Modelica-Based Control of a Delta Robot

Scott A. Bortoff

bortoff@merl.com

Mitsubishi Electric Research Laboratories, USA

This presentation describes the author's research to develop a delta robot that is useful for precise robotic assembly. A delta robot is a closed-chain, parallel link mechanism, and is used in industry almost exclusively for pick and place operations, because of its low cost and high speed. These applications require simple PID type control, which makes the robot very precise in terms of its position control, but also very stiff. However, the delta robot has mechanical characteristics that make it very attractive for use in more complex assembly operations, involving contact and collision, where high robot stiffness is not desirable. These characteristics include its low mass (because actuators are mounted at the base), high link stiffness, high mechanical precision and low joint friction (all because of its mechanical design), and low capital cost (because of its symmetry, giving a low parts count). Assembly applications require point to point motion control and also feature contact and collision between the robot end effector and objects it seeks to manipulate. Conventional robots move very slowly in these applications, so they do not transfer excessive energy or momentum, implying the dynamics of contact and collision can be neglected. To exploit the delta robot's low mass and high speed, the dynamics of contact and collision need to be considered. In these situations, it is desirable to modulate the robot impedance. This requires more sophisticated multivariable control algorithms.

Modelica is the ideal (and arguably the unique) language capable of modeling and simulating not only the robot dynamics, but also an assembly task including contact and collision, in addition to the control algorithms, ranging from low level feedback to higher level task management. The robot itself is a closed-chain manipulator, which can be represented as a set of high-index set of differential-algebraic equations (DAEs).

These are readily expressed in Modelica. Unlike serial-link robotic manipulators, the delta robot has two distinct Jacobians. One of these is useful for simulation of the effects of forces applied to the end effector on robot motion (and vis versa), and the other useful for control. There are good reasons for the distinction, which will be described. Impedance control can be achieved by feedback linearizing the robot dynamics. Unlike serial-link robotic manipulators, the feedback linearization must be computed algorithmically (iteratively). This is because the delta robot forward kinematics cannot be expressed analytically, but can be computed algorithmically. This implies that the control law must be expressed in discrete-time. Modelica's algorithm blocks, support of functions and arrays etc. allow for rapid realization of control algorithms, and its synchronous features allow for correct representation and simulation of the resulting closed-loop sampled data system, which is a feedback interconnection of the continuous-time robot dynamics (a high-index DAE) and a discrete-time controller.

The presentation will also describe efforts to model and simulate object contact and collision using a hybrid DAE modeling framework, which combines a finite state machine and continuous, high-index DAE. Again, the Modelica language, with its hybrid DAE model of computation, is ideal for representing contact and collision, and results in a non-stiff set of DAEs that does not suffer from odd behaviors that are associated with alternative methods of representing contact and collision, such as impulsive methods and those that are based on complementary formulations.

Finally, the presentation will show real-time control of a laboratory delta robot directly from the Modelica development environment. Using the device drivers library, with suitable but relatively straightforward modifications, we may take models of the control used for simulation, disconnect the dynamic model of the robot, plug in the experiment's inputs and outputs, recompile and instantly have a closed-loop laboratory experiment. Although this has some limitations, in part because the Modelica development environment runs as a single process and thread, it is an effective way to move from simulation to experiment without recoding. As such, Modelica should be considered as a platform not only for desktop modeling and simulation, but also for control system represen-tation (as the algorithms are developed throughout the lifecycle of a project) and also laboratory experiment. This may be particularly of interest to similar research laboratories and educational organizations.