Project MULE

Moon Utility Loader for Exploration

Space Exploration Project Winter Term 2023/2024

FH Aachen
March X, 2024

Contents

Index

List of Figures

List of Tables

1.4 Rover Parameters

The rover parameters used in further calculations are summarized in Table 1.4.

Parameter	Value	Unit	Description
D	0.8	m	Diameter of the rover’s wheels
b	0.35	m	Width of each wheel
h_g	0.03	m	Grouser height
n_g	8	m	Number of Grousers
b_{track}	2.80	m	Distance between the centres of the left and right wheels
$L_{\text{wheelbase}}$	4.13	m	Distance between the front and rear axles
h_{CoG}	1.04	m	Height of the rover’s centre of gravity
L_{Leg}	1	m	Length of each leg
m	1800	kg	Total mass of the rover
n_{w}	4	-	Number of wheels

Table 1.4: Preliminary Rover Parameters

1.5 Lunar Soil Parameters

Parameter	**Value**	**Units**	**Description**
c_b	170	$\frac{\text{N}}{\text{m}^2}$	Coefficient of soil/wheel cohesion
ϕ	33	$^\circ$	Soil/wheel internal friction angle
K	0.018	m	Fraction of maximum soil shear strength mobilized
n	1	-	Exponent of soil deformation
k_c	1400	$\frac{\text{N}}{\text{m}^2}$	Cohesive modulus of soil deformation
k_ϕ	830000	$\frac{\text{N}}{\text{m}^3}$	Frictional modulus of soil deformation
γ	2470	$\frac{\text{N}}{\text{m}^3}$	Lunar Soil Weight Density
N_q	32.23	-	Terzaghi’s bearing capacity factors
N_c	48.09	-	Terzaghi’s bearing capacity factors
N_γ	33.27	-	Terzaghi’s bearing capacity factors
K_c	33.37	-	Terzaghi’s bearing capacity factors
K_γ	72.77	-	Terzaghi’s bearing capacity factors

Table 1.6: Lunar Soil Trafficability Parameters [Akin 2022b]

1.6 Terrain Interactions

The following calculations model the interactions between the wheel and soil according to Bekker's model and shown in [1.5]

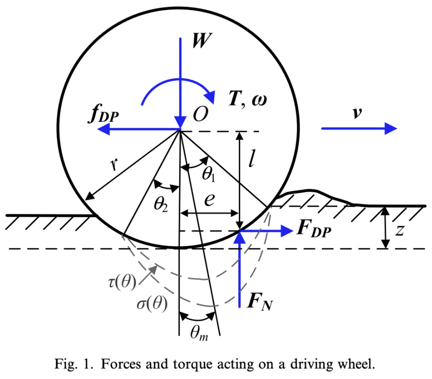


Figure 1.5: Wheel Soil Model, taken from: [Ding et al. 2011]

The Weight force of the whole vehicle on the lunar surface:

$$W_{vehicle} = \frac{1800\,\text{kg} \cdot 1.62\,\frac{\text{m}}{\text{s}^2}}{4} = 2916\,\text{N} \tag{1.1}$$

The Weight force of the rover acting on one wheel:

$$W_{wheel,normal} = \frac{F_G \cdot \cos(\theta)}{n_{wheel}} = \frac{2916\,\text{N} \cdot \cos(0°)}{4} = 729\,\text{N} \tag{1.2}$$

The maximum sinkage at the front wheels on flat terrain is calculated using [Ellery 2015, pg. 148]:

$$\begin{aligned} z &= \left(\frac{3 \cdot W_{wheel,normal}}{(3-n)\cdot(k_c + b\cdot k_\phi)\cdot\sqrt{D}}\right)^{\frac{2}{2n+1}} \\ &= \left(\frac{3 \cdot 729\,\text{N}}{(3-1)\cdot\left(1400\,\frac{\text{N}}{\text{m}^2} + 0.35\,\text{m}\cdot 830000\,\frac{\text{N}}{\text{m}^3}\right)\cdot\sqrt{0.8\,\text{m}}}\right)^{\frac{2}{2\cdot1+1}} \\ &= 0.026\,\text{m} \end{aligned} \tag{1.3}$$

With the sinkage depth known, we can calculate the pressure [Akin 2022b, pg. 8]:

$$\begin{aligned} p_{ground} &= \left(\frac{k_c}{b} \cdot k_\Phi\right) \cdot z^n \\ &= \left(\frac{1400\,\frac{\text{N}}{\text{m}^2}}{0.35\,\text{m}} \cdot 830000\,\frac{\text{N}}{\text{m}^3}\right) \cdot (0.026\,\text{m})^1 \\ &= 21.61\,\text{kPa} > 3\,\text{kPa} \qquad \checkmark \end{aligned} \tag{1.4}$$

As stated in [Berkelman et al. 1995, pg. 44], the Soviets determined after an examination of the performance of the Lunokhod I rover on the lunar soil, that a ground pressure of 3 kPa would provide sufficient support for lunar vehicles and should be used as a design parameter.
The slip ratio s is modelled using a sigmoid function fitted to data by the Mars Exploration Rover [Ellery 2015, pg. 119] assuming a minimum slip of 5 % on even terrain $\left(\theta_{slope} = 0^\circ\right)$ and maximum slip of 100 % at the angle of repose $\left(\theta_{slope} = 23^\circ\right)$ for the slope as shown in [1.6].

$$s = 0.0353 + \frac{1}{1 + 67 \cdot e^{\left(-0.32 \cdot \theta_{slope}\right)}} \tag{1.5}$$

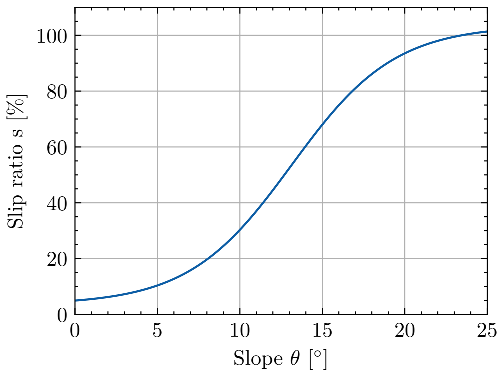


Figure 1.6: Slip Ratio over Slope Model

With the sinkage depth known, it is possible to calculate the entrance angle θ_1, the maximum stress angle θ_m, both of the levers l_{lever} and e_{lever} according to [Ding et al. 2011].

$$\theta_1 = \arccos\left(1 - \frac{z}{r}\right) = \arccos\left(1 - \frac{0.026\,\mathrm{m}}{0.4\,\mathrm{m}}\right) = 0.36\,\mathrm{rad} \tag{1.6}$$

The maximum stress angle is considered as a linear function of θ_1 depending on the slip ratio s:

$$\theta_m = (0.45 + 0.24 \cdot s) \cdot \theta_1 = (0.45 + 0.24 \cdot 0.05) \cdot 0.36\,\mathrm{rad} = 0.17\,\mathrm{rad} \tag{1.7}$$

$$l_{lever} = r\cos(\theta_1 - \theta_m) = 0.4\,\mathrm{m} \cdot \cos(0.36\,\mathrm{rad} - 0.17\,\mathrm{rad}) = 0.39\,\mathrm{m} \tag{1.8}$$

$$e_{lever} = r\sin(\theta_1 - \theta_m) = 0.4\,\mathrm{m} \cdot \sin(0.36\,\mathrm{rad} - 0.17\,\mathrm{rad}) = 0.08\,\mathrm{m} \tag{1.9}$$

The length of the wheel's circumference in contact with the soil $l_{contact}$ is calculated according to [Akin 2022b, pg. 44]

$$l_{contact} = \frac{D}{2}\arccos\left(1 - \frac{2z}{D}\right) = \frac{0.8\,\mathrm{m}}{2}\arccos\left(1 - \frac{2 \cdot 0.026\,\mathrm{m}}{0.8\,\mathrm{m}}\right) = 0.15\,\mathrm{m} \tag{1.10}$$

