Modelling and Simulation of the Locomotion of Humanoid Robots


*Istituto Italiano di Tecnologia (IIT), Genova 16163, Italy
(e-mail: gustavo.cerda@iit.it).

**Centre for Interdisciplinary Computational and Dynamical Analysis (CICADA), School of Mathematics, The University of Manchester, M13 9PL, Manchester, United Kingdom
(e-mail: houman.dallali@postgrad.manchester.ac.uk).

*** Control Systems Group, School of Electrical and Electronic Engineering, The University of Manchester, M13 9PL, UK, (e-mail: martin.brown@manchester.ac.uk)

Abstract: Modelling and simulation is a critical stage in the design and testing of complex systems, in particular humanoid robots. Our motivation to develop a new simulator is that existing tools do not provide the flexibility needed for modelling the C-Cub, the most recent open source humanoid robot with compliant joints developed at the Italian Institute of Technology (IIT). Furthermore, we are interested in a simulator where the user has the ability to customize the simulation in order to represent the actuators and sensors that are actually used. In this paper, a simulator based on Robotran, a symbolic multibody modelling software tool, and Matlab is described. The approach presented in the paper has been used to develop and test control systems for the C-Cub and an earlier version (iCub).

Keywords: Humanoid, Modelling, Simulation, Multibody, Compliance, C-Cub

1. INTRODUCTION

The development of realistic mathematical models and simulators is a time consuming and laborious process. However, it is a vital stage in the analysis, design and manufacture of complex systems. Mathematical models are also invaluable tools for the design of control algorithms and in this case, it is essential to include models for actuators and sensors. Rapid testing can be carried out in the simulator before implementation in the real hardware, thus avoiding costly damages to equipment. Ideally, the model and simulator should be a faithful representation of the real system so that there is little or no difference between running the simulation and testing (the control system) on the real hardware.

The focus of this paper is on the modelling and simulation of humanoid and bipedal robots. As such, models for rigid multibody systems need to be developed, together with the relevant actuators and sensors (electric, hydraulic, pneumatic motors, gearboxes, belt/cable drives, compliance, non-linear friction, sensor dynamics, noise/quantization). Models that include compliance are essential for humanoids that have been built and designed using compliant elements, e.g. Flame and C-Cub (Fig. 1). Modelling compliance in the robot drive system is also important since high compliance in the drives will have an impact on the design of control systems. One of our specific goals is to develop an open simulator for C-Cub. This will help to foster research and comparative studies using the same platform. A similar approach can also be used for the iCub, and earlier humanoid developed at IIT without compliant joints, Tsagarakis et al. (2007).

The paper is organized as follows. Section two presents a brief overview of existing rigid multibody modelling and simulation software packages and their shortcomings. In section three a brief description of Robotran, Samin et al. (2003), a powerful tool for multibody system modelling, is given. The Robotran-Matlab simulator that has been developed is described in section four. Conclusions and future work are given in the final section of this paper.

Fig. 1 C-Cub legs: Compliant joints in ankles and knees.
2. EXISTING SIMULATORS FOR HUMANOID ROBOTS

During the last twenty years numerous software packages for modelling multibody systems have been developed and some have been specifically written for humanoid robots.

OpenHRP3 (Open Architecture Human-centered Robotics Platform version 3) is a simulator and motion control library for humanoid robots developed at the National Institute of Advanced Industrial Science and Technology, Kanehiro et al. (2004). This is a free tool that can simulate the direct dynamics of open and closed chains of multibody systems. It includes a library of position, force/torque, vision, and inclination sensors. These are essential for developing control systems for the robots. The software also includes simple controllers for locomotion, balancing, a walking pattern generator as well as collision detection and avoidance schemes. Advanced controllers can be developed by the users. Besides standard tools for plotting simulation results, OpenHRP has a 3D visualization interface to display the simulation. The software developed in OpenHRP can be directly used in the HRP humanoid robots but this is not possible for other humanoid robots. The main reason is that several features in OpenHRP cannot be modified by the user. For example, the actuator models are fixed, and compliance in the robot joints or the robot links cannot be added to the model. Furthermore, there is very little documentation available for OpenHRP and since this is a free tool there is also no support from the developers.

