Physics-based Modeling and Simulation of Planetary Missions at JPL

8th NAFEMS MBD-WG Webinar

April 24th, 2013

Marco B. Quadrelli, marco.b.quadrelli@jpl.nasa.gov

Dynamics and Real-Time Simulation Lab
Jet Propulsion Laboratory
California Institute of Technology

http://dartslab.jpl.nasa.gov

© 2013 California Institute of Technology. Government sponsorship acknowledged.

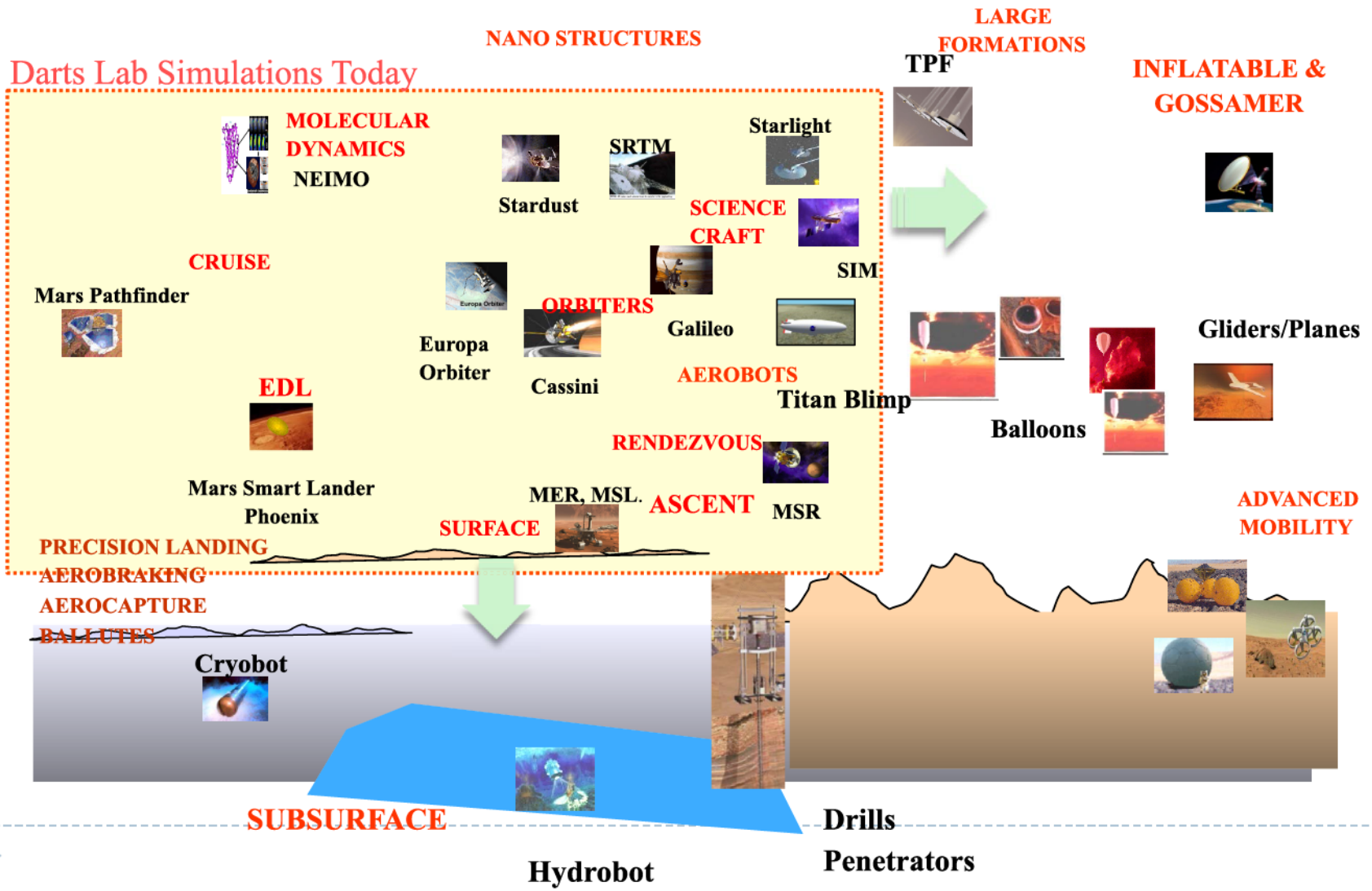
Outline

- Spacecraft Modeling / Simulation
- Multibody Dynamics Engine
- Some Examples

Focusing on two DARTS-Lab Tools:

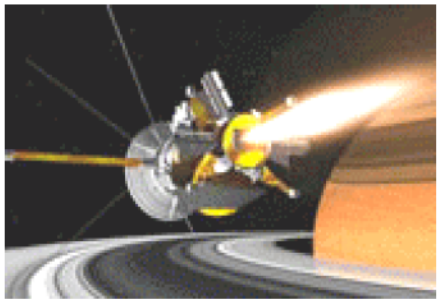
DARTS	Dynamics Algorithms for Real Time Simulation
DSENDS	Dynamics Simulator for Entry Descent and Surface landing

Mission Domain Applications

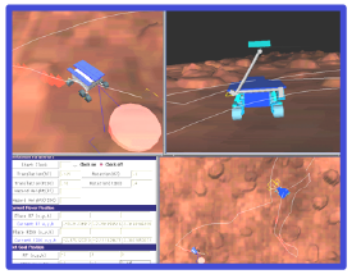


Examples of DARTS Lab Simulations

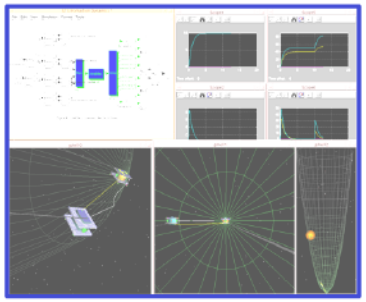
Free-Flyers
Orbiters & flyby spacecraft



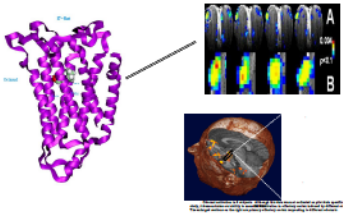
ROAMS
Planetary Rover Simulator



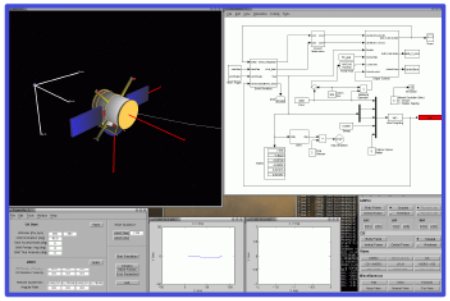
Formation Flying
Distributed Real-Time Simulations (Starlight)



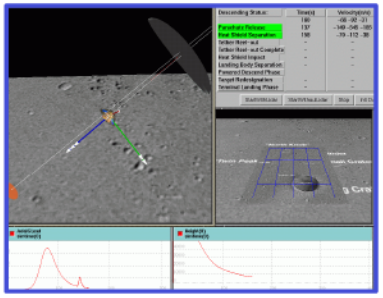
Molecular Dynamics
Large Scale Molecular Dynamics Simulation



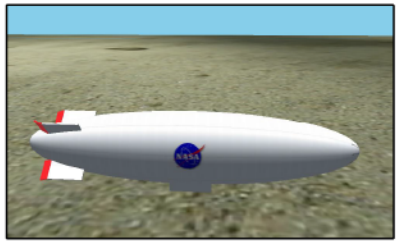
Rendezvous & Sample Capture
(Mars RSC, ST-6, CNES' 07)



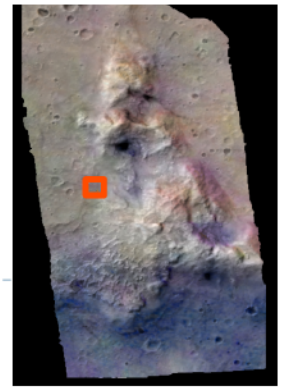
DSENDS
Entry, Descent & Landing Simulator



Aerobots
Titan Airship



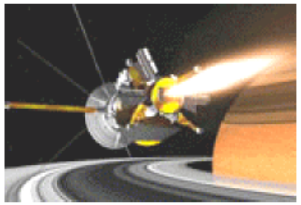
SimScape
Hierarchical Multi-Resolution Terrain Model Framework

