DynRPAT: A Novel Parametric Analytical Tool to Efficiently Simulate High-Speed or Low-Gravity Locomotion Conditions for Planetary Exploration Rovers

P. Oettershagen [1], A. Sager La Ganga [1], M. Goury du Roslan [1], C. Lambelet[1], S. Michaud [1]
(1) Beyond Gravity, Schaffhauserstrasse 580, 8052 Zurich, Switzerland,
philipp.oettershagen@beyondgravity.com

ABSTRACT

Locomotion performance is a major driver in the design and science output of planetary exploration rover missions. While current rovers (e.g. ESA ExoMars) traverse slowly, upcoming robots utilize more dynamic motion, either due to fast travel speed (e.g. NASA/ESA SFR), or motion in low gravity (e.g. DLR MASCOT). Simulations that can handle such dynamic motion, but are computationally efficient enough for parametric rover and mission analysis, are thus required. Therefore, this paper presents the Dynamic Rover Parametric Analytical Tool (DynRPAT), which extends Beyond Gravity's (former RUAG Space) existing RPAT. DynRPAT allows the simulation of highly-dynamic rover motion by efficiently implementing full 6-DOF rover kinematics, Newton Euler Equations of Motion, and accurate wheel-soil interaction. It also supports high-quality planetary terrain import (e.g. from NASA's MRO) and statistical mission analysis. The paper presents preliminary results and a comparison to ExoMars LVM rover test data, such as Wheel Drop and Egress tests performed at BeyondGravity Zurich.

1 INTRODUCTION & MOTIVATION

Locomotion analysis is key for planetary exploration rover missions. Early in the project, the locomotion analysis drives the mission (e.g. landing site and scientific targets) and rover design, and during operations it is used for traversability checks and motion planning.

The existing computational tools for locomotion analysis can be grouped into:

- Multi body simulation (MBS) [1,2]: Highly accurate rover dynamics modeling. Require comprehensive inputs. Their complexity prohibits quick parametric analysis (e.g. of the rover dimensions which are often represented by a complex CAD model) and results in low simulation speeds.
- Quasi-static simulation [3,4]: Trades off accuracy for simplicity and simulation speed. They can efficiently simulate and compare various rover configurations, e.g. for rover concepts / dimensioning, and are thus good for early rover design.

While quasi-static tools are well suited for traditional *quasi-static* locomotion (e.g. slow ExoMars 11mm/s speed), they can't support efficient iterative development of novel rover concepts with *dynamic* locomotion. The term "*dynamic*" can refer either to high-speed rovers such as NASA/ESA SFR [5] with speed up to 30x of ESA ExoMars, or to rovers in lunar or asteroid low-gravity environments such as DLR MASCOT [6] or the MINERVA-II rovers on JAXA's Hayabusa-2. Therefore, a novel simulation approach that combines (a) computational efficiency, (b) accuracy even in dynamic use cases, and (c) good correlation with test data, is needed. Such a modeling approach, with initial test comparison results, is presented in this paper.

2 CONTRIBUTIONS

This paper presents the methodology and initial results behind the Dynamic Parametric Analytical Tool (DynRPAT), Beyond Gravity's (former RUAG Space) novel simulation suite for dynamic planetary rover locomotion. It extends RPAT [4], Beyond Gravity's quasi-static rover simulation tool published in 2012. It keeps well-known RPAT technologies such as wheel-soil interaction (to predict drawbar pull from slip, load, and sinkage), DEM-based terrain modeling, and full mission post-processing, but extends it with a full dynamic rover model. Thus, it can now simulate:

- Rover acceleration / deceleration with changing wheel loads and slip-sinkage
- Wheel drop phase with short loss of soil contact after a large obstacle
- Long coasting phases, as would happen in micro-g environments such as on asteroids

3 IMPLEMENTATION DETAILS

To enable accurate yet efficient simulation of dynamic planetary rover locomotion, combined with comprehensive mission analysis and post-processing capabilities, DynRPAT implements the technologies presented in this section.

3.1 Dynamic Modelling Approach

Simulation loop design

The dynamic simulation is based on three principal models:

- wheel-soil interaction model, computing the external forces and moments seen by each wheel.
- equation of motion: from the external forces applied on the different bodies, it computes the acceleration of the system expressed in the general CS.
- kinematic model: from the general CS acceleration, it computes the state vector of each sub-system.