Other simulators have been developed by Honda, Sony, Fujitsu for specific Humanoid robots (for ASIMO, Qrio and HOPA-1). Commercial software packages like Webots, Adams and RoboWorks are useful for modelling multibody systems, but lack generality for modelling actuators, sensors, ground contacts and impacts. None of these packages provide full control over all the features of interested.

To overcome the limitations in OpenHRP and existing simulators, Reichenbach (2009) started developing a new dynamic simulator. This simulator will be tested in Archie, a humanoid robot developed at the Technical University of Vienna and the University of Manitoba. However, the simulator could also be used for different robots including robot arms and other mechanical systems. One important feature to be added in this simulator is the capability of hardware accelerated calculations. This would substantially reduce computational times. The developer also aims to provide users with full control to add elements and customize the simulator to suit their own applications. Currently, Reichenbach’s simulator is not widely available and is still under development.

Humanoid simulators have also been developed in Matlab, Simulink and SimMechanics for Silo2, Ponticelli et al. (2006). In this approach, modelling different types of actuators including compliance in the robot drive system is quite simple. Also a comprehensive set of toolboxes is available for the design of advanced control systems. The rigid multibody modelling is essentially implemented in SimMechanics which also has tools for linearization and 3D visualization. One problem using SimMechanics is that simulations become slow as the complexity of the system grows, particularly when schemes to detect ground contacts or impacts (event detection) are included in the simulation. Moreover, adding a good model of non-linear friction will substantially increase the simulation run-time. An alternative for rigid multibody modelling is to use Robotran, Samin et al. (2003), in place of SimMechanics. This has some advantages compared with SimMechanics. Robotran generates symbolic models rather than numerical models and this improves the numerical accuracy and speed of simulations. Furthermore, Robotran also has a graphical user interface which simplifies entering the robot data. Body parameters, e.g. mass, inertia, location of centre mass and geometry, are entered for each body by filling the corresponding information, as shown in Fig. 2. The overall multibody system configuration is constructed by simply drawing bodies and the corresponding joints connecting them. Figure 3 shows a 10 degree of freedom model for the legs of C-Cub.

3. ROBOTRAN: SYMBOLIC MULTIBODY MODELLING

Robotran is a symbolic multibody modelling and simulation software package developed at the Universite Catholique de Louvain in Belgium. The software is based on Matlab and Simulink and provides a user-friendly environment. The main features of Robotran are summarized at the end of this section. The Robotran software package has three main components, the graphical user interface (MBsysPad), a symbolic equation generator (MBsysTran) and an interface between Robotran and Matlab (MBsysLab). The user is provided with the graphical and Matlab interfaces, while the symbolic equation generator is accessed on line via the Robotran web server. The symbolic equations are generated in a few milliseconds and the user receives a set of Matlab M-files with the symbolic equations. A 3D visualization/animation tool is also provided in MBsysPad.
addition, for each sensor placed on a body, the user can also extract the relevant Jacobians. Therefore, Robotran provides:

- symbolic equations for dynamics and kinematics (planar/3D),
- models for open and closed chains,
- linearized model around any operating point,
- equations are available in Matlab syntax and can use Matlab toolboxes for further analysis,
- actuator dynamics, including compliance are added in Matlab,
- ground impact models can be added in Robotran or Matlab,
- a simple graphical user interface for data input,
- model can be converted to match hardware sensors and actuators polarities and
- linear/angular positions, velocities, accelerations, orientations and Jacobians of any chosen points of the multibody system.

