Recursive dynamics simulator (ReDySim): A multibody dynamics solver

Suril V. Shah,[1, a)] Paramanand V. Nandihal,[2, b)] and Subir K. Saha[2, c)]
[1)] *Department of Mechanical Engineering McGill University, QC H3A0C3, Canada*
[2)] *Department of Mechanical Engineering Indian Institute of Technology, New Delhi 110016, India*

(Received 10 September 2012; accepted 25 September 2012; published online 10 November 2012)

Abstract Recursive formulations have significantly helped in achieving real-time computations and model-based control laws. The recursive dynamics simulator (ReDySim) is a MATLAB-based recursive solver for dynamic analysis of multibody systems. ReDySim delves upon the decoupled natural orthogonal complement approach originally developed for serial-chain manipulators. In comparison to the commercially available software, dynamic analyses in ReDySim can be performed without creating solid model. The input parameters are specified in MATLAB environment. ReDySim has capability to incorporate any control algorithm with utmost ease. In this work, the capabilities of ReDySim for solving open-loop and closed-loop systems are shown by examples of robotic gripper, KUKA KR5 industrial manipulator and four-bar mechanism. ReDySim can be downloaded for free from http://www.redysim.co.nr and can be used almost instantly. [doi:10.1063/2.1206311]

Keywords ReDySim, multibody systems, dynamic modeling, recursive dynamics, DeNOC

Multibody dynamics find applications in robotics, automobile, aerospace and many other streams for analysis, simulation and control. It has evolved a lot in the last two decades and there is a huge scope of research for scientists and engineers. Computer-aided dynamic analysis of multibody systems has been the prime motive of engineers since evolution of high speed facilities using computers. Dynamic analysis involves force and motion analyses. Force or inverse dynamics analysis attempts to find the driving and reactive forces for the given input motion, whereas the motion analysis or forward dynamics, obtains system's configuration under the input forces. Force analysis helps in design and control of multibody systems, whereas motion analysis allows one to study and test a design virtually without really building a real prototype. An efficient framework is essential for the dynamic analysis of complex multibody systems.

There are several software and toolboxes available for simulating multibody systems. They are in the form of toolkit for MATLAB[1,2] and LabVIEW.[3] The commercially available tools such as ADAMS[4] and RecurDyn[5] are the general purpose software used for dynamic analysis. These software are well suited for some standard industrial applications, however, they fail to attract research community mainly due to their inflexibility at user's end for solving complex problems. The generic nature of this software many times compromise with accuracy of the results even for smaller system for longer simulation time.

In this context, recursive dynamics algorithms play an important role. They are attractive due to simplicity and computational uniformity regardless of ever growing complex multibody systems. Recursive formulations have significantly helped in achieving real-time computations and model-based control laws. ReDySim[6–8] is a MATLAB-based solver, which is a general purpose platform, essentially consisting of very efficient recursive order (n) inverse and forward dynamics algorithms for simulation and control of a tree-type system. ReDySim delves upon the decoupled natural orthogonal complement (DeNOC) approach[9] originally developed for serial-chain manipulators. The specialty of this solver exists in its recursive nature and flexibility in solving complex problems. The gain in computational time is more as the number of links and joints in the system increases.

This paper mainly addresses analysis of fixed-base open- and closed-loop systems using ReDySim. The floating-base module of RedySim is also available on http://www.redysim.co.nr, and can be used for analyzing legged robots[7] and space robots with mobile or free base. Details of floating-base module are not given due to space limitation.

Even though, the algorithms in ReDySim are meant for tree-type systems as shown in Fig. 1, it can effectively be used to solve closed-loop systems by simply providing constrained Jacobian matrix resulting out of loop-closure equations and letting ReDySim know how a closed loop system is cut-open. The flow chart showing inverse and forward dynamics algorithms[6] of the tree-type systems are given in Fig. 2. The physical configuration of the system is mainly defined by the Denavit-Hartenberg (DH)[10] parameters, as proposed by Khalil and Kleinfinger.[11] The user inputs for both the inverse and forward dynamics are discussed next.