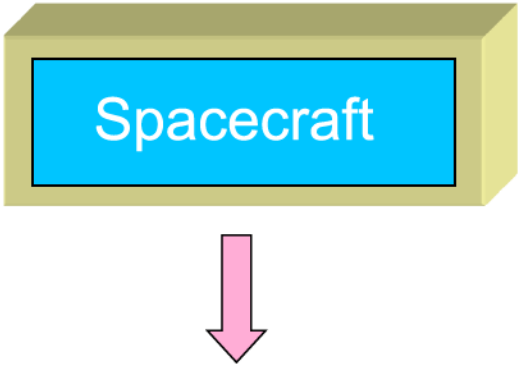


Modeling Focus across Mission Regimes

Deep Space

Predictable deep space environment, precision GNC

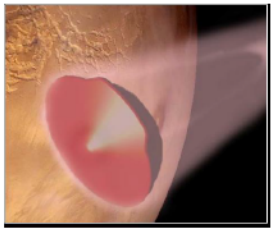


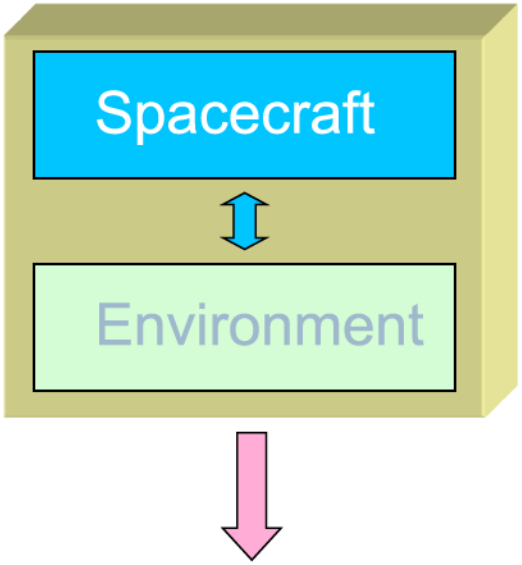


Virtual, High-Fidelity Vehicle Simulators

EDL

Uncertain environment, risk mitigation

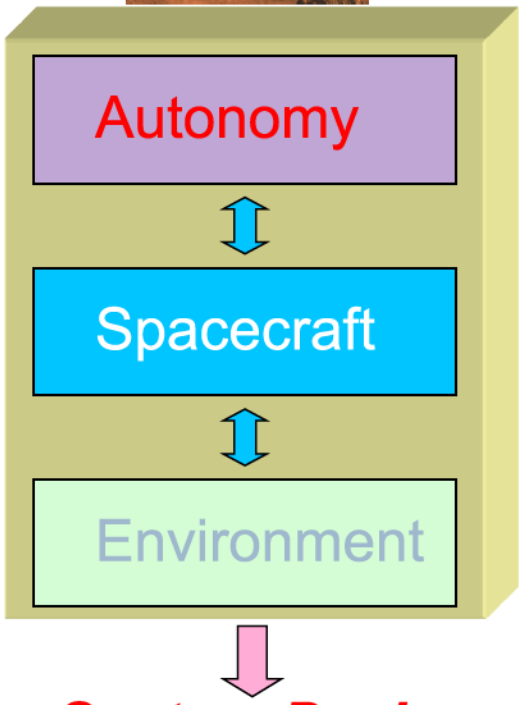




Performance & Risk Assessment

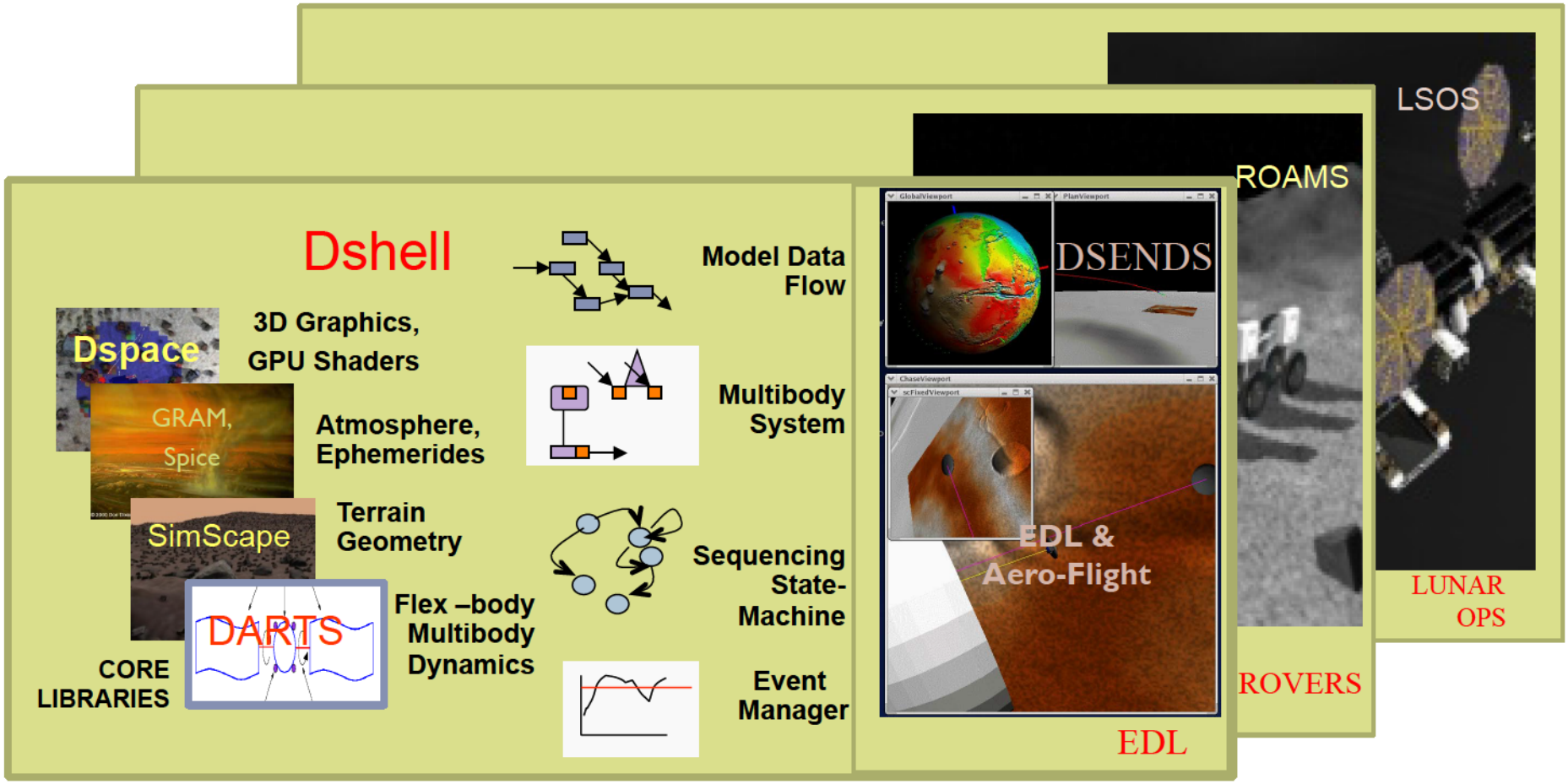
Surface

Unstructured environment, onboard reasoning



System Design, Reactive Control & Ops

DARTS Lab's Simulations for Surface & Near-Surface Robotic Systems

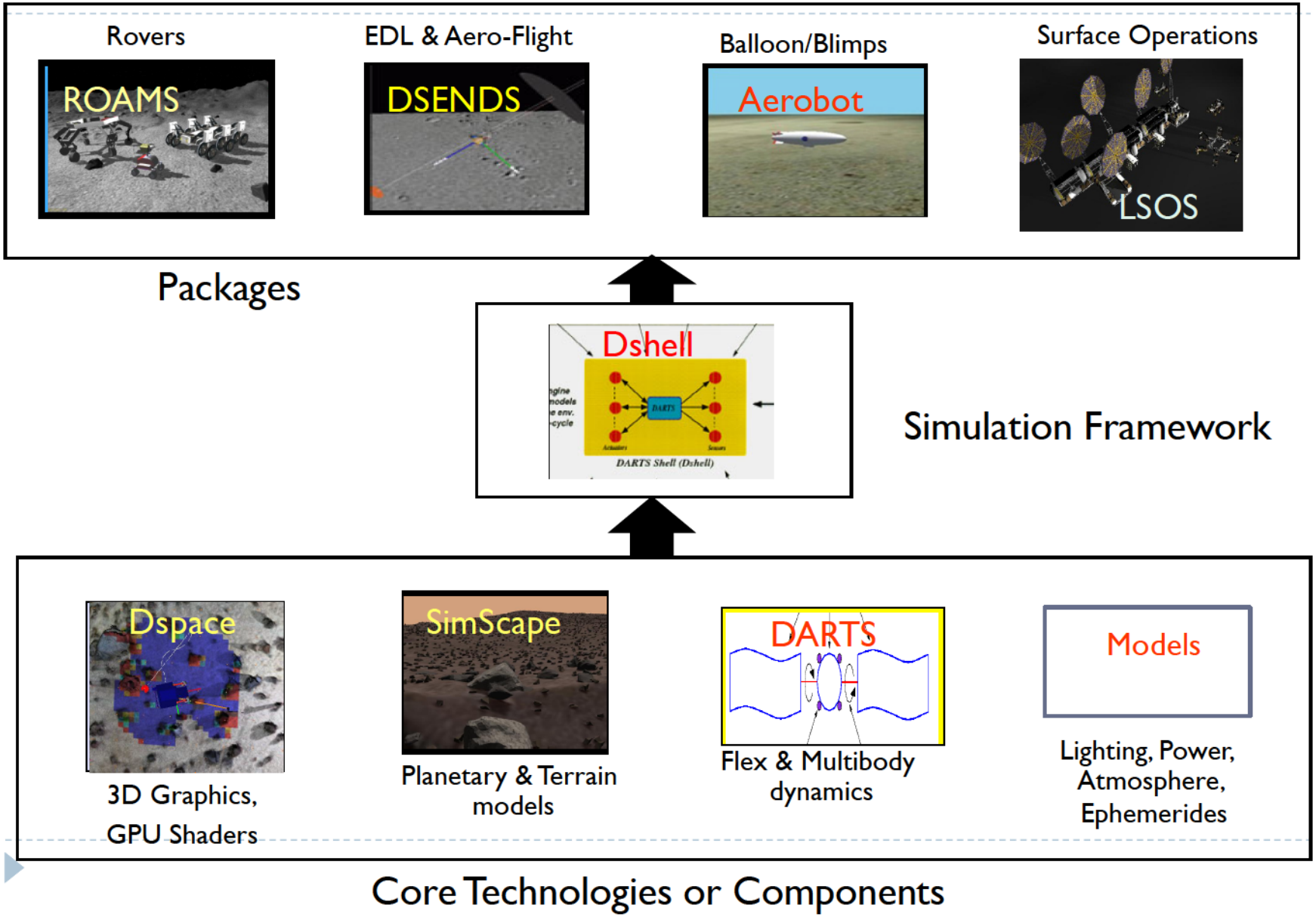


SIMULATOR DEPLOYMENTS

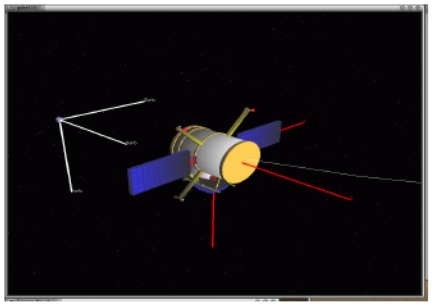
PRECISION LANDING
AEROBRAKING
ASCENT VEHICLES
AEROBOTS
RENDEZVOUS

Easy to adapt and extend to meet emerging/evolving needs.

Software Architecture

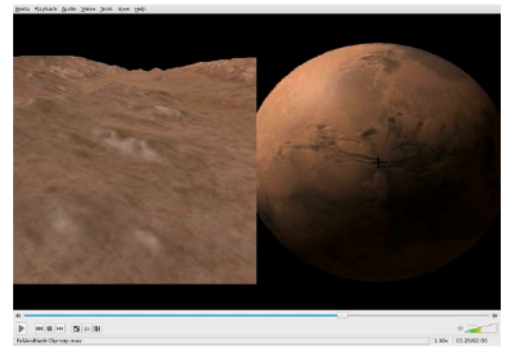


Some Physics-Based Simulations

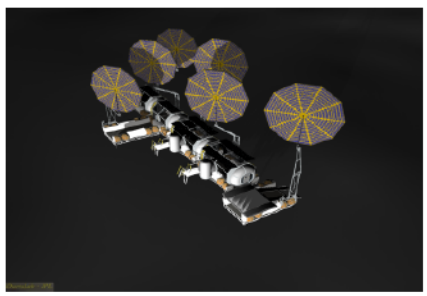


Rendezvous Sim

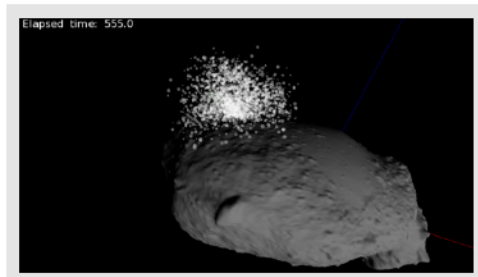
Aerobot Sim



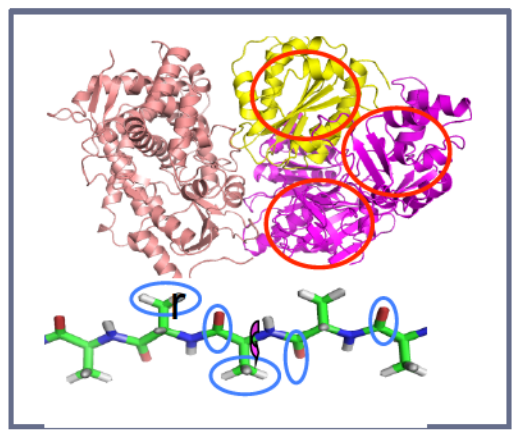
Large-scale terrain modeling & visualization



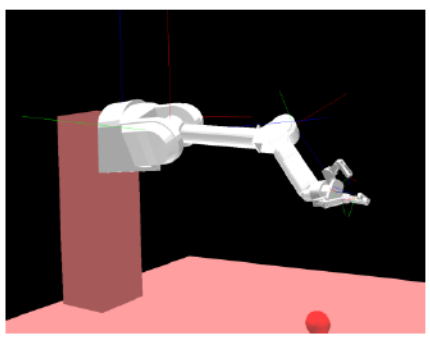
Lunar Habitat Sim



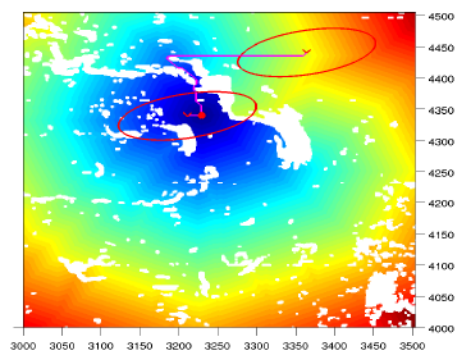
Ejecta Sim



Large-scale molecular dynamics

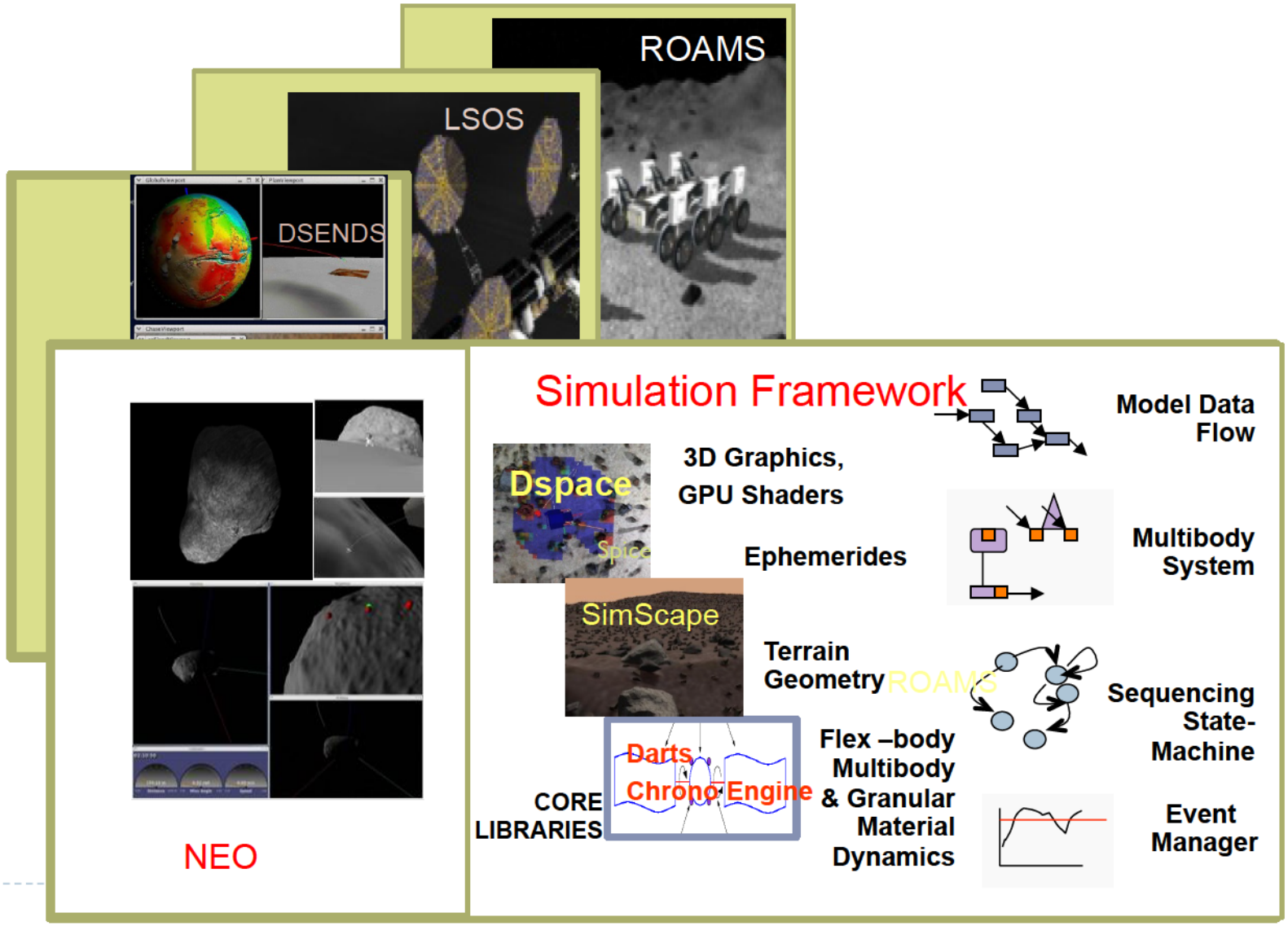


Manipulator dynamics & control

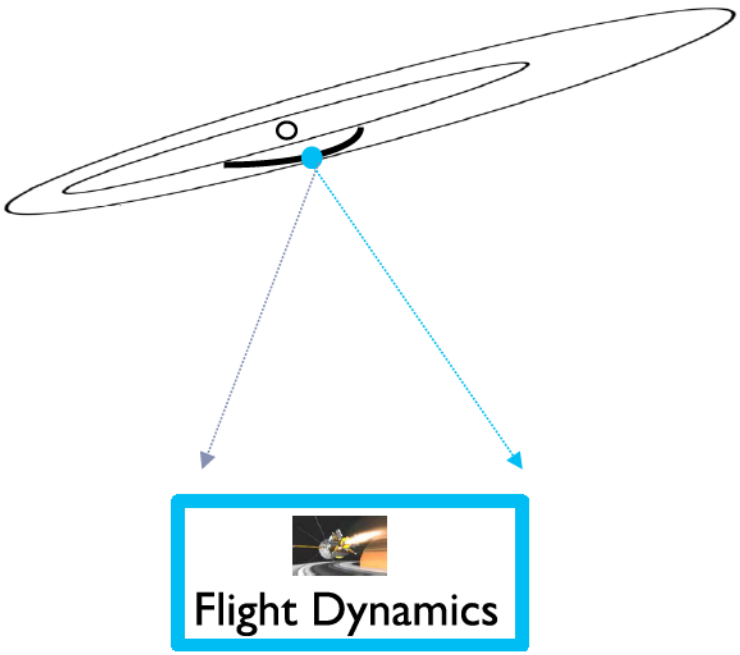


Mission landing vs. mobility trade analysis

Multi-mission Simulation Framework



Interplanetary Navigation & Mission Design vs Spacecraft Flight Dynamics



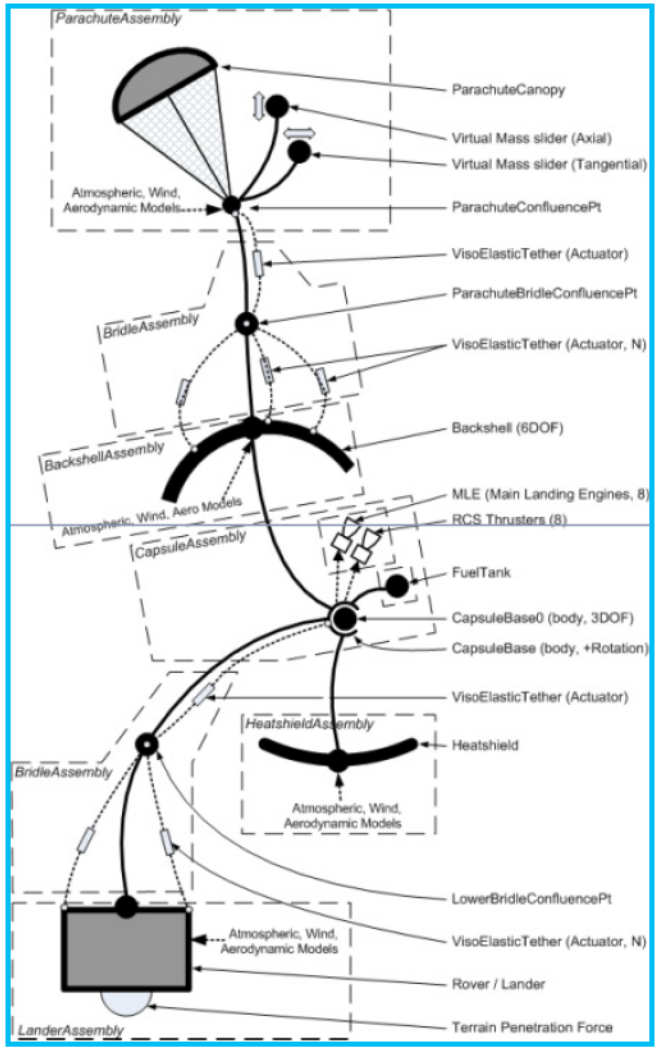
Interplanetary Navigation and Mission Design - *Model spacecraft "as a point"*

Spacecraft position and velocity states evolve in relation to solar system ephemerides, gravity and delta-V inputs, and are monitored using radiometric and/or optical navigation data

Flight Dynamics - *Model spacecraft "as a multi-body system"*

Spacecraft position, attitude, articulation and body flexibility states (and their rates) interact with gravity, atmospheres, terrain, and on-board s/c devices in response to ground commands and flight-software directed sensing and control actions.