4. A ROBOTRAN-MATLAB SIMULATOR FOR HUMANOID ROBOTS

In this section, the enhancements that have been developed to integrate actuator models into the symbolic equations generated by Robotran are described. For humanoid robot locomotion it is also necessary to take into account the hybrid nature of the system, i.e. changing between open and closed chain models (single/double support phases) as well as the modelling of ground impacts. Moreover, by introducing suitable joint coordinate transformations, we describe how the Robotran models become more realistic and Robotran can be used to map simulation results directly to C-Cub.

4.1 Actuator modelling

The equations of motion are derived using the recursive Newton-Euler formalism and the constraint forces are introduced via the Lagrange multipliers method. The equations of motion take the form:

\[
M(\theta)\ddot{\theta} + c(\theta, \dot{\theta}, f, \tau) = Q(\theta, \dot{\theta}, \dot{\theta}_n, \dot{\theta}_n) + J^T(\theta)\lambda
\]  

(1)
\[ h(\theta) = 0 \]  
\[ \dot{h}(\theta, \dot{\theta}) = J(\theta)\dot{\theta} = 0 \]  
\[ \dot{h}(\theta, \dot{\theta}, \ddot{\theta}) = J(\theta)\ddot{\theta} + J(\dot{\theta}, \dot{\theta}) = 0 \]  
\[ J_m \dot{\theta}_m + B_m \dot{\theta}_m = T_m - Q(\theta, \theta_m, \dot{\theta}_m, \dot{\theta}_m) \]  
Equations (1)-(4) are generated by the MBsysTran module in Robotran. In the case of an open chain model, the Lagrange multipliers, \( \lambda \), are zero and (2)-(4) can be eliminated. The user has complete freedom to specify the external link forces and torques \( f, \tau \) as well as the external joint forces and torques \( Q(\theta, \dot{\theta}, \theta_m, \dot{\theta}_m) \). For example, a compliant joint with damping can be modelled by:

\[ Q(\theta, \dot{\theta}, \theta_m, \dot{\theta}_m) = K_s (\theta_m - \theta) + B_s (\dot{\theta}_m - \dot{\theta}) \]  
When the joint torques are generated by motors, the user needs to add the relevant differential equations (5) in a user_Derivatives M-file. Additional equations may be required depending on the type of actuators (electric, hydraulic, pneumatic etc.). For the C-Cub, the actuators are voltage controlled electric motors, and neglecting the electrical dynamics gives:

\[ T_m = K_r R_m^3 V_n - K_s R_m^3 K_s \dot{\theta}_m \]  
The parameters \( K_r, K_s \) represent the equivalent torque and bemf motor constants (motor parameters reflected to the joint axis), respectively; \( R_m \) represents the motor armature resistance and \( V_n \) is the applied motor voltage.

The MBsysLab module in Robotran provides a set of M-files to carry out a number of tasks: linearization and modal analysis; direct and inverse dynamic calculations (required for finding equilibrium points); coordinate partitioning for closed chain models; direct kinematics. For inverse kinematics the user can solve this problem by selecting a suitable numerical method (for example Newton-Raphson) and implement this in Matlab. For C-Cub we have developed an M-file using the sensor data from Robotran and solved the inverse kinematics problem given the robot’s centre of mass position as well as the position/orientation of swing leg foot. The user also has the freedom to select other methods for closed loop chains and direct dynamics calculations. In this case the user can develop their own M-files and only call the relevant functions provided by Robotran containing the symbolic equations. For example, solving (1) and (4) we can write an expression for the Lagrange multipliers

\[ \lambda = \left[ J M^{-1} J^T \right]^{-1} \cdot \left\{ J M^{-1} (Q(\theta, \dot{\theta}, \theta_m, \dot{\theta}_m) - c(\theta, \dot{\theta}, f, \tau)) + J \dot{\theta}(\theta, \ddot{\theta}) \right\} \]  
Substituting (7) in (1) yields

\[ M(\theta)\ddot{\theta} = Q(\theta, \dot{\theta}, \theta_m, \dot{\theta}_m) - c(\theta, \dot{\theta}, f, \tau) \]

\[ - J^T(\theta) \left[ J M^{-1} J^T \right]^{-1} \cdot \left\{ J M^{-1} (Q(\theta, \dot{\theta}, \theta_m, \dot{\theta}_m) - c(\theta, \dot{\theta}, f, \tau)) + J \dot{\theta}(\theta, \ddot{\theta}) \right\} \]