In order to perform inverse dynamics, the following input parameters are required:

Model parameters:

(1) number of links (n_l);

(2) type of system (type), i.e., open-loop or closed-loop;

(3) degrees-of-freedom (dof) of the system;

[a)] Corresponding author. Email: surilvshah@gmail.com.
[b)] Email: paramanandn@gmail.com.
[c)] Email: saha@mech.iitd.ac.in.

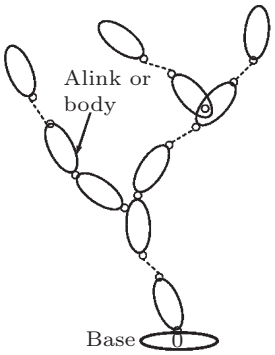

Fig. 1. Tree-type system.

(4) actuated joint in the system (a_{j});

(5) vector containing number of joint variables associated with each joint (n_{j});

(6) constant Denavit-Hartenberg (DH) parameters for revolute joints (a, α and b);

(7) parent of each link (β);

(8) vector $\boldsymbol{d}_k$ measured from origin O_k to the center-of-mass (COM) C_k of the kth link;

(9) mass of each link (m_k);

(10) vector of gravitational acceleration ($\boldsymbol{g}$) in the inertial frame;

(11) inertia tensor of each link about COM and represented in body-fixed frame ($\boldsymbol{I}_k^{\mathrm{C}}$);

(12) time period (T_{p}) and step size (dt).

The above input parameters are entered in the function file named inputs.m. The joint torques are then obtained by calling protected function file runinv.p. User may call function file plot_tor.m to see the output torques. The function file runinv.p also calculates Lagrange multipliers at the cut joints for closed-loop systems.

In order to generate inverse dynamics results, cycloidal joint trajectory is chosen as default. However a user can define any trajectory through the function file trajectory.m. Using the defined trajectory, the calculated joint torque values are stored in data file tor.dat. The numerical results of joint trajectories are stored in traj.dat. One can also visualize input joint trajectory by using file animate.m. The file animate.m is not generic and need to be modified for animating different systems.

In order to solve the inverse dynamics of a closed-loop system, first the trajectories of independent joints are entered in the function trajectory.m whereas the relationship between independent and dependent joint trajectories and Jacobian matrix are entered in the function file inv_kine.m. The inv_kine.m file is specific to a given system and the user is required to modify it depending on the type of system to be analyzed.

In order to perform forward dynamics, input parameters are provided in the file inputs.m. These input parameters are nothing but the first eleven entities of model parameters, as discussed earlier. In addition to these input parameters, the following parameters are required for the purpose of integration:

(1) initial conditions $y_0 = [\boldsymbol{q}^{\mathrm{T}} \;\; \dot{\boldsymbol{q}}^{\mathrm{T}} \;\; E_{\mathrm{act}}]^{\mathrm{T}}$ where $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and E_{act} are initial joint positions, joint rates, and actuator energy, respectively;

(2) initial time (t_{i}) and final time (t_{f}) of simulation, and step size (dt);

(3) relative tolerance (r_{tol}) and absolute tolerance (a_{tol}), and the type of integrator, note that one may use either adaptive solver ode45 (for non-stiff problem) or ode15s (for stiff problem) or fixed step solver ode5 by specifying the index 0, 1 or 2, respectively.

