74th International Astronautical Congress (IAC), Baku, Azerbaijan, 2-6 October 2023.

IAC-23,D1,4A,11,x77760

A semi-stochastic, numeric simulation tool in model based systems engineering for tumbleweed rovers

Markus Renoldner*, Julian Rothenbucher, Lucas Cohen, Mihir Kapadia, Furqan Mahmood, Darius Vicovan
All authors are affiliated to Team Tumbleweed
* Corresponding Author, markus@teamtumbleweed.eu

Abstract

Scientific exploration of the Martian surface via a wind-driven, low-cost swarm of rovers involves a systematic design of mission-level, as well as system- and subsystem-level concepts. Risks associated with design are mitigated through multiple demonstrator missions. However, the complexity and entanglement of mission- and system-level design parameters introduce cross-discipline complexities hindering the full realization of these demonstrator missions. Further, the semi-stochastic nature of the wind-driven rover creates high uncertainty in several key mission-, and system-level performance figures, which are absent in legacy missions to Mars. Finally, the historical lack of comprehensive model-based systems engineering for wind-driven spheroid rovers, makes this a unique systems challenge.

These factors lead to large uncertainty in early-stage mission planning, deviation of the system worst-case scenarios relevant to the sizing of spacecraft subsystems, and difficulty in the establishment of mass and power budgets. Furthermore, there is no research available on specific methods to estimate mission-level performance of Tumbleweed Missions employing system-level semi-stochastic actuation and motion. Lastly, no specific methods for sizing important subsystems of a Tumbleweed rover and for defining the operation of the mission are available.

To this end, we present a semi-stochastic, numeric simulation tool consisting of a 3-DOF physics model of the Tumbleweed Rover. Wind and climate data is taken from the Mars Climate Database (MCD) and topography data measurements from the Mars Orbiter Laser Altimeter (MOLA). This tool serves as a key component in addressing performance and operations of the mission and enables the evaluation of mission and system level design parameters, as well as the computation of realistic trajectories for a rover network. We also discuss how the tool can be applied in an existing Model-Based System Engineering framework to derive key mission parameters.

We conclude that improvements to numeric modelling are relevant for the design of swarm missions with high levels of system-level uncertainty and low-cost individual spacecraft.

Keywords: semi-stochastic simulation, numeric simulation, rover, wind-driven, Mars, Tumbleweed mission

1 Introduction

Historic exploration of the Martian surface is heavily based on single-vehicle, cost-intensive mission concepts. While benefits of this mission type clearly exist, it lacks the ability to cover long-term, large-scale, planetary observations, or to access difficult terrain. One alternative is the Tumbleweed mission, which intends to utilise a swarm of unactuated, wind-driven, tumbling, spherical rovers. The motion of this network of rovers is almost entirely dependent on the bearing and velocity of the imminent wind as well as local terrain topography. More details on the Tumbleweed mission can be found in an article presented at the IAC 2022 [1]. Details about demonstrator missions as a means to proof technology readyness can be found in the report of the Pre-Phase A study, [2].

Previous studies have not investigated and missions have not showed in detail the feasibility of such low-controllability planetary explorers. Traditional rover designs also do not require testing their simultaneous dependence on climatic properties and topographic features of the environment as much, while the evaluation of this interplay of external forces is central for the design of the Tumbleweed mission.

Further, the semi-stochastic nature of the wind-driven rover creates high uncertainty in several key mission-, and system level performance figures, which are absent in legacy missions to Mars. This paper presents a simulation tool that computes the trajectory of Tumbleweed Rovers across the Martian surface induced by Martian winds and local topography features. The objective is to show how mission- and system-level performance specifications and requirements of such types of network missions can be quantitatively evaluated and tested in the simulation environment, and how this tool facilitates and integrates with model based systems engineering. Limited physical access for testing of prototypes in the target environment is hence overcome.

1.1 Background information and literature review

The Tumbleweed mission cannot be easily classified into conventional architectures, because of several fundamentally different mission and system level parameters. In the early planning phase, a direct link can be established between the system design and mission level performance, with a clear mission outlook defined. Tumbleweed rovers will experience a wider range of possible conditions that cannot be precisely determined during the design phase using purely deterministic method, due to the stochastic spreading mechanism. Risk analysis for standard planetary missions can be done quantitatively through simulations and other models, finding risk vs. reward relationships. These are based on known risks and complications that can emerge with tested and established technology. Creating the same quantitative risk assessments for Tumbleweed is less straightforward, due to the more unpredictable nature of the mission.

Existing analyses on swarms of extra-planetary exploration vehicles focus on devising algorithms to efficiently manoeuvre on the surface. In particular, they develop methods to autonomously set a trajectory for each individual rover, accounting for the position of other rovers and obstacles, and ensuring a suitable spread. Saaj and Ibrahim, [3] have developed one such algorithm for rover navigation, relying on attraction and repulsion potential fields. Existing analyses further focus on verifying the success of the algorithm through simulations. Petritoli et al., [4] illustrate the verification process of the navigation algorithms. In contrast, the semi-stochastic, numeric simulation tool for Tumbleweed rovers aims to demonstrate the feasibility of using wind power to propel and disperse Tumbleweed rovers, with limited control on the trajectory of each rover.

1.2 Outline of the paper

This paper details the functionality of a semi-stochastic, numeric simulation tool for a wind-driven, low-cost swarm of Mars rovers and how it is applied within existing Model-Based System Engineering methods (MBSE). Starting in section 2, we describe how the rovers are modelled and how the software architecture looks like. We also discuss its relevance to MBSE. This is followed by the results of the simulation of both verification tests and an exemplary simulation section 3. In section 4, we discuss how results from the simulation flow into the sizing of important subsystems of a Tumbleweed rover and into a Model-Based System Engineering where it is used to derive key mission parameters. Finally, section 5 our conclusions are presented.

2 Methodology

In this section, we discuss the architecture of the modeling tool, including how it replicates physical behavior of a Tumbleweed rover under stochastic, external forces.

The overall objective of the simulation tool is to relate mission and system level parameters of swarm missions. Mission level parameters include the choice of landing point location, the degree to which the planet surface can be covered by a rover swarm and the time frame of the mission. (Sub)system parameters include technical variables like mass and aerodynamic cross-section. An important functional variables that can be evaluated and quantified is the ability to control the rovers movement at various degrees, like breaking, steering or even propelling the rover independent of wind and surface inclination. In the sample simulations described in subsection 3.2, we relate system level parameters to mission level parameters. In

other words, we evaluate the spreading behaviour based on a certain choice of technical variables. We aim to create a simulation capable of computing certain mission level parameters based on technical parameters of the systems.

As discussed later, the tool is heavily based on the Mars Climate Database (MCD), [5, 6, 7]. It is based on a General Circulation Model which is used to calculate the dynamic changes in a planet's atmosphere over time. It accomplishes this by solving equations governing motion and thermodynamics on a three-dimensional grid that encompasses the entire atmosphere. It is tailored to capture the unique physical processes occurring on Mars, spanning from the planet's surface to its outermost atmospheric layer. These processes include phenomena like the freezing of carbon dioxide in the atmosphere during polar nights, the formation and evolution of water ice clouds, the dispersion of airborne dust particles, and the multitude of chemical reactions that occur throughout the Martian atmosphere, among others. The topography information from MCD comes from a dataset collected by the Mars Orbiter Laser Altimeter (MOLA) of the Mars Global Surveyor (MGS) space probe.