Starting from the top of the diagram in Figure 1, wheel-soil interactions are solved for each wheel. The wheel-soil geometry module computes the hub-displacement based on the wheel's position and the terrain's DEM. The wheel-soil terramechanics module determines the soil's reaction force, Drawbar Pull and torque based on the wheel's slip and hub-displacement. Once the wheels' external forces are determined and expressed in the world CS, the Newton-Euler algorithm computes the rover's and bogies' acceleration. The kinematic algorithm solves the new state vector for each rover item. A new simulation loop starts.

Assumptions

The assumptions to simplify the model are:

- Air friction can be neglected relative to soil friction.
- The stiffness of the different items constituting the rover kinematic chain are considered infinite relative to the wheel and soil stiffness.
- No friction is considered at joints such as bogies.
- The wheel motor's step response is considered ideal

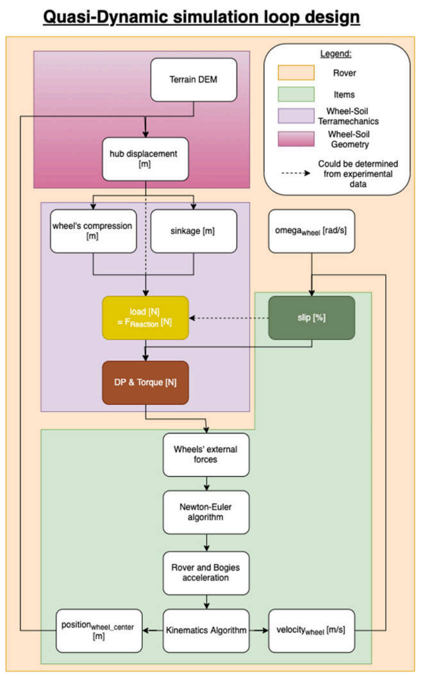


Figure 1: Simulation loop design

Newton-Euler equations and rover kinematics

Two distinct analytical approaches can solve the equation of motion of a multi-body dynamic system: Newton-Euler equations or Lagrange equations [10]. Given the interest in internal forces of the system, the Newton-Euler approach was selected. These equations accurately model the influence of external (e.g. gravity load and drawbar pull) forces and torques on the rover's elements. Given a full rover kinematic model, the external forces result in realistic internal forces so that the behavior of wheels, deployable legs, rotatable bogies, and the rover chassis can be simulated accurately. Newton-Euler was implemented with the equation

$$\begin{aligned}&\textstyle\sum_{i=0}^{nbParts}(J_{P_i}^T m_i J_{P_i} + J_{R_i}^T I_i J_{R_i})\overrightarrow{\ddot{q}} + \sum_{i=0}^{nbParts}(J_{P_i}^T m_i \dot{J}_{P_i})\overrightarrow{\dot{q}} \\ &= \textstyle\sum_{i=0}^{nbParts}(J_{R_i}^T \overrightarrow{M}_{EXT_i}) + \sum_{i=0}^{nbParts}(J_{P_i}^T \overrightarrow{F}_{EXT_i}) \\ &\quad + \textstyle\sum_{i=0}^{nbParts}(J_{P_i}^T \overrightarrow{F}_{G_i})\end{aligned} \tag{1}$$

Where J_P is the 3x6 Jacobian matrix for translation state vector and J_R for rotation state vector, M_{EXT} is the sum of external moments applied on the part, F_{EXT} external forces and F_G gravity, m is the mass and I the moment of inertia.

Newton Euler is solved in a recursive algorithm looping through each part of the multi-body system. The chain of parent/children parts is built directly from an external rover geometry database listing the properties of the different elements: relative position, mass, parent part.

Efficient Newton-Euler solver

Two solvers are implemented: A basic forward Euler integration with error *O(Δt)*, which already allows real-time simulation, as well as an improved solver using Beeman's method [7] with improved error *O(Δt³)*. Usually, the Newton-Euler equations are integrated using Euler's method. That is, for each timestep *t*, the acceleration $\vec{\ddot{q}}_t$ is solved in Equation (1) and then the velocities and positions are updated via

$$\begin{aligned}\vec{\dot{q}}_{t+1} &= \vec{\dot{q}}_t + \Delta t\,\vec{\ddot{q}}_t\\ \vec{q}_{t+1} &= \vec{q}_t + \Delta t\,\vec{\dot{q}}_t\end{aligned} \tag{2}$$