These parameters are entered in the function file named initials.m. The joint torque on each joint, which are input for forced simulation, can be entered in the function file torque.m. The default value of torque at each joint is kept zero which lead to the free simulation. However, in the case of force simulation, the user can define proportional (P), proportional and derivative (PD) or model-based control law for torque input as the current time (t), number of links (n_{l}), and vectors of joint positions ($\boldsymbol{\theta}$) and joint velocities ($\dot{\boldsymbol{\theta}}$) are passed to the function torque.m. One can also integrate any user defined control algorithm in this function file. Dynamic simulation of closed-loop system also requires entering Jacobain and its time derivative in the function file jacobian.m. It is worth noting that the vectors of current joint angels ($\boldsymbol{\theta}$) and joint velocities ($\dot{\boldsymbol{\theta}}$) are passed as the inputs to this function and outputs are Jacobian matrix and its time derivative.

Finally, simulation is performed by running protected function file runfor.p. The output joint motions are stored in data file statevar.dat whereas the time history is stored in timevar.dat. The joint motions can be plotted by using function file plot_motion.m. The function file for_kine.m can be used to calculate the position of the link origin and COM, velocity, angular velocity and the tip position. Moreover, the total energy can be calculated by running the file energy.p and plotted using plot_en.m. This can be used for the purpose of validating the simulation results. The system can also be animated using the file animate.m.

Dynamic analyses of serial, tree-type and closed-loop systems are presented next using ReDySim. The planar 4-dof robotic gripper and a spatial 6-dof industrial manipulator KUKA KR5[12] are selected as open-loop systems, whereas a 1-dof planar four-bar mechanism is selected as an example of closed-loop system.

A tree-type robotic gripper, as shown in Fig. 3, can hold objects to be manipulated by a robotic manipulator. Numerical results for inverse dynamics, i.e., to find the joint torques for a given set of input motions, were obtained using the inverse dynamics module of the ReDySim. The detailed steps are shown in Fig 2(a) where the definitions of all variables are available in Ref. 6. The motion for each joint for this purpose was computed using

$$\theta(t) = \theta(0) + \frac{\theta(T) - \theta(0)}{T} \cdot \left[t - \frac{T}{2\pi} \sin\left(\frac{2\pi}{T} t \right) \right], \tag{1}$$

