AIAA Atmospheric Flight Mechanics Conference and Exhibit
5-8 August 2002, Monterey, California

AIAA 2002-4413

System Engineering Challenges of Real-Time Simulation for Mars Smart Lander Entry, Descent and Landing

Bryan J. Martin*
Jet Propulsion Laboratory, Pasadena, CA
David. A. Henriquez†
J. Bob Balaram‡
Garett. A Sohl†
and
Marc. I. Pomerantz§

DSENDS is an in-development spacecraft simulator for Entry, Descent, and Landing, developed initially through DNP funding at JPL and currently slated for use in the Mars Smart Lander mission as an analysis and verification tool. Based on the JPL-developed* Dshell *simulator, the need to consolidate and combine tools from disparate sources and with varying degrees of completeness has presented unique challenges. This paper discusses some of those challenges and and their solutions.

Introduction

The Jet Propulsion Laboratory (JPL) is developing a high-fidelity, real-time spacecraft simulator for Entry, Descent and Landing (EDL) on planetary and small bodies. This simulator, DSENDS (Dynamics Simulator for Entry, Descent and Surface landing), is an EDL-specific extension of the JPL Darts/Dshell multi-mission spacecraft dynamics and devices simulation toolkit which is in use by missions such as Cassini, Galileo, SIM, and Starlight. The broad scope of possible missions for this tool requires that we maintain a high level of generality and flexibility in our system design. The first major test of the DSENDS simulator is in support of the Mars Smart Lander mission, which will use DSENDS in a real-time environment (**Mars-DSENDS**).

The DSENDS simulator interfaces with three general classes of tools. *COTS* tools such as StethoScope,[1] daVinci,[2] Graphviz,[3] and Matlab Simulink[4] require use of a stable and standardized API. *JPL internal* tools such as the visualization programs Dview and Dspace, are integrated with the Dshell environment yet run as external processes using a coordinated, semi-asynchronous form of IPC. Finally, *instrument, system component, and environmental models* developed by disparate sources within and without JPL each present challenges due to the need to develop a unique interface or wrapper that takes into account that they are typically mission components that are under development.

Within this paper we discuss the integration and system engineering issues for a selection of the above elements into the DSENDS simulator.

Spacecraft Dynamics

The multi-mission capabilities of Darts/Dshell[5] allow easy reconfiguration of DSENDS to simulate a wide range of spacecraft configurations. Mass properties, thruster sizes and thruster locations can be changed to reflect different design concepts. DSENDS simulations have been created based on an early '07 prototype design and the current Smart Lander reference design. Even though the current reference design is in flux, DSENDS provides the flexibility needed to evolve with any changes.

Dynamic Spacecraft Reconfiguration

One significant challenge in EDL simulation is managing the transitions between the multiple phases of EDL, which often require a structural change in the simulation model to reflect new spacecraft configurations. Heat shield separation and parachute deployment are examples of these transitions. These changes

*Senior Staff Engineer, Engineering and Science Directorate, Autonomy and Control Section, Simulation and Verification Group.
†Staff Engineer, Engineering and Science Directorate, Autonomy and Control Section, Simulation and Verification Group.
‡Senior Staff Engineer, Engineering and Science Directorate, Mobility Systems Concept Development Section, Telerobotics Research and Applications Group
§Member of Technical Staff, ARCO Service Corporation

are conceptually simple, but require automated methods to maintain the continuity and accuracy of the end-to-end simulation by copying appropriate subsets of the spacecraft state, often with affine transformations.

DSENDS allows dynamic reconfiguration of the spacecraft model and correctly propagates the old system state to the new model using components of a state machine (see section *Automation*). This ability to reconfigure the simulation model allows DSENDS to provide varying levels of fidelity during each phase by changing the complexity of the model, and also to track multiple objects as they separate from the main spacecraft. Integration step size can also be varied during the simulation, allowing the user to choose particular phases to simulate at high fidelity while maintaining continuity with the rest of the simulation.

Tether Dynamics

After parachute deployment, the flight train system includes several flexible tethers[6] which combine at a single point at one end (the *confluence point*). Our tests of simulation methods for these tether lines are based on a Pathfinder-like system, which includes a backshell-lander system involving three lines and a single confluence point (see Figure 1). A similar system is used for the backshell-parachute tether line system.

A visco-elastic model for each individual tether line is used, and single-ended constraint equations are used to maintain kinematic consistency. The constraint forces imposed by the individual tethers dictate the motion of the confluence point as the lines stretch, contract, and go slack. The high frequency dynamics of the tether lines requires a small integration step size to insure stability. Although variable step integrators allow the selection of error accuracy, they do not provide any guarantee of computational cost. This can prove unacceptable in a real-time environment, where upper-bounds on computation impose severe constraints on computation time.