However, it is well known that Euler's method has a global integration error of *O(Δt)*. We propose Beeman's predictor-corrector method [7] from the molecular dynamics literature as an alternative integration approach, as it achieves a global error of *O(Δt³)* while incurring very little extra memory and compute overhead. In practice, this means that our simulation errors are two orders of magnitude smaller for the same timestep if we use Beeman's method. Thus, Equation (2) becomes

$$\begin{aligned}\vec{\dot{q}}_t &= \vec{\dot{q}}_{t-1} + \frac{5}{12}\Delta t\,\vec{\ddot{q}}_t + \frac{2}{3}\Delta t\,\vec{\ddot{q}}_{t-1} - \frac{1}{12}\Delta t\,\vec{\ddot{q}}_{t-2}\\ \vec{q}_{t+1} &= \vec{q}_t + \Delta t\,\vec{\dot{q}}_t + \frac{2}{3}\Delta t^2\,\vec{\ddot{q}}_t - \frac{1}{6}\Delta t^2\,\vec{\ddot{q}}_{t-1}\\ \vec{\dot{q}}^{\,(\text{predicted})}_{t+1} &= \vec{\dot{q}}_t + \frac{3}{2}\Delta t\,\vec{\ddot{q}}_t - \frac{1}{2}\Delta t\,\vec{\ddot{q}}_{t-1}\end{aligned} \tag{3}$$

Where we use the predicted velocity $\vec{\dot{q}}^{\,(\text{predicted})}_{t+1}$ to compute the next step's acceleration, $\vec{\ddot{q}}_{t+1}$. With this method, we must only save in memory the acceleration of the latest three steps.

Vectorized processing

To speed up vector and matrix operations, DynRPAT leverages a custom Single Instruction Multiple Data (SIMD) implementation for each vector and matrix size combination (3x1, 6x1, 3x3, 3x6, and 6x6). Every vector-matrix mathematical operation, such as matrix multiplication and addition, was optimized to use Intel's x86 SSE instructions. Furthermore, we achieve a speedup of x30 on Jacobian matrix inversion compared to the ALGLIB numerical analysis library [10] by explicitly formulating all multiplications and additions and then vectorizing them.

Adaptive time step

To accelerate the simulation when the rover is evolving in flat surface and uniform soil, the solver can automatically adapt time step as function of wheel speed and maximum allowable hub displacement. During simulation, after completion of each step, it computes for each wheel the time step that will lead to a hub-displacement equal to a certain "max hub-displacement" parameter. In this process, the whole rover's dynamic is not taken into account. It is limited to the wheel-soil interaction, thus limiting computation efforts.

Improved dynamic wheel-soil interaction

The wheel-soil interaction is modelled through a serial double spring plus damper system, as seen in Figure 2. The system is described with the differential equations

$$\begin{cases} F_{Reaction} &= F_{R,wheel} = F_{R,soil}\\ u_{tot} &= u_{wheel} + u_{soil}\\ F_{R,wheel} &= k_{wheel}u_{wheel} + \beta_{wheel}\dot{u}_{wheel}\\ F_{R,soil} &= k_{soil}u_{soil} + \beta_{soil}\dot{u}_{soil}\end{cases} \tag{4}$$

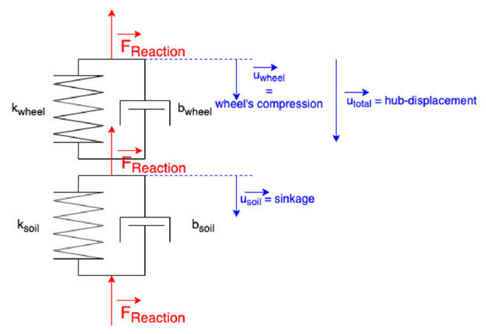


Figure 2: Soil serial double spring + damper reaction force model

Where *k* is the stiffness, *β* the damping coefficient, $F_{Reaction}$ the total reaction force, $F_{R,wheel}$ the force acting on the wheel, $F_{R,soil}$ the soil reaction force, and u_{wheel}, u_{soil}, and u_{tot} are the compressions of the wheel, soil, and the hub-displacement, respectively. At each timestep, given the hub-displacement u_{tot} as an input, we update u_{wheel} and u_{soil}, and solve for the total reaction force $F_{Reaction}$. From Equation (4) we derive

$$\dot{u}_{wheel} = \frac{-(k_{wheel} + k_{soil})u_{wheel} + k_{soil}u_{tot} + \beta_{soil}\dot{u}_{tot}}{\beta_{wheel} + \beta_{soil}} \tag{5}$$