Using the $l_{contact}$, the Area of the wheel in contact with the soil is calculated:

$$A_{wheel} = b l_{contact} = 0.35\,\mathrm{m} \cdot 0.15\,\mathrm{m} = 0.051\,\mathrm{m}^2 \tag{1.11}$$

The soil bearing capacity can be determined using Terzaghi's Theory. The safe wheel weight on the soil can be calculated as follows [Akin 2022b, pg.34]:

$$\begin{aligned} W_{wheel,safe} &= A_{wheel}\left(c_b N_c + \gamma z N_q + \frac{1}{2}\gamma b N_\gamma\right) \\ &= 0.051\,\mathrm{m}^2 \cdot \left(170\,\frac{\mathrm{N}}{\mathrm{m}^2} \cdot 48.09 + 2470\,\frac{\mathrm{N}}{\mathrm{m}^3} \cdot 0.026\,\mathrm{m} \cdot 32.23 + \frac{1}{2} \cdot 2470\,\frac{\mathrm{N}}{\mathrm{m}^3} \cdot 0.35\,\mathrm{m} \cdot 33.27\right) \\ &= 1249.43\,\mathrm{N} > W_{wheel} = 729\,\mathrm{N} \qquad \checkmark \end{aligned} \tag{1.12}$$

The safe soil pressure therefore is:

$$p_{safe} = \frac{W_{wheel,safe}}{A_{wheel}} = \frac{1249.43\,\mathrm{N}}{0.051\,\mathrm{m}^2} = 24.62\,\mathrm{kPa} \tag{1.13}$$

$$p_{real} = \frac{W_{wheel,normal}}{A_{wheel}} = \frac{729\,\mathrm{N}}{0.051\,\mathrm{m}^2} = 14.37\,\mathrm{kPa} < p_{safe} = 24.62\,\mathrm{kPa} \qquad \checkmark \tag{1.14}$$

The compression resistance is calculated next in accordance to [Akin 2022b]:

$$R_{compression} = \left(\frac{k_c + bk_\phi}{n+1}\right) z^{n+1} = \left(\frac{1400\,\frac{\mathrm{N}}{\mathrm{m}^2} + 0.35\,\mathrm{m}\cdot 830\,000\,\frac{\mathrm{N}}{\mathrm{m}^3}}{1+1}\right)\cdot 0.026\,\mathrm{m}^{(1+1)} = 98.53\,\mathrm{N} \tag{1.15}$$

The rolling resistance is calculated using the sinkage depth of the wheel according to [Akin 2022b] and models internal friction in tires, bearings, seals, etc. using a typical friction coefficient $c_f = 0.05$:

$$R_{rolling} = W_{wheel,normal}\cdot c_f = 729\,\mathrm{N}\cdot 0.05 = 36.45\,\mathrm{N} \tag{1.16}$$

The gravitational resistance for slopes is calculated for the edge case of $\theta_{slope} = 20°$:

$$R_{gravitational} = W_{wheel,normal,20°}\cdot c_f = 2916\,\mathrm{N}\cdot\cos(20°)\cdot 0.05 = 249.33\,\mathrm{N} \tag{1.17}$$

To calculate the bulldozing resistance, we calculate the angle of approach α as well as the length of soil ruptured by compression l_0:

$$\alpha = \arccos\left(1 - \frac{2\cdot z}{D}\right) = \arccos\left(1 - \frac{2\cdot 0.026\,\mathrm{m}}{0.8\,\mathrm{m}}\right) = 0.35\,\mathrm{rad} \tag{1.18}$$

$$l_0 = z\cdot\tan^2\left(\frac{\pi}{2} - \frac{\phi}{4}\right) = 0.026\,\mathrm{m}\cdot\tan^2\left(\frac{\pi}{2} - \frac{33°}{4}\right) = 0.007\,\mathrm{m} \tag{1.19}$$

The bulldozing resistance [Akin 2022b]:

$$\begin{aligned} R_{bulldozing} &= \frac{b\sin(\alpha+\phi)}{2\sin\alpha\cos\phi}\left(2zc_bK_c + \gamma z^2K_\gamma\right) + \frac{l_0^3\gamma}{3}\left(\frac{\pi}{2} - \phi\right) + c_b l_0^2\left[1 + \tan\left(\frac{\pi}{4} + \frac{\phi}{2}\right)\right] \\ &= 204.83\,\mathrm{N} \end{aligned} \tag{1.20}$$

The number of grousers in ground contact:

$$N_g = \frac{2\pi r}{n_g l_{contact}} = \frac{2\pi\cdot 0.4\,\mathrm{m}}{8\cdot 0.15\,\mathrm{m}} = 2.17 \tag{1.21}$$

The tractive force of a wheel with grousers is calculated using [Akin 2022c, pg. 21], but adapted to use the proper slip term for wheeled vehicles according to [Ellery 2015, pg. 117]:

$$\begin{aligned} H &= \left[A_{wheel}c_b\left(1 + \frac{2h_g}{b}\right)N_g + W_{wheel,normal}\tan(\phi)\left(1 + 0.64\frac{h_g}{b}\arctan\frac{b}{h_g}\right)\right]\left[1 - e^{\frac{-sl_{contact}}{K}}\right] \\ &= 176.99\,\mathrm{N} \end{aligned} \tag{1.22}$$

The drawbar pull required for the whole rover to drive equals to soil thrust minus the sum of the resistances. While the compression, gravitational and rolling resistance occur on all wheels, the bulldozing resistance only occurs on the two front wheels [Akin 2022b, pg. 46]. The drawbar pull is calculated for the case of flat terrain, so the gravitational resistance in this case equals to zero:

$$\begin{aligned} DP &= H - \sum R \\ &= 4H - (4R_{compression} + 2R_{bulldozing} + 4R_{gravitational} + 4R_{rolling}) \\ &= 707.96\,\mathrm{N} - 949.61\,\mathrm{N} \\ &= -241.65\,\mathrm{N} \end{aligned} \tag{1.23}$$

Using the required drawbar pull under the condition of maximum slip $s = 1$, we can calculate the maximum grade of negotiable slope [Ellery 2015, pg. 130]:

$$\theta_{max\ slope,calc} = \arctan\left(\frac{DP_{vehicle,s=1}}{W_{vehicle}}\right) = \arctan\left(\frac{1185.26\,\mathrm{N}}{2916\,\mathrm{N}}\right) = 22.12^\circ > 20^\circ \qquad \checkmark \tag{1.24}$$

The required torque is calculated according to [Ding et al. 2011, pg. 30] with an added constant torque of resistance by the seals:

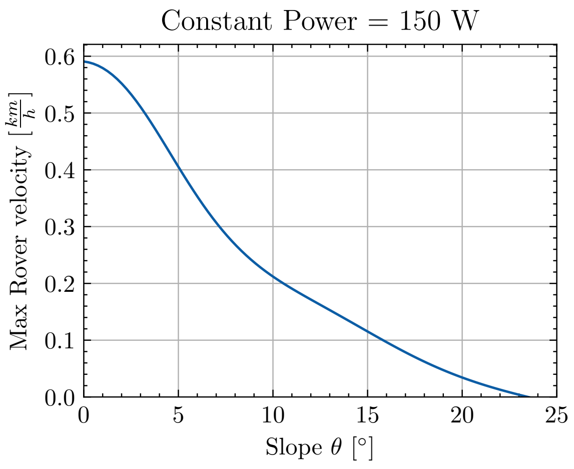
$$\begin{aligned} T_{wheel} &= \frac{DP \cdot l_{lever} + W_{vehicle} \cdot e_{lever}}{n_w} + T_{seals} \\ &= \frac{-241.65\,\mathrm{N} \cdot 0.39\,\mathrm{m} + 2916\,\mathrm{N} \cdot 0.07\,\mathrm{m}}{4} + 28\,\mathrm{Nm} \\ &= 60.79\,\mathrm{Nm} \end{aligned} \tag{1.25}$$

The maximum velocity using the maximum power on flat terrain is:

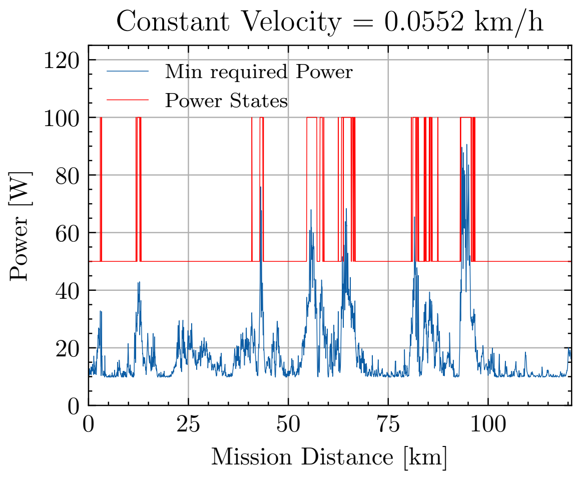
$$v_{rover} = \frac{P_{max}\eta_{gearboxes} r(1-s)}{n_w T_{wheel}} = \frac{150\,\mathrm{W} \cdot 0.7 \cdot 0.4\,\mathrm{m} \cdot (1 - 0.05)}{4 \cdot 60.79\,\mathrm{Nm}} = 0.59\,\frac{\mathrm{km}}{\mathrm{h}} \tag{1.26}$$

The maximum rover velocity with a constant electrical power of 150 W of the range of slopes is shown in Figure 1.7a. Calculating the minimum required average velocity to fulfil the mission requirement, the power profile over the mission distance can be calculated to determine the subsystem power states as shown in Figure 1.7b:

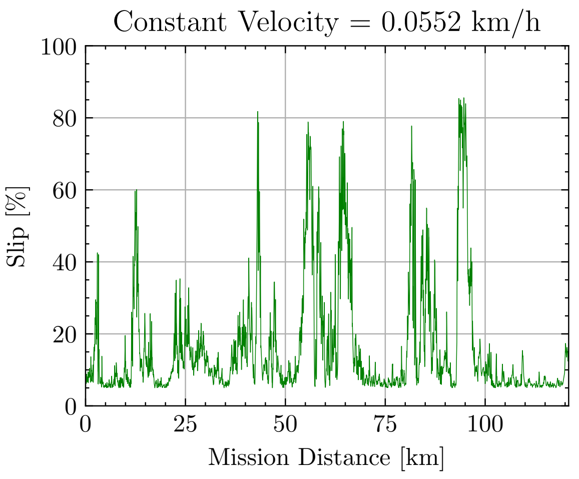
$$v_{min,mission} = \frac{d_{mission}}{(T_{mission} - T_{standing}) \cdot 365\,\frac{\mathrm{d}}{\mathrm{a}} \cdot 24\,\frac{\mathrm{h}}{\mathrm{d}}} = \frac{120.89\,\mathrm{km}}{(0.5\,\mathrm{a} - 0.25\,\mathrm{a}) \cdot 365\,\frac{\mathrm{d}}{\mathrm{a}} \cdot 24\,\frac{\mathrm{h}}{\mathrm{d}}} = 0.0552\,\frac{\mathrm{km}}{\mathrm{h}} \tag{1.27}$$



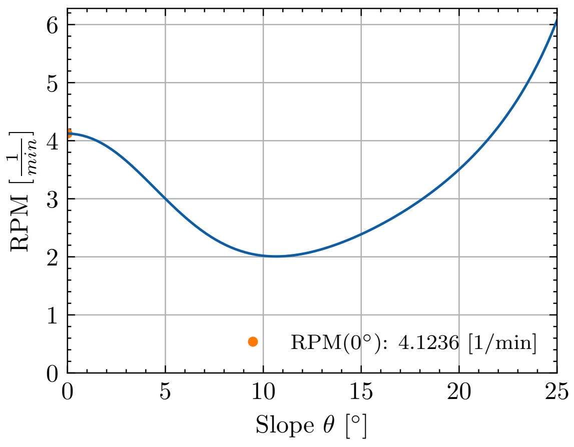
(a) Maximum Rover Velocity at Constant Power over Slope



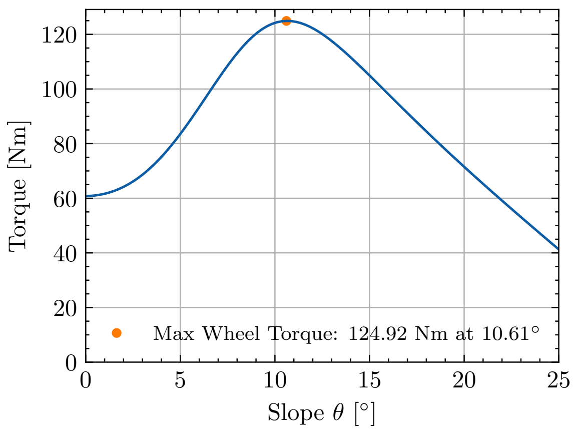
(b) Mission Power Profile and Locomotion Subsystem Power States