A set of M-files for all the actuators of C-Cub (5)-(7) have been developed. The M-files are then combined with the Robotran model (1)-(4) to produce the overall model for C-Cub. For linear control system design a similar approach is used except that the linearized Robotran model replaces (1)-(4). Discrete time control systems have been designed in the Robotran-Matlab simulator using LQR optimal control and state observers. These controllers have been implemented and tested on the real hardware. Initial tests were carried out on the legs and the upper body of ICub while the robot was off the ground. More recently a balancing controller was tested on C-Cub to keep an upright posture while the robot was on the ground (double support phase). The results of these tests have partially validated the modelling approach described in this section.

4.2 Ground impacts and exchange of support leg

Humanoid locomotion requires switching between closed and open chain models (exchange between single and double support phases). Inevitably this switching introduces ground impacts and the relevant models can be added to (1)-(5) directly in Robotran or Matlab. A simple model for ground impacts takes the form:

\[ \dot{\theta}^+ = \Phi(\dot{\theta}^-) \]  
Here, \( \dot{\theta}^- \) and \( \dot{\theta}^+ \) denote the joint velocities before and after the impact. Furthermore, when the role of support and swing legs is reversed, the joint positions and velocities in the model (also motor positions and velocities if applicable) also need to be swapped. The reason for this is that the Robotran equations assume that the support leg does not change

\[
\begin{bmatrix}
\theta^a_{\text{stance}} \\
\theta^b_{\text{stance}} \\
\theta^a_{\text{swing}} \\
\theta^b_{\text{swing}}
\end{bmatrix} =
\begin{bmatrix}
\theta^a_{\text{stance}} \\
\theta^b_{\text{stance}} \\
\theta^a_{\text{swing}} \\
\theta^b_{\text{swing}}
\end{bmatrix}
\]  
Here the superscripts \( a \) and \( b \) correspond to the event after and before the exchange of support.

Simulation of a full locomotion cycle therefore requires stopping the simulation when an impact is detected, carrying out the relevant calculations, swapping the role of legs in the model and re-starting the simulation. In this manner, it is possible to achieve a realistic simulation for several locomotion cycles. More sophisticated event detection schemes as those shown in Fig. 6 can also be implemented in Matlab.
4.3 Robotran model transformations

When building a model in Robotran, it is important to understand how the reference frames are chosen as well as the joint displacements convention used in Robotran. This is particularly important for mapping results in Robotran to the real hardware and developing control system that can be ported directly to the real system. When drawing the multibody system in Robotran all the body reference frames must be aligned. This is called “the standard configuration” and corresponds to zero displacements. For C-Cub the example shown in Fig. 2, the standard configuration corresponds to the upright position. In this example the reference frame $\text{XYZ}$ is chosen so that the positive x-axis points in the direction of forward walking, the positive z-axis points upwards and the positive y-axis is chosen to complete an orthogonal right handed coordinate system (Fig. 3). In Robotran all joint displacements are given as relative motions and the right hand rule convention determines the direction of positive rotations. Absolute displacements can be computed from the set of relative displacements and the corresponding time derivatives. For the particular case of C-Cub, the joint angles measurements for the stance leg are consistent with the convention in Robotran. However, the joint angles of the swing leg have the opposite direction to the Robotran convention. It is important to be aware of these differences, particularly when developing walking trajectories and control systems. To “convert” the Robotran model so that the angles in the model correspond to the angles in C-Cub we can define a suitable transformation. The transformation is quite simple to implement. The relation between the relative joint angles in Robotran and C-Cub is given by

$$\theta_n = W \theta_{\text{C-Cub}}$$

where $\theta_n$ denotes the joint angles in Robotran, $\theta_{\text{C-Cub}}$ are the angles in C-Cub and $W$ is a square invertible matrix

