DARTS - Multibody Modeling, Simulation and Analysis Software

Abhinandan Jain(✉)

Jet Propulsion Laboratory, California Institute of Technology,
4800 Oak Grove Drive, Pasadena, CA 91109, USA
jain@jpl.nasa.gov

Abstract. This paper describes the *Dynamics Algorithms for Real-Time Simulation (DARTS)* software for multibody dynamics modeling, analysis and simulation. DARTS is in use for closed-loop simulation for aerospace, ground vehicle, robotics applications and large, multi-scale molecular dynamics applications. DARTS is designed for high-fidelity multibody dynamics, fast computational speed, to handle run-time configuration changes, and to provide a broad family of computational algorithms for analysis and model based control. This paper describes DARTS capabilities, novel aspects of its architecture and design, and application examples.

1 Introduction

This paper describes the current generation of the *Dynamics Algorithms for Real-Time Simulation (DARTS)* software for multibody dynamics modeling, analysis and simulation [1]. DARTS adopts a minimal coordinates approach similar to the SimBody [2] and RBDL [3] tools for solving the equations of motion for time domain simulations. However the goals of DARTS are broader. It is designed to handle rigid/flexible multibody dynamics, arbitrary system topologies, smooth and non-smooth dynamics, run-time configuration changes, and also provide a full complement of computational algorithms needed for dynamics analysis and model based control with fast computational performance. While the DARTS object-oriented implementation is in C++, a rich Python interface is available for all the classes and methods in the system. This allows users full flexibility in defining and configuring the model as desired and to even modify the model topology and properties during run-time. DARTS is in use for the dynamics simulation for aerospace, ground vehicle, robotics and multi-scale molecular dynamics applications [4]. Section 2 describes the dynamics formulation approach, Sect. 3 the software architecture and key features, and Sect. 4 describes application examples. Due to space limitations, this paper limits itself to an overview of key aspects of DARTS, with details to follow in an expanded future publication.

A. Kecskeméthy and F. Geu Flores (Eds.): ECCOMAS 2019, COMPUTMETHODS 53, pp. 433–441, 2020.
https://doi.org/10.1007/978-3-030-23132-3_52

2 Structure-Based Dynamics

DARTS algorithms are based on minimal coordinates dynamics formulations. Minimal coordinate dynamics reduce, if not avoid, the need for explicit constraints, DAE integration methods and constraint stabilization techniques. DARTS is based on the *Spatial Operator Algebra (SOA)* [5] methodology which uses mathematical operator techniques to exploit the structure of minimal coordinate models for analysis and to develop low-cost recursive computational algorithms. Using SOA, analytical operator expressions can be derived for the factorization of the mass matrix and its inverse for arbitrary tree multibody systems. These expressions form the basis for a broad range of low-cost computational algorithms including the well known, $O(\mathcal{N})$ recursive methods for the low-cost solution of the equations of motion for tree-topology systems [5,6]. These algorithms are structure-based and consist of scatter/gather recursions that proceed across the bodies in the system topology. This property allows DARTS to easily handle run-time structural changes in the system topology such as from the attachment/detachment and addition/deletion of bodies. Such structural changes are common in aerospace separation and deployment scenarios, during robotics manipulation and in model coarsening strategies for large-scale molecular dynamics simulations. The algorithms accommodate such changes with recursions simply following the new system topology. DARTS supports general multibody system models including:

Serial/tree rigid/flexible systems: DARTS includes $O(\mathcal{N})$ *articulated-body inertia (ABI)* recursive method for solving the equations of motion of arbitrary tree-topology multibody systems [5,7]. The computational cost of these methods scales linearly with the number of bodies. SOA shows that these structure-based recursive methods for rigid multibody systems directly extend to flexible bodies as well. While some low level computational details change, the overall forward dynamics algorithm structure remains the same. The algorithm implemented within DARTS support rigid and flexible body dynamics. Assumed mode, small deformation models are used for flexible bodies.