2.1 Physical modeling of Tumbleweed rover

The main justification of developing said modeling tool is to provide a platform capable of replicating the behavior of a Tumbleweed rover or a swarm of such kind. The rover is subject to three major external forces: a force due to the Martian wind, $\boldsymbol{F}_W$, proportional to the squared velocity difference of the wind to the rover, a force resulting from a non-zero surface gradient being a measure of the steepness of terrain, $\boldsymbol{F}_G$, as well as the magnitude of a surface friction force, F_F, that results from the contact between the rover and the ground and is always oriented anti-parallel to the momentary velocity vector.

This model makes the following, simplifying assumptions:

- The ground is modelled as perfectly smooth, neglecting smaller-scale surface roughness and obstacles. This could lead to overestimation of performance as these features provide additional rolling resistance. The chosen approach is to account for this by increasing the rolling friction coefficient by a factor of 10 versus the expected rolling friction on smooth ground. The actual figure is highly dependent on the design of the actual rover, and must be experimentally determined in future work.
- The rover is modelled as a perfect sphere, with no preferential rolling axis or direction. This is not entirely true, as a real rover is likely to have non-uniform mass distribution. However, for the purpose of this simulation, the effects are considered to be small.

2.2 Code architecture of modeling tool

The code is organized as an object-oriented library that implements not only physical laws, like the computation of change of rover momentum due to the sum of external forces, but also data processing tools.

The digital representation of a rover is implemented in the `class Rover`. Its attributes store information about the environment, like surface gravity, terrain altitude, wind velocity, as well as information about the rover itself like coordinates of current and past positions, velocity values and the physical dimensions of the rover. Its methods implement functionality connected to the environment, to the rover propagation, as well as for data processing. The key members functions for environment related tasks are:

- `Rover::update_wind_mcd()`: updating the wind velocity vector as well as importing other environment variables, such as air density and local distance from the planet's center from the Mars Climate Database (MCD), [5], [6], [7],
- `Rover::get_gradient_mcd()`: updating the local surface gradient based on topography information from [5],

Key member functions of rover propagation related functionality of the tool include

- `Rover::get_forces()`: computing all external forces acting on the rover (i.e. forces due to wind, surface steepness and friction)
- `Rover::update_local_pos()`: updating the resulting change in position of the rover using "east-north-up"-coordinates (also known as "enu-coordinates"). These coordinates are a local coordinate basis of the planet surface, or in other words a coordinate basis of the plane tangent to the planet surface at the current global position. In this definition of the coordinate system, the planet surface is assumed to be flat, meaning that the normal vector of this tangent plane is aligned with the planet radius.
- `Rover::update_global_pos()`: computing the new global position from the local change in position.

Finally, a number of functions related to data processing store positional information in binary files, enabling fast access to the generated information produced by the modelling tool. This workflow is represented in Figure 1.

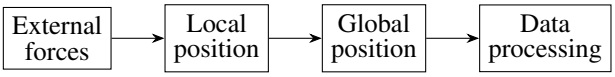

Figure 1: Software structure diagram.

A number of Rover objects are being instantiated at the beginning of the simulation. The main file calls all member functions necessary for updating a rover's position (i.e. members related to the environment as well as the rover propagation) iteratively. A time step size is chosen beforehand - this implies that we implicitly assume all external forces to be constant within one time step, which is a good enough approximation for time step sizes that are very small compared to characteristic times of the three forces. As an example, on many location on the planet surface, we observe a recurrent oscillation in the orientation of the wind velocity vector, with a period of 1 sol (=88775 seconds). Assuming that this is the dominating part of the dynamics of the wind vector, choosing a time step size of around 10^0 seconds is a reasonable choice.

2.3 Core functions of source code

In the following, we will discuss some implementation details concerning all class members, related to the rover propagation listed above.

The member `Rover::get_forces()` computes computes all external forces - i.e. wind, surface gradient and friction force - acting on the rover. They are mathematically defined as

$$\boldsymbol{F}_W \equiv \frac{1}{2}\rho\|\Delta\boldsymbol{v}\|\Delta\boldsymbol{v}Ac_D, \tag{1}$$

$$\boldsymbol{F}_G \equiv \begin{pmatrix} mg\sin(\alpha_e) & mg\sin(\alpha_n)\end{pmatrix}^\intercal, \tag{2}$$

$$F_F \equiv mg\mu\cos(\beta), \tag{3}$$

with ρ being the atmospheric density, $\Delta\boldsymbol{v}$ being the difference of the velocity of the wind and the rover, $\|\cdot\|$ being the Euclidean norm, A being the rovers active aerodynamic cross section, c_D being a coefficient of aerodynamic drag, m being the total rover mass, g being the magnitude of the planetary gravity field (assumed to be $3.72m/s^2$ and constant), α_i being the two angles between the local horizontal plane (normal vector aligned with planet radius) and the respective coordinate line in the local tangent plane, μ being a coefficient of friction, which relates the normal force to the friction force, and β being the angle between the local horizontal plane and the local tangent plane meaning that it is the angle of steepest descent.

Note that the bold symbol forces and the bold velocity defined above are coordinate vectors with respect to the local "enu" coordinate basis, as we solely need them to compute a local change in position. Also we neglect the third component of "enu", as the rover movement is restricted to the planetary surface. As visible, we only use the magnitude of the friction force, as its direction is always implied be the local rover velocity vector. Further, it was assumed, that the wind velocity vector is tangential to the planet surface.

We add, that α_i are being provided by the MCD, [5], [6], [7], and β is being computed using the following subroutine. First, note that the unit normal vector of the local tangent plane defined by the two angles, α_e, α_n is given by

$$\boldsymbol{e}_n \equiv \begin{pmatrix}\cos(\alpha_e)\sin(\alpha_n)\\ \sin(\alpha_e)\\ \cos(\alpha_e)\cos(\alpha_n)\end{pmatrix}.$$

The angle β can be found by computing the angle between $\boldsymbol{e}_n$ and the normal vector of the horizontal plane $\boldsymbol{e}_h$ in the enu-coordinate basis

$$\beta = \arccos(\boldsymbol{e}_n \cdot \boldsymbol{e}_h),$$

as all vectors have unit length. Observing the above force definitions, (1) - (3), and realizing that the terrain model provides "only" 32 data points per degree longitude which is interpolated linearly, one can conclude that locally, the surface is flat and does not include any features with dimensions comparable to the rovers size, i.e. $\approx 10^0 - 10^1 m$.