$$W = \begin{bmatrix} I_{5 \times 5} & 0_{5 \times 5} \\ 0_{5 \times 5} & -I_{5 \times 5} \end{bmatrix}$$

where $I_{5 \times 5}$, $0_{5 \times 5}$ denote $5 \times 5$ identity and zero matrices respectively. When the actuators are modelled, a similar transformation is required for the motor angles. In this manner, we can “transform” the Robotran model into a model that matches the real system sensors readings. The transformation $W$ and the inverse transformation can be incorporated into the relevant symbolic equations in the M-files generated by Robotran. A similar procedure can be implemented to cater for differences in the actuators signal polarities.

5. ROBOTRAN-MATLAB SIMULATOR DEMO MODELS

Two case studies for the Robotran-Matlab simulator have been developed.

The first case study is based on a compass gait walker Karssen et al. (2008). This is a simple two legged un-actuated system walking down a slope with curved feet and including a model for ground impacts. To model the slope in Robotran, the gravity vector is defined as a vector with vertical and horizontal components (along the slope direction). The ground impact model equations were initially derived in a terms of a different angle convention to that of the symbolic model in Robotran. In this case we need to implement a transformation similar to that described in section 4.3. After converting the Robotran angles (and angular velocities), the impact model equations are used to obtain the joint velocities after the impact. The result is then converted back to the Robotran angle convention and the legs are swapped using (11). The Compass gait demo simulator will run for several steps and the simulation results are displayed in a Matlab animation.

The second case study is the C-Cub simulator. The current version is only for 10 degrees of freedom (the two legs). The model parameters (masses, inertias and geometry) were obtained from a CAD package. This is the first simulator for C-Cub that includes the dynamic equations of motion. The relevant actuator models for the legs are included as Matlab M-files. The Robotran model in the demo is for the single support phase (stance right leg, swing left leg). This model is unstable and a stabilizing LQR controller is provided in the demo. A Matlab file is also available for solving a simple inverse kinematic problem and generate walking trajectories for the demo. As the project evolves the software will be updated. The goal is that the simulator will become a good representation of the real system. Our current interest is to use the simulator for designing, testing and comparing different control systems, Dallali et al. (2010). Users that are
interested in methods for trajectory generation can also use this model.

The demonstration simulators can be downloaded from:
http://www.cicada.manchester.ac.uk/research/icub/

6. CONCLUSIONS

We have outlined a generic Robotran-Matlab simulator for humanoid robots. The simulator can be used for any robotic system and the user has complete freedom to specify and design actuators and suitable control systems. We have partially validated the modelling approach and simulator by designing control systems that have been tested on two Humanoid platforms C-Cub and ICub. Additional work is needed to develop a set of M-files for solving a variety of inverse kinematics problems for trajectory generation. Also, further work is needed to improve the portability of the control system designed in Matlab to the real system. Some initial attempts in this respect were unsuccessful. The new release of Maple4Sim seems to provide the capability of generating stand alone C-code and this can provide a good solution for exporting the controller to the real system. Furthermore, Maple4Sim can integrate the multibody modelling and the actuator models and produce symbolic models. Maple4Sim is currently under evaluation as an alternative to the Robotran-Matlab simulator.

![Diagram of event detection scheme](image)

**Q1)** Swing  
**Q2)** Push-off 1  
**Q3)** Push-off 2  
**Q4)** Push-off 3  

**Q1)** Event 1: heel impact (leading leg)  
**Q2)** Event 2: toe impact (leading leg)  
**Q3)** Event 3: heel leaves ground (trailing leg)  
**Q4)** Event 4: toe leaves ground (trailing leg)

Fig. 6. Four event detection scheme.

ACKNOWLEDGEMENTS

This research was supported by the CICADA project EPSRC grant EP/E050441/1, The University of Manchester and the Italian Institute of Technology.

REFERENCES