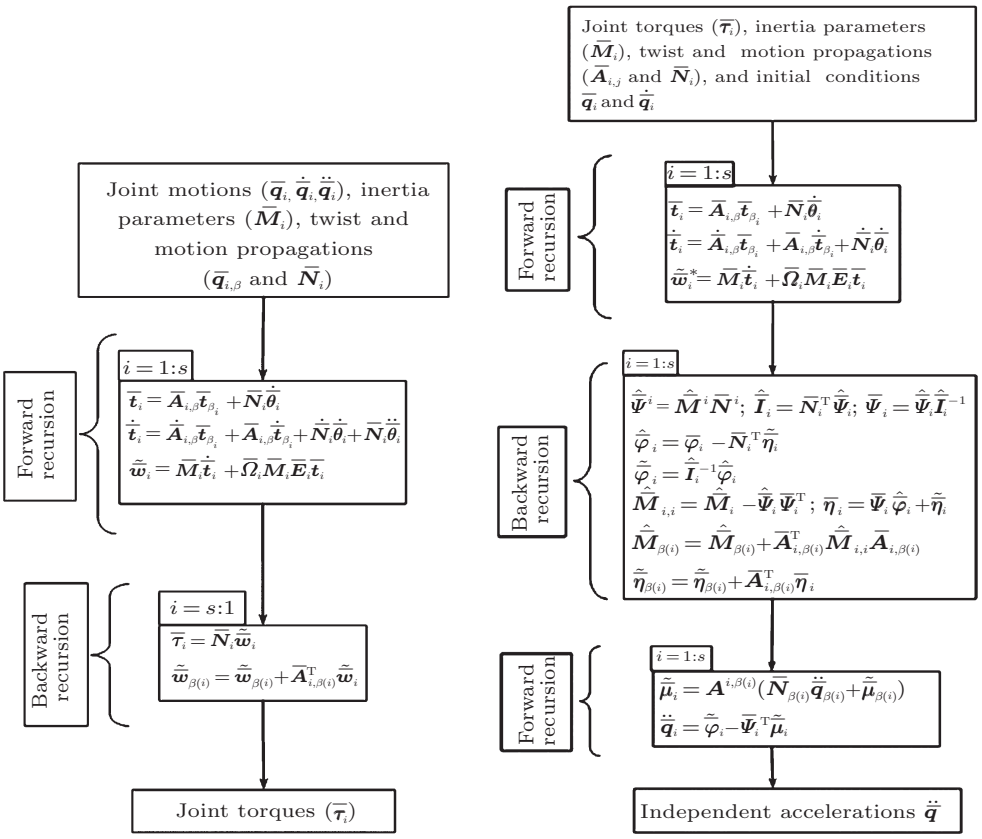

Fig. 2. Recursive dynamics algorithms.[6]

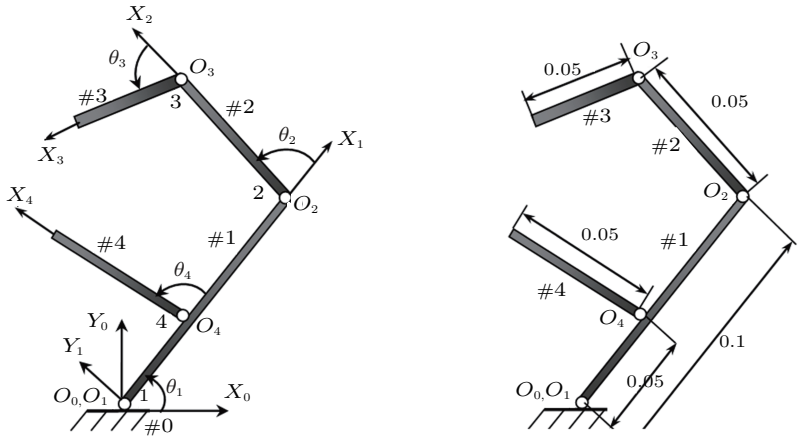

Fig. 3. Robotic gripper.

where $\theta(0)$ and $\theta(T)$ denote the initial and final joint angles, respectively, as given in Table 1. The joint trajectory in Eq. (1) is so chosen that the initial and final joint rates and accelerations for all the joints are zero. These ensure smooth joint motions. The lengths and masses of all links were taken as $l_1 = 0.1$ m, $l_2 = l_3 = l_4 = 0.05$ m, $m_1 = 0.4$ kg, and $m_2 = m_3 = m_4 = 0.2$ kg. The sample joint torques for the robotic gripper are then plotted in Fig. 4. In order to validate the results, a CAD model of the gripper was developed in ADAMS[4] software and used for the computation of the joint torques. The results were superimposed in Fig. 4, which show close match with the values obtained using the proposed inverse dynamics algorithm of the ReDySim. ReDySim took only 0.025 s on Intel T2300@1.66 GHz computing system. The ADAMS software, however, took 1.95 s, which is longer and is expected as it is general purpose software.

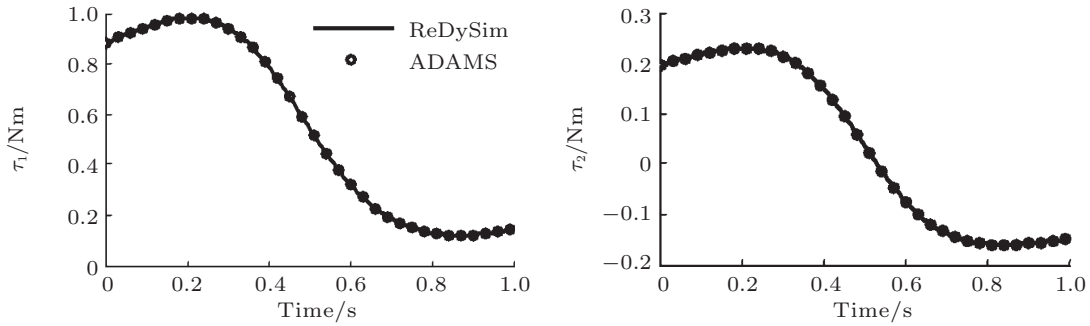

Fig. 4. Joint torques for robotic gripper.