The member `Rover::update_local_pos()` takes the resulting force given by the previous function in order to compute the change in position in the enu system. Mathematically, and in this order, the method implements the following:

$$\boldsymbol{a}_0^i \equiv \frac{1}{m}(\boldsymbol{F}_W + \boldsymbol{F}_G),$$
$$\boldsymbol{v}_0^i \equiv \boldsymbol{v}^{i-1} + \boldsymbol{a}_0^i\Delta t.$$

The superscript index indicates the time step, i.e.

$$x(i\cdot\Delta t) \equiv x(t^i) \equiv x^i,$$

Δt denotes the chose time step size, and $\boldsymbol{a}_0$, and $\boldsymbol{v}_0$ denotes the acceleration, and velocity, respectively, both without taking into account the friction force. As we compute values iteratively, initial values for the position as well as the velocity have to be prescribed.
Next, we compute

$$v_1^i \equiv \|\boldsymbol{v}_0^i\| - \frac{F_F\Delta t}{m}, \tag{4}$$

which is the velocity magnitude taking into account the surface friction force defined in (3). This friction force is only valid, if the rover moves forward. In cases, where $\boldsymbol{v}_0^i$ is basically zero, we have to model static friction, otherwise the friction force might result in a movement antiparallel to the previous movement direction. That means,

$$\text{if } v_1^i < 0, \text{ set } v_1^i \equiv 0.$$

The rover velocity is then

$$\boldsymbol{v}^i \equiv \frac{\boldsymbol{v}_0^i}{\|\boldsymbol{v}_0^i\|}v_1^i. \tag{5}$$

To clarify, in case $v_1^i \geqslant 0$, we do not overwrite v_1 meaning that it then contains the value as in (4), after which we again use (5) to get the 2-component rover velocity in the enu system. Moreover, the subscripts 0, and 1 indicate temporary values, while we use no subscript for final or "physical" values.

The local change in position simply follows the rule

$$\Delta \boldsymbol{x}_{enu}^i \equiv \begin{pmatrix} \Delta q_e^i & \Delta q_n^i \end{pmatrix}^\intercal \equiv \boldsymbol{v}^i \Delta t,$$

where $\Delta \boldsymbol{x}_{enu}^i$ contains the two step lengths in east and north direction.

The member `Rover::update_glob_pos()` handles all coordinates using a latitude and a longitude value. These can be understood as "global coordinates" of the planet surface. It takes the previously computed east and north step as well as the old global position as an input to compute the new global position. Now it is clear why the local enu system has its merits: the local east coordinate, q_e, is aligned with the global longitude coordinate, q_{lon}, and the local north coordinate, q_n, is aligned with the global latitude, q_{lat}. The "local-to-global" mapping is realised using the following relations:

$$q_{lat}^i \equiv q_{lat}^{i-1} + \Delta q_n^i \frac{360}{2\pi R},$$
$$q_{lon}^i \equiv q_{lon}^{i-1} + \Delta q_e^i \frac{360}{2\pi R_{lon}},$$

where R is the planetary radius and R_{lon} is the curvature radius of the coordinate line representing the longitude coordinate. It takes into account the fact that the Mars' radius varies with latitude due to its spherical shape and it is computed as follows

$$R_{lon} \equiv R \cos\left(\frac{q_{lat}^{i-1}}{180\pi}\right).$$

We emphasize that the coordinates q_{lat} and q_{lon} have to be interpreted as degrees, due to the traditional use of the terms "longitude" and "latitude". This is not a necessity, the definitions of q_{lat} and q_{lon} can of course equivalently be established using radians/arc lengths.

In order to maintain global coordinate values to stay in their defined ranges, i.e. $q_{lat} \in (-90, 90), q_{lon} \in (0, 360)$, we employ the following exception handling clauses:

$$\text{if } q_{lat} > 90, \text{ set } q_{lat} \equiv 180 - q_{lat}, \text{ and } q_{lon} += 180,$$
$$\text{if } q_{lat} < 90, \text{ set } q_{lat} \equiv -180 - q_{lat}, \text{ and } q_{lon} += 180,$$
$$\text{if } q_{lon} > 360, \text{ set } q_{lon} -= 360,$$
$$\text{if } q_{lon} < 360, \text{ set } q_{lon} += 360.$$

2.4 Data structures

The simulation generates N global positions each containing a longitude and a latitude value. These are simply stored in an array. Tracking and saving time stamps, velocity, or even acceleration values can be avoided, as the mentioned quantities can easily be recovered from the position values and the fact, that Δt remains constant in the current software version. We employ binary files for data storage that both speed up the simulation itself, and also enable fast post processing compared to text-based file formats.

3 Results

In order to show "correctness" of the code, a verification and validation campaign has been employed, where the term "verification", refers to tests that evaluate the tool based on base specifications, and where "validation" referes to tests evaluating the fulfilment of user requirements. This definition implies a possible overlap of validation over verification - i.e. validation implying verification.

We first verify the code using unit tests, which we will not explicitly cover here. Then, to verify the overall system function, and validate that the modelling tool accurately represents the real world within the given assumptions, we perform additional validation in the form of several sample runs.

This consists of three major steps: firstly we validate the integrated physics simulator through testing using five simplified validation cases, representative of different situations, expressed through wind, surface gradient and initial velocity. Then, we validate the overall tool, which includes the interface to read out wind and surface gradient from MCD, as well as the propagator that computes the trajectory. Below, Table 1 shows how the tests, from 1 to 5, validate the tool using a variety of continuous and discontinuous surface gradient (G) and wind (W) inputs, and validate transitions both from rolling to stationary and from stationary to rolling, with an "X" indicating that the test covers a certain test scenario (rolling to stationary or vice versa).

Table 1: Validation objectives for each test

Objective	1	2	3	4	5
Continuous forces		W			
Discontinuous forces			G	W	W,G
Rolling to stationary	X		X	X	X
Stationary to rolling		X	X	X	X

Finally, we validate the tool as a whole through inspecting the paths the rovers take qualitatively and quantitatively, and comparing them with expected behavior based on global terrain and wind patterns. To this end, we run the simulation to generate several hundred rover ground paths given a variation of initial conditions.

3.1 Results of simplified test case validation

The simplified test scenarios should show both that the code behaves as expected, and that the model is convergent: if the discretisation in the time domain is made finer, the error should decrease. We will now present the five tests performed, and assess the validity of our model

Test 1: Zero gradient, zero wind, non-zero initial velocity

The goal of the first test is to validate the transition from a rolling state to a stationary state, in the absence of gradient forces. This validates that a) the model accurately models aerodynamic forces, b) the transform of variables in the local coordinate system to the global one is done correctly, c) the rolling friction of the rover is modeled accurately, and d) the transition from rolling to static friction is modelled correctly.

To this end, we consider a rover with a nonzero initial velocity of $\|\boldsymbol{v}^0\| \equiv 10m/s$, zero terrain gradient, and we compute the deceleration due to surface friction and aerodynamic drag

We will work with the following set of parameters, described in Table 2.

Parameter	Value	Unit
Δt	0.1	s
N	300	
ρ	0.0070872	kg/m^3
r	2.5	m
g	3.72	m/s^2
R	$3.3895 \cdot 10^6$	m
m	20	kg
μ	0.05	
c_D	1.3	
$\|\boldsymbol{v}^0\|$	10	m/s
$\|\boldsymbol{v}_W\|$	0	m/s
α	0	rad

Table 2: Parameters for test 1.