, which has the form of an explicit ODE, $\dot{u}_{wheel} = f(u_{wheel},\ u_{tot},\ \dot{u}_{tot})$. In practice, high soil and wheel stiffnesses result in high-frequency modes which are numerically unstable. Thus, we use a novel left-edge stencil for the finite-difference approximation [12] of u_{wheel} to leverage past values and decrease integration errors. We first define the following approximations of order-*n*

$$\dot{u}_{tot} \simeq \frac{1}{\Delta t}\sum_{i=1}^{n} c_i u_{tot}^{i}$$

$$\dot{u}_{wheel} \simeq \frac{1}{\Delta t}\sum_{i=1}^{n} c_i u_{wheel}^{i} \qquad \forall i,\ c_i \in \mathbb{R}$$

where *i* indicates the timestep, *n* being the current one, and *1* being *n-1* steps ago. The coefficients c_i are calculated using the online tool [8]. After substituting and rearranging terms, we get the current time step value for the wheel compression

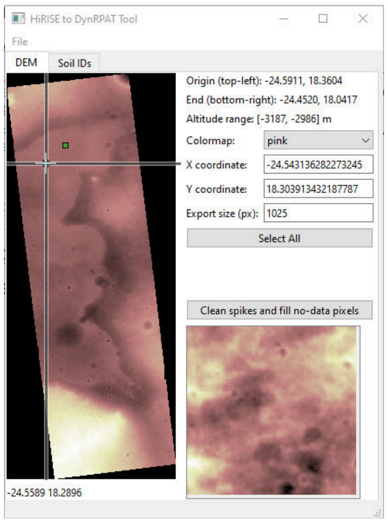
$$u_{\text{wheel}}^{n} = \frac{-(\beta_{\text{wheel}} + \beta_{\text{soil}})\sum_{i=1}^{n-1} c_i u_{\text{wheel}}^{i} + \Delta t\ k_{\text{soil}} u_{\text{tot}}^{n} + \beta_{\text{soil}} \sum_{i=1}^{n} c_i u_{\text{tot}}^{i}}{(k_{\text{wheel}} + k_{\text{soil}})\Delta t + (\beta_{\text{wheel}} + \beta_{\text{soil}})c_n}$$

Intuitively, our method converges for some timestep value below $(\beta_{wheel} + \beta_{soil})/(k_{wheel} + k_{soil})$, which can be seen from Equation (5) when considering a constant u_{tot}. If this requirement is satisfied, and since this is a general integration method of order-*n*, the integration error is $O(\Delta t^n)$ while we only keep in memory *2n* variables. In particular, *n=2* is equivalent to the well-known backward Euler method. To achieve a high simulation speed, we set $\Delta t=3$ [ms] in our experiments, which does not converge under our damping coefficient and stiffness values, thus requiring us to use the lowest *n=2*. However, Δt can be decreased to convergence and *n* increased to minimize errors whenever precision is required.

3.2 Usability Features

Importing Realistic Terrains

DynRPAT includes a GeoTIFF terrain import module for high-resolution realistic terrains, supporting digital elevation models (DEMs) and including an editor to specify the heterogeneous soil types. In particular, we explore terrains as recorded by HiRISE on the Mars Reconnaissance Orbiter. Figure 3 shows the import tool, where the terrain to be imported is selected using planetary coordinates. The 1,5GB high-resolution terrain of the proposed landing site for the ExoMars rover in the Oxia Planum is displayed in its entirety. Terrain DEMs are imported into DynRPAT in GeoTIFF format using GDAL. The soil type is considered by a soil map, for which all ESA planetary soil simulants (ES1, ES2, ES3, and ES4) are supported.


HiRISE to DynRPAT Tool

File

DEM | Soil IDs

Origin (top-left): -24.5911, 18.3604

End (bottom-right): -24.4520, 18.0417

Altitude range: [-3187, -2986] m

Colormap: pink

X coordinate: -24.543136282273245

Y coordinate: 18.303913432187787

Export size (px): 1025

Select All

Clean spikes and fill no-data pixels

-24.5589 18.2896

Figure 3: Terrain selected in Oxia Planum for import into DynRPAT

This terrain import module includes an algorithm for terrain cleaning and filling in missing values. This algorithm is necessary since high-resolution terrain topography reconstruction using HiRISE stereo pairs of images can result in noisy terrain with missing values, especially around extreme or blocking terrain such as craters. This algorithm is divided into three steps: 1) *outlier* detection; 2) interpolation by *kriging* [9]; 3) correction of discontinuities.