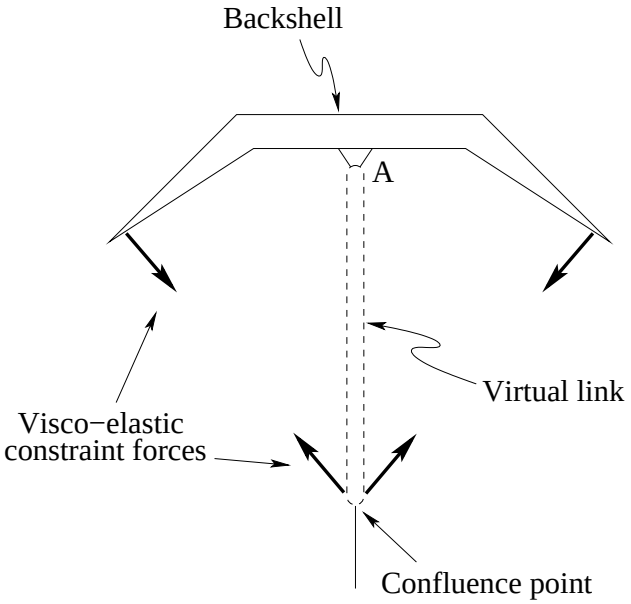

Fig. 1 Confluence point dynamics

Our early testing of visco-elastic tether systems showed similar qualitative results for both fixed- and variable-step integrators, when the fixed-step integrators used sufficiently small step sizes. These qualitative behaviors may be adequate for general simulation or for limited real-time testing, although integration error bounds are significantly higher. Shown in Figure 2 are some simulation results for a triple bridle system.

Simulation speed of the variable-step techniques is highly dependent upon the state of the tethers (see Figure 3). Variable step integrators reduce the step size when a tether transitions between slack and taut (which occurs mostly between $t = 1$ and $t = 5$ in Figure 2). This provides increased accuracy at the cost of simulation speed, making the variable step integrators slower than a fixed step near these transitions. However, as the system settles and the tether lines remain taut, a variable step integrator can increase the step size and achieve faster performance compared to a fixed step method ($t > 6$ Figure 2).

These spring-damper tether models have been incorporated into the DSENDS simulation for use in modeling the tether dynamics of the parachute and aft shell. DSENDS currently uses a fixed step integrator and requires a manual reduction in step size when using the high fidelity tether models. This step size modification is automated by the state machine (see section *Automation*).

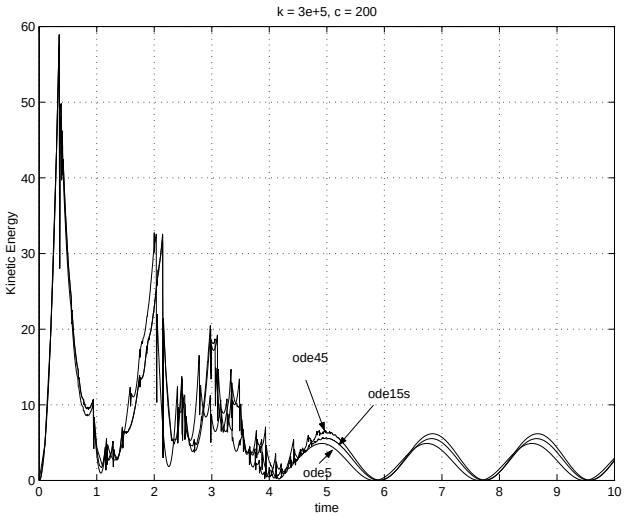

Fig. 2 Tracking results for different integration schemes.

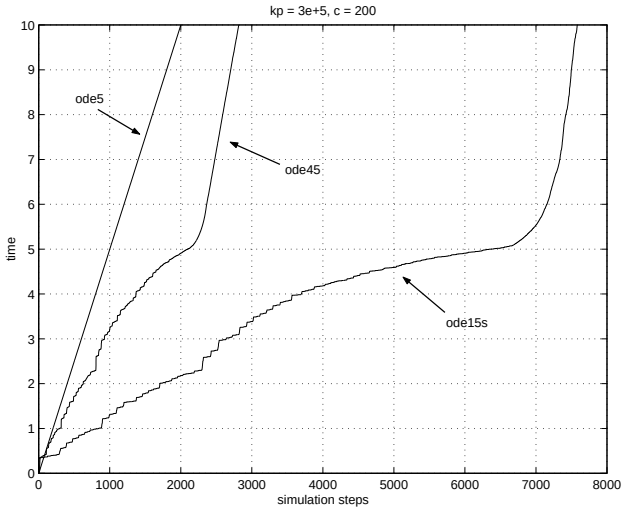

Fig. 3 Integration effort for different integration schemes.

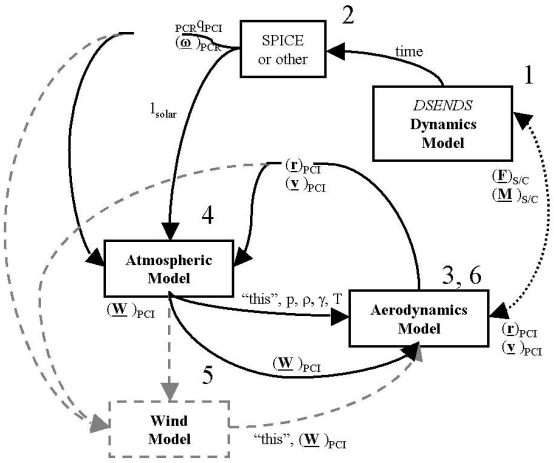

Fig. 4 Connectivity between aerodynamic models.

Aerodynamic Modeling

Accurate aerodynamic forces and moments are fundamental for an Entry, Descent and Landing simulator. During EDL the system will experience aerodynamic forces as it traverses from the thin outer atmosphere (governed by free molecular flow mechanics) to the atmosphere at the landing site (governed by continuum fluid flow mechanics). Also, the aerodynamic system decelerates from hypersonic velocities (in which molecular dissociation dominates the boundary layer) to subsonic velocities on the order of minutes. Thus, the DSENDS simulator must accommodate such disparate flow and velocity regimes in a real-time environment without imposing artificial discontinuities.