Example Model (MSL)



Model spacecraft "as a multi-body system"

States – position, attitude, articulation, flex modes states/rates

Dynamics – 10's to 100's Hz; configuration changes, separations

Sensing – IMU, star-tracker, encoders, radar, lidar, imager, radiometrics

Actuation – thrusters, motors, gimbals, linkages

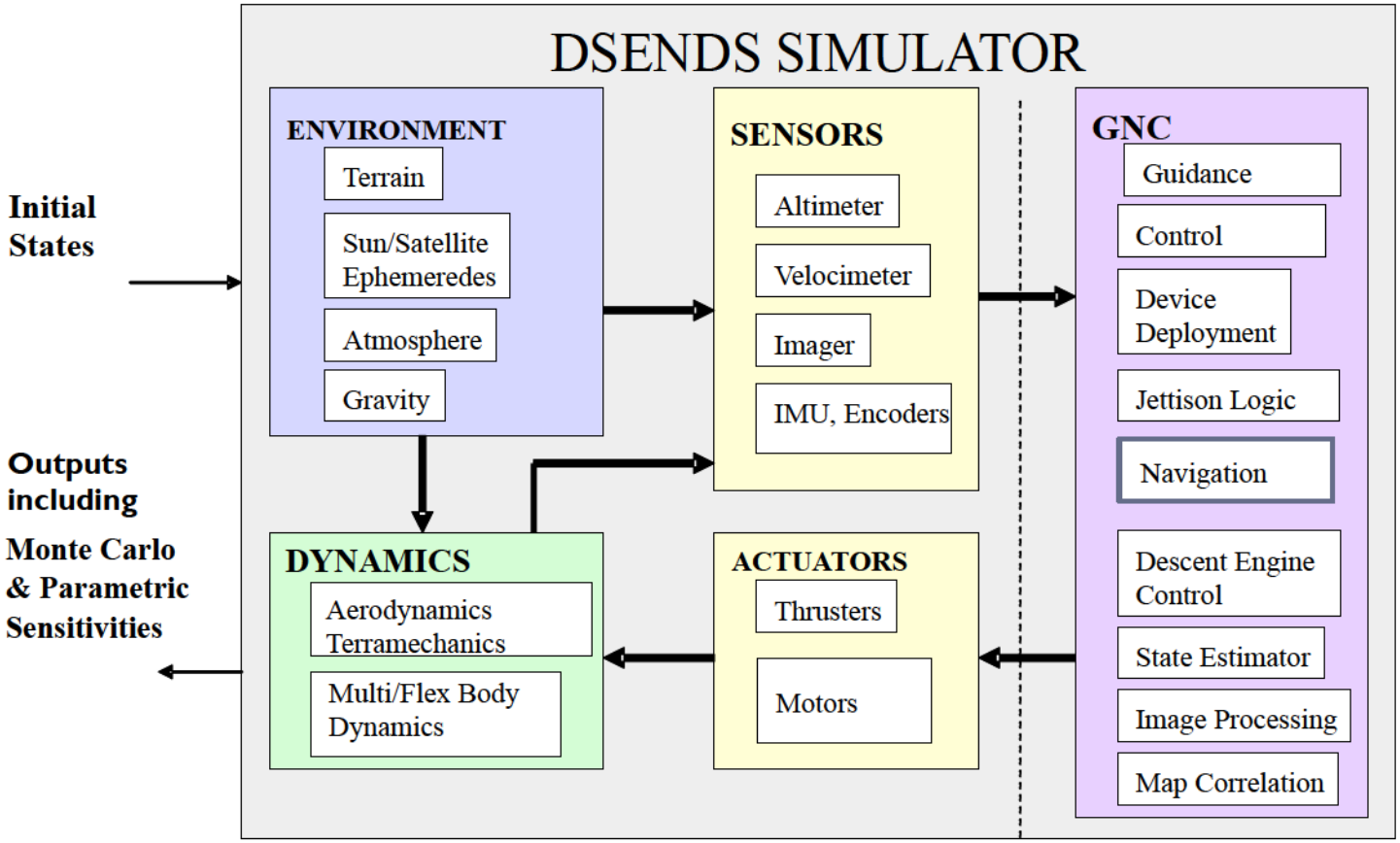
Environment – gravity, aerodynamics, atmosphere, aero-thermal, terrain shape, lighting/ shadowing

Operation – flight software in-the-loop; ground and autonomous modes

Addressing Critical Questions

- What is the EDL timeline and constraints ?
- What targeting related TCM's need to be performed ?
- What are the mission/system elements needed e.g. separate lander probe ?
- How robust is the proposed mapping/sampling strategy ?
- What combination of flight-like algorithms, software and avionics succeeds in achieving mission success with acceptable risk ?
- What are contact forces during a docking operation ?
- What are the attitude control implications (e.g. maximum attitude rates, deadbands, overturning moments) resulting from forces arising from the interactions of a multi-body spacecraft and terrain ?
- What are the communications and lighting performance and constraints ?
- What are the visibility and operational implications arising from material disturbed and ejected from the surface of an asteroid ?
- What anchoring concept would allow for in-situ implantation of an instrument on the surface ?
- What are the dynamics of a tether system used to secure an instrument ?

DSENDS Functional Block Diagram



- **Virtual spacecraft for use in testbeds**
- **System-level performance assessment tool**

DSENDS Functions

Mission & System Description

Environment Description

Simulation

Physics
Ephemerides
Kinematics & Multibody dynamics
Flex body & Contact dynamics
Terrain shape
Aerodynamics, Terramechanics
Lighting, Scene Generation
Line-of-Sight Geometry

Parametrics / Monte-Carlo

Computing
Clusters
GPU accelerators
Cloud Computing

Performance:
F(system/env state & state-history)

Trajectory times	Delta-V	Fuel
Visibility	Lighting	Collision
Power	Energy	
Temperature	Radiation	

Parametric performance *F(p)*
Sensitivity *dF/dp*
Dispersed state & parameter performance

Products
Data
3-D Visualizations
Movies
Trade Space Visualization
Client/Server Queries
Embedded Models
Risk / Failure Probabilities

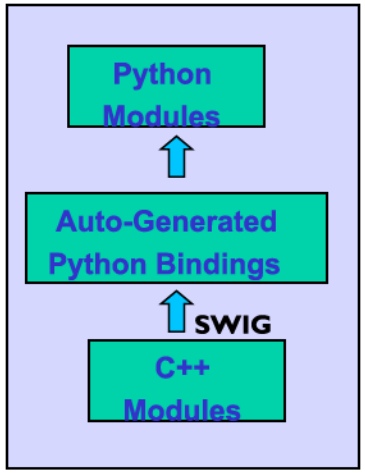
DSENDS Capabilities

Vehicle Dynamics & Kinematics
Entry Capsules; Landers; Ascent Vehicles, Rovers, Balloon Airships
Parachutes; Bridles, Viscoelastic lines; Tethers & Reel-out devices; Anchors; Wheels; Legs; Suspensions; Arms; Masts.
Body separation, CG shift, Fuel depletion, fuel slosh, hard/soft contacts
3-dof, 6-dof, multi-body dynamics; flex-body dynamics, kinematic constraints
Astrodynamics
Spherical harmonics & Polyhedral gravity; Ephemeredes (Spice, Masar);
Aerodynamics
General aerodynamics with aero-tables, C/Fortran aero libraries, or user-provided Python callback functions
Para inflation models
Geometry
Collision detection, line-of-sight analysis; Lighting models; Shadows

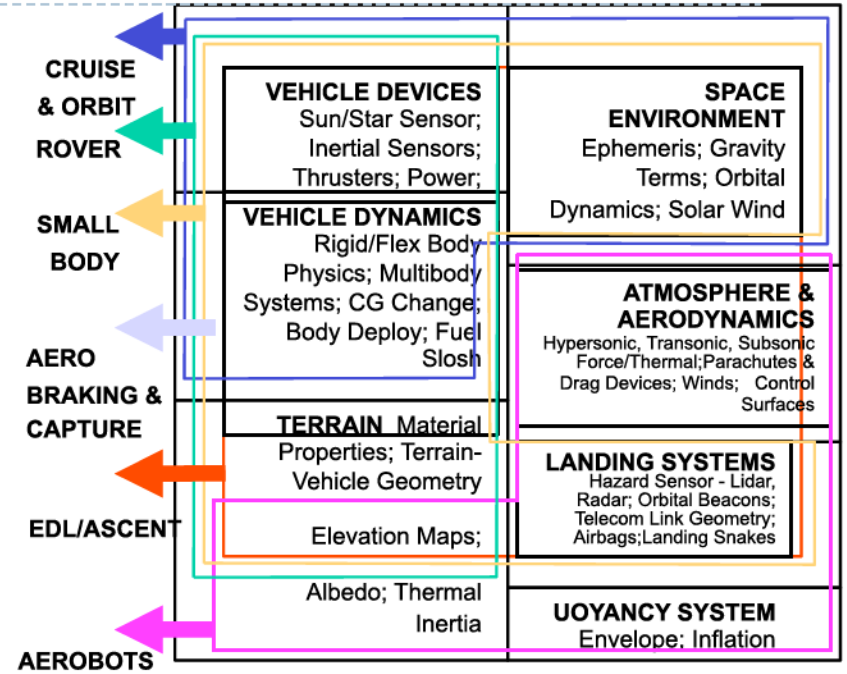
Sensor
IMU;Altimeter, Velocimeter
Descent camera, Lidar, Landmarks
Encoders
Mission Design
EDL entry states i/f; Arrival targeting;
Radiation models; Resource usage
Comm. link budgets
Software
Python scripting; C++ modules
Interoperable with Monte/Masar
Control/Guidance stubs
VxWorks/Linux; Simulink i/f;
Checkpoint, Data logs, 3-d graphics
Monte-Carlo & Parametric simulations with automated variate generation
Cycle time and call frequency settings
Data playback & reconstruction
RK4/CVODE adaptive/stiff integrator

Actuators
Throttled thrusters & engines
Motors, gimbals, gears, springs
Atmosphere Models
Mars/Earth/Venus GRAM models for density, pressure, temperature & winds
Meso scale spline models for pressure, temperature and winds (steady & gust)
Terrain Modeling
Planet shape e.g. Mars MOLA Terrain
Site multi-gigabyte terrain DEM/texture
General mesh shape models for irregular objects e.g. asteroids
Soil Dynamics
Wheel-soil interaction
Interfaces
i/f to C, C++, Fortran, Matlab code
Flight s/w i/f
Real-time telemetry i/f

DSENDS Architectural Features



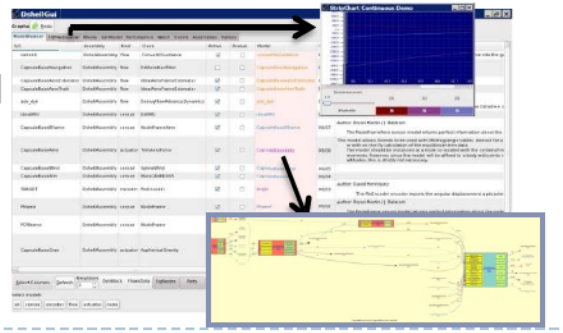
To Users DSENDS is a Python language extension that they can adapt, customise and extend. Modern scripting language (**Python**) is used to provide data-driven building of applications at run-time and allows easy configuration , initialization, automated testing, and the user to rapidly evaluate and iterate through designs



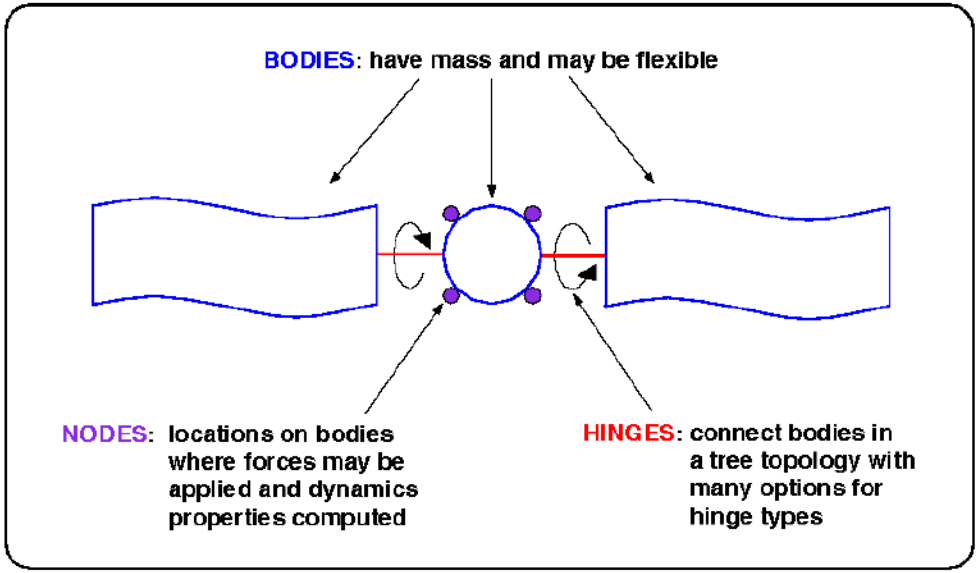
High Performance from C++ implementation of module functions and fast algorithms

Functionality organized as independent **reusable modules & libraries.**

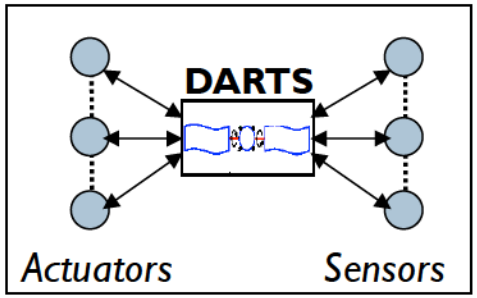
Visualization and Introspection dynamically generated



DARTS Dynamics Engine



Recipient of the 1997 NASA Software of the Year Award.



Dshell – DARTS Shell

- Implements highly efficient **Spatial Operator Algebra (SOA)** O(N) multibody dynamics algorithm – provides real-time performance and scales well to 100's and 1000's of degrees-of-freedom
- Uses minimal dof, internal coordinate formulation – has superior numerical properties as most constraints are implicit
- Implements flex-body dynamics - high performance & high fidelity to capture both rigid and flexible modes

DARTS Capabilities

	Feature
Multibody System	Rigid and Flexible Bodies
	Variety of Kinematic Joints
	Sensor and Actuator Nodes
	Recursive Calculation of CM, CM Velocity and other system properties
	Multiple tree topologies
	Local kinematic loops
	Geared Systems
	Prescribed Joints
	Vary system topology during simulation
	System Jacobians and Joint Jacobians

	Feature
Algorithms	Kinematics
	Forward Dynamics
	Inverse Kinematics
	Inverse Dynamics
	Direct Embedding for constraints
Architecture	C++ object-oriented
	SWIG Wrapped
	Python scripting
	VxWorks/Linux
Testing	Suite of regression tests for kinematics, dynamics etc.

DARTS : Rigid/Flexible Multibody Dynamics Engine

- Kinematics and Dynamics of rigid/flex multibody systems

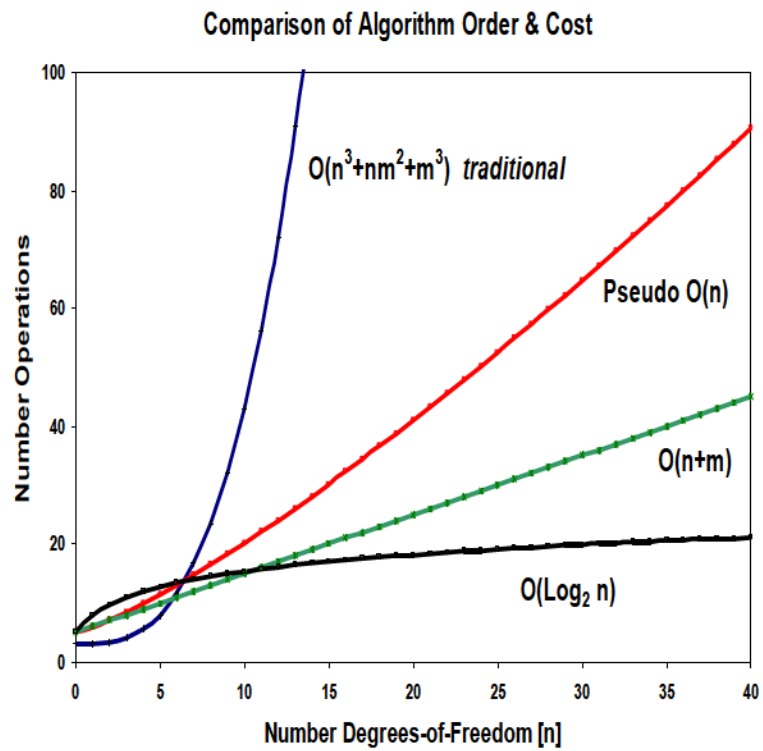
- Uses minimal dof, **internal coordinate** formulation – eliminates constraints, is an **ODE** approach, and has superior numerical properties;

- Implements highly efficient **O(N)** multibody dynamics algorithm in contrast with the more commonly used $O(N^3)$ algorithm (No explicit calculation/inversion of mass matrix);

- Based on **Spatial Operator Algebra** mathematical framework for multibody dynamics (Articulated body inertia based formulation, Recursive calculation of generalized accelerations);

- General purpose with model data driven interface;

- Models **multi-flexible** body systems and captures nonlinear rigid/flex nonlinear coupling.