As an example, Figure 4 shows an untreated terrain with missing values. The first step is to detect *noisy* values in the terrain. We apply the *outlier* detection kernel seen in Figure 5 by convolving it with the terrain image. It is composed of a Gaussian kernel whose 70x70 pixel center is first set to 0, and then the values are normalized so the kernel sums to 1. Then, we compute the difference between the original image and the convolved image, and we consider pixels with a difference above *T=0.7* [m] as outliers. We optimized these parameters for HiRISE terrains. Intuitively, we give importance to the difference with close neighbors, but also take into consideration further pixels.

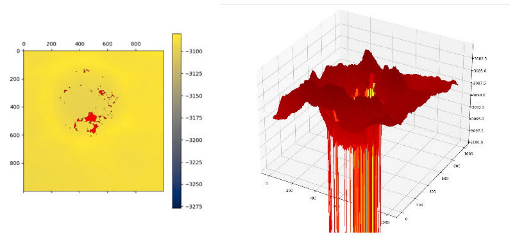


Figure 4: Terrain with missing values

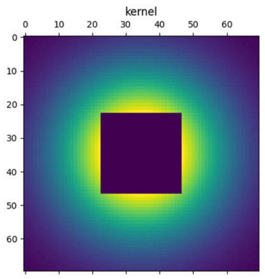


Figure 5: Outlier detection kernel

In the second step, we perform *kriging*, that is, we train a *Gaussian process regression* (GPR) [9] model on the terrain to interpolate the outliers and missing pixels. Since we have high-resolution terrains and it is very computationally expensive to train a GPR model, we sample 1'000 points uniformly at random from the image and use them for training. We use a Matérn kernel with parameter $\nu = {}^{3}/_{2}$.

Third, we correct discontinuities. Since GPR has some error, the predicted heights are disconnected from the rest of the terrain. To solve this, we estimate the *bias* by convolving an averaging 7x7 kernel repeatedly on the boundary of missing and outlier pixels, where we take the difference between the original height of the terrain and the predicted height of GPR. This estimates how much we must increase the height in the predicted terrain, eliminating all discontinuities at the boundary.

Figure 6 shows a terrain before and after applying the algorithm. This crater is an example of an extreme terrain in the HiRISE data where the high-resolution topological reconstruction introduces noise and where this algorithm is applicable and valuable. Generally, this algorithm can clean any planetary terrain with missing and noisy pixels after appropriately tuning the parameters.

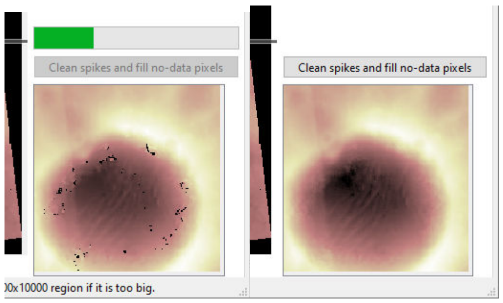
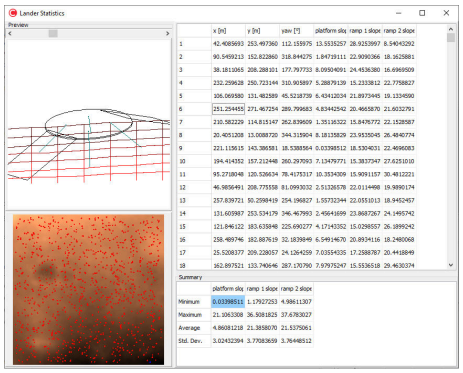

Clean spikes and fill no-data pixels
0x10000 region if it is too big.
Clean spikes and fill no-data pixels

Figure 6: Terrain cleaning algorithm applied on noisy terrain with missing values

Mission Analysis

On top of the analysis of individual locomotion cases, DynRPAT aims to parametrically trade-off different missions against each other. In particular, it is able to estimate the instability of lander egress by retrieving statistics on likely slope angles the rover would be subject to in the lander. It achieves this by placing the lander on randomly sampled points in realistic terrain from the HiRISE/MRO data. Then, it calculates the lander body angle and ramp angles which can be used to estimate how likely the rover is to have a successful egress.