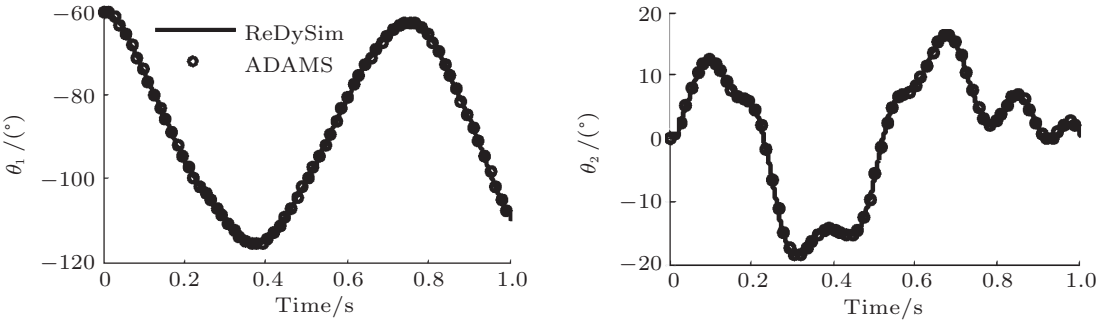

Fig. 5. Simulated Joint angles for robotic gripper (Joints 1 and 2).

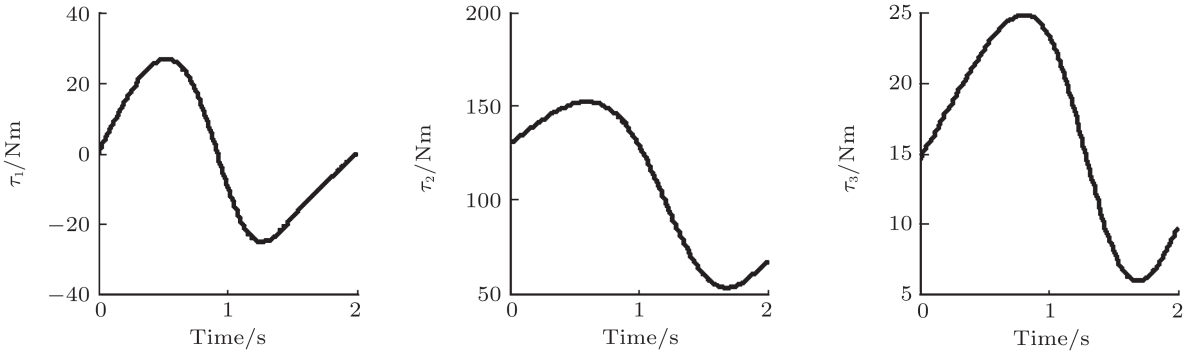

Fig. 6. Joint torques for KUKA KR5.