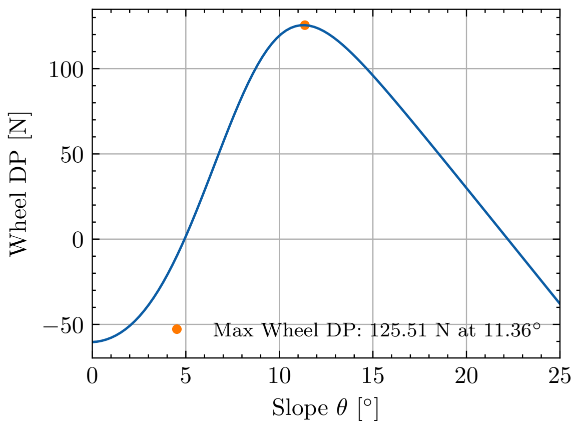
(c) Mission Slip Profile



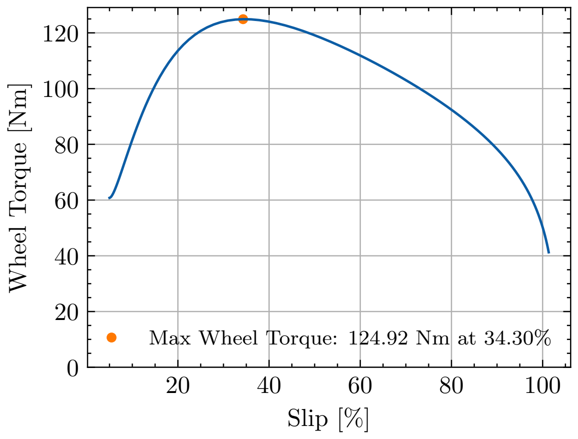
(d) RPM over Slope



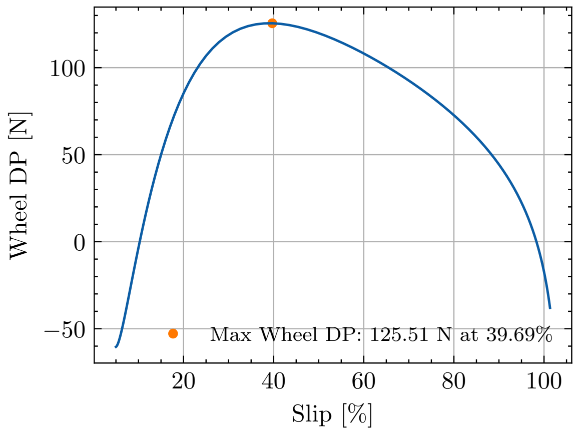
(e) Torque over Slope



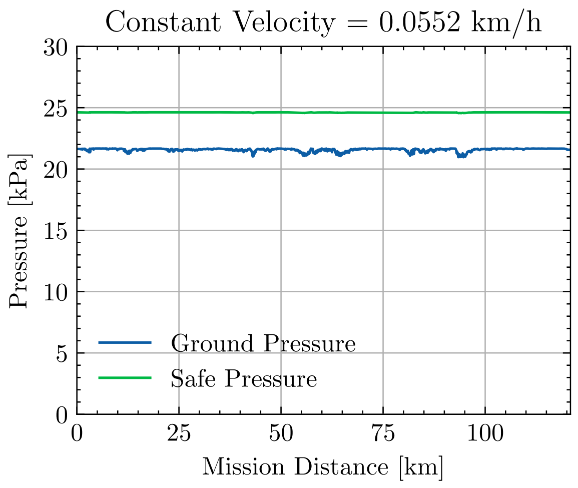
(f) Drawbar Pull over Slope



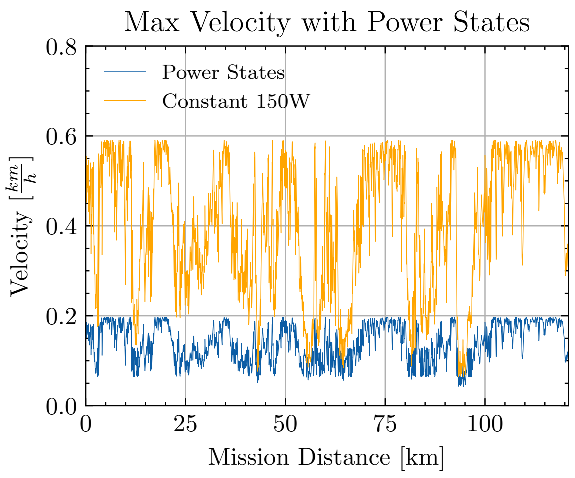
(a) Torque over Slip



(b) Drawbar Pull over Slip



(c) Pressure over Mission Distance



(d) Max Velocity Profile with Power States

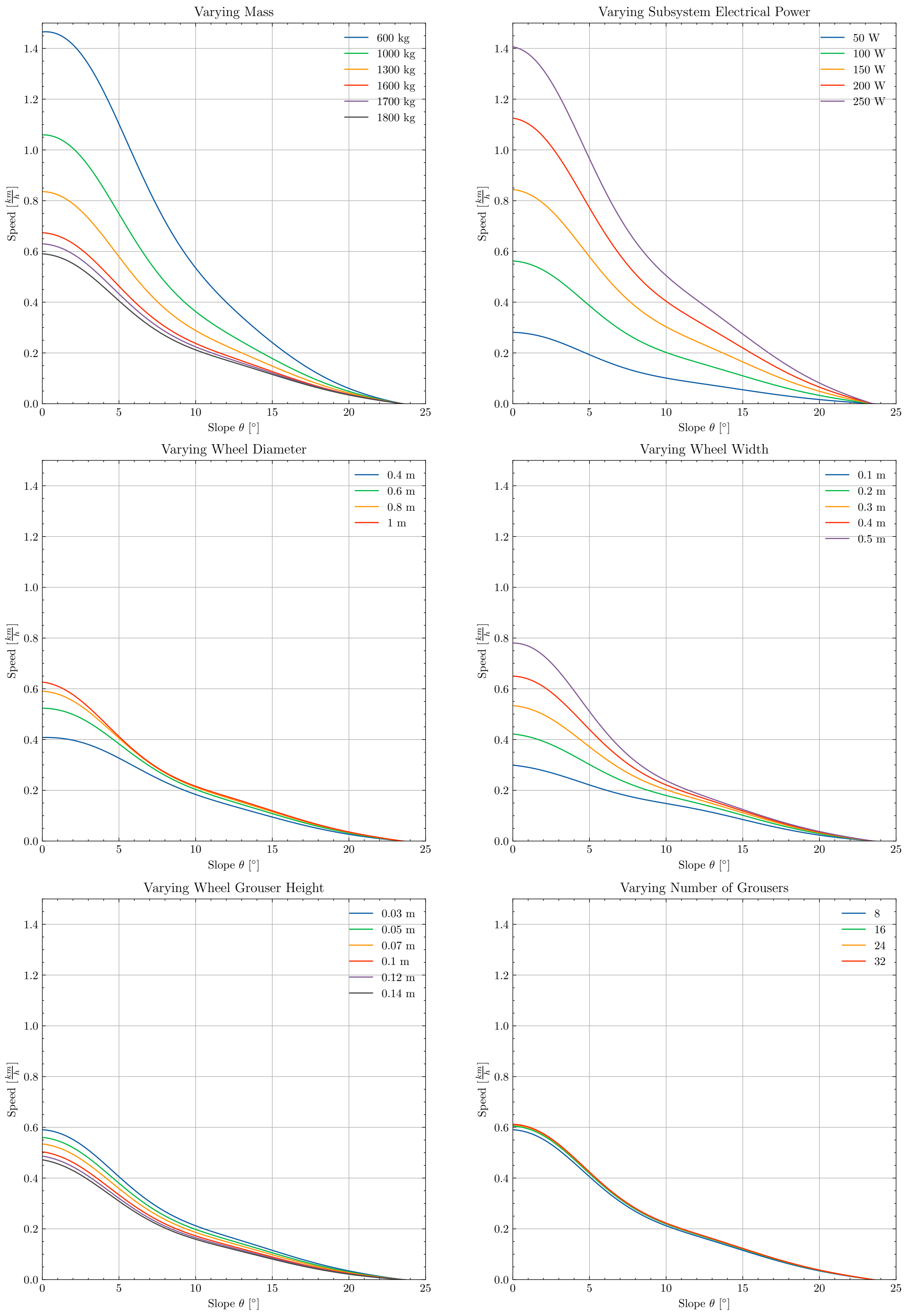


Figure 1.9: Terrain Simulation Varying Parameters

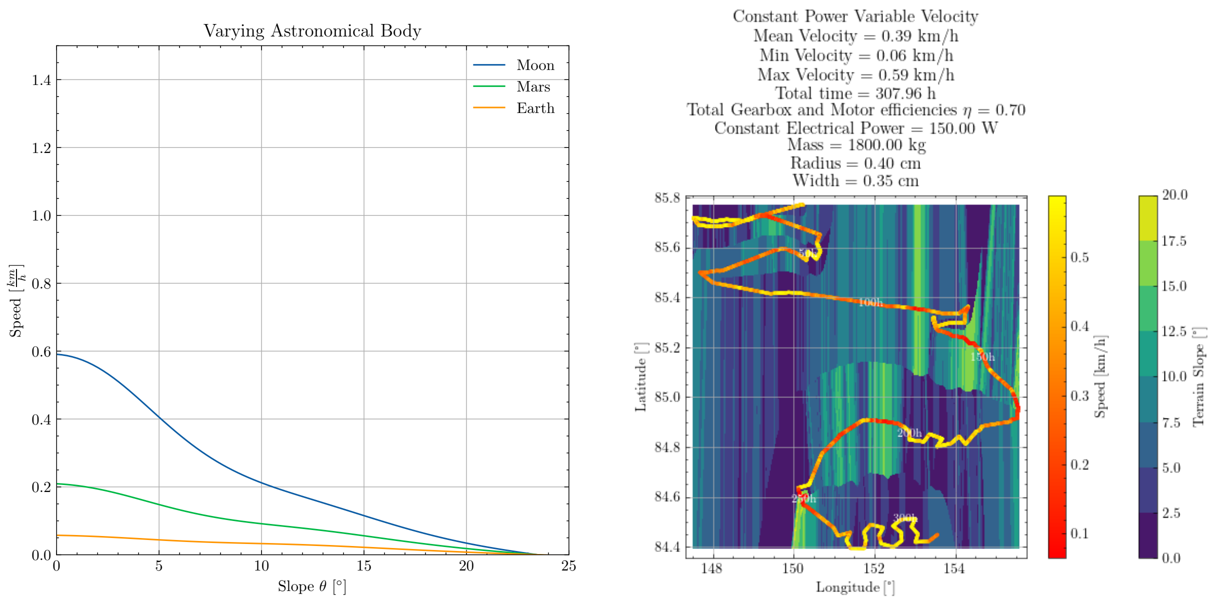
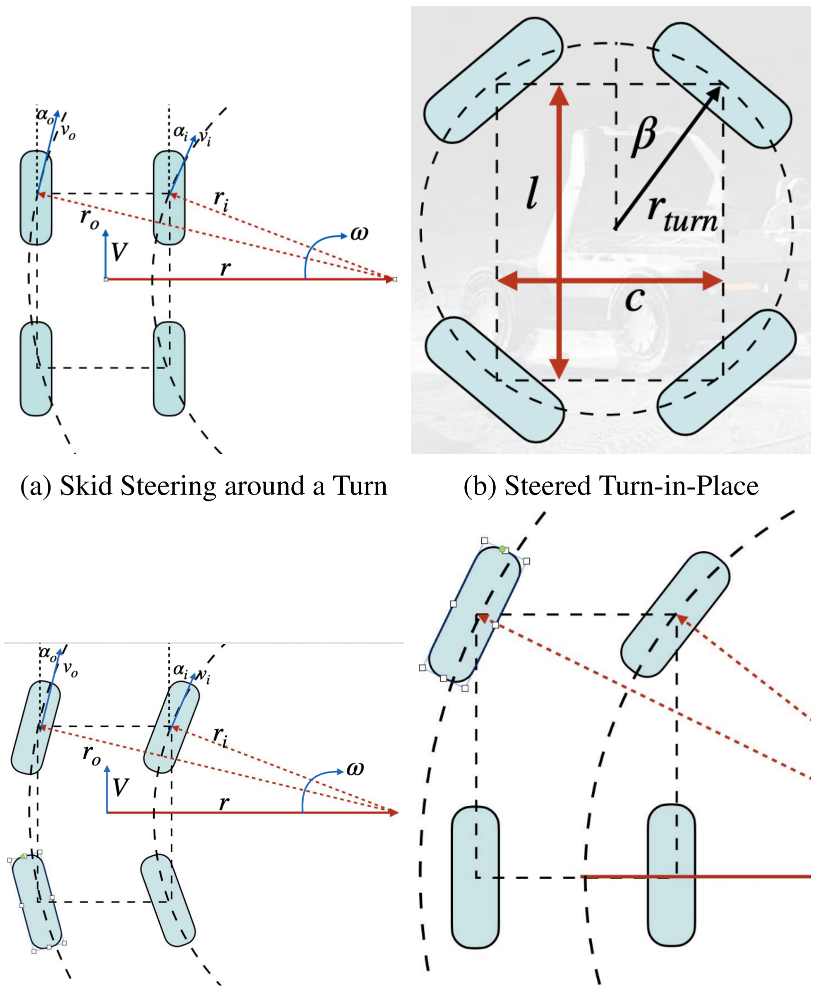


Figure 1.10: Terrain Simulation Varying Parameters and Map

1.7 Steering

Steering is performed using Skid Steering around a Turn as shown in Figure 1.11a.Other steering methods such as a steered Turn-in-Place or Double- or Single Ackermann Steering would have required additional actuators and mechanisms, increasing complexity and adding possible failure points.



(a) Skid Steering around a Turn (b) Steered Turn-in-Place

(c) Double Ackermann Steering (d) Single Ackermann Steering

Figure 1.11: Steering Methods, taken from [Akin 2022a]

1.8 Static Stability

$$\theta_{static,pitchover} = \tan^{-1}\left(\frac{L_{wheelbase}}{2 \cdot h_{CoG}}\right) = \tan^{-1}\left(\frac{4.13\,\mathrm{m}}{2 \cdot 1.04\,\mathrm{m}}\right) = 63^\circ > \theta_{max} = 20^\circ \qquad \checkmark \tag{1.28}$$

$$\theta_{static,rollover} = \tan^{-1}\left(\frac{2.8\,\mathrm{m}}{2 \cdot 1.04\,\mathrm{m}}\right) = 53^\circ > \theta_{max} = 20^\circ \qquad \checkmark \tag{1.29}$$

Bibliography

Bibliography

[Aki22a] Dave Akin. *Steering Forces/Slopes and Static Stability*. https://spacecraft.ssl.umd.edu/academics/788XF22/788XF22L10.steering-slopesx.pdf. Accessed: 2023–01-10. 2022.

[Aki22b] Dave Akin. *Terramechanics 1: Passive Rolling Resistance*. https://spacecraft.ssl.umd.edu/academics/788XF22/788XF22L06.terramechanics1_r1x.pdf. Accessed: 2023–01-06. 2022.

[Aki22c] Dave Akin. *Terramechanics 2: Traction*. https://spacecraft.ssl.umd.edu/academics/788XF22/788XF22L07.terramechanics2x.pdf. Accessed: 2023–01-06. 2022.

[Ber+95] Peter Berkelman et al. *Design of a Day / Night Lunar Rover*. Tech. rep. CMU-RI-TR-95-24. Pittsburgh, PA: Carnegie Mellon University, June 1995.

[BWW08] P. W. Bartlett, David S. Wettergreen, and William Whittaker. "Design of the Scarab Rover for Mobility & Drilling in the Lunar Cold Traps". In: 2008. URL: https://api.semanticscholar.org/CorpusID:9428456.