Figure 7 shows the exemplary ExoMars lander and ramp angles retrieved with this tool when using 1'000 samples. The resulting distribution of lander and ramp angles is shown in Figure 8. The distribution can be used to assess which percentage of lander placement cases are covered by ExoMars LVM test data, and will thus likely result in successful egress.



	x [m]	y [m]	yaw [°]	platform slo	ramp 1 slope	ramp 2 slope
1	42.4085693	253.497360	112.155975	13.5535257	28.9253997	8.54043292
2	90.5459213	152.822860	318.844275	1.84719111	22.9090366	18.1625881
3	38.1811065	208.288101	177.797733	8.09504091	24.4536380	16.6969509
4	232.259628	250.723144	310.905897	5.28879139	15.2333812	22.7758827
5	106.069580	131.482589	45.5218739	6.43412034	21.8973445	19.1334590
6	251.254455	271.467254	289.799683	4.83442542	20.4665870	21.6032791
7	210.582229	114.815147	262.839609	1.35116322	15.8476772	22.1528587
8	20.4051208	13.0088720	344.315904	8.18135829	23.9535045	26.4840774
9	221.115615	143.386581	18.5388564	0.03398512	18.5304031	22.4696083
10	194.414352	157.212448	260.297093	7.13479771	15.3837347	27.6251010
11	95.2718048	120.526634	78.4175317	10.3534309	15.9091157	30.4812221
12	46.9856491	208.775558	81.0993032	2.51326578	22.0114498	19.9890174
13	257.839721	50.2598419	254.196827	1.55732344	22.0551013	18.9452457
14	131.605987	253.534179	346.467993	2.45641699	23.8687267	24.1495742
15	121.846122	183.635848	225.690277	4.17143352	15.0298557	26.1899242
16	258.489746	182.887619	32.1839849	6.54914670	20.8934116	18.2480068
17	25.5208377	209.228057	24.1264259	7.03554335	17.2588787	20.4418849
18	162.897521	133.740646	287.170790	7.97975247	15.5536518	29.4630374

Summary

	platform slo	ramp 1 slope	ramp 2 slope
Minimum	0.03398511	1.17927253	4.98611307
Maximum	21.1063308	36.5081825	37.6783027
Average	4.86081218	21.3858070	21.5375061
Std. Dev.	3.02432394	3.77083659	3.76448512

Figure **7**: Lander placement analysis tool

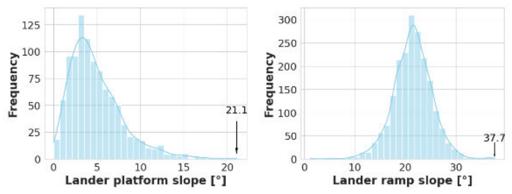


Figure 8: Retrieved lander and ramp angles after 1'000 placements. Maximum values are indicated

State-of-the-art visualization

DynRPAT includes a visualization program to realistically visualize in real-time simulated rover traverses, including complex situations such as ExoMars on its landing platform egressing onto Martian terrain (Figure 9). This visualization program is compiled with the

Unigine 3D engine, and the rover and lander state are transferred to this program via TCP/IP while the simulation runs. Additionally, the executed simulation can easily be replayed and visualized.

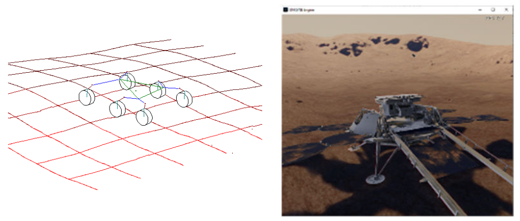


Figure 9: DynRPAT live simulation of ExoMars rover on Martian terrain with obstacles (left) and DynRPAT visualization of ExoMars using Unigine 3D (right)

4 PRELIMINARY RESULTS

4.1 Computational speed

Comparing DynRPAT to the original quasi-static approach of RPAT [1] yields:

- The quasi-static original RPAT [1] allowed ca. 150x real-time speed during simulations.
- DynRPAT's initial release, which used the simple Newton Euler forward integration also leveraged by other simulation tools, was 5x slower than real-time.
- The speedup due to SIMD vectorized processing is x3 on the overall simulation.
- DynRPAT's final release, with the computational speedups presented in this paper (Beeman's method, wheel-soil accurate finite-difference approximation, and SIMD support), can simulate ~8x faster than real-time on a typical consumer CPU.
- Adaptive timestep was implemented on a simplified model and speeds up computation by a factor 4x.