One significant challenge in this area has been to leverage and incorporate existing aerodynamic models from legacy code used for trajectory simulations (e.g. POST[7]). These are well suited for high-fidelity Monte Carlo mission analysis but were never intended to operate in a real-time simulation environment. Although these packages tend to be monolithic, it is possible to isolate the critical functions in these validated and trusted codes that are used to compute aerodynamic forces and moments. These functions can then be imported into the DSENDS simulator, so long as they do not violate the real-time requirement and can be hosted as a model in the DSENDS aerodynamic modeling architecture.

Consequently, the DSENDS team had to architect the manner in which aerodynamics are modeled to allow real-time operation of inherited aerodynamics models. In order to meet these requirements, DSENDS aerodynamics modeling architecture decomposed the aerodynamics into three separate models: *Aerodynamics Models* which encapsulate the aerodynamic coefficients for the given aerodynamic system, *Atmospheric Models* which encapsulate the ambient atmospheric conditions, and *Wind Models* which produce any local wind velocity not captured in the Atmospheric Model. Common interfaces and functional requirements have been made for these three groups of models so that they can correctly inter-operate. The fidelity and computational overhead of an aerodynamic model then depends on which set of models were instantiated for a given DSENDS EDL simulation. Also, many different kinds of legacy codes can be encapsulated in one of these three types of models.

Currently, DSENDS has the following catalog of models:

- Aerodynamics Models
  - Linearized, Axisymmetric Aerodynamics
  - Interpolated, Axisymmetric Aerodynamics
  - Viking-based, Hypersonic Aerodynamics developed at LaRC (includes Mars GRAM 3.37)
- Atmospheric Models
  - Fitted Reference Martian Atmosphere
  - Mars GRAM
- Wind Models
  - Under Development

The Viking-based hypersonic model listed above was intended for use in extremely high-fidelity atmospheric entry simulations, and was not optimized for use in a real-time environment, although we fully expected it to be efficiently designed and implemented. Our integration and testing efforts therefore concentrated

on efficient use of the library as well as validation and timing tests.

Our preliminary tests incorporate the LaRC hypersonic model into Mars-DSENDS using an axis-symmetric body model. The flight path and time-dependent dynamic quantities (acceleration, dynamic pressure, etc) were compared against the pre-existing low fidelity models previously used in DSENDS . These tests have been performed on hardware running a non-real-time OS (Solaris), and have been used only to measure theoretical or relative performance. The low-fidelity model showed qualitatively similar results but did not have the frequency content of the LaRC model.

The following sections discuss the requirements and implementation of the DSENDS aerodynamics modeling architecture.

System Engineering Issues Related to Aerodynamics Modeling

Requirements for Atmospheric Models

Atmospheric models are either simple algebraic models or based on a database of atmospheric properties (density, pressure, temperature, wind velocity, etc.) that vary with time. Spatial-temporal variations in the local atmospheric properties are due to the uneven, radiative heating of the planet and the influence of planetary terrain, and consequently vary with local time (e.g. planetary attitude), solar longitude (i.e. season) and with planetary latitude and longitude.

Although terrain information may be convolved into an atmospheric model, the atmospheric models are not expected to produce terrain elevation data. Synchronization between atmosphere and terrain models will be performed externally to the atmospheric model and is expected to be solely through relative position of the aerodynamic system (latitude, longitude and height) with respect to a planet-centered, planet-centered rotating frame (PCR).

Since atmospheric models are relative to the planetary surface (i.e. vary with latitude, longitude and height above the surface), each atmospheric model must have access to a function or object to convert its planet-centered, rotating frame (PCR) data into a planet-centered, inertial frame (PCI), which makes the data more compatible with the rest of the DSENDS EDL simulation.

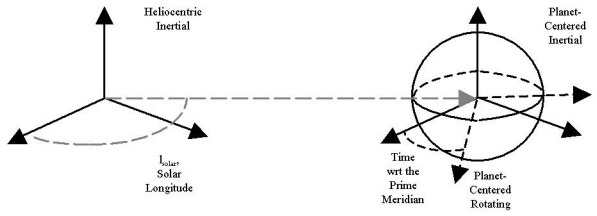

Fig. 5 Celestial body frames.

As mentioned above, atmospheric models are expected to have data that varies with solar longitude and local time. However, atmospheric models are also not expected to maintain planetary attitude and rate, which are instead supplied by an external object that maintains the system states (e.g. a planetary attitude dynamics object in DSENDS or SPICE object).

Atmospheric models may produce wind velocities as a function of the relative position of the aerodynamic system above the planetary surface. As mentioned earlier, the atmospheric model will have access to a function or object to transform the wind velocities into a standard PCI frame. The atmospheric model must also correctly add the planetary rotation to wind velocity so that the inertial wind velocity represents the total kinetic energy of the flow field encountered by the aerodynamic system.

There are classes of atmospheric phenomena that can be modeled by a set of ODEs. The real-time requirement for the Mars-DSENDS simulator is not expected to support the use of high-fidelity, reactive, ODE-driven models. However, the spatial-temporal data produced by ODE-based atmospheric models may be used as an input to atmospheric models that interpolate spatial-temporal data. In other words, the DSENDS atmospheric models only require an algebraic dependence on inputs, regardless of whether the underlying atmospheric model uses interpolated data or an algebraic equation.

Requirements for Wind Models

Simple wind models are those that model the diurnal and semi-diurnal cycles in the wind (due to diurnal solar radiative heating) as the sum of sinusoidal functions in time. High fidelity wind models in the atmospheric sciences use a set of spatial-temporal partial differential equations that are entered into an estimation process (such as a Kalman Filter) where atmospheric and terrain measurements are used to refine the results so that a realistic wind field is produced with spatial-temporal variations.