The more common and tranditional approach uses a DAE formulation

Pros

- Full descriptor formulation
- Ability to handle any arbitrary constraint
- Diagonal mass matrix
- Conceptually easy to understand

Cons

- Computationally expensive
- Inexact constraint satisfaction
- Numerical issues of DAEs - stability and non-physical oscillations, convergence, singularity

Spatial Operator Algebra (SOA) Formulation

- SOA defines a ***new theoretical paradigm*** for MBS analysis and computational dynamics algorithms.
- SOA is the basis for the state-of-the-art **DARTS** flexible MBS dynamics software.
- SOA embodies algorithms and formulations for
  - Mass matrix factorization and inversion,
  - Efficient Matrix determinant
  - Operational Space Dynamics,
  - Diagonalized Dynamics,
  - Linearization
  - Sensitivity Analysis
- New technology directions not possible by any other alternative approaches
- SOA has formed the basis of Matlab's SimMechanics multibody dynamics toolbox, and several molecular dynamics softwares at Stanford, NIH, Scripps & in Europe.
- Publications: 550 page monograph; ~20 papers in top journals (IEEE transactions, JGC etc.) and ~35 conference papers.
- Pioneering research and technology development for physics based dynamics modelling of complex applications such as spacecraft, rovers vehicles, robotics and bio-molecular systems

DARTS: System Level Equations

Diagonal Spatial Inertia $$M = diag[M(1) \ \cdots \ M(n)]$$

Shift Matrix $$\phi \triangleq \begin{pmatrix} I & 0 & \dots & 0 \\ \phi(2,1) & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \phi(n,1) & \phi(n,2) & \dots & I \end{pmatrix}$$

Joint Motion Map $$H = diag[H(1) \ \cdots \ H(n)]$$

Velocity Equation $$V = \phi^* H^* \beta$$

Acceleration Equation $$\alpha = \phi^*(H^*\dot{\beta} + a)$$

Newton Euler Equations of a Multibody System $$f = \phi(M\alpha + b) = \phi M \phi^* H^* \dot{\beta} + \phi(M\phi^* a + b)$$

Projected into minimal coordinate space $$T = Hf = H\phi M \phi^* H^* \dot{\beta} + H\phi(M\phi^* a + b)$$

Equation of Motion $$\mathcal{M}\dot{\beta} + \mathcal{C} = T \quad \text{where} \quad \mathcal{M} = H\phi M \phi^* H^*$$

DARTS: Mass Matrix Inverse via the SOA

Mass Matrix $\mathcal{M} = H\phi M\phi^* H^*$ *non-square factors*

Analytical Newton-Euler factorization of the mass matrix

Mass Matrix Factorization $\mathcal{M} = [I + H\phi K]D[I + H\phi K]^*$ *square factors*

Analytical Innovations factorization of the mass matrix

The Factorization Intermediate $[I + H\phi K]^{-1} = [I - H\psi K]$

Mass Matrix Inversion $\mathcal{M}^{-1} = [I - H\psi K]^* D^{-1}[I - H\psi K]$

Analytical operator expression for the mass matrix inverse ➡ O(N) articulated body forward dynamics algorithm

O(n) Solution of EOM $\dot{\beta} = \mathcal{M}^{-1}\bar{T} = [I - H\psi K]^* D^{-1}[I - H\psi K]\bar{T}$

- The above shows a LDU Decomposition of the mass matrix & its inverse
- The factorization of the mass matrix and the intermediate identity show system level operator factorization of the mass matrix
- The system level operators map into physical component level identities
- These component level identities form the basis for the implementation of the algebra through Innovations Factorization implemented in 3 sweeps

DARTS: Recursive Sweep 1

- Kinematics Sweep - from base body to tip
  - Use recursive relations to incrementally calculate kinematic quantities using parent-child relations

$$
\begin{cases}
V(n+1) = 0, \quad \alpha(n+1) = 0 \\
\textbf{for } \boldsymbol{k} = \boldsymbol{n} \cdots \boldsymbol{1} \\
\quad V(k) = \phi^*(k+1,k)V(k+1) + H^*(k)\beta(k) \\
\quad \alpha(k) = \phi^*(k+1,k)\alpha(k+1) + H^*(k)\dot{\beta}(k) + a(k) \\
\textbf{end loop}
\end{cases}
$$

$$
\begin{cases}
f(0) = 0 \\
\textbf{for } \boldsymbol{k} = \boldsymbol{1} \cdots \boldsymbol{n} \\
\quad f(k) = \phi(k,k-1)f(k-1) + M(k)\alpha(k) + b(k) \\
\quad T(k) = H(k)f(k) \\
\textbf{end loop}
\end{cases}
$$

DARTS: Recursive Sweep 2

- Articulation Sweep - from tip to base
  - Use recursive relations to incrementally augment articulated inertias felt at each joint
  - Calculate Coriollis and Gyroscopic loads

$$\mathcal{M} = [I + H\phi K]D[I + H\phi K]^*$$

$$\begin{cases} P^+(0) = 0 \\ \textbf{for } k = 1 \cdots n \\ \quad P(k) = \phi(k,k-1)P^+(k-1)\phi^*(k,k-1) + M(k) \\ \quad D(k) = H(k)P(k)H^*(k) \\ \quad G(k) = P(k)H^*(k)D^{-1}(k) \\ \quad K(k+1,k) = \phi(k+1,k)G(k) \\ \quad \overline{\tau}(k) = I - G(k)H(k) \\ \quad P^+(k) = \overline{\tau}(k)P(k) \\ \quad \psi(k+1,k) = \phi(k+1,k)\overline{\tau}(k) \\ \textbf{end loop} \end{cases}$$

DARTS: Recursive Sweep 3

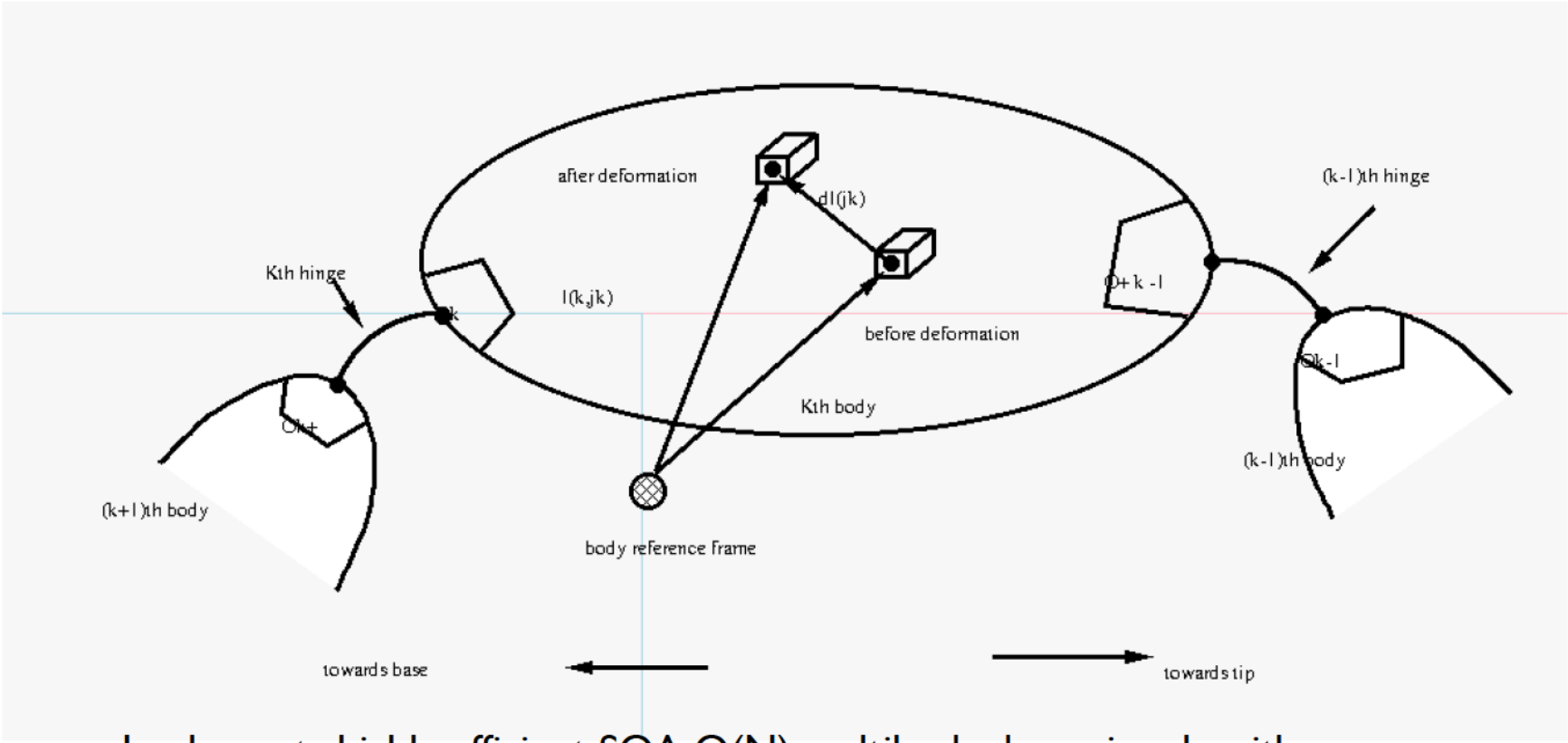
- Solution Sweep - from base to tip
  - Use recursive relations that use entities generated during the the kinematics and articulation sweep to solve equations of motion

$$\mathcal{M}^{-1} = [I - H\psi K]^* D^{-1} [I - H\psi K]$$

$$\dot{\beta} = \mathcal{M}^{-1}\bar{T} = [I - H\psi K]^* D^{-1} [I - H\psi K]\bar{T}$$

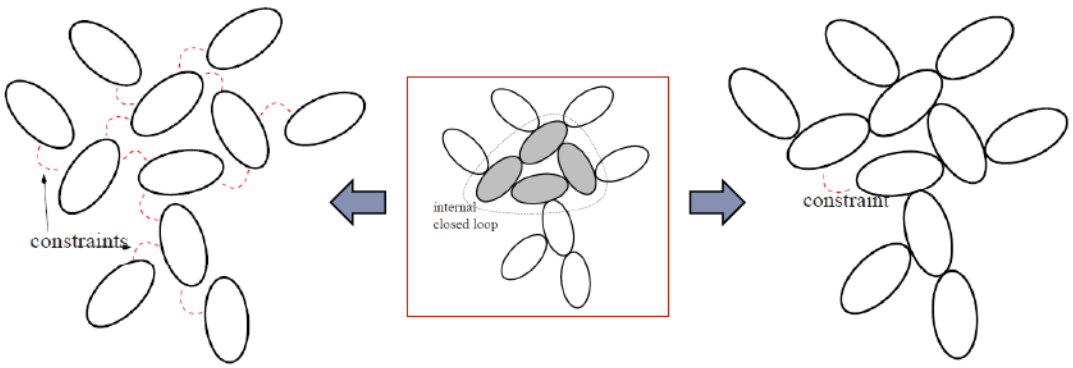
$$\left\{ \begin{array}{rcl} & & z^+(0) = 0 \\ \textbf{for } \boldsymbol{k} & = & \boldsymbol{1 \cdots n} \\ z(k) & = & \phi(k, k-1) z^+(k-1) \\ \epsilon(k) & = & \bar{T}(k) - H(k) z(k) \\ \nu(k) & = & D^{-1}(k)\epsilon(k) \\ z^+(k) & = & z(k) + G(k)\epsilon(k) \\ \textbf{end loop} & & \end{array} \right. , \left\{ \begin{array}{rcl} & & \alpha(n+1) = 0 \\ \textbf{for } \boldsymbol{k} & = & \boldsymbol{n \cdots 1} \\ \alpha^+(k) & = & \phi^*(k+1, k)\alpha(k+1) \\ \dot{\beta}(k) & = & \nu(k) - G^*(k)\alpha^+(k) \\ \alpha(k) & = & \alpha^+(k) + H^*(k)\dot{\beta}(k) \\ \textbf{end loop} & & \end{array} \right.$$

DARTS: Flexible Body Dynamics



- Implements highly efficient SOA O(N) multibody dynamics algorithm – provides real-time performance and scales well to 100's and 1000's of degrees-of-freedom
- Assumed modes are used to model body (small) deformation
- Includes rigid/flex dynamics coupling (full set of modal integrals)

DARTS: Minimal Coordinate Dynamics Model for Closed-chain Dynamics



$$\mathcal{M}(\theta)\ddot{\theta} + \mathcal{C}(\theta,\dot{\theta}) + G_c{}^*(\theta,t)\lambda = \mathcal{T}$$
$$G_c(\theta,t)\dot{\theta} = \mathfrak{U}(t)$$

Redundant coordinates

- All bodies are treated as independent
- Bilateral constraints used for all hinges
- Larger model, but simple to set up

Minimal coordinates

- Minimal hinge coordinates
- Bilateral constraints only for cut joint
- Configuration dep. mass matrix

$$\begin{aligned}
\ddot{\theta}_f &\triangleq \mathcal{M}^{-1}(\mathcal{T} - \mathcal{C}) \\
\lambda &= \boxed{-\left[G_c \mathcal{M}^{-1} G_c^*\right]^{-1}} \gamma \\
\ddot{\theta}_\delta &\triangleq \mathcal{M}^{-1} G_c^* \lambda
\end{aligned}$$

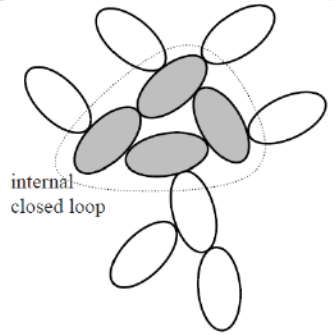
Schur complement

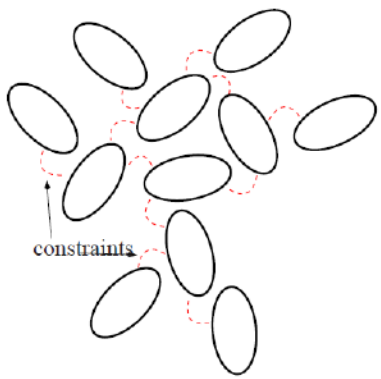
$$\ddot{\theta} = \ddot{\theta}_f + \ddot{\theta}_\delta$$

Procedure

- Solve tree system dynamics
- Use constraint accel error to compute missing constraint forces: Requires reflected mass matrix at constraint modes (Schur complement). Expensive and complicated.
- Correct free solution with effect of constraint forces

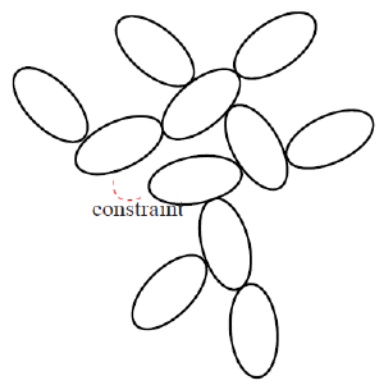
DARTS: Non-Tree (Graph) Topology Systems





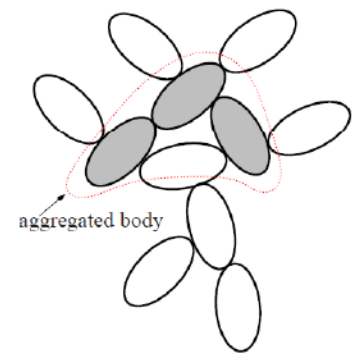
Fully Augmented (FA) model

Non-minimal coords
+ constraints



Tree-Augmented (TA) model

Minimal tree coords
+ constraints



Constraint Embedding (CE) model

Minimal coords

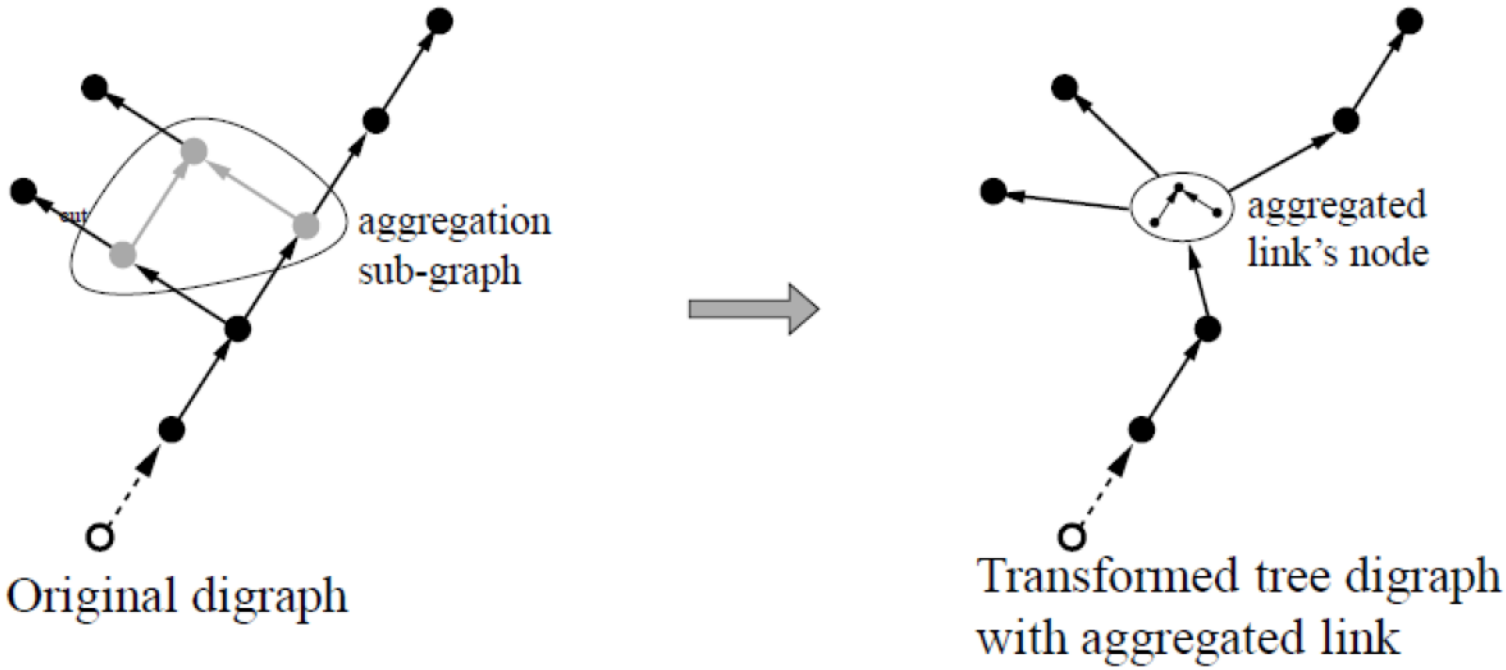
- *Simple setup*

Better for large loops

Optimal for small loops

DARTS: Constraint Embedding Closed-Chain Model

- Constraint embedding is a technique for transforming a non-tree graph into a tree graph by aggregating internal loops