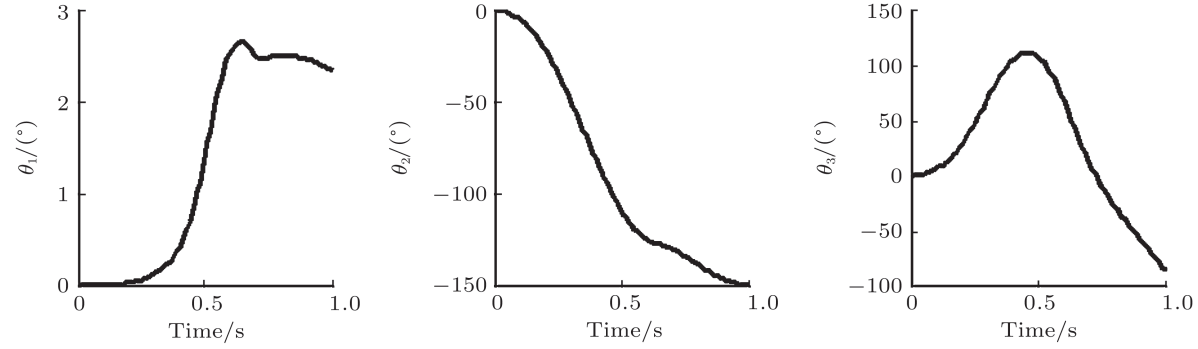

Fig. 7. Simulated joint angles of the KUKA KR5 (Joints 1, 2 and 3).

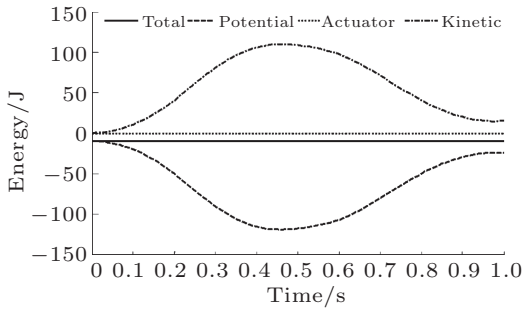

Fig. 8. Energy plot of KUKA KR5.

Table 1. Initial and final joint angles for robotic gripper.

Joints	1	2	3	4
$\theta(0)$	0°	0°	0°	90°
$\theta(T)$	60°	80°	80°	120°

The forward dynamics simulation of the robotic gripper was then carried out using forward dynamics module of ReDySim. Numerical results for the acceleration were obtained for the free-fall of the gripper, i.e., it was left to move under gravity without any external torques applied at the joints. The accelerations were then numerically integrated twice using the ordinary differential equation (ODE) solver ode45 (default in ReDySim) of MATLAB. The ode45 solver is based on the explicit Runge-Kutta formula given in Dormand and Prince.[13] The initial joint angles and rates were taken as $\theta_1 = -60°$, $\theta_2 = \theta_3 = \theta_4 = 0°$, and $\dot{\theta}_1 = \dot{\theta}_2 = \dot{\theta}_3 = \dot{\theta}_4 = 0$ rad/s respectively. Figure 5 shows the comparison of the simulated joint angles and the same obtained in ADAMS (by using RKF45 solver) over the time duration of 1 s with the step size of 0.001 s. The results obtained using ReDySim is in close match with those obtained using ADAMS. The ReDySim took 0.29 s in contrast to 39 s required by ADAMS.

Dynamic analysis of a spatial 6-dof industrial manipulator KUKA KR5 was performed next. To study the dynamic behavior of the KUKA KR5, its kinematic architecture[12] was used with user-defined mass and inertia properties. The DH parameters considered for KUKA KR5 and its assumed mass and inertia properties are given in the Table 2.

First, inverse dynamics of the KUKA KR5 robot was performed for initial and final joint angles of all the six joints as 0° and 60°, respectively. Once again cycloidal trajectory of Eq. (1) was used as the input joint motions. The joint torques at first three joints are shown in Fig. 6. Note that the torque required at joint 1 is zero at the beginning and at the end, as joint 1 is not affected by the gravitational acceleration. Moreover, the torque requirement at joint 2 is maximum, which is very obvious as this joint is required to overcome the effect of the gravity on links 2, 3, 4, 5 and 6.

Next, forward dynamics of the KUKA KR5 robot was performed for the free-fall under gravity. For this, both initial joint angles and velocities are assumed to be zero. Figure 7 shows the plot of joint angles for the time period of 1 s. The law of conservation of energy was used to validate the simulated results. Since there is no dissipation in the system, the total energy is plotted in Fig. 8, which remained unchanged throughout the simulation time. Hence, the simulation results are validated.