Results are shown in Figure 2.

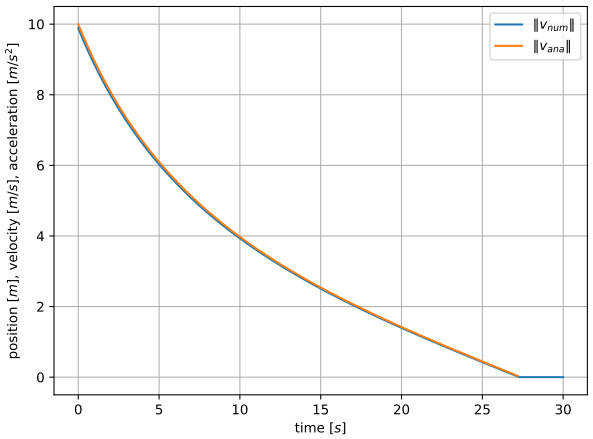

Figure 2: Results from test 1.

The simulation result was compared to an analytical solution from computer algebra. Inspecting Figure 2, we observe a decreasing acceleration over time, as velocity decreases due to the quadratic dependence of the wind force on the velocity difference. Moreover, we see a non-smooth acceleration curve when the rover comes to rest, with the final non-zero value approximately being the acceleration caused by rolling friction only. All these are as expected and show correct modelling of rolling and aerodynamic forces. Error convergence is presented in Figure 3 and Table 3.

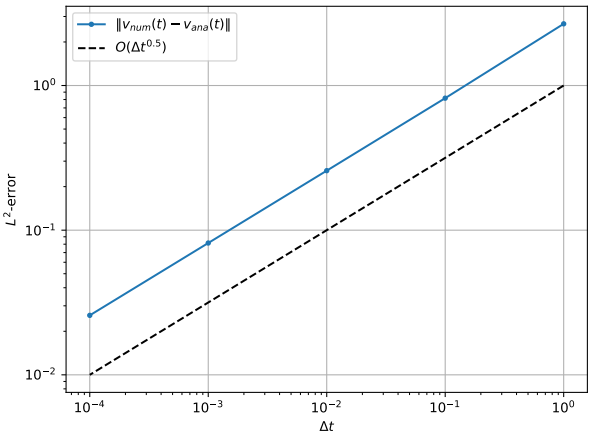

Figure 3: Convergence plot of test 1.

Δt	N	$\|\boldsymbol{v}_{num}(t) - \boldsymbol{v}_{ana}(t)\|$
1	30	2.669
0.1	300	0.8177
0.01	3000	0.2578
0.001	30000	0.0815
0.0001	300000	0.0258

Table 3: Numeric results from test 1.

As mentioned in subsection 2.3, the symbol $\|\cdot\|$ used in Figure 3 and Table 3 denotes the Euclidean norm. In case of Figure 2, the norm is applied to the velocity at every timestep to yield the velocity magnitude.

Test 2: Zero gradient, constant wind speed, zero initial velocity

The goal of the second test is to validate the response of the rover to wind, and the transition from stationary to a rolling condition, and the correct implementation of the physics model in a static setup. To that end, we simulate a constant wind speed during the whole simulation. The terrain gradient remains flat, and we analyse how the rover responds to this changing wind environment.

We will work with the following set of parameters, described in Table 4.

Parameter	Value	Unit
Δt	$0.1s$	s
N	5000	
ρ	0.0070872	kg/m^3
r	2.5	m
g	3.72	m/s^2
R	$3.3895 \cdot 10^6$	m
m	20	kg
μ	0.05	
c_D	1.3	
$\|\boldsymbol{v}_W\|$	10	m/s
α	0	rad

Table 4: Parameters for test 2.

Results are shown in Figure 4.

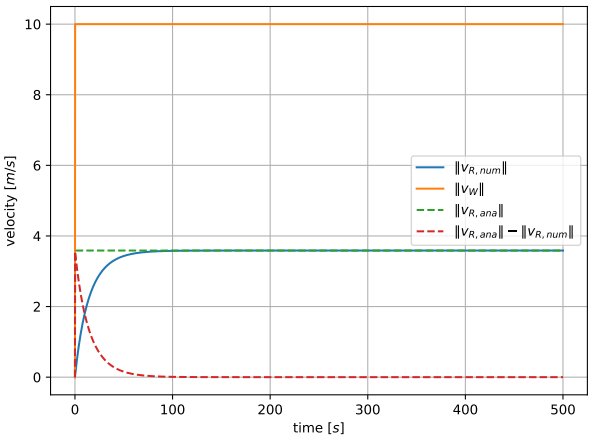

Figure 4: Results from test 2.

The results are visible in Figure 4. The value v_{ana} is the analytical value of the rover speed in the steady state limit. It is value was determined by assuming that the system assumes a quasi-static state after the initial acceleration. A Force balance of wind and friction yields

$$v_{ana} = \|\boldsymbol{v}_W\| - \sqrt{\frac{2mg\mu}{\rho A c_D}}.$$

The numerical value gets arbitrarily close to the analyical one for $t \to \infty$.

Test 3: Rectangular gradient profile, zero wind, zero initial velocity

The goal of the third test is to validate the interaction of the rover with a gradient. This validates not only the accurate modeling of the force acting on the rover as a result of the gradient, but also the transition from stationary to rolling friction, and back to stationary friction regime. For this purpose, the aerodynamic forces are deactivated so that the gradient force modelling may be evaluated on its own. The test simulates a rectangle profile of the gradient over time, which is equivalent with the rover descending down a slope that then levels out after a set time.

The choice of parameters for test 3 is listed in Table 5.

Parameter	Value	Unit
Δt	$0.1s$	s
N	500	
ρ	0.0070872	kg/m^3
r	2.5	m
g	3.72	m/s^2
R	$3.3895 \cdot 10^6$	m
m	20	kg
μ	0.05	
$\boldsymbol{v}_W$	$\boldsymbol{v}_R$	m/s
α	0.2	rad

Table 5: Parameters for test 3.

Due to the choice $\boldsymbol{v}_W \equiv \boldsymbol{v}_R$, we get a scenario with vanishing wind force. The result of the test is shown in Figure 5.

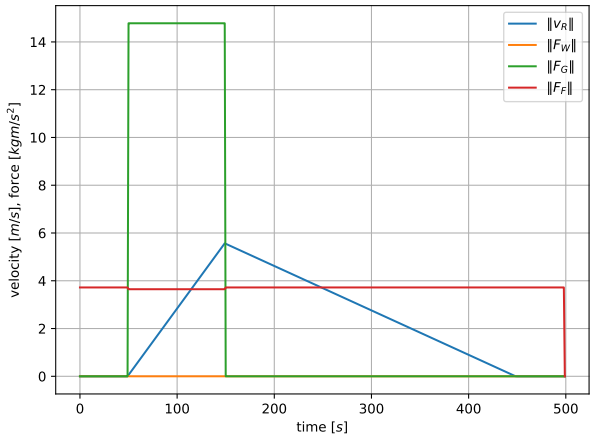

Figure 5: Results from test 3.