[Din+11] Liang Ding et al. "Experimental study and analysis on driving wheels' performance for planetary exploration rovers moving in deformable soil". In: *Journal of Terramechanics* 48.1 (2011), pp. 27–45. ISSN: 0022-4898. DOI: https://doi.org/10.1016/j.jterra.2010.08.001. URL: https://www.sciencedirect.com/science/article/pii/S0022489810000601.

[Ell15] Alex Ellery. *Planetary Rovers: Robotic Exploration of the Solar System*. 1st. Springer Publishing Company, Incorporated, 2015. ISBN: 3642032583.

[Gar21] John D. Garrett. "garrettj403/SciencePlots". Version 1.0.9. In: (Sept. 2021). DOI: 10.5281/zenodo.4106649. URL: http://doi.org/10.5281/zenodo.4106649.

[GB17] Nikola Georgiev and Joel Burdick. "Design and analysis of planar rotary springs". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 4777–4784. DOI: 10.1109/IROS.2017.8206352.

[Ula+23] Stephan Ulamec et al. "Science objectives of the MMX rover". In: *Acta Astronautica* 210 (2023), pp. 95–101. ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2023.05.012. URL: https://www.sciencedirect.com/science/article/pii/S0094576523002448.

Terrain Simulation

January 9, 2024

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scienceplots
plt.style.use('science')

#plt.rcParams['figure.dpi'] = 600
#plt.rcParams['savefig.dpi'] = 600

n_wheel = 4 # number of wheels
diameter = 0.8
radius = diameter/2
b_width = 0.35 # wheel width in m
h_grouser = 0.03 # grouser height
n_grouser = 8 # number of grousers
eta = 0.7 # gearbox efficiency
power_constant = 150*eta # constant power for the whole locomotion subsystem
v_rover_constant = 0.05 # km/h
c_b = 170 # N/m^2 coefficient of soil/wheel cohesion
n = 1 # exponent of soil deformation
k_c = 1400 # N/m^2 cohesive modulus of soil deformation
k_phi = 830000  # N/m^3 frictional modulus of soil deformation
K = 0.018  # coefficient of soil slip in m
gamma = 2470 # N/m3 lunar soil density
phi = 33 # deg soil internal friction angle
phi_rad = np.deg2rad(phi) # rad soil internal friction angle
mu = np.tan(phi_rad) # static friction coefficient
N_q = 32.23
N_c = 48.09
N_gamma = 33.27
K_c = 33.37
K_phi = 72.77
K_gamma = 81.93
mass = 1800 # mass of the rover
gravity = 1.62 # gravitational constant of the moon
slope = np.linspace(0, 25, 100)
#slope = 0
```

```
slip = 0.0353 + 1/(1+67*np.exp(-0.32*slope))
#slip = 1
k = (k_c / b_width) + k_phi
c_f = 0.05 # internal rolling friction coefficient (bearings)