Closed-chain topology systems: Recursive methods are not directly usable for system topologies that include loops. One approach is to introduce loop-cuts to obtain a spanning tree. The low-cost $O(\mathcal{N})$ recursive method are used for the spanning tree dynamics, followed by additional steps to correct the solution for the loop-cuts. Within DARTS, this *tree-augmented (TA)* method also uses a SOA-based low-cost method for computing the operational space inertias needed for the loop-cuts during the correction phase [8]. An alternative to the non-minimal coordinate TA approach is the SOA-based *constraint embedding (CE)* technique for closed-loop systems. The CE method transforms the graph-topology systems into a minimal coordinate tree-topology system using compound, variable-geometry bodies. Unlike projection methods, the transformation is structure-preserving so that the $O(\mathcal{N})$ ABI recursive methods and ODE integrators can be used even for closed-loop systems. Both the CE and TA dynamics solution methods are available within DARTS.

Contact/collision dynamics: For non-smooth dynamics involving contact and collisions, DARTS includes support for penalty-based methods. Also available are impulsive complementarity based solution methods that exploit minimal coordinate to reduce the size of the complementarity problem to the number of loop-cut and unilateral contact constraints [9]. Both linear and non-linear complementarity solution algorithms are available in DARTS [10].

3 Architectural Features

Frames layer: The set of *frames* of interest within a simulation can easily number in the hundreds or more. DARTS includes general purpose frame transforms layer that supports the on demand computation of relative pose, velocity and accelerations for any pair of frames in the system. This layer removes the need for special purpose methods for specific frame pair properties. A notable feature is that the values are computed only when requested to avoid the cost of keeping the frames tree fully updated. Transform values are cached and reused, and recomputed only when they are no longer current. The frames layer allows reattachment, as well as the creation and deletion of frames at run-time.

Multibody elements: The key elements of a multibody model are the set of *bodies*, connector *hinges* and *nodes* (physical points of interest on bodies). Within DARTS, each of these entities are distinct C++ objects with associated data and methods. Body and node classes are derived from the frame base class. In addition to supporting a full set of hinge types, a *custom* hinge type allows users to go beyond the pre-defined types and create hinges consisting of an arbitrary sequence of articulation degrees of freedom. Bilateral loop constraints are defined via a hinge that specifies the permissible constrained motion.

Structural changes: DARTS allows run-time structural changes to the system topology such as from the detachment and reattachment of bodies. Also bodies can be deleted and added on the fly. These features are needed for robotics manipulation scenarios, and for events such as heat-shield separation and deployment in aerospace applications. The structure-based SOA algorithms continue to work by simply switching to the new multibody system topology.

Subgraphs: While dynamics computations are usually for the full multibody system, the SOA structure-based algorithms can be limited to connected *subgraphs* of bodies within the system. Important examples of such subgraphs are individual vehicles and robot limbs arms. Within DARTS, the full multibody system is itself a subgraph containing all the bodies. Virtually all of the computational algorithms (including forward dynamics) can be invoked on just the bodies in a subgraph - while ignoring all external bodies. Maintaining the distinction between the multibody model and the computational subgraph has other important benefits. The CE method for closed-loop systems relies on a transformation of the multibody graph into a tree topology system. Within DARTS, the transformed graph is just another subgraph, and using

the CE method consists of simply invoking the forward dynamics algorithm on this transformed subgraph.

Python interface: DARTS uses the open-source SWIG tool [11] to auto-generate a comprehensive Python interface for all the C++ classes and their methods. The Python interface allows users to conveniently set up simulations using Python scripts without losing the speed benefits from the compiled C++ layer. The comprehensive Python interface gives users a high degree of flexibility in tailoring the simulation to individual simulation needs and interacting with it during run-time.

Dynamics solvers and integrators: As discussed above, DARTS supports multiple methods for solving the equations of motion. While the $O(\mathcal{N})$ ABI algorithm can be used for tree-topology forward dynamics, the TA algorithm requires a different procedure since it uses non-minimal coordinates. Only integration schemes compatible with the selected solution method can be used. Thus while a minimal coordinates solution method can be used with (any) ODE integrator, a non-minimal coordinates method requires a DAE solver or a Baumgarte like constraint error management scheme. DARTS includes solver C++ classes to support the pairing of a solution method with a compatible numerical integrator.