Many of these effects are available with varying degrees of fidelity in current atmospheric models such as Mars GRAM. It is for this reason that DSENDS wind models need only compute local perturbing wind velocities for a specified ground-fixed atmospheric volume.

This approach facilitates correlation between a wind model and a high fidelity atmospheric model without the same wind effects being doubly applied or erro-

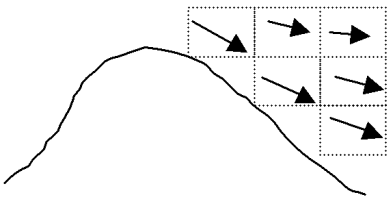

Fig. 6 Semi-diurnal flow near a mountain.

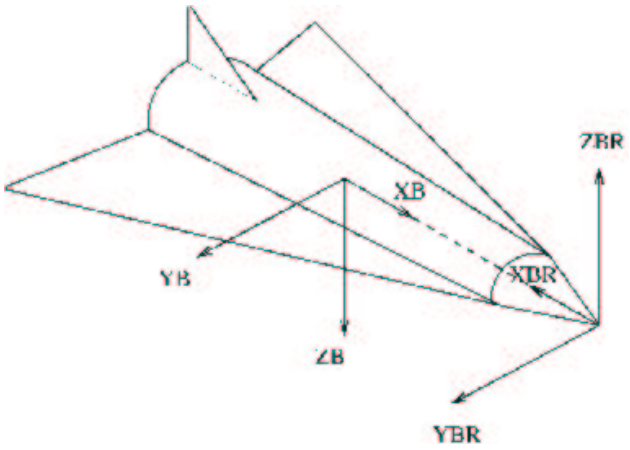

Fig. 7 Aerodynamic body reference frames

neously canceled. Furthermore, DSENDS wind models maintain parameters that define the PCR-fixed region of influence for which the wind model is valid. The DSENDS wind models also provides public functions or object to return these parameters so that the perturbed wind velocity can be correlated with the planetary terrain for the same PCR-fixed volume. These parameters can either be hard-coded into the models or set by the user. Each wind model is passed to a function or object that converts its PCR velocity vectors into the PCI frame, so that the output is compatible with the DSENDS environment. Since only perturbations are computed in DSENDS wind models, they are not *required* for running a DSENDS EDL simulation. Spatial-temporal wind models have a set of ODE's that must be solved for the entire wind field throughout its extent at various grid points. The perturbed local wind velocity is then queried for a specific location. If the specified location is between the grid points, an interpolated value is computed. Such a model could be a large computational burden for DSENDS and may violate the real-time requirement for the DSENDS simulator. However, since these models are required for modeling localized wind effects near mountains, craters, or valleys, DSENDS wind models use a bounded region of influence such that the computation overhead does not violate the DSENDS real-time constraints. In addition, DSENDS wind models are given access to functions or objects that allow maintenance of continuous states and that allow interpolation between grid point solutions.

The subsequent discussion will focus on the requirements for implementing a simple spatial wind model, which does not preclude the implementation of more complex wind models. A perturbing wind model can be developed using a PSD that only varies with altitude and does not require solving a set of ODE's. After the coefficients to the discrete Fourier series are computed, the wind model can convert a PCI position into an altitude and compute the perturbing wind at that altitude. Additionally, truncating the Fourier series easily bounds the computational overhead of this wind model, although at the cost of model fidelity. The values used to compute the Fourier series (e.g. the fundamental frequency, number of harmonics, etc.) can be model parameters, thereby making such a wind model well suited for generic application.

Requirements for Aerodynamics Models

Aerodynamic coefficients are modeled as a set of dimensionless coefficients that are relative to a body-fixed reference frame. The reference frame defines a coordinate system in which forces and moments are applied, and is very likely to be different from the mechanical reference frame. Therefore, aerodynamics models maintain an affine transform that allows aerodynamic forces to be mapped from the aerodynamic reference frame into the mechanical reference frame.

Figure 7 shows aerodynamic reference frame, in which the X-axis is along the axis of symmetry and is nominally anti-parallel to the free stream velocity when the total angle of attack is zero (i.e. X points into the "wind"). The Y-axis is defined as the "pitching axis" and points such that a positive rotation (i.e. counterclockwise) about Y shall produce a positive angle of attack. Z is the cross product of X and Y.

So far, the discussion on modeling requirements for DSENDS aerodynamics models has been general. The following discussion will focus on the requirements needed for modeling axisymmetric aerodynamic systems. Axisymmetric aerodynamics models are a class of models that are not sensitive to an angular displacement and angular rate about the axis of symmetry. Although a seemingly limiting choice of aerodynamic systems, axisymmetric aerodynamics models can be used to model most EDL aerodynamic subsystems (e.g. entry vehicle, parachute, etc.). Therefore, DSENDS development focused on developing these models, but

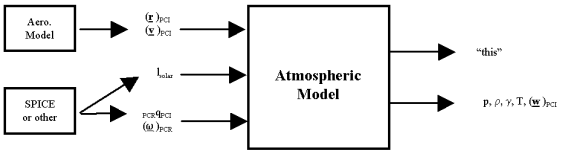

Fig. 8 Atmospheric model interfaces

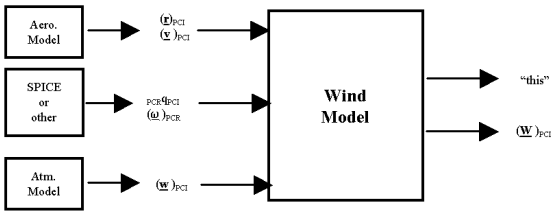

Fig. 9 Wind model interfaces

the interfaces and design of the DSENDS atmospheric and wind models does not preclude future lifting body (asymmetric) aerodynamics models.