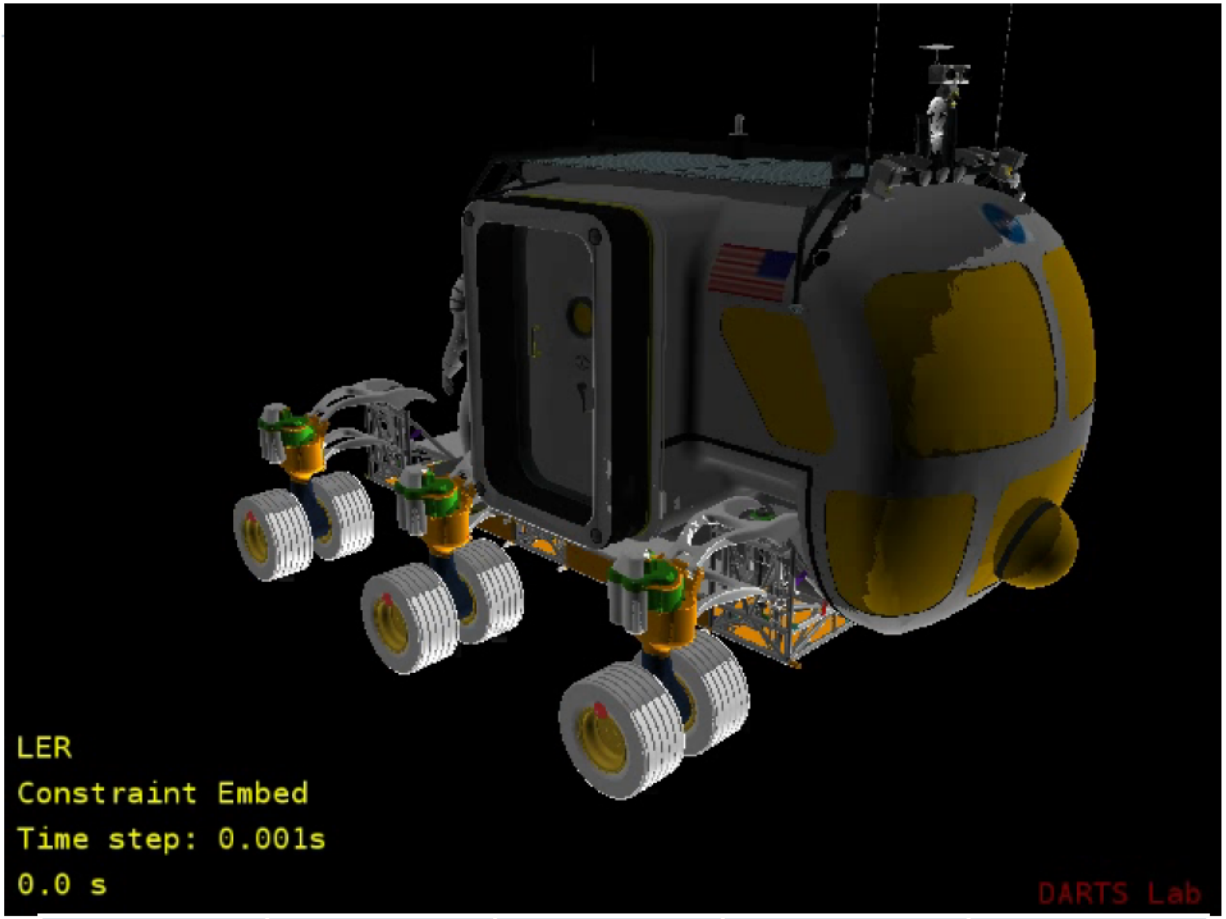


- Aggregated bodies have variable geometry
- Allows use of full **tree minimal coordinates** model for closed-chain system without bilateral constraints
- Can use **tree O(N) algorithm**

DARTS: Performance Comparison LER vehicle simulation

LER rover model

- 6 wheels with independent wishbone suspensions
- Each wishbone suspension represents a spring-loaded 4- bar linkage
- 31 bodies, 24 effective dofs
- Sinusoidal normal forces applied at each wheel

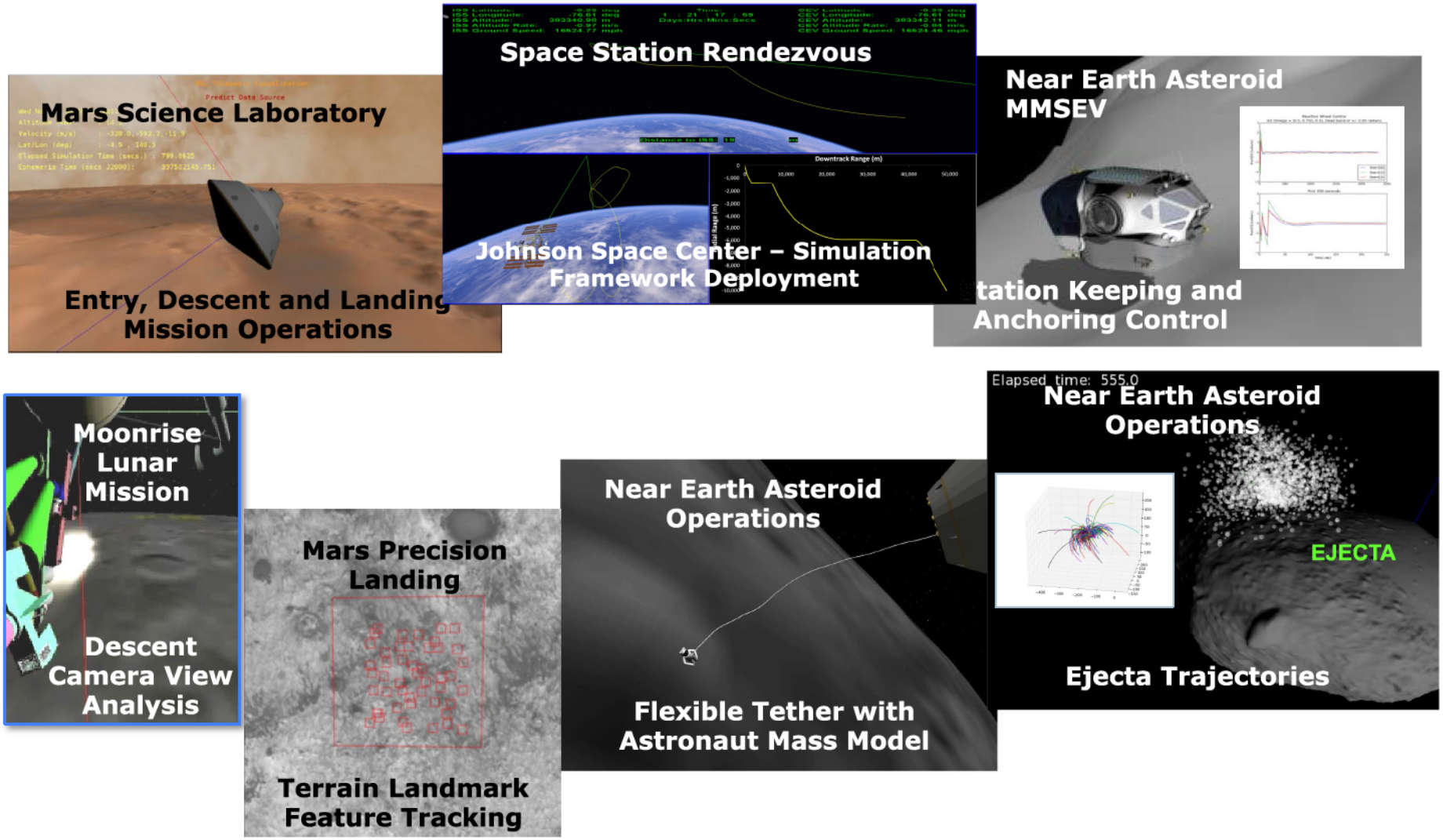


Comparison metrics:

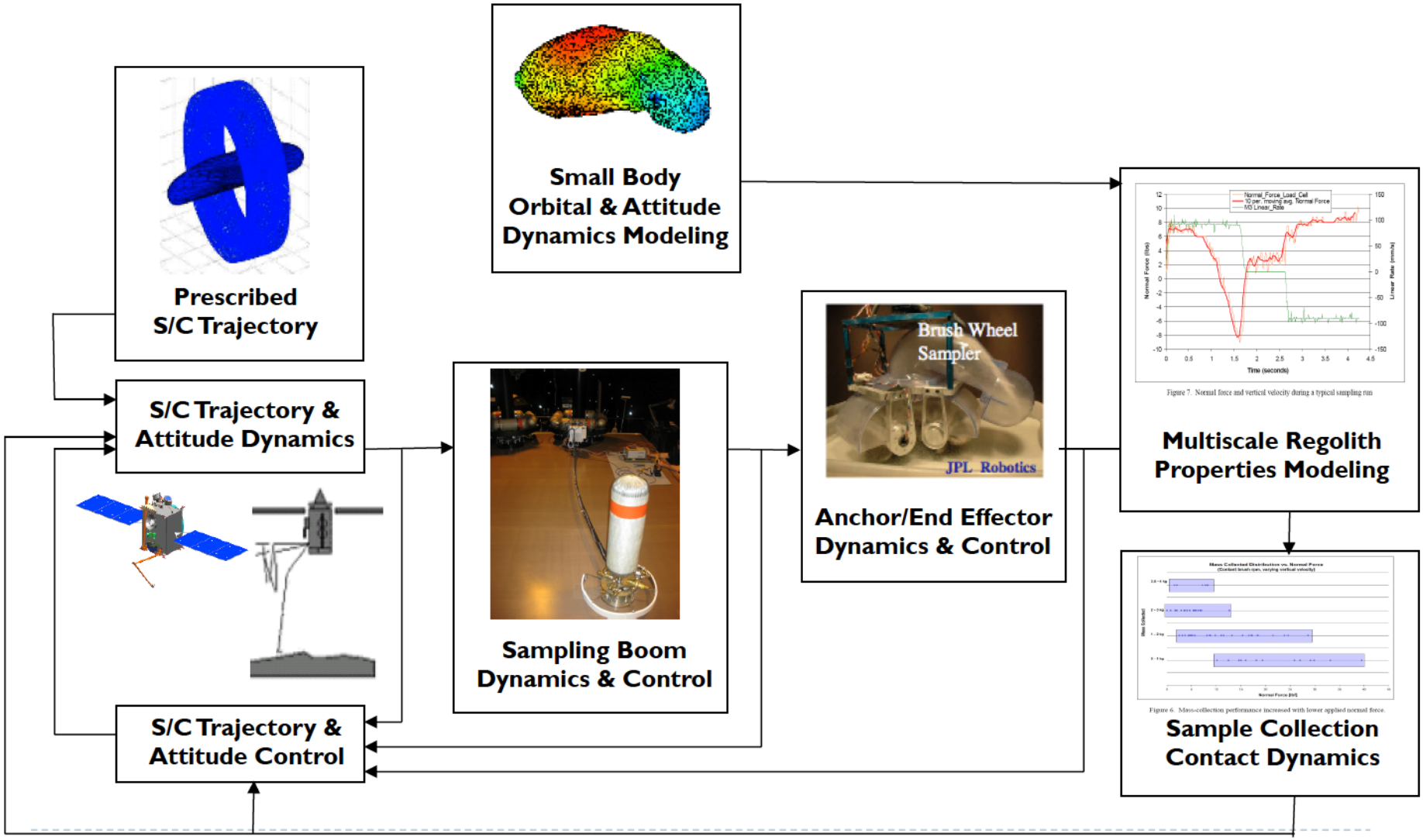
- Constraint error build up
- Fidelity error
- Computational speed.

Method	Tree dofs	Constraint dofs	Augmented size	Sim time ratio
CE	24	0	24	1
TA	36	12	48	1.6
FA	186	162	348	59.4

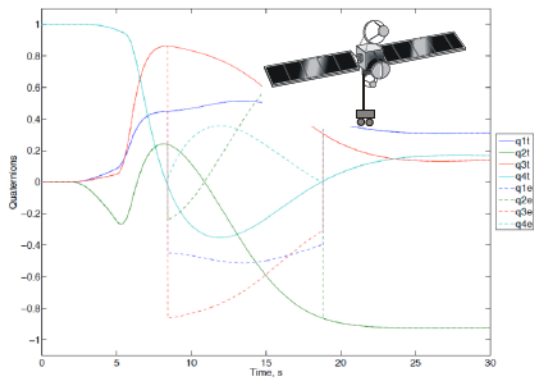
DARTS/DSENDS: Examples of Use



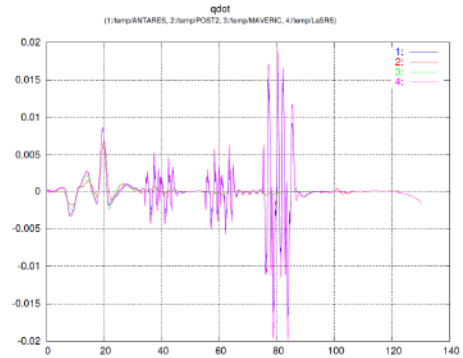
Integrated Modeling and Simulation for Primitive Body Sampling



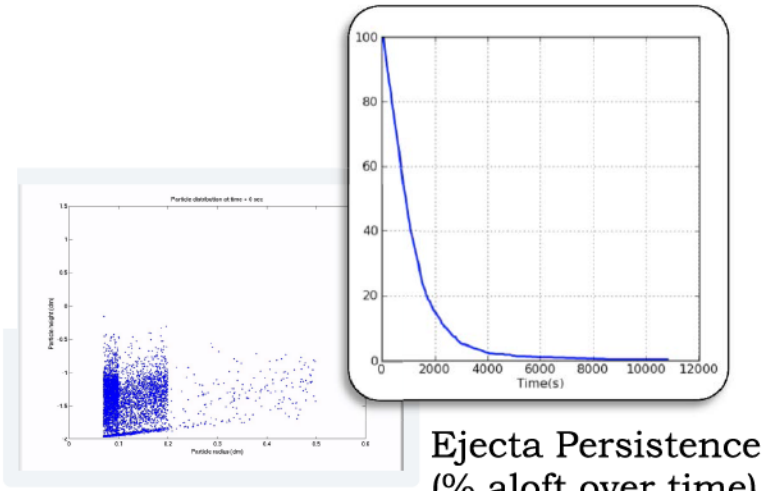
Orbits, Trajectories



Touch-and-go attitude control profiles

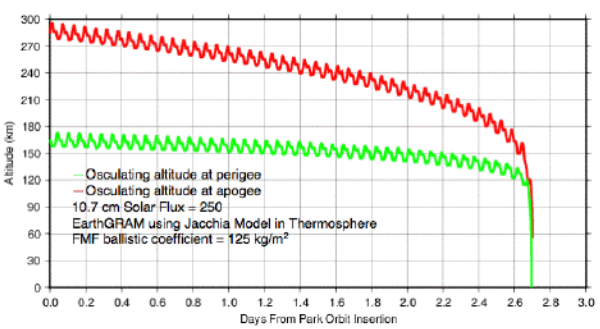


Ares I-X trajectory
(courtesy JSC-MOD)