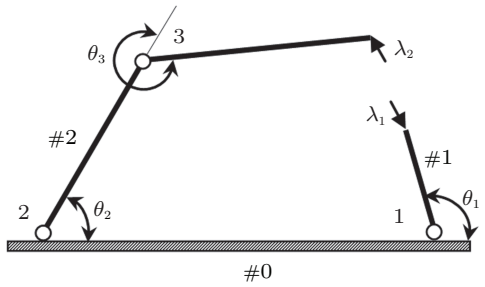

Fig. 9. A four-bar mechanism.

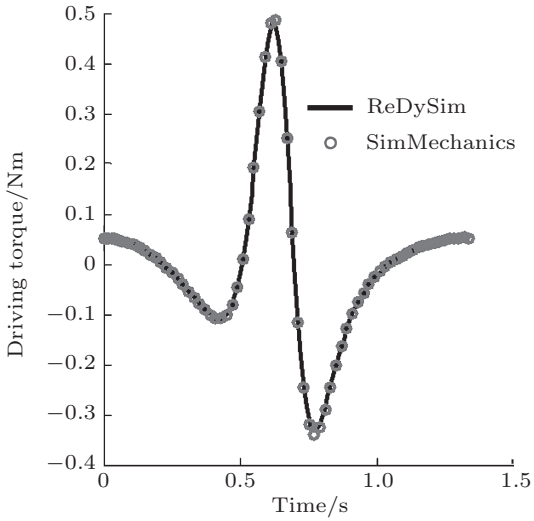

Fig. 10. Inverse dynamics results for four-bar mechanism.

Dynamic analysis of a closed-loop planar four-bar mechanism was carried out next. First, inverse dynamics was carried out using ReDySim's inverse dynamics module. The link length of crank, output link, coupler and fixed base were taken as 0.038 m, 0.115 2 m, 0.115 2 m and 0.089 5 m respectively. Also, the mass of the crank, output link and coupler were taken as 1.5 kg, 3 kg and 5 kg respectively. Inertia tensor of each link about its COM was calculated by assuming each link as a slender rod. In order to solve the four-bar mechanism, it was cut at an appropriate joint, as shown in Fig. 9, to form a tree-type system.[14] The opened joint was substituted with suitable constraint forces (λ) known as Lagrange multipliers. For this purpose, the Jocobian matrix was provided in inv_kine.m function file. The constant angular velocity of 4.712 4 rad/s was provided as the input to the joint 1 of the four-bar mechanism. The constant angular velocity value was provided in

Table 2. DH parameters and inertia properties of KUKA KR5.

i	a_i/m	$\alpha_i/(^\circ)$	b_i/m	$\theta_i/(^\circ)$	m_i/kg	$I_{i,xx}/(\mathrm{kg}\cdot\mathrm{m}^{-2})$	$I_{i,yy}/(\mathrm{kg}\cdot\mathrm{m}^{-2})$	$I_{i,zz}/(\mathrm{kg}\cdot\mathrm{m}^{-2})$
1	0	0	0.400	θ_1	16.038	0.176 9	0.266 5	0.262 2
2	0.180	90	0	θ_2	7.988	0.271 6	0.276 8	0.021 8
3	0.600	0	0	θ_3	12.937	0.388 9	0.376 5	0.104 1
4	0.120	90	0.620	θ_4	2.051	0.004 7	0.010 1	0.012 1
5	0	–90	0	θ_5	0.811	0.000 7	0.001 7	0.001 8
6	0	90	0	θ_6	0.008	0.000 003	0.000 001	0.000 001

