Recursive Dynamic Simulator (ReDySim): Floating-base (Legged-robots) Instruction Manual for Inverse Dynamics

Getting started

1. Require MATLAB 2009a or higher version in order to use this module.

Run demo of a Quadruped robots

1. Run function file *run_me.m*. This will perform inverse dynamics of the quadruped robot and generate the plots for the input joint motions and joint torques.
2. Run file *animate.m* in order to animate the system using the simulation data.
3. User may also copy all the files from the folder *biped* or *hexapod* in the main folder containing the file *run_me.m* in order to perform inverse dynamics of eiterh of them.

Inverse dynamics using *run_me.m*

1. Function prototype: `run_me()`
2. Enter the input parameters such as type modified-DH parameters (See appendix A), inertia tensors, masses etc., in the file *inputs.m* and save the file.
3. Enter the initial conditions and integration parameters such as initial state variable, integration tolerances, etc. in the file *initials.m* and save the file.
4. Enter joint trajectories in the file trajectory.m.
5. Ensure that the parameters are properly entered in the files *inputs.m*, *initials.m*, and *trajectory.m*.
6. Run file *run_me.m* in order to perform inverse dynamics and to generate the plots for joint torques, input joint motions, base motion, energy and momentum. It actually runs *runinv.p* (for inverse dynamics), *motionplot.m* (for plotting input joint motion and base motion), *torplot.m* (for plotting joint torques), *energy.p* (for energy calculation), and *ploten.m* (for energy plot).
7. Ensure that both the *timevar.dat* and *statevar.dat* files are in the program folder containing file *animate.m*. Run file *animate.m* in order to animate system using the simulation data. This file is specific to the system under study and hence need to be modified depending on the system or mechanism.

Details of the functions used in the simulation above

1. Input functions

These function files are required as the input.

inputs.m

- Function prototype: `[n dof type alp a b bt dx dy dz m g Icxx Icyy Iczz Icxy Icyz Iczx aj T steps]=inputs()`
- The number of links (n), Degrees-of-Freddom of the system (dof), type of system (type), i.e., closed or open , model parameters such as modified DH parameters (alp, a, b), parent of each

link (bt), the X, Y and Z coordinates of position of Center-of-Mass (COM) form the origin (dx, dy, dz), masses (m), vector of gravitational acceleration (g), elements of inertia tensors about COM (Icxx, Icyy, Iczz, Icxy, Icyz, Iczx), details of actuated joints (aj), time period (T) and time steps (steps) are provided in this file.

trajectory.m

- Function prototype: `[thi dthi ddthi]= trajectory(tim)`
- The trajectories of independent joints are provided in this function. The independent joint positions, velocities and accelerations are output of this function.
- Note that all the joint motions are independent in the case of open-loop systems, where, closed-loop systems have independent joint motions equal to it degree-of-freedom.

inv_kine.m

Function prototype: `[th dth ddth J]=inv_kine(thi, dthi, ddthi)`

- The relations between independent and dependent joint angles, and jacobian are provided in this function for the closed-loop system. This function is not required for open-loop systems.
- Output of this function is vector of independent and dependent joint position (th), velocity (dth) and acceleration (ddth) and jacobian (J).

2. Output function

This file generates output in the forms of either data files or plots.

runinv.m

- Function prototype: `runinv()`
- The file *runinv.m* performs inverse dynamics and generate output data files *torvar.dat* and *timevar.dat statevar.dat* containing joint torques and Lagrange multipliers at the cut joints (for closed-loop), time variables, and state variable, respectively, the give time period.
- First *n*, *n* being number of link in the open-type system, columns in file *statevar.dat* contain positions associated with joint variables, and next *n*, i.e., (*n*+1) to (2*n*), columns contain time rates, i.e., velocities, associated with joint variables, and final *n* columns contain joint accelerations.

plot_motion.m

- Function prototype: `plot_motion()`
- The file *motionplot.m* show the plots containing input joint positions, velocities, and accelerations for both independent and dependent joints using data files *timevar.dat* and *statevar.dat*.

plot_tor.m

- Function prototype: `plot_tor()`
- The file *plot_tor.m* plots the joint torques and Lagrange multipliers (in the case of closed-loop systems only) .

animate.m

- Function prototype: `animate()`
- This function file animate the system using the simulation data. This file is specific to the system under study and hence need to be modified depending on the system or mechanism. It uses function *for_kine.m* to obtained Cartesian co-ordinate of the point on a link from joint angles saved in data file statevar.dat using forward kinematics relationships. User has full access to both the *animate.m* and *for_kine.m*.

Note: All the above files can be run by using *run_me.m* or independently in the order shown above.

3 Some other important functions

Some other important functions are outline below:

invdyn_tree_eff.m

- Function prototype: `[tu] = invdyn_tree_eff(th,dth,ddth,n,alp,a,b,bt,dx,dy,dz,m,g,Icxx,Icyy,Iczz,Icxy,Icyz,Iczx)`
- The function is used to calculate joint torques (tu) for open loop systems. For closed-loop systems it provides sum of inertia and bias forces.

for_kine.m

- Function prototype: `[so sc vc w st]=for_kine(th,dth,n,alp,a,b,bt,dx,dy,dz)`
- This function calculates position of the link origin (so) and COM (sc), velocity of the COM (vc) of the link and angular velocity (w) of the link and the tip position (st).
- As users have access to this function, other points on the link can also be taken as output depending on the system.

Note:

Users have full access to functions *inputs.m*, *inv_kine.m*, *plot_tor.m*, *plot_motion.m*, *for_kine.m* and *animate.m* whereas the rest are protected codes (pcodes) which can only be used as function.