Visualization: DARTS includes 3D graphics visualization of the multibody model. Even without any graphics data, there is a built in capability to generate a stick figure 3D graphics model that can be animated and articulated. This is useful for debugging and adjusting the model during the early model definition stages. In addition, primitive geometry shapes and CAD parts can be attached to nodes on the bodies to provide a richer graphical representation. Different graphics engines (e.g. Ogre3D, Blender) can be used to render the graphics.

DARTS also includes features such as built in nonlinear solvers for solving loop constraints and state initialization, methods for linearizing the dynamics model for use in control design and analysis, and GUIs for browsing, adjusting and interactively animating the model.

3.1 Computational Algorithms and Features

In addition to the forward dynamics solution methods described above, DARTS includes several other algorithms and features with key ones listed below:

Jacobians: The Jacobian matrix is commonly used in robotics. It maps generalized velocities into the spatial velocity of frames attached to the mechanism. DARTS provides methods to compute the Jacobian matrix map from any subset of degrees of freedom to any set of frames attached to the multibody system.

Hybrid dynamics: The forward dynamics problem solves for the generalized accelerations resulting from specified generalized forces on the system. DARTS supports a more general $O(\mathcal{N})$ recursive hybrid dynamics algorithm

where the inputs for any subset of the degrees of freedom can instead be generalized accelerations, and the algorithm will compute the unknown generalized forces for these degrees of freedom. Specifying a degree of freedom to be of *prescribed* type reverses the input/output computation for a degree of freedom. The hybrid algorithm reduces to the standard O($\mathcal{N}$) ABI forward dynamics algorithm when none of the degrees of freedom are prescribed, and to the O($\mathcal{N}$) Newton-Euler inverse dynamics algorithm when all degrees of freedom are prescribed. The prescribed property of a degree of freedom can be changed on the fly.

Operational space inertia: For a body node, the *operational space inertia (OSC)* matrix represents the mass matrix of the system reflected to the node. This is an important quantity for task-space and whole-body motion control in robotics, as well for the TA closed-chain dynamics solution method. The SOA algorithms provide a low-cost recursive algorithm for computing OSC matrices [5,12], and DARTS uses it for computing the OSC for any set of nodes specified by the user.

Composite body inertias, kinetic energy, momentum: DARTS includes methods for computing the configuration dependent *composite body*, i.e. combined spatial inertia, momentum and kinetic energy properties for any sub-set of bodies in the system. These methods are useful for computing the properties of individual vehicles, robotic arms etc.

CM frame: In certain applications, it is important to keep track of the center of mass (CM) location of a subgraph. DARTS automatically creates a CM frame for each subgraph that tracks the CM location of the bodies as the configuration evolves with time. This allows users to use the standard frame level transform methods to easily track the location of the CM for any subgraph.

Interbody forces: The recursive O($\mathcal{N}$) ABI forward dynamics algorithms do not require, nor compute, interbody spatial forces. However there are times when these force values are needed. The SOA algorithm provides an inexpensive expression for computing this value for any pair of connected bodies [13] that is implemented within DARTS.

Pruning bodies: The bodies within a complex multibody model typically match the structure of the mass property and kinematics input deck for the system. This can result in more bodies than are essential to the dynamics computations, and add to the computational cost. DARTS allows users to prune the bodies in a multibody model by freezing and coalescing rigidly connected pairs of bodies.

DCA algorithm: The divide-and-conquer algorithm (DCA) is an alternative forward dynamics methods that is amenable to parallelization [14]. DARTS includes an implementation of this method for use with large multibody models (e.g. molecular dynamics).

Linearization: Since multibody dynamics models are inherently nonlinear, linearized dynamics models are often required for control system design and analysis. DARTS provides methods for automatically computing and exporting linearized models for multibody models.

Statistical dynamics: Statistical properties play a key role in molecular dynamics simulations. The use of multibody methods for such simulations requires generalizations of the classical *equipartition principle* to systems with constraints in order to distribute the thermal energy evenly across the coupled system degrees of freedom. DARTS implements SOA techniques for such a generalized equipartition principle [15]. Also, it is known that the use of multibody methods introduces biases in the statistical properties which can be overcome with the additional use of a *Fixman potential*. DARTS implements SOA methods for incorporating this potential within molecular dynamics simulations [16].