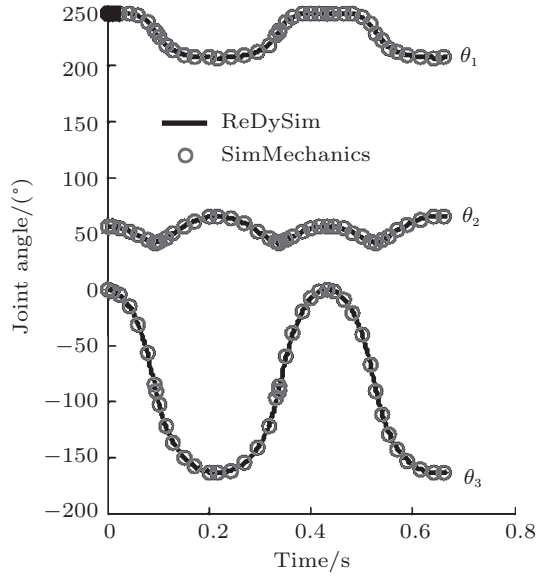

Fig. 11. Simulated joint motions for the four-bar mechanism.

the function file trajectory.m. The inverse dynamics was carried out for the time period of 1.33 s. The joint torque at joint 1 was calculated by executing function file runinv.p. The joint torque at joint 1 is plotted in Fig. 10.

Next, simulation of the four-bar mechanism was carried out using ReDySim's forward dynamics module. The simulation was done for free-fall under gravity for time period of 0.68 s without any external torque applied at the joint 1. The Jacobian matrix was provided in *jocobian.m* function file. By executing function file runfor.p, the joint angles were calculated, which are shown in Fig. 11.

The results of the ReDySim were validated with MATLAB's SimMechanics as shown by circular markers in Figs. 10 and 11.

An efficient Recursive Dynamics Simulator (ReDySim) has been developed in MATLAB for analyses of multibody systems. Capability of the ReDySim is shown for serial, tree-type and closed-loop systems whose results are validated with the other commercial software. The ability of ReDySim in including desired trajectory and control law, provide flexibility to researcher in incorporating their customized algorithms. The control aspect is not reported here due to space limitation and will be communicated in future. ReDySim does not require building model in the software environment before simulation; users can simply provide the input parameters in the MATLAB environment. ReDySim showed considerable improvement over commercial software such as ADAMS and algorithms available in literature in terms of the computational time. ReDySim can be downloaded free from http://www.redysim.co.nr. The users are encouraged to send their comments and suggestions to redysim@gmail.com or the authors.

1. P. I. Corke, Robot. Automat. Mag. IEEE **3**, 24 (1996).
2. M. Toz, and S. Kucuk, Comput. Appl. Eng. Edu. **18**, 319 (2010).
3. T. J. Mateo Sanguino, and J. M. A. Márquez, Simulation Tool for Teaching and Learning 3D Kinematics Workspaces of Serial Robotic Arms with up to 5-dof. Computer Applications in Engineering Education, 2010.
4. Automated Dynamic Analysis of Mechanical System (ADAMS), Version 2005.0.0, MSC Software (2004).
5. RecurDyn. FunctionBay Inc. 2009.
6. S. V. Shah, Modular framework for dynamic modeling and analysis of tree-type robotic systems [PhD Thesis]. (Indian Institute of Technology Delhi, Delhi, 2011).
7. S. V. Shah, S. K. Saha, and J. K. Dutt, ASME J. Nonlinear Computat. Dyn. **7**, (2012).
8. S. V.Shah, S. K. Saha, and J. K. Dutt, Mech. Mach. Theory **49**, 234 (2012).
9. S. K. Saha, ASME J. Appl. Mech. **66**, 986 (1999).
10. J. Denavit, and R. S. Hartenberg, ASME J. Appl. Mech. 215 (1955).
11. W. Khalil, and J. Kleinfinger, IEEE Int. Conf. on Robotics and Automation, 1174 1986.
12. KUKA KR5, Technical specification, KUKA Robotics website, http://www.kuka.com, last accessed on 20 May 2012.
13. J. R. Dormand, and P. J. Prince, J. Comput. Appl. Math. **6**, 19 (1980).
14. H. Chaudhary, and S. K. Saha, ASME J. Mech. Design **129**, 1234 (2007).