Although most aerodynamic systems experience both positive and negative total angles of attack, most aerodynamic coefficients databases for axisymmetric vehicles tend to only list coefficients for positive angle of attack. This convention assumes that the aerodynamic forces and moments are applied in a frame such that the total angle of attack is positive. As a result, an axisymmetric aerodynamics model determines its aerodynamic reference frame at runtime. The frame maintains X along the axis of symmetry but Y is chosen such that the total angle of attack the angle is positive; Y and Z directions will vary with the incident flow field. The transform maintained by the axisymmetric aerodynamics model is used to define the X with respect to the mechanical reference frame.

The axisymmetric aerodynamics model is given access to functions or objects that allow it to ascertain the inertial state of the aerodynamic system (6 DOF). Furthermore, the axisymmetric aerodynamics model has access to functions or objects that allow it convert its 6 DOF state knowledge into total angle of attack within the aerodynamics reference frame.

Software Engineering for Atmospheric Models

DSENDS defines a C++ model class called a flow model that only has an algebraic dependence on its input and no continuous states. The atmospheric models communicate with the aerodynamics models via inherited class types called flowIns and flowOuts (as do all aerodynamics models described in the next sections). These types are defined in the DSENDS model base class, from which the DSENDS flow model class is derived, and allow DSENDS models to read and write data to signal buffers. Two DSENDS models can communicate with each other if they are connected to the same set of signal buffers. Therefore, the DSENDS atmospheric model reads the PCI position and velocity vectors, the quaternion from PCI to PCR and the angular rate of the PCR frame from its inputs, and it writes its address, the local density, the local temperature, the local pressure, the local specific heat ratio and the local wind velocity in the PCI frame to its outputs. (See Figure 8)

The DSENDS atmospheric model is associated directly with with an DSENDS aerodynamics model via a pointer. The aerodynamics model is a forcing function on an aerodynamic system and in order for it to compute the proper forces and moments to apply, it must be supplied the local atmospheric properties for the current inertial state of the aerodynamic system. As will be further explained below, the atmospheric model will be executed by a direct command from the aerodynamics model (i.e. via a public member function). The aerodynamics model always has the latest local atmospheric properties, so long as the aerodynamics model and atmospheric model are connected to the same set of signal buffers and the aerodynamics model writes its PCI position and velocity prior to executing the atmospheric model.

Software Engineering for Wind Models

The DSENDS wind models were implemented using a base class that allows the model to maintain continuous states. The DSENDS sensor model base class was selected because it was assumed that the variations in the local perturbing wind velocity have time scales that allow it to be loosely coupled with the aerodynamic forces and moments (i.e. minutes or hours). Consequently, there is no need to simultaneously solve a wind model ODE and the equations of motion of the aerodynamic system.

The wind models communicate directly with the aerodynamics models, and read the PCI position and velocity vectors, the quaternion from PCI to PCR, the angular rate of the PCR frame and the atmospheric wind velocity from its inputs, writes its address, and the local perturbed wind velocity in the PCI frame to its outputs. (See Figure 9)

The DSENDS wind model is associated directly with an aerodynamics model. The aerodynamics model is a forcing function on an aerodynamic system and it must have the local wind velocity vector for the current inertial state of the aerodynamic system in order

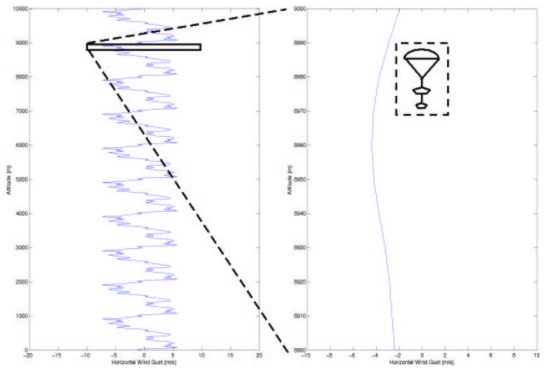

Fig. 10 Wind model interfaces

to correctly compute the free stream velocity vector and therefore the correct angle of attack. If the wind model is connected to an aerodynamics model, the aerodynamics model will execute the wind model via a public member function. The aerodynamics model must execute an atmospheric model prior to executing the wind model. If no wind model is connected (i.e. NULL address or address equals the atmospheric model address) the aerodynamics model will skip execution of the wind model and will simply read the wind velocity on its inputs.

Software Engineering for Aerodynamic Models

The DSENDS aerodynamics models were implemented using the DSENDS actuator model base class, which allows the model to apply forces and moments on the dynamics model.

The aerodynamics model communicates directly with the atmospheric models and wind models, reading the address of the atmospheric model, the address of the wind model, the local density, the local temperature, the local pressure, the local specific heat ratio and the local wind velocity in the PCI frame from its inputs.

The aerodynamics model also writes its PCI position and velocity vectors to its outputs.

The aerodynamics model uses the addresses of the atmospheric model and the address of the wind model in order to associate and communicate with them. After the aerodynamics model gets the 6-DOF inertial state of the aerodynamic system from the dynamics model via a function call, it extracts its PCI position and velocity and writes it to its outputs, thereby providing inputs to the atmospheric and wind models. The aerodynamics model then proceeds to check its inputs for valid atmospheric and wind model addresses. If the address of the wind model is null, the aerodynamics model skips execution of the wind model. However, if the address of the atmospheric model is null, the aerodynamic model will print a critical error message and force the DSENDS simulator to exit.