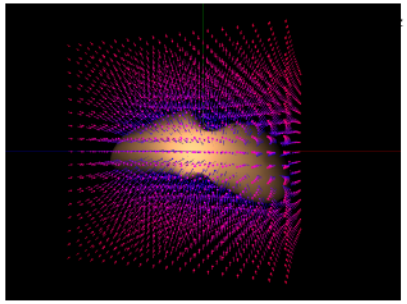


Ejecta Persistence
(% aloft over time)

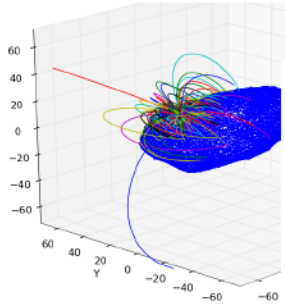
Regolith Granular Dynamics



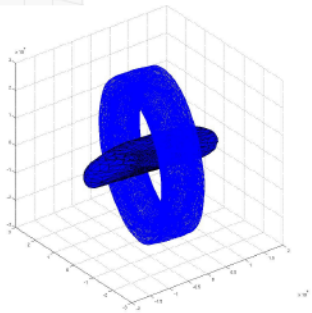
MSL Earth parking orbit analysis



Small-Body
Polyhedral Gravity



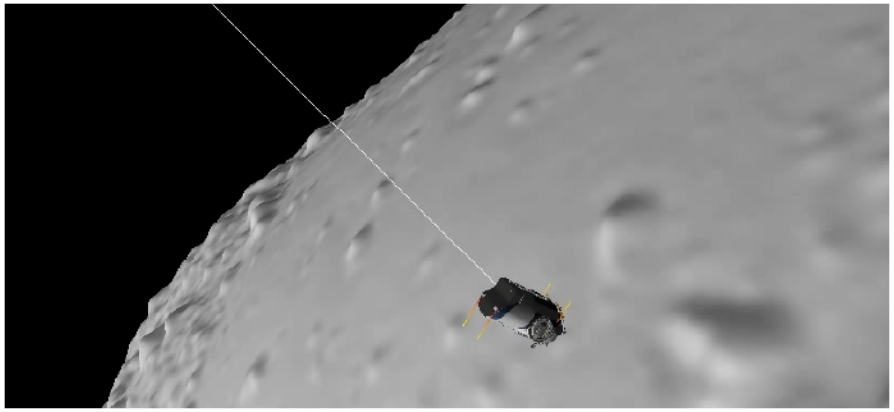
Small-Body
Ejecta Trajectories



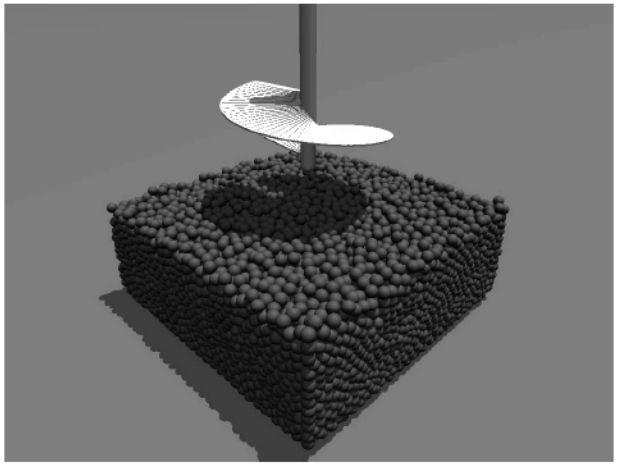
Small-Body
Stable Orbit Bundles

NEO Missions (MMSEV at Itokawa)

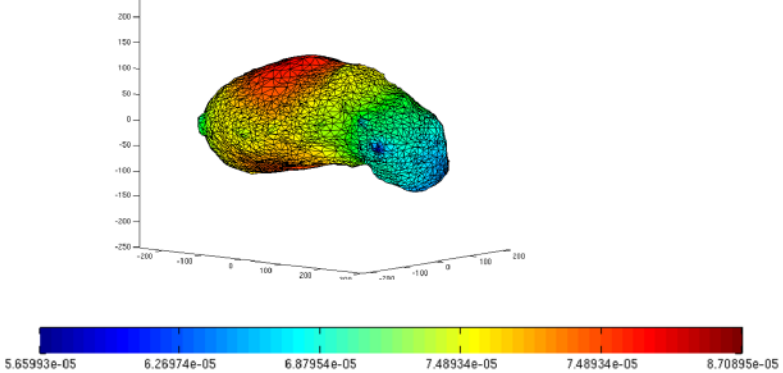
Station Keeping



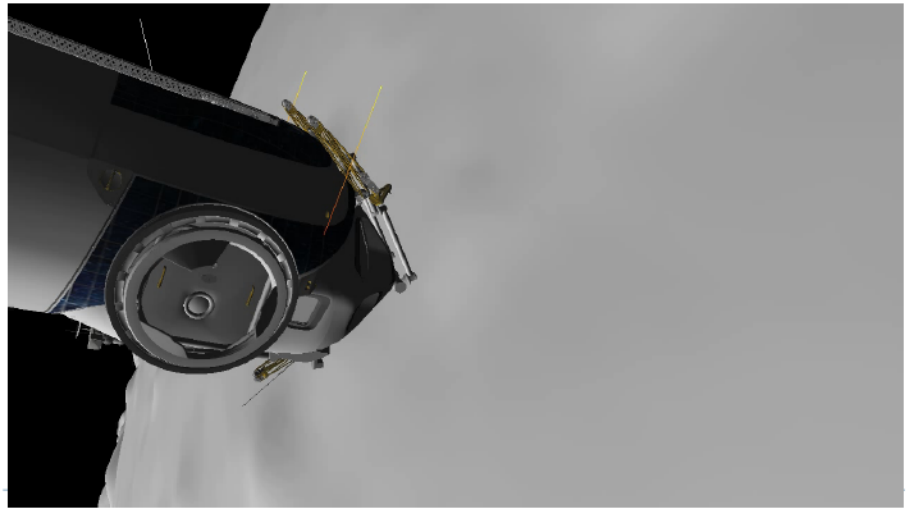
Anchoring Modeling



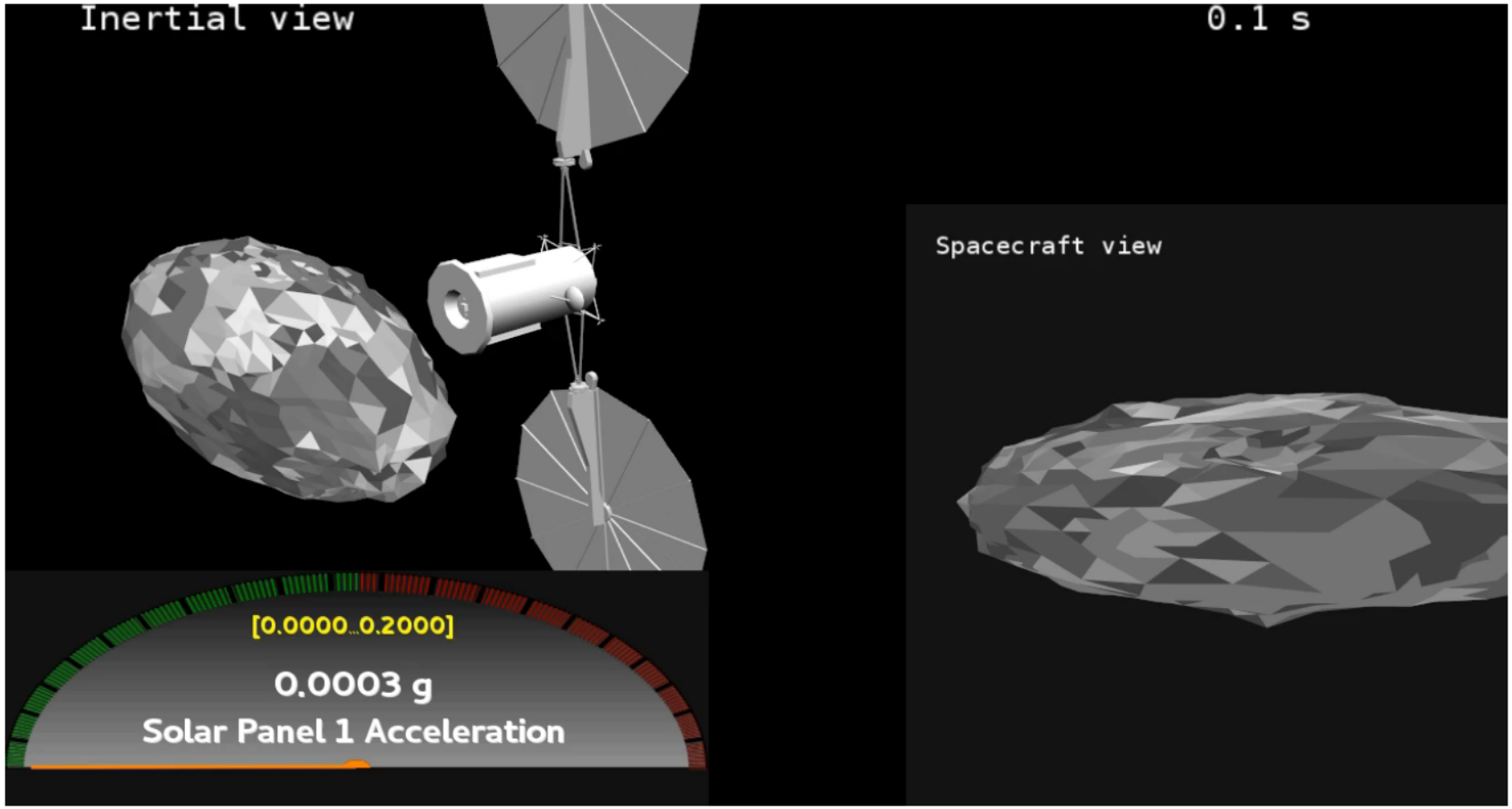
Surface acceleration on Itokawa asteroid



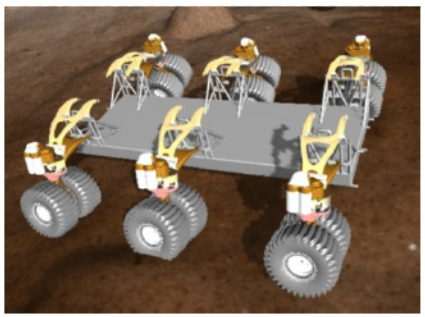
Anchoring Control

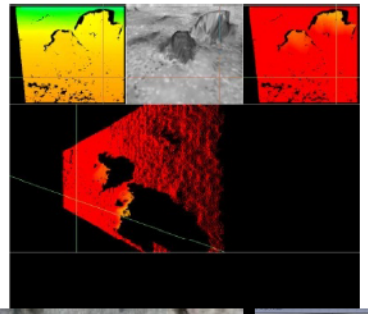


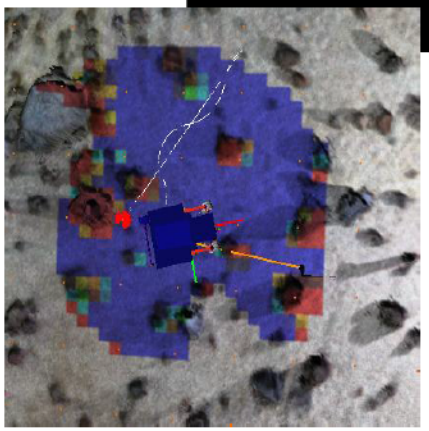
NEO Capture


Inertial view
0.1 s
Spacecraft view
[0.0000...0.2000]
0.0003 g
Solar Panel 1 Acceleration

ROAMS Rover Simulator

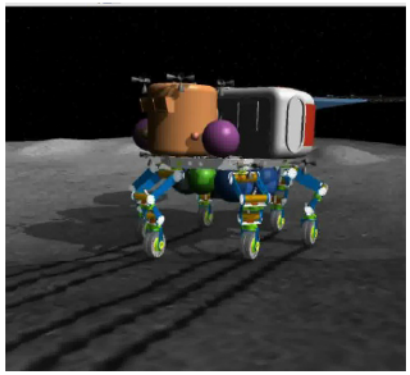


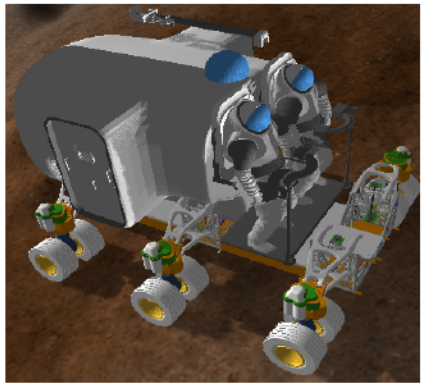


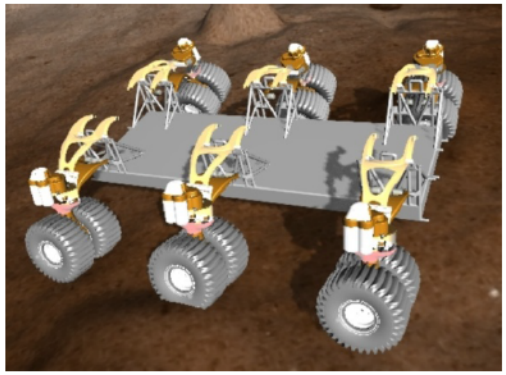


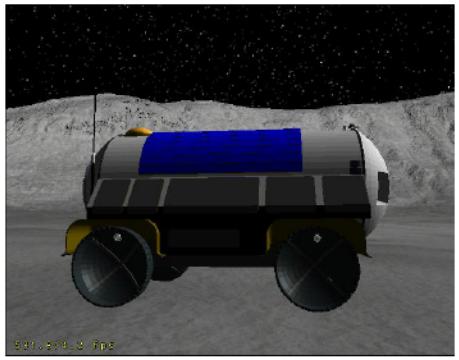
- **Vehicle Platforms:** Single and multi-vehicle simulations; parameterized model templates
- **Motion:** Vehicle mobility, arm models, wheel/soil dynamics – slippage/sinkage
- **Hardware models:** Kinematics, dynamics, motors, encoders, IMU, inertial sensors
- **Camera sensors:** Image synthesis for cameras with non-idealities, rover and terrain shadows
- **Environment:** SimScape synthetic, empirical & analytic terrains, ephemerides interface for sun position
- **Closed-loop visualization:** Dspace 3D graphics (CAD/auto-generated vehicle models), data monitoring
- **Closed-loop interface:** C++ & Python interface for sim. configuring and closing the loop with software; Stand-alone Monte-Carlo capability.
- **Faster than real-time:** 6x dynamics, sub-second camera image synthesis
- **White and black box simulation modes**