In Figure 5, we first observe a linear increase and then decrease of speed, as expected due to the chose forces. Furthermore, we observe a slight decrese of the rolling friction force due to the reduced normal force as a consequence of the incline. Also, the non-smooth velocity curve indicates that the tool properly responds to discontinuous inputs in gradient. The value of the maximum velocity is compared to analytic results in Table 6. We denote the error with $\Delta \boldsymbol{v}_{max}$. Error convergence is presented in Figure 6 and Table 6.

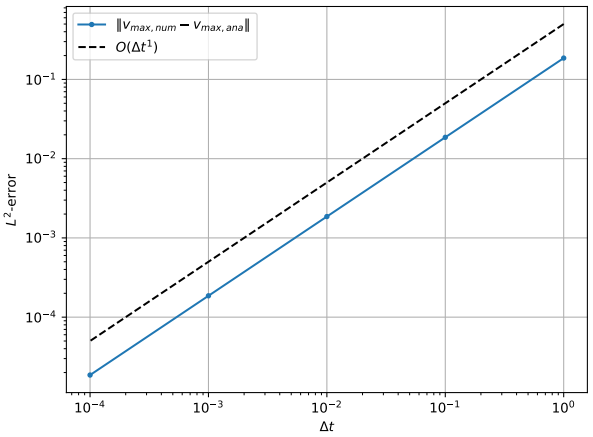

Figure 6: Convergence plot of test 3.

Δt	N	$\|\boldsymbol{v}_{max,num} - \boldsymbol{v}_{max,ana}\|$
1.0	50	0.186
0.1	500	0.0186
0.01	5000	0.00186
0.001	50000	0.000186
0.0001	500000	0.0000186

Table 6: Numeric results from test 3.

We observe firs order convergence with the time step size.

Test 4: Rectangular wind profile, zero gradient, zero initial velocity

This test validates that the rover responds to wind as expected even, if the inputs are discontinuous, and the rover is not approximately in a steady state, but accelerating and decelerating. This includes the transition between stationary and rolling regimes, and vice versa. In this test, the rectangle wind profile over time accelerates the rover from a state of rest, and then returns it to a state of rest upon once the wind turns back 0 again - this is essentially a simplified model of a wind gust.

The choice of parameters is shown in Table 7.

Parameter	Value	Unit
Δt	$1s$	s
N	100	
ρ	0.0070872	kg/m^3
r	2.5	m
g	3.72	m/s^2
R	$3.3895 \cdot 10^6$	m
m	20	kg
μ	0.05	
c_D	1.3	
α	0	rad
$\|\boldsymbol{v}_W\|$	*see (6)	

Table 7: Parameters for test 4.

The wind magnitude is given as

$$\|\boldsymbol{v}_W\| : t \mapsto \begin{cases} 10m/s, & t \in (10, 30) \\ 10m/s, & t \in (60, 80) \\ 0, & else \end{cases} \tag{6}$$

Results are illustrated in Figure 7.

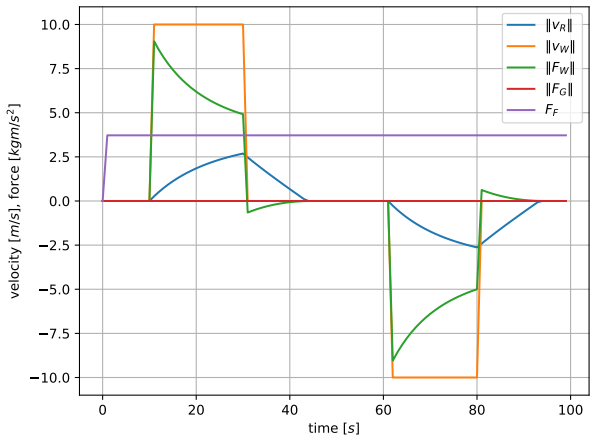

Figure 7: Results from test 4.

The results for the fourth test show that the rover reacts to the wind profile as expected: as can be seen in Figure 7in the beginning of the wind pulse, the rover accelerates more quickly than at the end, as the relative wind speed decreases. It then decelerates in a near-linear fashion, as the deceleration is dominated by the rolling friction. Furthermore, the transitions from stationary to rolling, and back to stationary are positive.

Test 5: Rectangle gradient, rectangle wind, zero initial velocity

The goal of this test is to validate an integrated case, combining wind and gradient forces in different combinations. This validates that the interactions, including the implementation of relative wind, are done correctly. The test incorporates the rover's descent down a hill, modelled as a rectangle wave in the gradient, while concurrently simulating a wind profile . The wind profile is oriented a) parallel, b) antiparallel, or c) orthogonal to the terrain gradient, thereby allowing for a comprehensive analysis of the rover's dynamic response to wind direction relative to the terrain.

The choice of parameters is shown in Table 8.

Parameter	Value	Unit
Δt	$1s$	s
N	100	
ρ	0.0070872	kg/m^3
r	2.5	m
g	3.72	m/s^2
R	$3.3895 \cdot 10^6$	m
m	20	kg
μ	0.05	
c_D	1.3	
α	0	rad
$\|\boldsymbol{v}_W\|$	*see (7)	

Table 8: Parameters for test 5.

The wind magnitude is given as

$$\|\boldsymbol{v}_W\| : t \mapsto \begin{cases} 10m/s, & t \in (3, 13) \\ 10m/s, & t \in (50, 60) \\ 10m/s, & t \in (70, 80) \\ 0, & else \end{cases} \tag{7}$$

The implemented profile is shown in Figure 8.

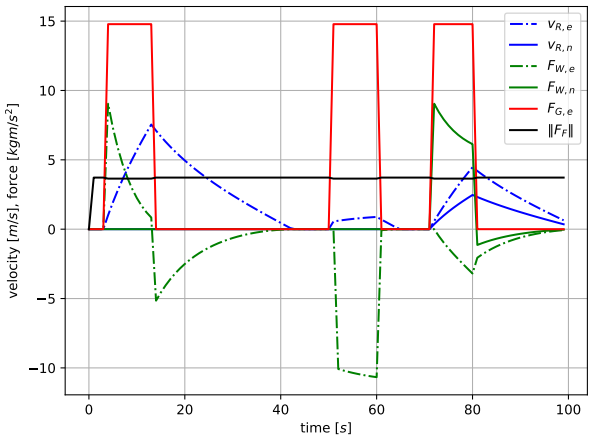

Figure 8: Results from test 5.

Lastly, for the fifth test, the results show that integrating wind and gradient force has no unexpected effects: in

the first pulse shown in Figure 8, gradient and wind are acting in the same direction, which leads to a significantly higher acceleration than for wind or gradient alone (cf. Figure 5 and Figure 7). In the second pulse, wind is acting the gradient, leading to them roughly cancelling out. As a result, the velocity of the rover remains approximately constant. In the last pulse, the wind is orthogonal to the gradient, leading to the introduction of a second velocity component sideways.

Overall, the result of these tests indicate that the rover behaves as expected in these synthetic test cases, implying that the simulation tool is valid.