After verifying that it is connected to a valid atmospheric model, the aerodynamics model executes the atmospheric model and then the wind model (if valid) using public member functions of those models. The aerodynamics model then reads the ambient atmospheric conditions and wind velocity from its inputs, uses its inertial velocity and wind velocity to compute the free stream velocity and angles of attack, and the uses ambient atmospheric conditions and the free stream velocity to compute the Mach number of the aerodynamic system. If the aerodynamics model uses an aerodynamic coefficient database, then the aerodynamics model must use some interpolation functions to determine the aerodynamic coefficients for the given aerodynamic system state. If the aerodynamic model uses some parametric aerodynamic coefficient model, the aerodynamic model then queries the coefficient model by some function call to get the salient aerodynamic coefficients. However the aerodynamics model accesses its aerodynamic coefficients, it takes the coefficients for the given angles of attack and Mach number and then computes the aerodynamic forces and moments to apply to the aerodynamic system.

Terrain modeling

Terrain models are used by DSENDS for a variety of purposes, including monitoring the height of the spacecraft over the terrain, determining instrument field-of-views, generating sensor responses for Imaging, Lidar and Phased-Array Radar terrain sensing devices, evaluating the spacecraft kinematic and dynamic response during touchdown, and as a visualization aide. The extent and resolution of the terrain needed by DSENDS varies by several orders of magnitude during the descent profile. At entry, a large swath of terrain is required at coarse resolution. As the spacecraft descends, narrower extents of terrain at increasing resolution are required. Furthermore, the specific locations at which the terrain is required varies as a result of changes in the spacecraft trajectory and pointing. For example, during hypersonic entry at Mars an unguided spacecraft has an a trajectory uncertainty of typically over 100km. Similarly, during descent on a parachute, the pointing of the spacecraft can vary by many 10's of degrees as the parachute interacts with the descent-induced and atmospheric wind.

The terrain models supported in DSENDS are comprised of Digital Elevation Maps (DEMs). DEMs are essentially 2.5 D data and suffice for EDL simulation needs. The DEM data come from various sources, and can combine real and simulated elevation data. At one level is planet-wide data that may have been obtained

by remote sensing from orbit. Topographic data is available, for example, for Mars using the MOLA data set or for Venus using the Magellan data set. At another level is purely synthetic data. This data, using simulated features such as craters and rocks overlayed on a base topography, is useful for performing system tests where real data is not available. Such synthetic data is available from a variety of sources.[8–10] Finally, we have data that is intermediate between the two, where low-resolution real terrain is synthetically enhanced to high-resolution. Enhanced data has been made available for certain specific landing sites of interest. In addition to DEM's we must also mention data relevant to defining planetary geoid data, which is required to support atmospheric models. DSENDS supports both a DEM type representation of geoid data as well as those based upon a spherical harmonics expansion.

A brute-force approach to preload a DEM of an entire region at high-resolution is not feasible. (A swath of terrain 200 km by 100 km at 10 cm resolution is $2 * 10^{14}$ pixels). Instead a phased approach is required, where terrain patches are dynamically loaded into the simulation based upon the needs of the particular phase of the simulation. The phasing requires a number of components:

- Dynamic generation of terrain patches at appropriate resolutions.
- Fetching of generated terrain for use by the simulation.
- Loading into special memory to facilitate real-time device response generation. This loading and unloading must be transparent to the real-time process implementing the device simulation. In DSENDS this is achieved by means of utilizing shared-memory segments between the real-time process and a another thread responsible for terrain memory management.
- Look-ahead prediction of required terrain segments to allow generation, fetching and loading requests to be queued and terrain to be available in a timely manner.

DSENDS currently implements an Instrument Terrain architecture to support the needs identified above. The architecture is shown in Figure 2.

Instrument Terrain Server

Terrain products are required within the EDL simulation to support a number of applications such instrument simulations, data monitoring modules, and 3-D visualization of the simulation. Examples of these are respectively, a terrain scanning Lidar instrument simulation, a monitor of the spacecraft height over the ground, and a 3-D view of the spacecraft approach to the landing site. The location, extent and spatial resolution of the terrain segments required to support these applications varies and is a function of the boresight, field-of-view, and the fidelity desired. For example, a Lidar with a steering mirror could require terrain anywhere within the field-of-regard provided by the mirror at a resolution that is a function of the instantaneous field-of-view (IFOV) of each pixel in the Lidar detector. A monitor of spacecraft altitude would require a small terrain segment located vertically below the spacecraft. A mouse-driven viewpoint generator for 3-D visualization would require terrain anywhere in the scene as the simulation user moves the view-point.

In all these cases terrain must be provided in a timely manner to the EDL simulator. This is especially important during real-time operations where instrument responses must be generated in synchronization with a real-time clock with no possibility of cycle-slips and consequent data loss. The option of having all of the terrain resident in memory for immediate access by the requesting EDL application is not feasible because of the sheer size of the data set required. Instead, a process of terrain generation (or enhancement in the case of synthetically augmented natural terrain) must be combined with terrain segment transport to the EDL simulator, and upload to the EDL simulator's memory. Each of processes identified has inherent delays that make timely access difficult. Fast generation of terrain using multiple processors, timely transport of data using fast network hardware and protocols, and real-time buffer and shared-memory segment management within the simulator are all ingredients in achieving timely terrain access. Moreover, as the segments needed by the application change, successive terrain segments must be generated as needed, uploaded to the simulator, and placed into memory in a timely and seamless fashion.