4 Applications

Engineering simulations include vehicle dynamics together with models for sensor/actuator devices, the environment and closing the loop with control software. For such simulation applications, DARTS serves as the vehicle dynamics module within the *DARTS Shell (Dshell)* component-based simulation framework [17]. Key application areas for DARTS and Dshell are described below.

Aerospace: The *DSENDS* tool is an adaptation of the DARTS/Dshell toolkit for spacecraft flight dynamics which uses component models for aerodynamics, engines, thrusters, gravity, fuel consumption, ephemerides etc. DSENDS is in use for closed-loop guidance and control for orbiters, landers and launch vehicle simulations for NASA missions such as Cassini, Mars Pathfinder, Mars Science Laboratory, Phoenix, InSight, Mars 2020, Space Launch System [18]. A recent application has been for rotorcraft simulation for the upcoming NASA Mars Helicopter technology demonstration mission [19].

Ground Vehicles: The *ROAMS* tool is an adaptation of the DARTS/Dshell toolkit for the simulation of autonomous ground vehicles such as NASA planetary rovers [20]. ROAMS includes models for vehicle suspensions, wheel/soil interaction, autonomy sensors such as cameras and lidars, and motion control software. ROAMS has also been used for simulating terrestrial vehicles such as the HMMWV, MRZR4 and other robot mobility platforms [21]. The constraint embedding methods have been especially useful for modeling the dynamics of the double wishbone and trailing arm suspensions in these vehicles.

Robotics and Embedded Use: DARTS has been used for simulating robotics systems such as manipulators and legged/humanoid mobile robots. Some of these applications use the contact/collision dynamics capabilities within DARTS. Another application area within robotics is its use as a modeling layer embedded within the robot control software. The large variety of algorithms supported by the SOA framework makes it especially suitable for addressing a broad and diverse set of model-based computations using low-cost recursive methods. This adaptation of DARTS, called *RoboDarts*, serves as a fast, cross-cutting versatile layer that computes model-based data for the control, execution, perception, estimation and planning modules within robot control software [22,23]. The queries can range from basic frame to frame pose transforms, gravity compensation computations, load balancing for multi-arm manipulation, trajectory

and grasp planning, and feed forward control inputs for gait management. The structure-based algorithms are also able to easily accommodate the time-varying configuration and constraints that are common in robotic tasks.

Computational Workbench: *PyCraft* is a multibody dynamics analysis computational workbench based on DARTS [24]. The goal of PyCraft is to provide an interactive environment for the numerical evaluation of operators and operator expressions from SOA theory. With DARTS providing the modeling layer, PyCraft implements C++ operator classes for all the key operators in SOA. Overloaded arithmetic operations for these operators are available for evaluating mathematical operator expressions from the command line. Thus for instance, operator expressions for the mass matrix and mass matrix inverse from SOA can be evaluated directly for any multibody model loaded into DARTS. The PyCraft computational workbench enables the easy evaluation and testing of results from SOA mathematical analysis. PyCraft supports the full range of computations provided by SOA including advanced techniques for computing sensitivities of operators and the mass matrix.

Molecular Dynamics: Another research application area for DARTS has been that of molecular dynamics simulations of bio-molecular systems. Conventional atom-level simulation methods suffer from small time steps dictated by the stiff bond stretching degrees of freedom. By eliminating these degrees of freedom, DARTS enables the use of larger simulation time steps. The *GNEIMO* [25] methods and software make use of DARTS for these applications and include multi-scale methods that significantly increase the simulation duration and sampling for these very large dynamical systems. Segments of the molecular model can be frozen and thawed on the fly to manage the coarseness of the model dynamics. The emphasis of molecular dynamics simulation is usually on statistical properties. Statistically correct initialization of the system state is based on an extension of the equipartition principle [15]. Furthermore, GNEIMO also supports the use of the Fixman potential for correcting statistical biases introduced by the internal constraints [16].

