Report on Rover Simulation Software Discussion

1. Overview of NASA ROAMS
   • NASA ROAMS stands for Rover Analysis, Modeling, and Simulation.
   • It is a high-fidelity simulation tool that models a rover's mechanical dynamics, instrument arms, actuators, sensors, power subsystems, and terrain interactions.
   • ROAMS is used to support design, testing, and operations of planetary rovers.
   • It is built on NASA's simulation frameworks and is available only for federal employees and contractors (i.e. not open and not free for all).

2. Key Features and Underlying Technologies of ROAMS:

- Comprehensive physics-based simulation of rover dynamics and multibody interactions.
- Detailed terramechanics modeling to simulate wheel-soil contact, slip, and sinkage.
- Modeling of rover subsystems (mechanical, sensor, actuator, power, thermal).
- Integration with onboard software for closed-loop and hardware-in-the-loop testing.
- Real-time and Monte Carlo simulation capabilities for performance assessment.
- Modular design allowing customization for different rover configurations and mission environments.
- High-fidelity 3D visualization of both rover and terrain.
- Capability to simulate mission scenarios (e.g., autonomous navigation, obstacle avoidance).

Underlying technologies/SDKs include:

- Extensions of NASA's DARTS/DSHELL simulation frameworks.
- Integration with Matlab/Simulink for algorithm development.
- Custom numerical solvers (such as Newton–Raphson) for solving inverse kinematics.
- Use of SPICE toolkit for orbital and environmental data.
- Support for Monte Carlo studies to evaluate statistical performance.

Sources referenced included NASA Technology Transfer documents and published papers from conferences such as the IAC.

3. Similar Systems/Software Used by Other Agencies

   Roscosmos/RSA:

- Marsokhod, a Russian rover prototype and its associated "virtual environment control system" used in early rover technology tests.
- In-house simulation environments and use of common tools like the SPICE toolkit.

  ESA:

- 3DROV – a planetary rover simulation environment built on the SIMSAT framework. It integrates realistic terrain (using DEM data), atmospheric and illumination models, and orbital information via the SPICE toolkit.

- ROCC/ROCS – systems for rover operations control that support telemetry, planning, and command sequencing.
- EcosimPro – used for simulating complex physical processes (e.g., thermal, fluid, and mechanical subsystems) within ESA.

  JAXA:

- LOCO – a specialized simulation tool for the Martian Moons eXploration (MMX) mission. It is used to test the rover's locomotion system under milligravity conditions on Phobos and supports autonomous operation given long communication delays.
- In addition, JAXA makes use of common tools (e.g., the SPICE toolkit) for orbital and instrument data.

4. Features That Such Simulation Software Should Have
   • Accurate physical dynamics modeling including multibody dynamics and terramechanics.

   • Realistic environmental modeling of terrain, illumination (sun angles, shadows), thermal conditions, and atmospheric properties.

   • Integrated sensor and actuator simulation to mimic real-world responses (cameras, IMUs, LiDAR, etc.) including noise and operational limits.

   • Support for testing control systems and autonomous navigation algorithms, accounting for communication delays.

   • High-fidelity 3D visualization to render the rover and its environment for analysis, design validation, and training.

   • A modular and flexible architecture that allows for easy updates or substitution of components (dynamics, sensors, control, etc.).

   • Interoperability with external data sources (e.g., DEMs, climate data, SPICE toolkit outputs) and simulation frameworks.

   • Robust numerical solvers and the capability to perform scenario testing (both deterministic and stochastic/Monte Carlo simulations).

   • A user-friendly interface with logging and telemetry analysis tools for debugging and performance evaluation.

5. How to Build Such a Simulation System Yourself
   a. Define your requirements and scope. Decide which aspects of the rover to simulate (dynamics, sensors, control, environment) and set clear use cases.

   b. Learn the necessary background in rover dynamics, terramechanics, control theory, and environmental modeling.

   c. Choose your tools and libraries:

   – Use open-source physics engines (Bullet, ODE, or PhysX) for rigid-body dynamics.

   – Consider robotics simulation frameworks like ROS/Gazebo for rapid prototyping.

   – Utilize game engines (Unity, Unreal Engine, or Ogre3D) for high-fidelity 3D visualization.

– Integrate external toolkits (e.g., NASA's SPICE) for realistic orbital and environmental data.
– Decide on programming languages (C++ for performance or Python for prototyping).

d. Design a modular software architecture with clearly defined interfaces between modules (dynamics, sensors, environment, control). This enables flexibility and future expansion.

e. Develop and integrate core components:

– Build the dynamics and terramechanics module for rover motion.

– Create the environmental model using DEMs, climate data, and simulation of illumination/thermal effects.

– Develop sensor simulation models that include noise and physical limitations.

– Implement control and autonomy algorithms, testing them in simulation.

f. Build visualization and user interface components for real-time 3D rendering and for logging simulation data.

g. Validate and test your system:

– Use theoretical models or available real-world data to validate your simulation.

– Conduct iterative development with scenario testing.

h. Leverage existing open-source projects and tutorials to learn best practices (e.g., NASA's open-source rover projects, ROS/Gazebo tutorials).

6. Conclusion
   Developing a rover simulation system is a multifaceted project that requires knowledge of physics, robotics, and software engineering. By following a structured, modular, and iterative approach—starting from clearly defined requirements and leveraging existing open-source tools and libraries—you can build a simulation system that accurately models rover dynamics, environmental interactions, sensor behavior, and control algorithms. Such a system will be instrumental in optimizing rover design, testing autonomous navigation, and de-risking planetary exploration missions.

End of Report