3.2 *Results of integrated validation using sample simulation runs*

Now, we present the results of the validation runs of the entire simulation using global Mars topography and climate models.

First run: 100 rovers, random starting location

As a first sample simulation, we will compute the trajectories of 100 randomly spawned rovers with a fixed lifetime of 80 Martian sols. In Table 9, the choice of simulation parameters is visible. Here the symbol t_R denotes the mobile lifetime of a rover. The mobile lifetime of a rover is the time it is capable of traversing over the planet surface while being fully operational. It is set to be 80 Martian sols, which is $t_R \equiv 80 \cdot 88775s = 7.102 \cdot 10^6 s$.

Parameter	Value	Unit
Nr. of rovers	100	
t_R	80	sols
g	3.72	m/s^2
R	$3.3895 \cdot 10^6$	m
m	20	kg
r	2.5	m
μ	0.05	
c_D	1.3	

Table 9: Parameters of first sample run.

The symbol r in Table 9 denotes the active aerodynamic radius of the rover, corresponding to the active aerodynamic cross section A introduced in (1). It is defined by $A \equiv r^2\pi$.

As discussed in subsection 2.3, we require initial conditions for both position and velocity. For any rover, we set the initial position, $\boldsymbol{x}^0_{glob} \equiv \begin{pmatrix} q^0_{lat} & q^0_{lon} \end{pmatrix}^\intercal$, in the global coordinate system to be a sample of a probability distribution that is uniform across the planet surface. We note that while for q^0_{lon} this is relatively easy, as we only require a uniform distribution given by $f : q \mapsto 1/360$ for $q \in (0, 360)$ and 0 otherwise, we have to employ the so called inverse cumulative distribution function (CDF) method to compensate for the curved planet surface towards the poles. For this we use the inverse CDF

$$F^{-1} : x \mapsto \arcsin(2x - 1),$$

and then draw uniformly distributed samples $x \in (0, 1)$ to yield suitable values for q^0_{lat}. The initial velocity is set to be $\boldsymbol{v}^0 \equiv \begin{pmatrix} 0 & 0 \end{pmatrix}^\intercal$.

The computed trajectories are visualized over the Martian topography map in Figure 9. A zoomed in version of a trajectory can be seen in Figure 10.

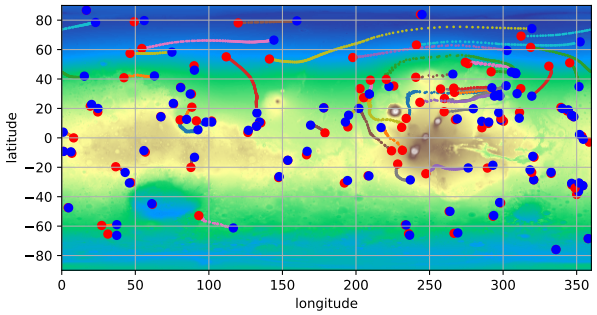

Figure 9: Results sample run with 100 rovers, 80 sols fixed lifetime.

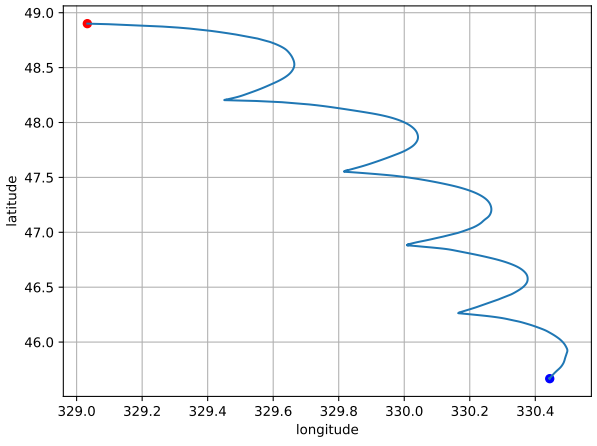

Figure 10: Detailed trajectory artefact.

The motion of rovers on Mars shows expected behaviours:

- Rovers tend to roll downhill.
- Rovers closer to the poles seem to cover larger distances, due to the map projection.

- Rovers get stopped by craters
- Rovers follow the daily shifts in wind directions. This is due to the day-night cycle of the wind velocity. During the night, the wind is slower, and during the day we can observe how the wind velocity vector changes direction periodically.

Second run: 100 rovers, 80 sols average lifetime

Taking into account stochastically distributed lifetimes of different rovers yields a more realistic simulation scenario. In the following simulation run, the modelled lifetime follows an exponential probablity distribtution with the probability density function

$$f : t \mapsto \frac{1}{t_E} \exp\left(-\frac{t}{t_E}\right). \tag{8}$$

Here we have used the symbol t_E for the expected lifetime of a rover. The rest of the simulation parameters remains unchanged, as in Table 9.

The computed trajectories are visualized over the Martian topography map in Figure 11.

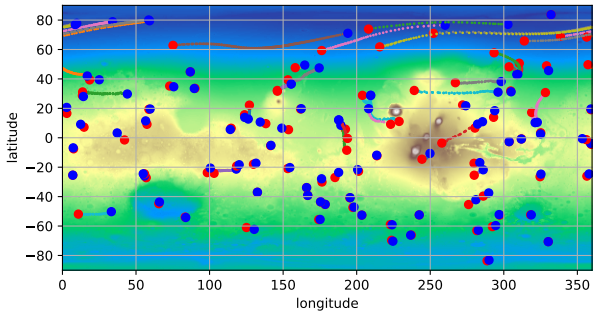

Figure 11: Results of sample run with 100 rovers, 80 sols average lifetime.

Based on the trajectory information of this simulation run, we can now for each rover compute average, maximum and minimum values of travelled distance, lifespan and speed. The results are shown in Table 10.

Parameter	Value	Unit
s_{avg}	421.757	km
s_{max}	2836.303	km
s_{min}	$0km$	km
$t_{R,avg}$	73.409	sols
$t_{R,max}$	268.384	sols
$t_{R,min}$	2.17	sols
$\|\boldsymbol{v}_{avg}\|$	0.119	m/s
$\|\boldsymbol{v}_{max}\|$	1.01	m/s
$\|\boldsymbol{v}_{min}\|$	0	m/s

Table 10: Analysis of sample run 2.

With an average (arithmetic) of roughly 73 Martian sols, the rovers traversed on average a distance of around $422km$ at an average speed of $0.1m/s$. Of course, this average includes the night time, where there is usually no wind.

3.3 Performance of code architecture

To give a rough estimate on the runtime, 100 trajectories, with a mean lifetime of 80 sols could be generated within around 20 hours on a single Intel Core i5-6300U CPU at around 2.4 GHz using a non-parallelized version of the software. As the trajectories of individual rovers are completely independent, the problem solved by the code is embarrassingly parallel up to the numbers of rovers and can therefore easily be exploited to yield a fraction of the runtime mentioned. The above mentioned simulation case will be discussed later in this section.

4 Discussion

In this section we will discuss the presented results and their implications in more detail. First we will discuss the results and the performance of the tool. We will also have a more in depth look into the application of the tool into an existing MBSE method and system design.