The Instrument Terrain Server (ITS) in the EDL simulator provides these functions to the various applications in the simulator. It incorporates the following functional elements:

- The ITS has a number of shared memory buffers which contain overlapping terrain segments. As the application requests terrain in the overlapping areas, buffers are switched in real-time to allow the application to access terrain in the new segment in a seamless fashion.
- The ITS uses a predictive model of terrain usage (called an Oracle) that allows it to predict

the extent and resolution of terrain segments required by the application. These predictive models are usually based upon a nominal EDL scenario and the current location and velocity of the ground "footprint" of the instrument/viewer field-of-view.

- The ITS uses the predictions from the Oracle, knowledge of terrain generation times, data transport times, and buffer sizes to sequence the generation, transport and upload of appropriately overlapping segments of terrain into the EDL simulator. The ITS manages the use (and reuse) of the real-time buffers, the extent of overlap, and provides a level of cache management (e.g. keep adjacent terrain segments in memory in case they are needed) to relieve the simulator from frequent interactions with the terrain generation/transport process. (Note that terrain generation can take many seconds, transport is usually a fraction of a second, and buffer management/swapping is done at simulation rates e.g 50 ms.)

- In addition the ITS provides backup terrain (with lower resolution and larger spatial extent) in case the generation/transport process fails to achieve the times predicted by its model, or if the Oracle prediction of anticipated application terrain request turn out to be wrong (e.g. if an unanticipated spacecraft motion causes the Lidar steering mirror to hit a hard-stop and thereby cause the Lidar to suddenly be viewing a ground area far removed from the normal scenario).

A prototype version of the ITS is operational incorporating a Terrain software object, shared memory buffers that allow seamless ITS and application terrain buffer access, and a preliminary version of the Oracle (implement in Tcl). The ITS is being used to support a single instrument simulation application at this point (the Lidar simulation). Planned work will allow the ITS to support multiple EDL applications and refine the Oracle's predictive capabilities.

Automation

Maintaining multiple, independent or interdependent, states of various spacecraft components can be a daunting task, especially if the mission design is either in flux or not yet complete. DSENDS requires an automated method which simplifies and partitions the various control, data gathering, visualization, and model switching functions involved with different components and mission phases. In addition, one of the common uses for end-to-end simulation is form performing Monte-Carlo or trade studies, which requires an automated mechanism for generating initial conditions, initializing and running the simulation, and gathering data. The next two sections briefly describe our tools for solving these challenges.

State Machine

The design of the DSENDS State Machine (SM) development was driven originally by the need to simplify dealing with different spacecraft stages as they separate, perform actions, or pass through different dynamic regimes, although its applicability has grown to include all automated functions involving the simulation. Since multiple, separate spacecraft models can be active simultaneously, and the SM must maintain the state of each spacecraft or spacecraft component independently while managing transitions between them, the SM has been designed to maintain multiple, simultaneous states. Our SM design is based on providing a subset of common capabilities in existing commercial products that provide similar functionality, while integrating seamlessly into the DSENDS environment. The primary abilities desired were:

- Automatic execution of user-functions during state execution and state transitions
- User-provided functions for testing of state transitions
- Multiple, simultaneous state capability
- Ease of definition and integration with DSENDS

In addition, the **GraphViz**[3] graph visualization package was used to provide automatic display of the connectivity of the state machine, as well as to show the current active states in real time. Figure 11 shows an subset of the SM driving the demonstration DSENDS simulation, showing names of the states and transition test functions, special functions like initialization (diamond) and termination of a state (rectangle), as well as the currently active states (green).

Batch Manager

DSENDS batch processing capabilities are centered around performing automatic data collection for regression testing, trade-space analysis, and automated testing. The batch processor can parallelize execution of cases over a heterogeneous collection of interconnected processors, resulting in fast execution and data collection limited only by the number of processors available.

The Batch Manager does for data collection what the DSENDS SM does for the simulation: it streamlines and simplifies the process of running simulations for the purpose of data collection and analysis, and