Although slower than the original RPAT [1], DynRPAT is more accurate and flexible in modeling dynamics while still being much faster than real-time.

4.2 Preliminary results and test correlation

In this section, we present initial results and a preliminary comparison with ExoMars LVM tests performed at Beyond Gravity's Zurich Mars Yard facility.

Gradeability Testing

We do a preliminary correlation of the rover slip as a function of the terrain slope. Figure 10 shows the results of the ExoMars LVM rover on ES4 soil for different tilt angles, giving an absolute error of below 7% slip.

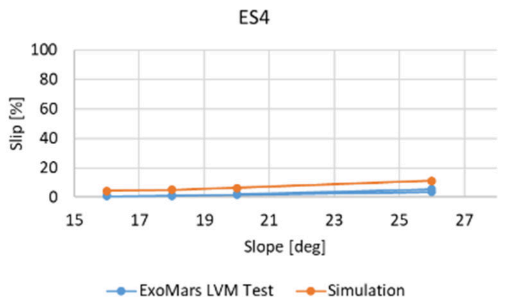
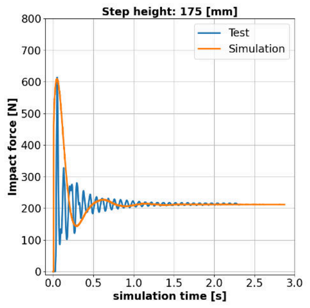
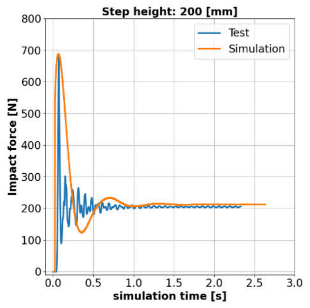


Figure 10: Slip versus tilt on ES4 soil

Wheel Drop Testing

To demonstrate the dynamics of the simulation, we correlate wheel drop tests on the ExoMars LVM rover. The tests, performed at Beyond Gravity, consist in dropping the rear wheels of the rover at different step heights (175mm, 200mm, 225mm) into impacting plates, recording the interaction force of the wheels to the ground. Figure 11 shows how the simulation compares to the tests when the wheels impact hard terrain from different heights. We plot the simulated reaction force and the recorded impact force from the different tests. The maximum force and steady state load correlate between simulation and test. Frequency response has not been correlated, as DynRPAT is not designed for structural modal analysis.





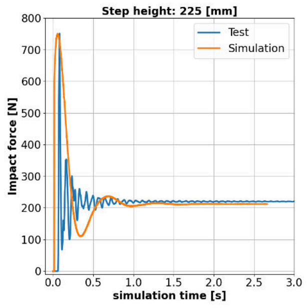


Figure 11: Wheel drop correlations

Egress from Landing Platform

DynRPAT can also model dynamics between the rover wheels and a lander to simulate lander egress. Figure 12 shows the ExoMars LVM rover egressing from its lander model. As seen, the trajectory of the rover is simulated from the starting platform down to the terrain. Step drops from the ramp to the soil are also simulated and all relevant variables, such as wheel impact forces, hub-displacements, bogie orientations, and bogie accelerations, are recorded.

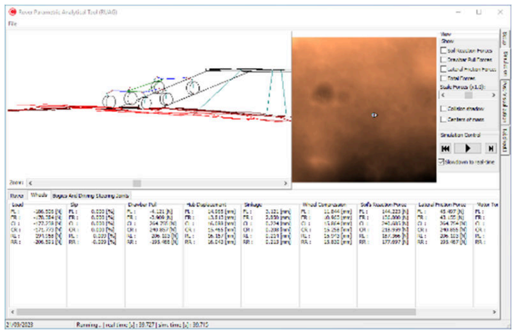


Figure 12: Simulated lander egress

Dynamic Egress ("Crane") testing

Given the ability to model dynamic motion, DynRPAT is also suited for simulation of dynamic landing on other celestial bodies. For example, DynRPAT can be used to model "sky-crane" deployments as used by NASA for its Mars Science Laboratory (MSL) rovers. Figure 13 shows such a situation. DynRPAT is able to then simulate accelerations, bogie motion and orientation during free fall, and reaction forces and dynamics upon impact into soft or hard soil.