4.1 Interpretation of research results

The central focus of this project was to develop and evaluate the utility of a semi-stochastic, numeric software tool for simulating the ground tracks of rovers on the Martian surface. The results of this study shed light on several key aspects concerning that.

This tool not only works effectively, i.e. accurately and in a user friendly way, as it can be run on an average personal computer, but also demonstrates its potential as a valuable resource for addressing critical aspects of mission planning and subsystem sizing in MBSE workflows.

The ability to perform major mission trades and accurately size subsystems is a crucial requirement in mission

planning. Various mission scenarios can easily be implemented in order to evaluate corresponding impacts on the coverage of the planet surface, and on the degree to which certain highly inaccessible geolocations can be reached.

Apart from that, the simulation provides fascinating insights into the potential of the concept of a Tumbleweed rover. It supports the main hypothesis behind all development efforts, as it shows the ability to travel vast distances that clearly outperform any historic mission on any extraterrestrial body. The impact of this technology on advancing scientific research in the fields of geophysics and astrophysics is currently under active investigation.

4.2 Current shortcomings of the tool

One notable weakness in the current research lies in the modeling of climatic variations. The tool primarily relies on a mean case scenario, which, while valuable for initial simulations, may underestimate the complexities of Martian weather patterns, as MCD is a climate rather than a weather model. Implementing a more comprehensive representation of climatic variations around the mean case would enhance the tool's ability to utilize the simulation tool's methods effectively across a singular location and point in time. This refinement could lead to a more robust simulation, providing a more accurate representation of real-world scenarios.

We further recognise weaknesses in the friction model that was described in subsection 2.3, when deriving an expression for the rover velocity, see equation (5). It turns out, that verification tests reveal positive results - still a higher fidelity model, which includes static friction would certainly improve the simulation results.

Currently, the rover propagation does not take into account any small scale obstacles, like small rocks or holes on the planet surface. These could significantly change simulation results, in areas of the surface, where large enough surface features are obstructing rover movement substantially.

The results obtained from the semi-stochastic, numeric simulation tool, as presented in this research, appear robust and consistent with the principles of physics and planetary science. While the results appear correct, it is important to recognize that their interpretation and application may still require careful consideration in the context of specific mission objectives and requirements. Therefore, it is essential to exercise prudence and discernment when translating these results into practical mission planning.

4.3 Relation to literature

As discussed in section subsection 1.1, this semi-stochastic, numeric simulation tool fills the research gap associated with the computation of trajectories of spacecraft which are employed as parts of missions with

- network or swarm based architecture,
- low costs per spacecraft,
- and where the individual vehicles are actuated by semi-stochastic external forces.

Practical implications

One avenue for future research is the optimization of the semi-stochastic, numeric simulation tool to further enhance its efficiency. This can be achieved through code optimization, migration to a lower-level programming language, and employing high performance computing techniques, like parallelization. By continuously improving the tool's computational speed, researchers can conduct more extensive simulations, explore a wider range of scenarios, and gain deeper insights into mission planning and subsystem sizing. This includes planned features as the previously discussed modelling of small scale surface features, which currently result in a substantially higher computational cost.

Researchers and mission planners can utilize the tool to fine-tune rover designs, enhancing their ability to navigate Martian terrains effectively. This optimization can result in more efficient mission outcomes, improved scientific data collection, and extended rover lifetimes. An example of this would be the design trade between several controllability options of a Tumbleweed rover. Questions concerning the necessity of steering or self-propulsion, along with the traditional wind-driven design, can now be answered based on data sets generated from the simulation tool. Another example is the possibility to identify suitable landing sites that align with mission objectives and rover capabilities. This informed decision-making process further reduces mission risks. Lastly, guidance laws and operational principles may be developed and optimised using this tool.

4.4 Application to MBSE

To get a good understanding of the possibilities and limitations of the simulation tool, it is important to look at how it is applied in MBSE methods, as well as how it can be applied to the system design of the Tumbleweed mission concept in specific.

Theoretical Implementation

For this paper, the application of the simulation tool into MBSE methods is done by looking at the integration with the Arcadia method. Arcadia (Architecture Analysis & Design Integrated Approach) is an architecture-centric MBSE framework developed by Thales and incorporated

in the modeling tool Capella [8]. Capella was initially developed internally by Thales but then continued in the public domain by Eclipse Foundation [9].

For the purposes of the readability of this paper, there are a few key concepts of the Arcadia method that need to be touched upon first. First, Arcadia is architecture-centeric approach. That means that the architecture, and the accompanying system- and subsystem interactions and boundaries are at the heart of the method. Justification and Tractability of the architecture is also captured in the model by relating the elements [9]. Next, Arcadia operates in four primary perspectives: Operational Analysis, System Needs analysis , Logical architecture and physical architecture. These layers are interrelated trough allocation of one layers elements to another [9]. For example, Functions from the Functional Analysis layer might be allocated over behavioural components in the System Meed and logical layers. Finally, to enable the trade between different architectures whilst adhering to constraints, viewpoints are defined. Viewpoints formalise the way (sets of) constraints can impact the system architecture. [10] Common viewpoints are Requirements, and department specific analysis or reporting tools, such as structural analysis models and communication link budget tools. Viewpoints are engineering-speciality specific layers acting orthogonal to the previously mentioned perspectives. This means that they can interact simultaneously with the architecture on Operational-, System Needs-, Logical- and physical level. Viewpoints can be used in short-loop design iterations, where many viewpoints are considered at the same time, as well as in depth analysis of the architecture in specific viewpoint, where one viewpoint does detailed (and often tedious) analysis independent from other viewpoints.

The simulation tool can be neatly integrated with Arcadia when it is viewed as a engineering viewpoint. To be precise, it is a swarm performance viewpoint, enabling investigation of the behaviour and performance of the swarm as a whole using the rover design. By treating this tool as a viewpoint it means that the stochastic nature of the analysis can be dis-entangled from the System Engineering workflow. As a result the performance model of the swarm can be used in architecture design as would a conventional (non-stochastic) performance model for conventional missions.

Since this model focuses on quantifying primarily mission level parameters and emergent behaviour of sets of systems (rover swarm) within the mission, a primary use of the tool within MBSE is as a short-loop analysis tool part of a multidisciplinary concurrent engineering study. Concurrent engineering sessions with system architects, system engineers and department specific engineering viewpoints experts allow rapid development and iteration of the system design. The simulation tool here plays a key role in assessing the effect of configuration changes to the mission objectives. Note that using the tool in this manner emphasises the important of producing reliable results in short time frames and by extend the compute-efficiency of the tool. This will be explored in more detail in subsection 4.5

Another potential use of the swarm performance viewpoint is as a integration, verification and validation tool as an in-depth analysis tool. Using the results of the simulation, physical test results can be compared to validate the mission level design architecture. However, in its current state the simulation is not ready to be used in such a manner. To be able to product reliable data for the comparisons between simulation and real world measurements the tool needs to be expanded on in certain key aspects such as allowing real measured climate or weather data to be loaded into the simulation, as well as micro-level topography or terrain for tests at smaller scales. This will also be discussed in more detail in subsection 4.5.

