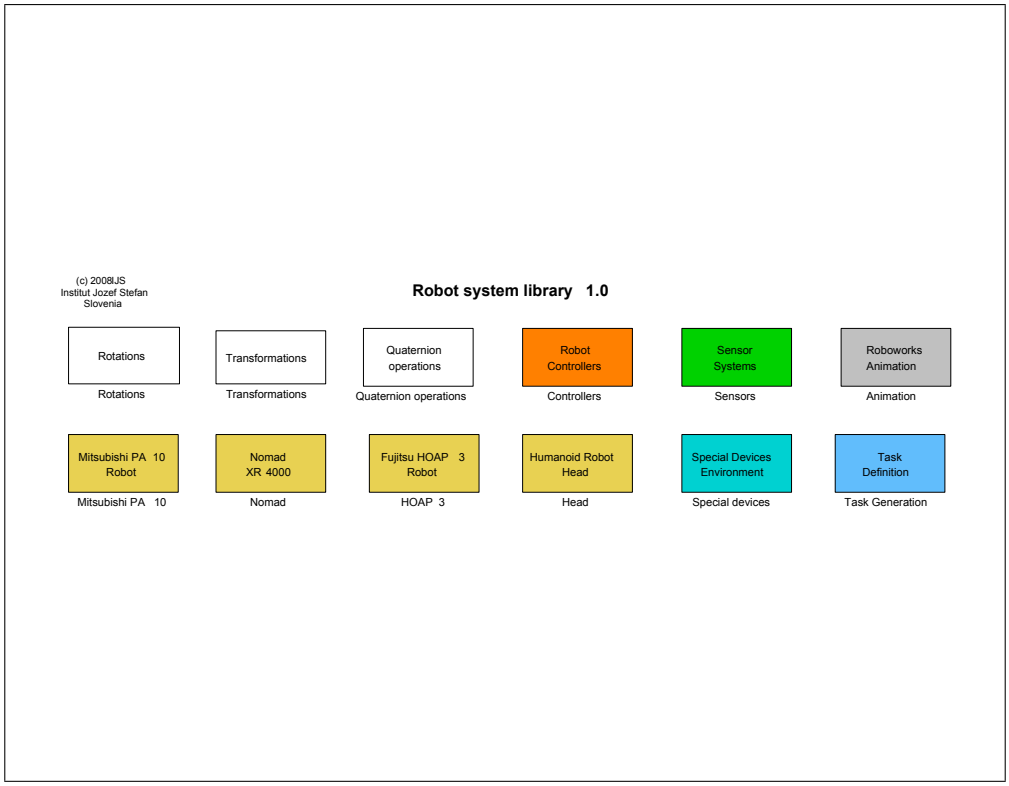



Fig. 20. Simulink Robot system library

library includes some blocks with standard subsystems like task space controllers, trajectory generation modules, etc.

### 3.3 Integration of sensors

Advanced robotics is characterized by the variety of complex sensory systems, e.g. vision sensors, force sensors, acoustic sensors, laser scanners, proximity sensor, etc. Therefore it is extremely important to apply as accurately as possible the sensor models into the simulation environment. The models of sensors are completely transparent to the design environment, i.e. real sensor can be substituted with the simulated one and vice versa in the control loop.
The integration of sensors depends on their characteristics. Complex sensor systems like vision and acoustic sensors, or more advanced laser proximity sensors require relatively high computational power for signal processing. In many cases, it is difficult to accomplish all required data processing on the local computer. Often we have to apply a remote computer or even a remote computer cluster in order to obtain required computational power. In such a case, the subsystems are connected through ethernet with UDP protocol. We have developed a special protocol classes for different sensors, actuators and other subsystems. However, the performance is also affected by the communication delays. Therefore, it is favourable to pro-