W_vehicle = mass * gravity
W_vehicle_normal = W_vehicle * np.cos(np.deg2rad(slope))
W_wheel = mass * gravity / n_wheel
W_wheel_normal = W_wheel * np.cos(np.deg2rad(slope))

z_sinkage = (3 * W_wheel_normal / ((3 - n) * (k_c + b_width * k_phi) * np.
  ↪sqrt(diameter))) ** (2/(2*n+1))
pressure = k * z_sinkage ** n

theta_1 = np.arccos(1 - z_sinkage / radius)
theta_m = (0.45 + 0.24 * slip) * theta_1
lever_e = radius * np.sin(theta_1 - theta_m)
lever_l = radius * np.cos(theta_1 - theta_m)
l_contact = (diameter / 2) * np.arccos(1 - (2 * z_sinkage / diameter))
A = b_width * l_contact
alpha = np.arccos(1 - 2 * z_sinkage / diameter) # angle of approach
l_0 = z_sinkage * np.tan(np.pi / 4 - phi_rad / 2) ** 2 # distance of rupture

R_compression = ((b_width * k) / (n+1)) * z_sinkage ** (n+1)
R_bulldozing = (b_width*np.sin(alpha+phi_rad) / (2*np.sin(alpha)*np.
  ↪cos(phi_rad))) * (2*z_sinkage*c_b*K_c + gamma*K_gamma*z_sinkage**2) +␣
  ↪((gamma*l_0**3)/3) * (np.pi/2 - phi_rad) + (c_b*l_0**2) * (1+np.tan(np.pi/4␣
  ↪+ phi_rad/2))
R_gravitational = W_wheel * np.sin(np.deg2rad(slope))
R_rolling = W_wheel_normal * c_f # internal (bearings)

sum_R = 4* R_compression + 2* R_bulldozing + 4* R_gravitational + 4* R_rolling
n_grouser_contact = 2 * np.pi * radius / (n_grouser * l_contact)

sum_H_grouser = 4* (A*c_b*(1+2*h_grouser/b_width)*n_grouser_contact +␣
  ↪W_wheel_normal*mu*(1 + (0.64*h_grouser/b_width) * np.arctan(b_width/
  ↪h_grouser))) * (1-np.exp(-slip * l_contact / K))

W_wheel_safe = A * (c_b*N_c + gamma*z_sinkage*N_q + gamma*b_width*N_gamma/2)
pressure_soil_safe = W_wheel_safe / A
pressure_real = W_wheel / A
pressure_margin = pressure_real / pressure_soil_safe
weight_margin = W_wheel / W_wheel_safe

drawbar_pull = sum_H_grouser - sum_R
torque = (((drawbar_pull * lever_l) + (W_vehicle_normal * lever_e))/n_wheel) +␣
  ↪28
```

```
omega = power_constant / (torque * n_wheel)
rpm = omega * 60/ (2 * np.pi)
v_rover = omega * radius * 3.6 * (1-slip)
power_variable = (v_rover_constant * torque * n_wheel) / (radius * 3.6 *␣
↪(1-slip))

theta_max = np.rad2deg(np.arctan(drawbar_pull / W_vehicle))
```

[ ]:
```
plt.plot(slope, v_rover)
plt.xlabel('Slope $\\theta$ [$^\circ$]')
plt.ylabel('Max Rover velocity [$\\frac{km}{h}$]')
plt.xlim(0, 25)
plt.ylim(0,)
plt.title('Constant Power = {:.0f} W'.format(power_constant/eta))
plt.grid()
plt.savefig("max_rover_velocity_slope.pdf")
plt.show()
```

[ ]:
```
plt.plot(slope, power_variable)
plt.xlabel('Slope $\\theta$ [$^\circ$]')
plt.ylabel('Variable Power [W]')
plt.xlim(0,)
plt.ylim(0,150)
plt.grid()
plt.savefig("power_variable_slope.pdf")
plt.show()
```

[ ]:
```
# Find the index of the maximum torque
max_torque_index = np.argmax(torque)
max_torque_value = torque[max_torque_index]
max_torque_theta = slope[max_torque_index]
plt.plot(max_torque_theta, max_torque_value, 'o', color=(255/255, 120/255, 0/
↪255), markersize=3)
plt.plot(slope, torque)
plt.xlabel('Slope $\\theta$ [$^\circ$]')
plt.ylabel('Torque [Nm]')
plt.xlim(0, 25)
plt.ylim(0,)
plt.grid()
plt.legend([f'Max Wheel Torque: {max_torque_value:.2f} Nm at {max_torque_theta:.
↪2f}$^\circ$'], loc='lower right', fontsize='small')
plt.savefig("max_torque_slope.pdf")
plt.show()
```

[ ]:
```
# Find the index of the maximum torque
max_dp_index = np.argmax(drawbar_pull)
max_dp_value = drawbar_pull[max_dp_index]/n_wheel
```

```
max_dp_slip = slip[max_dp_index]*100
plt.plot(max_dp_slip, max_dp_value, 'o', color=(255/255, 120/255, 0/255),␣
↪markersize=3)
plt.plot(slip*100, drawbar_pull/n_wheel)
plt.xlabel('Slip [\%]')
plt.ylabel('Wheel DP [N]')
plt.grid()
plt.legend([f'Max Wheel DP: {max_dp_value:.2f} N at {max_dp_slip:.2f}$\%$'],␣
↪loc='lower right', fontsize='small')
plt.savefig("max_drawbar_pull_slip.pdf")
plt.show()
```

```
[ ]: # Find the index of the maximum torque
max_torque_index = np.argmax(torque)
max_torque_value = torque[max_torque_index]
max_torque_slip = slip[max_torque_index]*100
plt.plot(max_torque_slip, max_torque_value, 'o', color=(255/255, 120/255, 0/
↪255), markersize=3)
plt.plot(slip*100, torque)
plt.xlabel('Slip [\%]')
plt.ylabel('Wheel Torque [Nm]')
plt.grid()
plt.ylim(0,)
plt.legend([f'Max Wheel Torque: {max_torque_value:.2f} Nm at {max_torque_slip:.
↪2f}$\%$'], loc='lower right', fontsize='small')
plt.savefig("max_torque_slip.pdf")
plt.show()
```

```
[ ]: # Find the index of the maximum torque
max_dp_index = np.argmax(drawbar_pull)
max_dp_value = drawbar_pull[max_dp_index]/n_wheel
max_dp_theta = slope[max_dp_index]
plt.plot(max_dp_theta, max_dp_value, 'o', color=(255/255, 120/255, 0/255),␣
↪markersize=3)
plt.plot(slope, drawbar_pull/n_wheel)
plt.xlabel('Slope $\\theta$ [$^\circ$]')
plt.ylabel('Wheel DP [N]')
plt.xlim(0, 25)
plt.grid()
plt.legend([f'Max Wheel DP: {max_dp_value:.2f} N at {max_dp_theta:.
↪2f}$^\circ$'], loc='lower right', fontsize='small')
plt.savefig("max_drawbar_pull_slope.pdf")
plt.show()
```

```
[ ]: plt.plot(0, rpm[0], 'o', color=(255/255, 120/255, 0/255), markersize=3)
plt.plot(slope, rpm)
plt.xlabel('Slope $\\theta$ [$^\circ$]')
plt.ylabel('RPM [$\\frac{1}{min}$]')
plt.xlim(0,25)
plt.ylim(0,)
plt.grid()
plt.legend([f'RPM($0^\circ$): {rpm[0]:.4f} [1/min]'], loc='lower right',␣
 ↪fontsize='small')
plt.savefig("max_rpm_slope.pdf")
plt.show()
```

```
[ ]: plt.plot(slope, slip*100)
plt.xlabel('Slope $\\theta$ [$^\circ$]')
plt.ylabel('Slip ratio s [\%]')
plt.xlim(0, 25)
plt.ylim(0, 110)
plt.grid()
plt.savefig("sigmoid_slip.pdf")
plt.show()
```