Practical Implementation into the Tumbleweed Mission

to complete the discussion on the implementation of the simulation tool into MBSE methods, we turn from a purely theoretical to a practical discussion through the lens of the Tumbleweed mission concept.

In the current system engineering efforts of Team Tumbleweed there are several noteworthy applications planned for the simulation tool. Each assesses a different aspect of the mission. These investigations are likely to occur for many different configurations of the mission architecture. For that reason future development of the spreading simulation tool might include direct integration with Capella as a viewpoint to enable (semi)automated evaluation of the current architecture configuration. This can be done by using '*Python4Capella*' [11] add-on to create a direct interface between the simulation tool and the model Capella.

- One application is the development of guidance laws for the rover. By making use of emergent behaviour the distribution of the swarm can be controlled using simple laws applied for each rover independently. The simulations enables the testing of different guidance laws by measuring the changes and controlability of the swarm distribution and rover trajectories compared to other laws or a baseline design. This enables a trade-off study to be performed.
- Another application is to aid in the determination of selecting an optimal landing region for the tumbleweed mission. Depending on the topography and atmospheric patterns different landing locations can effect the performance of the mission. The simula-

tion can also be used to asses the importance and effects of the initial landing distribution of the rovers.

- Finally, the simulation is planned to be used to gauge the scientific return of the mission. Using the simulation the distance and regions covered on Mars by the rovers can be investigated. Together with the scientific return analysis for different mission configurations and traversed regions the effect of the rover parameters on the scientific return and by extend the value of the mission can be quantified.

4.5 Future work

In order to fully realize the potential of the tool, there are important next steps to be taken:

- As mentioned, code optimization, and employing high performance computing techniques, like parallelization, will enable modelling of small scale surface features, and is therefore planned to be utilized in the future.
- Similarly, incorporating variability modeling from the (MCD) [5] into the tool is essential. By accounting for climatic variations around the mean case, researchers can conduct simulations that take into account more extreme weather scenarios relevant for long term missions.
- The implementation and refinement of a start-stop controllability function or even the option to self-propel individual rovers are crucial for understand the importance of the manoeuvrability with respect to the mission as a whole. Future work should explore and validate the effectiveness of various design options in enhancing the rover's ability to navigate challenging Martian terrains.
- To enhance code modularity and maintainability, future work can involve separating environment-related software objects into distinct classes. This organizational structure can streamline code development and facilitate easier integration of additional features and environmental models.
- Research efforts should focus on refining the friction model used in the simulation. Improving the accuracy of the friction model will result in more realistic rover behavior and better alignment with actual mission conditions on Mars. In particular, future work should take into account static friction and its implications on slow rover speeds.
- Using the tool to generate more informative data sets. This can include reachability tests of certain inaccessible geolocations, coverage analysis of the planet surface as a whole, and studying the dependence of mission level performance indicators on the mean time to failure and its statistical variance between single rovers.

This additional work will make the tool much more useful, and move it out from a developmental stage to a stage where it can be easily and effectively used by any engineer designing Tumbleweed missions.

5 Conclusion

In conclusion, the work presented here lays the foundation for effective mission design of a Tumbleweed mission: the semi-stochastic and swarm nature of the mission profile requires a tool that can accurately model the ground tracks taken by the rovers in order to inform both mission-level and system-level considerations.

On a mission level, this tool will enable selection of landing locations and times, mission performance analysis, sizing of rover swarms for specific objectives, worst-case analyses, allow for mission planning and operations, and to better inform mission objectives.

On a system (rover) level, this tool will allow for the sizing of Telemetry, Telecommand and Control, Power and Onboard Data Handling subsystems, that all depend on the distance covered and location of the rover. It will allow for the optimisation and testing guidance laws for the rover.

Beyond these limited use cases, this tool is an enabler of MBSE as conventionally used on missions, as it will supply engineers with the ability to rapidly iterate and investigate different design options, as is the standard in today's engineering workflows.

While there is still much work left to do on the integration and refinement of the tool, especially in terms of accurately modeling climatic variations and on runtime optimization, it still is key to unlocking the potential that Tumbleweed rovers have to make Mars exploration accessible to all.

References

[1] J. Rothenbuchner, L. Cohen, F. Abel, D. Buryaka, K. Cuervo, J. Kingsnorth, O. Mikulskyte, A. Phillips, M. Renoldner, and M. Sandrieser, "The Tumbleweed Mission: Enabling Novel Mars Data Sets through Low-Cost Rover Swarms, IAC-22,A3,IP,x72458," in *73rd International Astronautical Congress (IAC), Paris, France, 18-22 September 2022.*, 2022.

[2] J. Rothenbuchner, J. Kingsnorth, K. Cuervo, M. Renoldner, M. Kapadia, P. Wiesen, S. Harris, and M. Resinck, "Initial Mission Base-

line Report," 2022. [Online]. Available: https://www.teamtumbleweed.eu/research/

[3] C. M. Saaj and H. Ibrahim, "ROBUST TRACTION CONTROL AND PATH PLANNING ALGORITHMS FOR PLANETARY MICRO-ROVER SWARMS," publication Title: ESA. [Online]. Available: https://robotics.estec.esa.int/ASTRA/Astra2017/0.%20Tuesday%2020%20June/Poster%20Session/P10_Saaj.pdf

[4] E. Petritoli, M. Cagnetti, and F. Leccese, "Simulation of Autonomous Underwater Vehicles (AUVs) Swarm Diffusion," Sep. 2020.

[5] E. Millour, F. Forget, and A. Spiga, "The Mars Climate Database Version 6.1," *Europlanet Science Congress 2022, Granada, Spain*, 2022. [Online]. Available: https://doi.org/10.5194/epsc2022-786

[6] F. Forget, F. Hourdin, R. Fournier, C. Hourdin, O. Talagrand, M. Collins, S. R. Lewis, P. L. Read, and J.-P. Huot, "Improved general circulation models of the Martian atmosphere from the surface to above 80 km," , vol. 104, no. E10, pp. 24 155–24 176, Oct. 1999.

[7] E. Millour, F. Forget, A. Spiga, M. Vals, V. Zakharov, L. Montabone, F. Lefèvre3, F. Montmessin3, J.-Y. Chaufray, M. A. López-Valverde, F. González-Galindo, S. R. Lewis, P. L. Read, M.-C. Desjean, F. Cipriani, and the MCD development team, "The mars climate database (version 5.3)," *Scientific Workshop: "From Mars Express to ExoMars"*, Feb. 2018.

[8] E. Foundation, "Capella." [Online]. Available: https://www.eclipse.org/capella/

[9] J.-L. Voirin, *Model-based system and architecture engineering with the arcadia method*, ser. Implementation of model based system engineering set. London: ISTE Press ;, 2018. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=1204247

[10] ——, "Arcadia Engineering Landscape," 2023. [Online]. Available: https://www.eclipse.org/capella/arcadia-reference.html

[11] ylussaud, Sophie-pi, Aurelien-Pin, mbats, jgo-a4t, arnauddieumegard, and mypsycho, "python4capella," Sep. 2023. [Online]. Available: https://github.com/labs4capella/python4capella