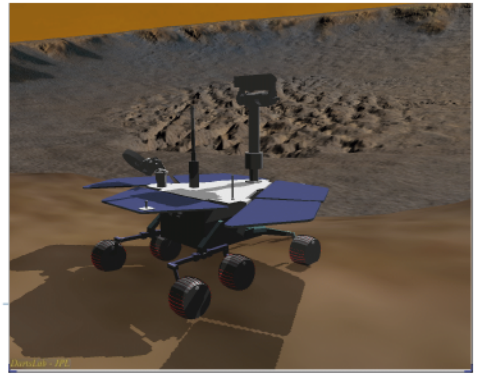
Example Vehicle Models

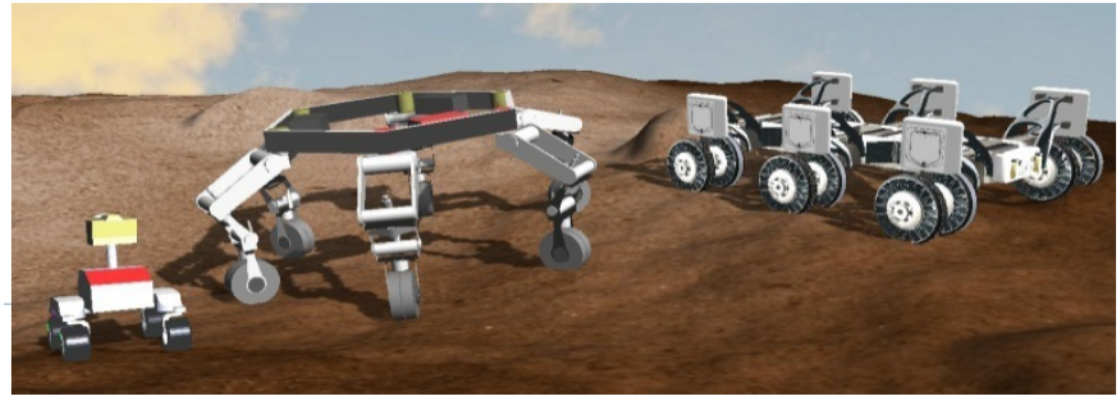






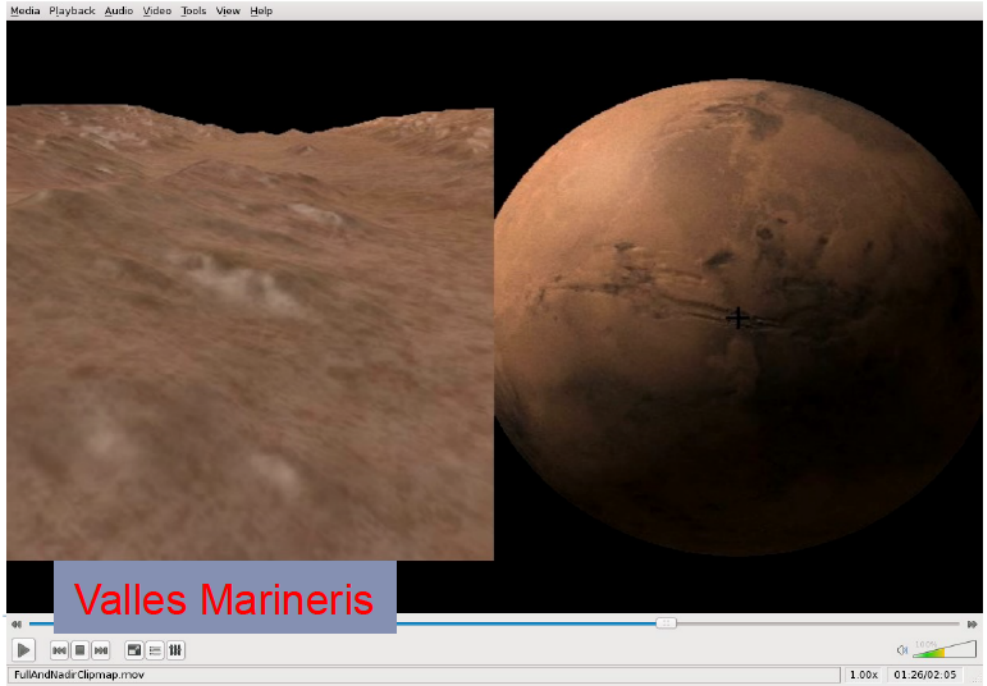






Continuous Level of Detail Visualization of Large Terrains

- Planet scale terrain models at meters level resolution
- Continuous level of detail for run-time management of graphics polygon complexity
- Victoria Crater data above is 1m resolution.
- Full Mars MOLA data is approximately 500m resolution – 2 billion sample points.



Closed-Loop Mobility & Manipulation Simulator

MSL Rover: 29 degrees of freedom (10 commandable actuators, 6 drive and 4 steer), full rocker bogey suspension, differential bar

MSL Arm and Mast: 13 dof simulation, Actuator models (DC motors, Gearing, Brakes, Friction, Backlash).

Faster than real time performance with flight software in the loop

MSL rover

ATHLETE Simulator:

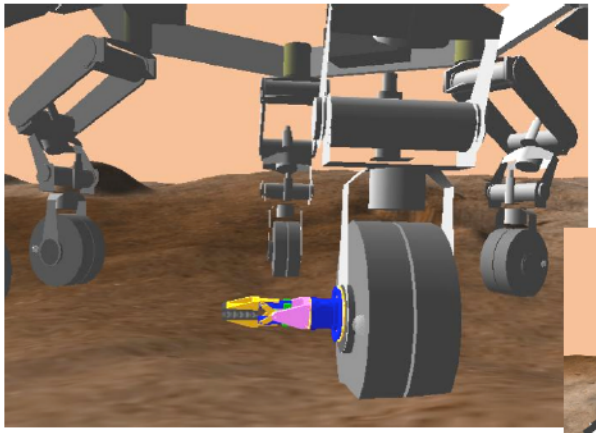
Athroams simulation created for ATHLETE task to provide high-fidelity vehicle simulation.

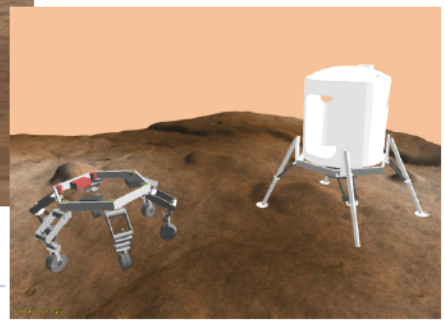
Interfaces implemented so that Flight Software can close the loop with Athroams as if it was a real ATHLETE rover.

O(20) degrees of freedom.

Operates at up to 20x real time.

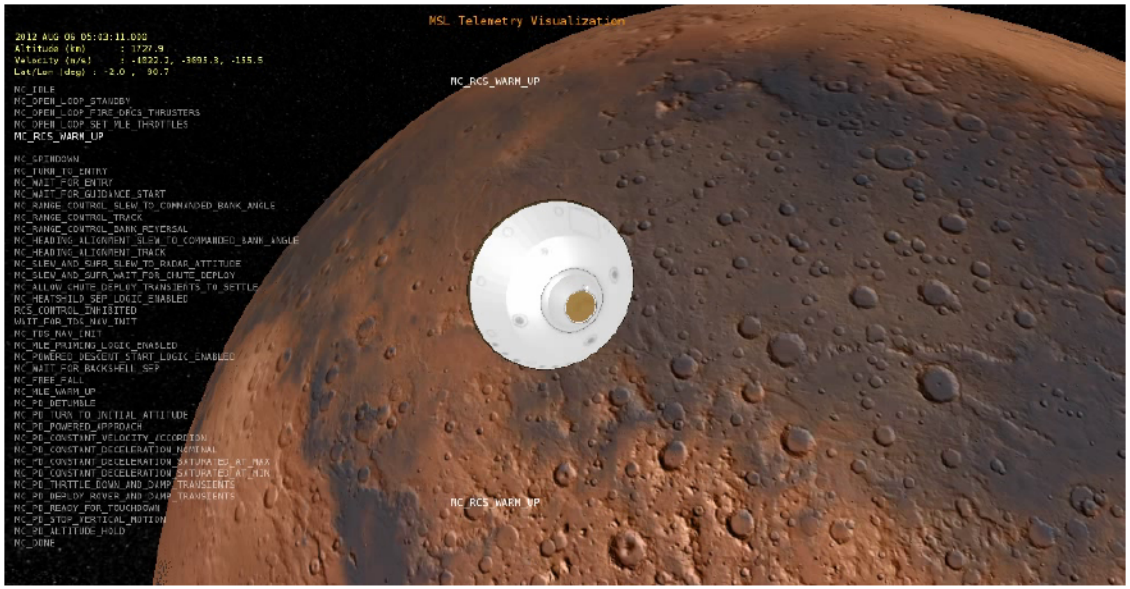
ATHLETE rover

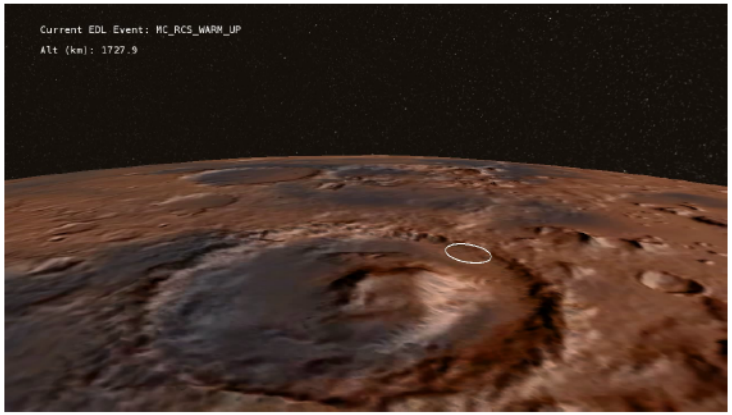




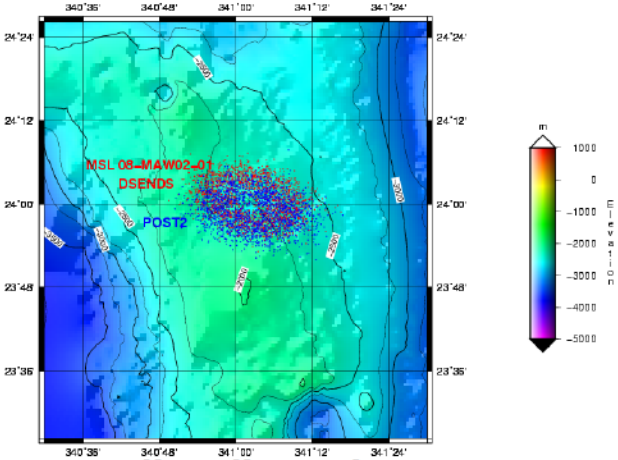
Mars Missions

MSL Entry, Descent and Landing

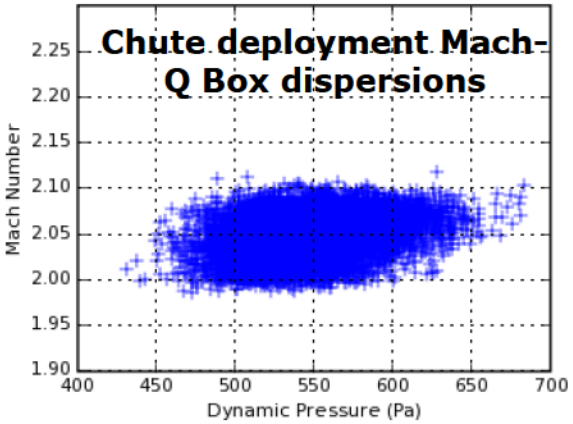




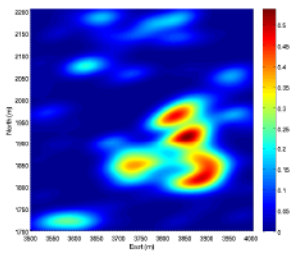
MSL Entry Guidance Performance



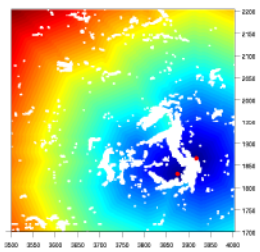
Landing dispersions at Mawrth



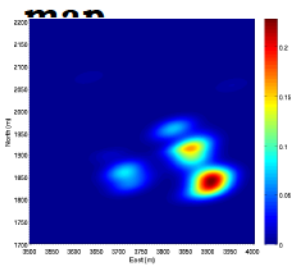
Landing/Mobility Performance & Risk Analysis



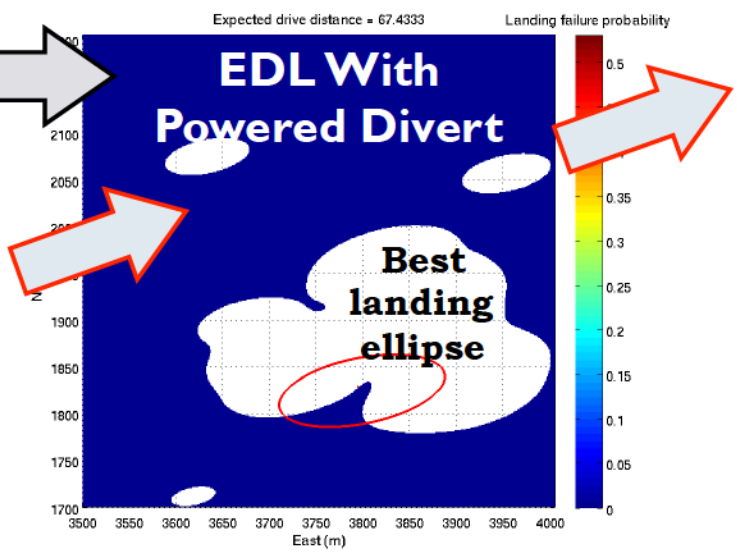
Landing Failure probability map



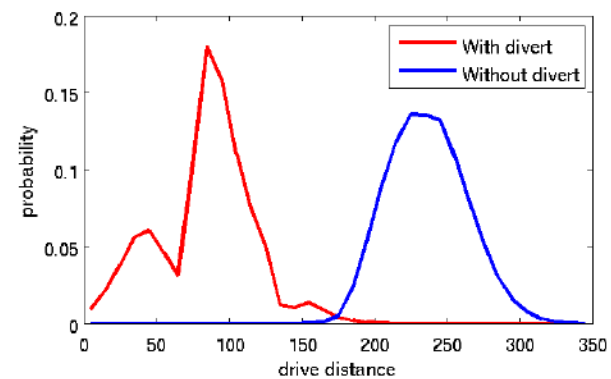
Mobility cost-to-go map



Landing Failure probability map

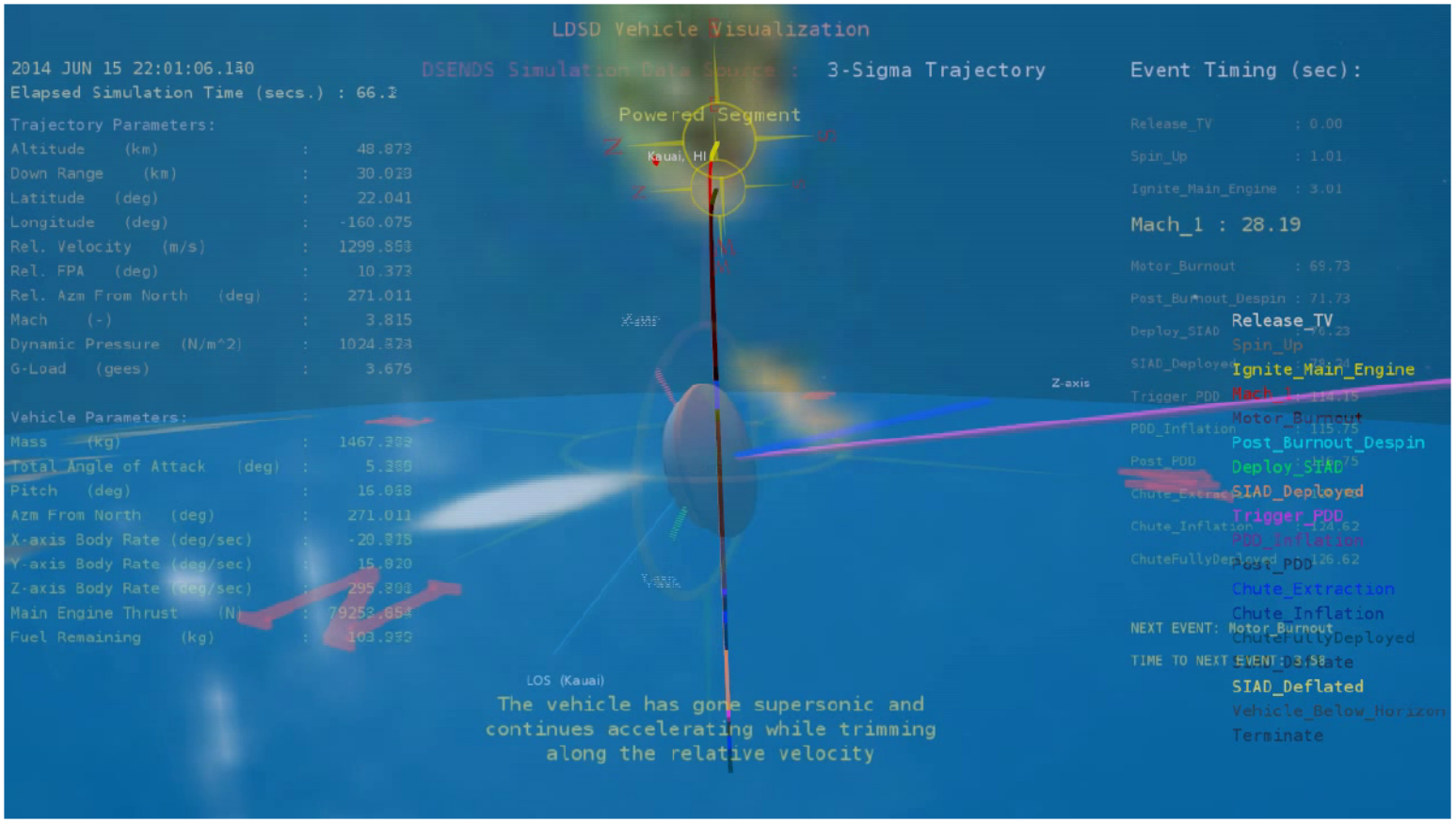


Distribution of rover drive distance



➔ Significant reduction with divert (71% in this example)