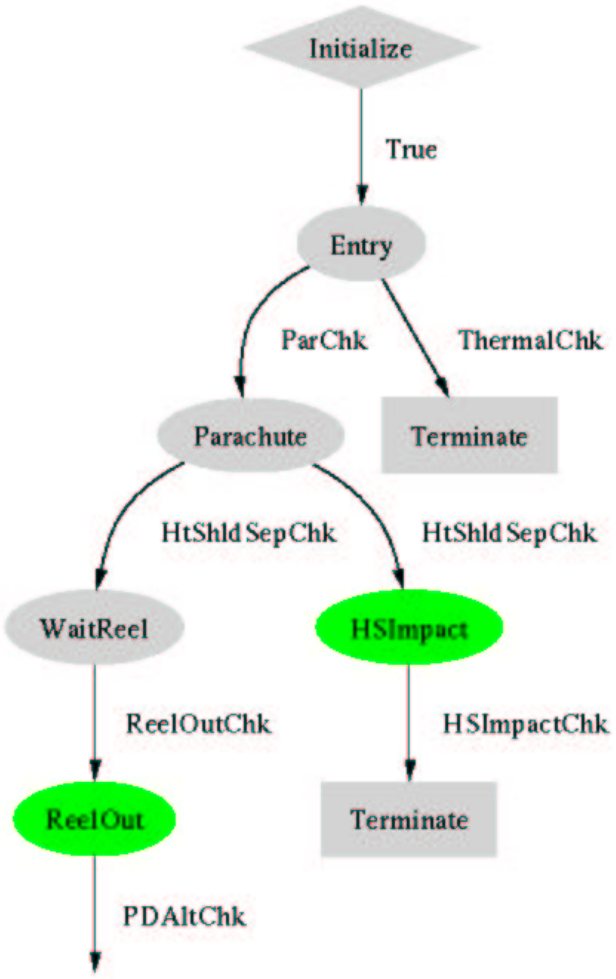

Fig. 11 State Machine Visualization

of generating answers to what-if questions. The batch manager provides a simple text-based interface that defines initial simulation states that are to be varied, such as mass of spacecraft components, fuel on board, entry angle, or spin rate. Then one defines the desired results to extract, which can range from fuel consumption, landing location, or landing error, to total horizontal delta-V, and so-on. Finally, a list of available resources on which to run the simulations is provided. The input and output categories both use user-defined readable names to identify the cases to run and the output data.

The Batch Manager executes one simulation on each computer, then monitors the simulations and collects data as each completes. Once a resource is free, the next available case is run on it. This continues until all cases have been run and all data extracted, then the batch manager shuts down. Data is saved to an output file along the way, in case the process is interrupted, and also displayed to the screen to show the

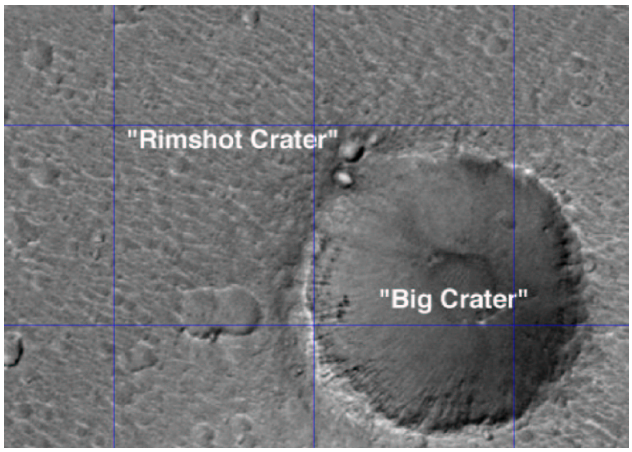

Fig. 12 Sample Terrain

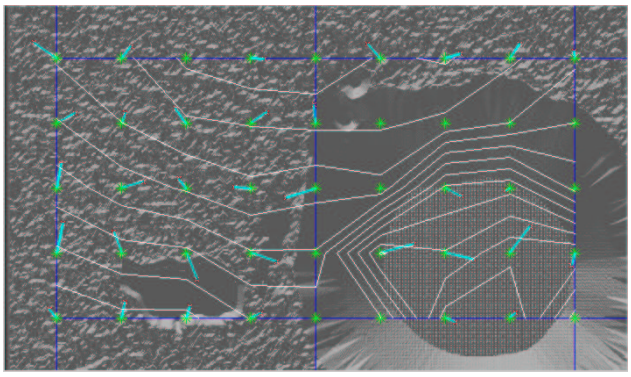

Fig. 13 Batch Manager Output Analysis

progression of each test case.

An example of the output is show in Figure 13. In this sample, a 1 by 2 km cratered area was gridded with desired landing sites (+). A detailed spacecraft model was used that included a guided powered landing capability, with some hazard avoidance. A digital elevation map from the JPL MIPL laboratory was used that corresponded to the imagery. The output picture combines fuel consumption contours (white lines) and deflection maneuvers (cyan lines). The output data also indicated which sites were infeasible.

Concluding Remarks

DSENDS will grow in the coming years to become a complete, end-to-end simulator for studying and validating missions involving entry, descent, and landing on planetary bodies. Although the main customer for this capability is currently Mars Smart Lander, DSENDS is also in use as a testbed for the JPL Miniature Optical Correlator, and as a concept-testing and validation tool in the PDC. The next year of development will focus on furthering implementation of the capabilities described in this paper, and completing validation of the tool and its components.

Acknowledgements

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration. References herein to any specific commercial product, process or service by trademark, manufacturer or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

References

[1]Real-Time Innovations, "StethoScope," http://www.rti.com.

[2]b-novative, "daVinci," http://www.b-novative.com/ products/daVinci/daVinci.html.

[3]AT&T, "GraphViz," http://www.research.att.com/ sw/ tools/ graphviz/.

[4]The Mathworks, "Matlab," http://www.mathworks.com.

[5]Jain, A. and Simulation & Verification Group, "Darts/Dshell - a spacecraft mission simulator," http://dartslab.jpl.nasa.gov.

[6]Smith, K. S., Peng, C.-Y., and Behboud, A., "Multibody Dynamics Simulation of Mars Pathfinder Entry, Descent and Landing," JPL D-13298, April 1974.

[7]Center, L. R., "POST: Program to Optimize Simulated Trajectories," http://www.larc.nasa.gov.

[8]Gaskell, R. W., Collier, J. B., Hussman, L. E., and Chien, R. L., "Synthetic Environments for Simulated Missions," *IEEE Aerspace Conference Proceedings, Big Sky, Montana*, March 2001.

[9]Gaskell, R., Collier, J., Husman, L. E., and Chen, R. L., "Synthetic Terrain for Simulated Missions developed in a collaboration between Parallel Applications Technologies Group and Optical Navigation." *NASA Science Information Systems Newsletter*, , No. 60, July 2001.

[10]Lee, M., Weidner, R., and Lu, W., "Design-based Mission Operation," *IEEE Aerospace Conference, BigSky, Montana*, March 2000.