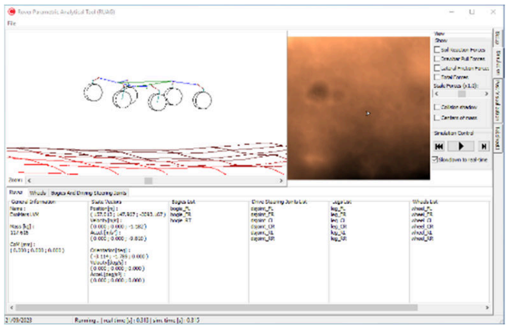


Figure 13: Simulated dynamic egress deployment

5 FUTURE WORK

This paper has presented a preliminary comparison of DynRPAT results with ExoMars LVM test data. Beyond that, planned future work includes:

- A full correlation with ExoMars test data, including more dynamic step-shape obstacle drop tests.
- A correlation with test data from faster rovers, such as ESA/NASA's SFR, for which prototype locomotion tests have been performed at Beyond Gravity's Zurich Mars Yard test facility.
- An improvement of DynRPAT's wheel-soil contact point modeling, which can still occasionally show unstable behavior during which the contact point quickly switches between the hard soil (e.g. on a step shape obstacle) and soft soil, thus negatively impacting the performance metrics such as the drawbar pull.
- Given DynRPAT's computational speed, it can be easily upgraded to retrieve mission statistics not only on the presented lander egress, but also on traversability. Using random rover placements and related motion paths on realistic HiRISE/MRO terrains, DynRPAT could calculate traversability by measuring the percentage of cases where the rover successfully reaches its target. This would support locomotion optimization and mission landing site selection.

Overall, the computational speed and flexibility of the simulation leaves DynRPAT room to easily improve and add complex features on top of the existing system.

6 CONCLUSION

This paper has presented the Dynamic Parametric Analytical Tool (DynRPAT), a simulation suite for planetary exploration rover locomotion. Initial comparison with ESA ExoMars LVM rover test data show a good agreement with gradeability tests, wheel drop tests, and Egress tests from lander platforms. DynRPAT features a combination of computational efficiency and medium-accuracy dynamic modeling, which place it right in the middle between the existing fast but low-accuracy quasi-static approaches, and the high-accuracy but mostly slow and complex multi-body simulators. DynRPAT is thus considered well suited to support medium-accuracy iterative use-cases, such as preliminary design or operations support of future high-speed planetary exploration rovers.

REFERENCES

1. A. Gibbesch, B. Schäfer, S. Michaud. *3D Simulation and Validation of RCL-E and MER Rover Types Mobility*. ASTRA 2006, ESTEC, the Netherlands.
2. Bernd Schäfer, Andreas Gibbesch, Rainer Krenn and Bernhard Rebele. *Planetary rover mobility simulation on soft and uneven terrain.* Journal of Vehicle System Dynamics, 2010.
3. A. Krebs et al. *Performance Optimization of All- Terrain Robots: A 2D Quasi-Static Tool.* IEEE International Conference on Robots and Intelligent Systems, 2006, Beijing, China.
4. P. Oettershagen. S. Michaud. *Development and Validation of a Modular Parametric Analytical Tool for Planetary Exploration Rovers*, I-SAIRAS 2012, Turin, Italy.
5. B. K. Muirhead, A. Nicholas, J. Umland. *Mars Sample Return Mission Concept Status*, IEEE Aerospace Conference (2020)
6. T. Ho et al. *MASCOT—The Mobile Asteroid Surface Scout Onboard the Hayabusa2 Mission*, Space Science Reviews 208, pp. 339-374 (2017).
7. David Beeman, *Some multistep methods for use in molecular dynamics calculations.* Journal of computational physics 20(2), pp. 130-139 (1976).
8. Cameron R. Taylor, *Finite Difference Coefficients Calculator*, https://web.media.mit.edu/~crtaylor/calculator.html (2016).
9. Margaret A. Oliver, and Richard Webster, *Kriging: a method of interpolation for geographical information systems.* International Journal of Geographical Information Systems 4(3), pp. 313-332 (1990).
10. ALGLIB Project, *ALGLIB - C++/C#/Java numerical analysis library,* https://www.alglib.net/ (2023).
11. Marco Hutter, Roland Siegwart, and Thomas Stastny. *Robot dynamics lecture notes*, 2017. Robotic Systems Lab, ETH Zurich.
12. Francis Begnaud Hildebrand, *Introduction to numerical analysis,* Courier Corporation (1987).