Earth Missions

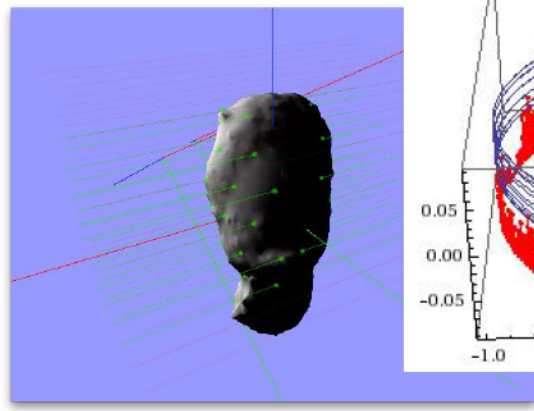


LDSD Test Flight Program

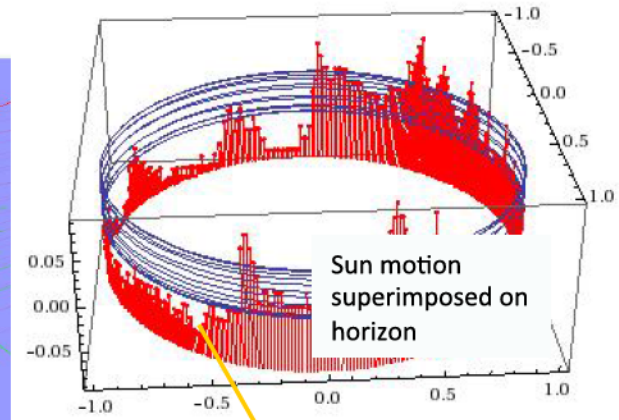
Geometric Analysis

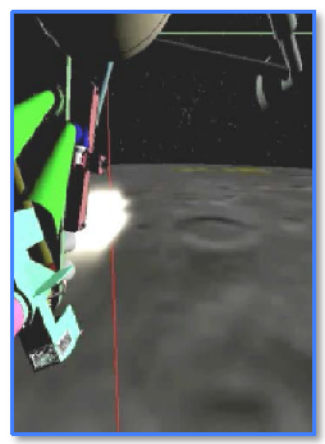
Line-of-Sight, View, Collision & Horizon Analysis

- Surface description comprised of irregular meshes as well as Digital Elevation Maps
- Complex multi-body spacecraft solid models
- Synthetic generation of navigation cameras views and lidar
- Sun angle, horizon analysis
- Communication link analysis

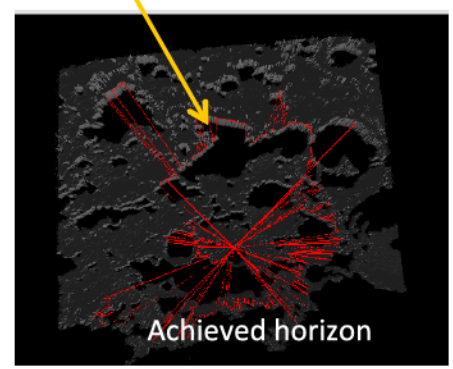


Ray-to-body intersections



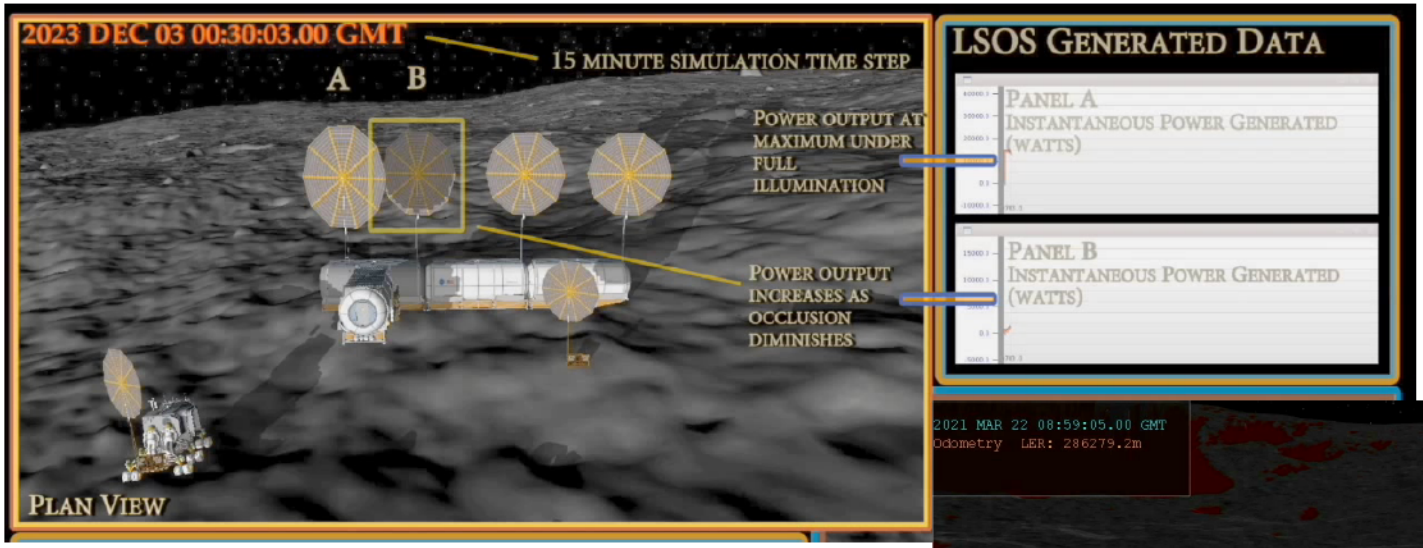


Descent camera view

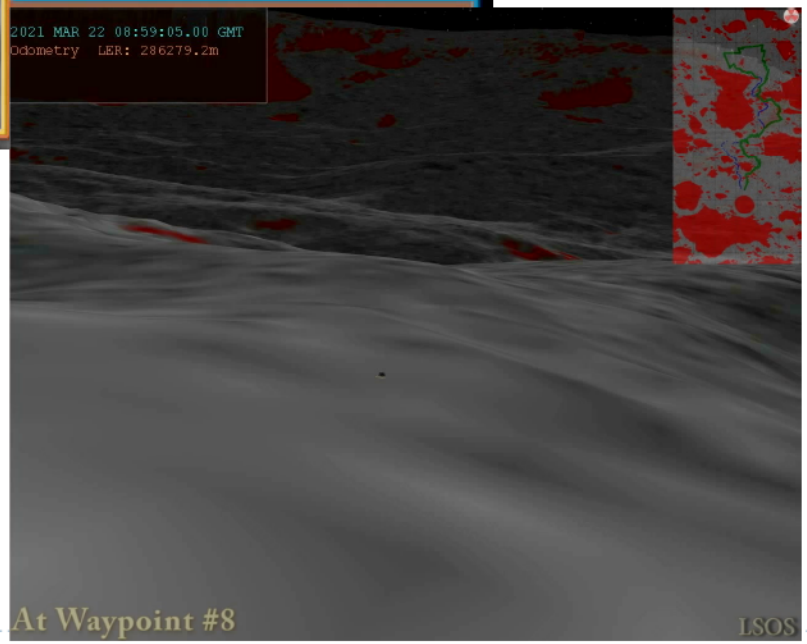


Lunar Surface Operations

Lunar Habitat Power Profile



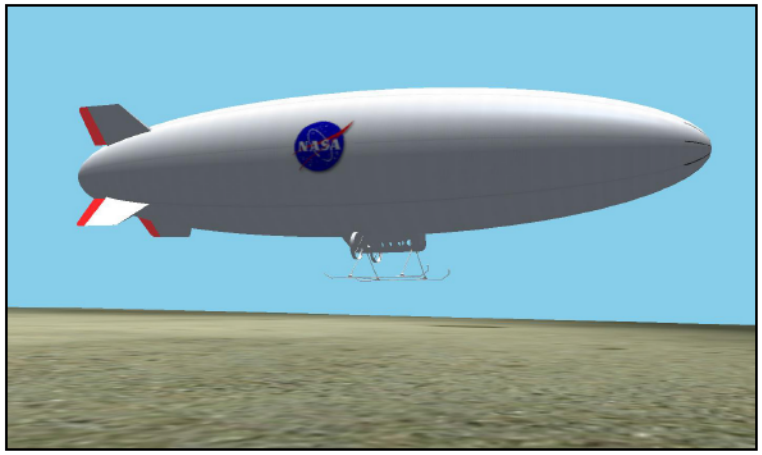
Lunar South Pole Rover Traverses

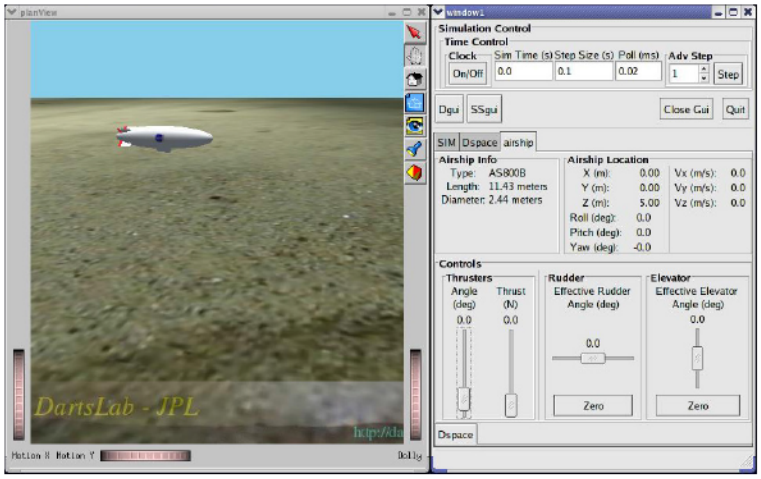


- Model surface systems (geometry, kinematics, dynamics, power, radiation, communications, temperature, etc.)
- Model lighting/shadows
- Simulate operational scenarios, assess technologies, plans
- Produce visualizations for review, analyses and simulation data
- Support analog field trial operations

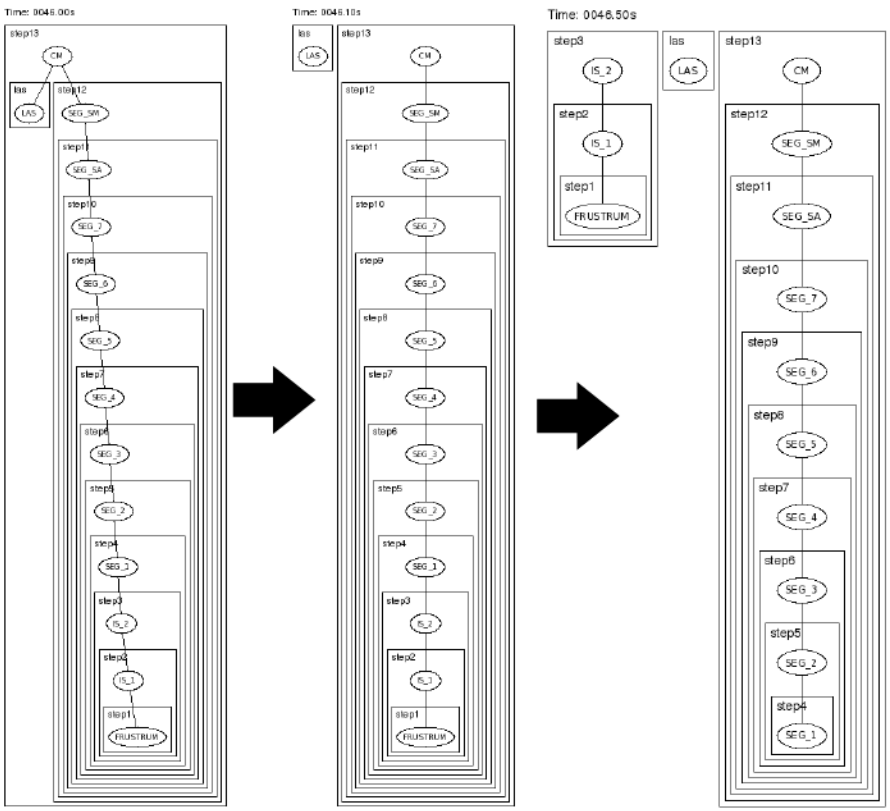
Airship Simulation

- Airship simulation created for Titan Aerobot Study
- Uses new Dshell aerodynamic models for lift, fin forces, buoyancy, and virtual mass effects.
- Includes interface (GUI) to control airship like hand-held RC controller.
- Implements on-board down looking cameras

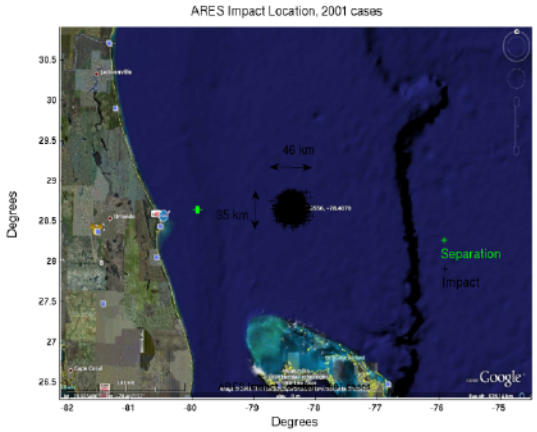




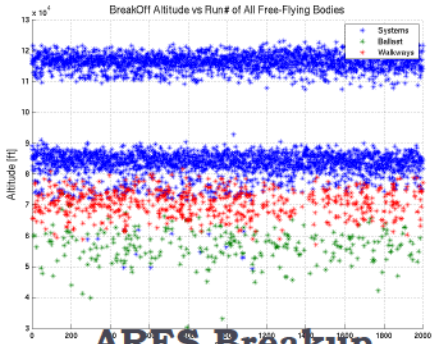
Launch Vehicle Breakup Analysis



ESMD/ARES Vehicle Breakup from one to hundreds of pieces



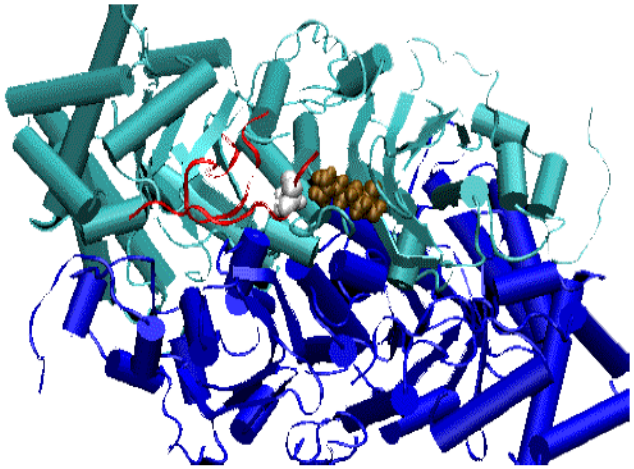
ARES Impact Footprint
(Nominal Separation Monte Carlo)



ARES Breakup Altitudes
(Nominal Separation Monte Carlo)

Coarse Grained Molecular Dynamics

- First applicability of SOA MBS approach to coarsen and speed up Molecular Dynamics simulations.
- Developed ***NEIMO*** internal coordinates molecular dynamics tool
- Extensively used in protein structures and dynamics, materials and nanotechnology (NSF Grand Challenge Project)
- Underlying algorithms in use at NIH, Scripps, Rice, Stanford etc.
- Several journal articles that advanced the state of the art
  - Constrained Noose-Hoover, Fixman's Potential etc.
- Continued collaboration with City of Hope
- Collaborated with Sandia National Laboratories to enhance their classical MD tool ***LAMMPS*** to include multibody based methods.
- Currently funded by NIH



Conclusions

- DARTS/DSENDS is a modular, flight-tested, physics-based modeling/simulation tool used for :
  - Standalone simulations & testbeds
  - Large-scale Monte-Carlo/Parametrics
  - Visualization
- Highly efficient, robust, verified kinematics and dynamics engine with faster than real time performance
- Python-based user interface, C++ object oriented code
- State of the art algorithms for rigid and flexible body dynamics
- Reusable architecture of multi-mission support
- Easy interface to sensor and actuator models
- Significant body of work in multibody dynamics
- Diverse applications ranging from rovers, arms, spacecraft to molecular dynamics systems
- Multi-Mission Use
- In use at JPL, JSC-MOD

marco.b.quadrelli@jpl.nasa.gov

Special thanks to DARTSLAB team members:
Abhi Jain, Bob Balaram,
Jonathan Cameron, Chris Lim,
Steven Myint, Havard Grip, Hari Nayar,
Marc Pomerantz, Calvin Kuo