```
[ ]: data = pd.read_csv('slope.csv')
distance = data['position']
longitude = -data['lon']
latitude = -data['lat']
slope = data['TerrainSlope']

def func(slope, v_rover_constant, mass, diameter, power_constant, b_width,␣
 ↪gravity, h_grouser, n_grouser):
    n_wheel = 4 # number of wheels
    #diameter = 0.8
    radius = diameter/2
    #b_width = 0.35 # wheel width in m
    #h_grouser = 0.03 # grouser height
    #n_grouser = 8 # number of grousers
    eta = 0.7 # gearbox efficiency
    #power_constant = 150*eta # constant power for the whole locomotion␣
 ↪subsystem
    #v_rover_constant = 0.05 # km/h
    c_b = 170 # N/m^2 coefficient of soil/wheel cohesion
    n = 1 # exponent of soil deformation
    k_c = 1400 # N/m^2 cohesive modulus of soil deformation
    k_phi = 830000  # N/m^3 frictional modulus of soil deformation
    K = 0.018  # coefficient of soil slip in m
    gamma = 2470 # N/m3 lunar soil density
    phi = 33 # deg soil internal friction angle
    phi_rad = np.deg2rad(phi) # rad soil internal friction angle
    mu = np.tan(phi_rad) # static friction coefficient
    N_q = 32.23
    N_c = 48.09
    N_gamma = 33.27
    K_c = 33.37
    K_phi = 72.77
    K_gamma = 81.93
    #mass = 1800 # mass of the rover
    #gravity = 1.62 # gravitational constant of the moon
    #slope = np.linspace(0, 25, 100)
    #slope = 0
    slip = 0.0353 + 1/(1+67*np.exp(-0.32*slope))
    #slip = 1
    k = (k_c / b_width) + k_phi
    c_f = 0.05 # internal rolling friction coefficient (bearings)

    W_vehicle = mass * gravity
    W_vehicle_normal = W_vehicle * np.cos(np.deg2rad(slope))
    W_wheel = mass * gravity / n_wheel
    W_wheel_normal = W_wheel * np.cos(np.deg2rad(slope))

    z_sinkage = (3 * W_wheel_normal / ((3 - n) * (k_c + b_width * k_phi) * np.
↪sqrt(diameter))) ** (2/(2*n+1))
    pressure = k * z_sinkage ** n

    theta_1 = np.arccos(1 - z_sinkage / radius)
    theta_m = (0.45 + 0.24 * slip) * theta_1
    lever_e = radius * np.sin(theta_1 - theta_m)
    lever_l = radius * np.cos(theta_1 - theta_m)
    l_contact = (diameter / 2) * np.arccos(1 - (2 * z_sinkage / diameter))
    A = b_width * l_contact
    alpha = np.arccos(1 - 2 * z_sinkage / diameter) # angle of approach
    l_0 = z_sinkage * np.tan(np.pi / 4 - phi_rad / 2) ** 2 # distance of rupture

    R_compression = ((b_width * k) / (n+1)) * z_sinkage ** (n+1)
    R_bulldozing = (b_width*np.sin(alpha+phi_rad) / (2*np.sin(alpha)*np.
↪cos(phi_rad))) * (2*z_sinkage*c_b*K_c + gamma*K_gamma*z_sinkage**2) +␣
↪((gamma*l_0**3)/3) * (np.pi/2 - phi_rad) + (c_b*l_0**2) * (1+np.tan(np.pi/4␣
↪+ phi_rad/2))
    R_gravitational = W_wheel * np.sin(np.deg2rad(slope))
    R_rolling = W_wheel_normal * c_f # internal (bearings)

    sum_R = 4* R_compression + 2* R_bulldozing + 4* R_gravitational + 4*␣
↪R_rolling
    n_grouser_contact = 2 * np.pi * radius / (n_grouser * l_contact)

    sum_H_grouser = 4* (A*c_b*(1+2*h_grouser/b_width)*n_grouser_contact +␣
↪W_wheel_normal*mu*(1 + (0.64*h_grouser/b_width) * np.arctan(b_width/
↪h_grouser))) * (1-np.exp(-slip * l_contact / K))
```

```
    W_wheel_safe = A * (c_b*N_c + gamma*z_sinkage*N_q + gamma*b_width*N_gamma/2)
    pressure_soil_safe = W_wheel_safe / A
    pressure_real = W_wheel / A
    pressure_margin = pressure_real / pressure_soil_safe
    weight_margin = W_wheel / W_wheel_safe

    drawbar_pull = sum_H_grouser - sum_R
    torque = (((drawbar_pull * lever_l) + (W_vehicle_normal * lever_e))/
 ↪n_wheel) + 28
    omega = power_constant / (torque * n_wheel)
    rpm = omega * 60/ (2 * np.pi)
    v_rover = omega * radius * 3.6 * (1-slip)
    power_variable = (v_rover_constant * torque * n_wheel) / (radius * 3.6 *␣
 ↪(1-slip))
    power_variable_step = np.clip(50 * np.ceil(np.clip(power_variable,30,100) /␣
 ↪30), 50, 100)
    return v_rover, power_variable, power_variable_step, drawbar_pull, torque,␣
 ↪rpm, pressure, pressure_soil_safe, pressure_real

solution = func(slope, v_rover_constant, mass, diameter, power_constant,␣
 ↪b_width, gravity, h_grouser, n_grouser)
v_mean = np.mean(solution[0])
slope_mean = np.mean(slope)
end_distance = distance.iloc[-1]
total_time = end_distance/v_mean
```

[ ]:
```
v_mean
```

[ ]:
```
slope_mean
```

[ ]:
```
end_distance
```

[ ]:
```
total_time
```

[ ]:
```
v_rover_constant = distance.iloc[-1] / ((0.5 - 0.25) * 365 * 24)
v_rover_constant
```

[ ]:
```
plt.plot(distance, func(slope, v_rover_constant, 1800, 0.8, 150*0.7, 0.35, 1.
 ↪62, 0.03, 8)[1], linewidth=0.3, label='Min required Power')
plt.plot(distance, func(slope, v_rover_constant, 1800, 0.8, 150*0.7, 0.35, 1.
 ↪62, 0.03, 8)[2], color='red',  linewidth=0.3, label='Power States')
plt.xlabel('Mission Distance [km]')
plt.ylabel('Power [W]')
plt.xlim(0, distance.iloc[-1])
plt.ylim(0, 125)
plt.title('Constant Velocity = {:.4f} km/h'.format(v_rover_constant))
plt.grid()
plt.legend(loc='upper left', fontsize='small')
plt.savefig("rover_power_over_mission_distance.pdf")
plt.show()
```

[ ]:
```
power_states = func(slope, v_rover_constant, 1800, 0.8, 150*0.7, 0.35, 1.62, 0.
↪03, 8)[2]
plt.plot(distance, func(slope, v_rover_constant, 1800, 0.8, power_states*0.7, 0.
↪35, 1.62, 0.03, 8)[0],  linewidth=0.3, label='Power States')
plt.plot(distance, func(slope, v_rover_constant, 1800, 0.8, 150*0.7, 0.35, 1.
↪62, 0.03, 8)[0], color='orange', linewidth=0.3, label='Constant 150W')
plt.xlabel('Mission Distance [km]')
plt.ylabel('Velocity [$\\frac{km}{h}$]')
plt.xlim(0, distance.iloc[-1])
plt.ylim(0, 0.8)
plt.title('Max Velocity with Power States')
plt.grid()
plt.legend(loc='upper left', fontsize='small')
plt.savefig("power_states_over_mission_distance.pdf")
plt.show()
```

[ ]:
```
slip = 0.0353 + 1/(1+67*np.exp(-0.32*slope))
plt.plot(distance, slip*100, linewidth=0.3, color='green')
plt.xlabel('Mission Distance [km]')
plt.ylabel('Slip [\%]')
plt.xlim(0, distance.iloc[-1])
plt.ylim(0, 100)
plt.title('Constant Velocity = {:.4f} km/h'.format(v_rover_constant))
plt.grid()
plt.savefig("slip_over_mission_distance.pdf")
plt.show()
```

[ ]:
```
plt.plot(distance, func(slope, v_rover_constant, 1800, 0.8, 150*0.7, 0.35, 1.
↪62, 0.03, 8)[6]/1000, linewidth=1, label='Ground Pressure')
plt.plot(distance, func(slope, v_rover_constant, 1800, 0.8, 150*0.7, 0.35, 1.
↪62, 0.03, 8)[7]/1000, linewidth=1, label='Safe Pressure')
plt.xlabel('Mission Distance [km]')
plt.ylabel('Pressure [kPa]')
plt.xlim(0, distance.iloc[-1])
plt.ylim(0, 30)
plt.title('Constant Velocity = {:.4f} km/h'.format(v_rover_constant))
plt.grid()
plt.legend()
plt.savefig("pressure_over_mission_distance.pdf")
plt.show()
```

[ ]:
```
slope = np.linspace(0, 25, 100) # range of slope values
```