Acknowledgement. The research described in this paper was performed at the Jet Propulsion Laboratory (JPL), California Institute of Technology, under a contract with the National Aeronautics and Space Administration[1].

References

1. Jain, A., Man, G.K.: Real–time simulation of the Cassini spacecraft using DARTS: functional capabilities and the spatial algebra algorithm. In: 5th Annual Conference on Aerospace Computational Control. Jet Propulsion Laboratory, Pasadena, CA, August 1992
2. Sherman, M.A., Seth, A., Delp, S.L.: Simbody: multibody dynamics for biomedical research. Procedia IUTAM **2**, 241–261 (2011)
3. Felis, M.L.: RBDL: an efficient rigid-body dynamics library using recursive algorithms. Autonomous Robots, 1–17 (2016)

4. Dynamics and Real-Time Simulation (DARTS) Lab (2019). http://dartslab.jpl.nasa.gov/
5. Jain, A.: Robot and Multibody Dynamics: Analysis and Algorithms. Springer, Berlin (2011)
6. Featherstone, R.: Rigid Body Dynamics Algorithms. Springer, New York (2008)
7. Jain, A.: Unified formulation of dynamics for serial rigid multibody systems. J. Guidance Control Dyn. **14**(3), 531–542 (1991)
8. Jain, A., Crean, C., Kuo, C., Quadrelli, M.B.: Efficient constraint modeling for closed-chain dynamics. In: The 2nd Joint International Conference on Multibody System Dynamics, Stuttgart, Germany (2012)
9. Jain, A.: Contact dynamics formulation using minimal coordinates. In: Terze, Z. (ed.) Multibody Dynamics, pp. 93–121. Springer (2014)
10. Mylapilli, H., Jain, A.: Evaluation of complementarity techniques for minimal coordinate contact dynamics. In: Proceedings of the ASME Design Engineering Technical Conference, vol. 6 (2014)
11. Simplified Wrapper and Interface Generator (SWIG) (2019). http://swig.org/
12. Rodriguez, G., Jain, A., Kreutz-Delgado, K.: Spatial operator algebra for multibody system dynamics. J. Astronaut. Sci. **40**(1), 27–50 (1992)
13. Jain, A.: Computing inter-body constraint forces in recursive multibody dynamics. In: The 5th Joint International Conference on Multibody System Dynamics, Lisboa, Portugal (2018)
14. Featherstone, R.: A divide-and-conquer articulated-body algorithm for parallel O(log(n)) calculation of rigid-body dynamics. Part 1: basic algorithm. Int. J. Robot. Res. **18**(9), 867–875 (1999)
15. Jain, A., Park, I.-H., Vaidehi, N.: Equipartition principle for internal coordinate molecular dynamics. J. Chem. Theory Comput. **8**(8), 2581–2587 (2012)
16. Jain, A., Kandel, S., Wagner, J., Larsen, A.B., Vaidehi, N.: Fixman compensating potential for general branched molecules. J. Chem. Phys. **139**(24), 244103 (2013)
17. Cameron, J.M., Balaram, J., Jain, A., Kuo, C., Lim, C., Myint, S.: Next generation simulation framework for robotic and human space missions. In: AIAA SPACE Conference and Exposition 2012 (2012)
18. Cameron, J.M., et al.: DSENDS: multi-mission flight dynamics simulator for NASA missions. In: AIAA SPACE 2016, Long Beach, CA (2016)
19. Grip, H., et al.: Flight control system for NASA's mars Helicopter. In: Proceedings of the AIAA Science and Technology Forum and Exposition (2018)
20. Jain, A., Balaram, J., Cameron, J.M., Guineau, J., Lim, C., Pomerantz, M., Sohl, G.: Recent developments in the ROAMS planetary rover simulation environment. In: IEEE 2004 Aerospace Conference, Big Sky, Montana (2004)
21. Jain, A., Guineau, J., Lim, C., Lincoln, W., Pomerantz, M., Sohl, G., Steele, R.: Roams: planetary surface rover simulation environment. In: International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2003), Nara, Japan, May 2003
22. Jain, A.: Structure based modeling and computational architecture for robotic systems. In: 2013 IEEE International Conference on Robotics and Automation (2013)