```
# Define values for mass, slip, radius, and power
mass_values = [600, 1000, 1300, 1600, 1700, 1800]
h_grouser_values = [0.03, 0.05, 0.07, 0.1, 0.12, 0.14]
diameter_values = [0.4, 0.6, 0.8, 1]
power_values = [ 50, 100, 150, 200, 250]
width_values = [0.1, 0.2, 0.3, 0.4, 0.5]
gravity_values = [1.62, 3.71, 9.81]
gravity_name_values = ['Moon', 'Mars', 'Earth']
n_grouser_values = [8, 16, 24, 32]

# Create subplots for Varying Mass
fig1, ax1 = plt.subplots(figsize=(6, 6))
for mass_n in mass_values:
    ax1.plot(slope, func(slope, v_rover_constant, mass_n, diameter,␣
 ↪power_constant, b_width, gravity, h_grouser, n_grouser)[0], label=f"{mass_n}␣
 ↪kg")
ax1.legend()
ax1.set_title("Varying Mass")
ax1.set_xlabel('Slope $\\theta$ [$^\circ$]')
ax1.set_ylabel('Speed [$\\frac{km}{h}$]')
ax1.grid(True)
ax1.set_xlim(0, 25)
ax1.set_ylim(0, 1.5)
plt.savefig("tire_sim_varying_mass.pdf")
plt.show()

# Create subplots for Varying Grouser Height
fig2, ax2 = plt.subplots(figsize=(6, 6))
for h_grouser_n in h_grouser_values:
    ax2.plot(slope, func(slope, v_rover_constant, mass, diameter,␣
 ↪power_constant, b_width, gravity, h_grouser_n, n_grouser)[0],␣
 ↪label=f"{h_grouser_n} m")
ax2.legend()
ax2.set_title("Varying Wheel Grouser Height")
ax2.set_xlabel('Slope $\\theta$ [$^\circ$]')
ax2.set_ylabel('Speed [$\\frac{km}{h}$]')
ax2.grid(True)
ax2.set_xlim(0, 25)
ax2.set_ylim(0, 1.5)
plt.savefig("tire_sim_varying_grouser_height.pdf")
plt.show()

# Create subplots for Varying Radius
fig3, ax3 = plt.subplots(figsize=(6, 6))
for diameter_n in diameter_values:
    ax3.plot(slope, func(slope, v_rover_constant, mass, diameter_n,␣
  ↪power_constant, b_width, gravity, h_grouser, n_grouser)[0],␣
  ↪label=f"{diameter_n} m")
ax3.legend()
ax3.set_title("Varying Wheel Diameter")
ax3.set_xlabel('Slope $\\theta$ [$^\circ$]')
ax3.set_ylabel('Speed [$\\frac{km}{h}$]')
ax3.grid(True)
ax3.set_xlim(0, 25)
ax3.set_ylim(0, 1.5)
plt.savefig("tire_sim_varying_diameter.pdf")
plt.show()

# Create subplots for Varying Power
fig4, ax4 = plt.subplots(figsize=(6, 6))
for power_n in power_values:
    ax4.plot(slope, func(slope, v_rover_constant, mass, diameter, power_n,␣
  ↪b_width, gravity, h_grouser, n_grouser)[0], label=f"{power_n} W")
ax4.legend()
ax4.set_title("Varying Subsystem Electrical Power")
ax4.set_xlabel('Slope $\\theta$ [$^\circ$]')
ax4.set_ylabel('Speed [$\\frac{km}{h}$]')
ax4.grid(True)
ax4.set_xlim(0, 25)
ax4.set_ylim(0, 1.5)
plt.savefig("tire_sim_varying_power.pdf")
plt.show()

# Create subplots for Varying Width
fig5, ax5 = plt.subplots(figsize=(6, 6))
for width_n in width_values:
    ax5.plot(slope, func(slope, v_rover_constant, mass, diameter,␣
  ↪power_constant, width_n, gravity, h_grouser, n_grouser)[0],␣
  ↪label=f"{width_n} m")
ax5.legend()
ax5.set_title("Varying Wheel Width")
ax5.set_xlabel('Slope $\\theta$ [$^\circ$]')
ax5.set_ylabel('Speed [$\\frac{km}{h}$]')
ax5.grid(True)
ax5.set_xlim(0, 25)
ax5.set_ylim(0, 1.5)
plt.savefig("tire_sim_varying_width.pdf")
plt.show()

# Create subplots for Varying Planet
fig6, ax6 = plt.subplots(figsize=(6, 6))
```

```
for gravity_val, gravity_name in zip(gravity_values, gravity_name_values):
    ax6.plot(slope, func(slope, v_rover_constant, mass, diameter,␣
  ↪power_constant, b_width, gravity_val, h_grouser, n_grouser)[0],␣
  ↪label=f"{gravity_name}")
ax6.legend()
ax6.set_title("Varying Astronomical Body")
ax6.set_xlabel('Slope $\\theta$ [$^\circ$]')
ax6.set_ylabel('Speed [$\\frac{km}{h}$]')
ax6.grid(True)
ax6.set_xlim(0, 25)
ax6.set_ylim(0, 1.5)
plt.savefig("tire_sim_varying_gravity.pdf")
plt.show()

# Create subplots for Varying Number of Grousers
fig5, ax7 = plt.subplots(figsize=(6, 6))
for n_grouser_n in n_grouser_values:
    ax7.plot(slope, func(slope, v_rover_constant, mass, diameter,␣
  ↪power_constant, b_width, gravity, h_grouser, n_grouser_n)[0],␣
  ↪label=f"{n_grouser_n}")
ax7.legend()
ax7.set_title("Varying Number of Grousers")
ax7.set_xlabel('Slope $\\theta$ [$^\circ$]')
ax7.set_ylabel('Speed [$\\frac{km}{h}$]')
ax7.grid(True)
ax7.set_xlim(0, 25)
ax7.set_ylim(0, 1.5)
plt.savefig("tire_sim_varying_number_of_grousers.pdf")
plt.show()
```

```
[ ]: slope = np.linspace(0, 20, 100)
solution = func(slope, v_rover_constant, 1800, 0.8, 150*0.7, 0.35, 1.62, 0.03,␣
  ↪8)
# Calculate required values
slope_min, slope_min, slope_max = np.min(slope), np.mean(slope), np.max(slope)
drawbar_pull_min, drawbar_pull_mean, drawbar_pull_max = np.min(solution[3]), np.
  ↪mean(solution[3]), np.max(solution[3])
v_min, v_mean, v_max = np.min(solution[0]), np.mean(solution[0]), np.
  ↪max(solution[0])
torque_min, torque_mean, torque_max = np.min(solution[4]), np.
  ↪mean(solution[4]), np.max(solution[4])
P_min, P_mean, P_max = np.min(solution[1]), np.mean(solution[1]), np.
  ↪max(solution[1])
rpm_min, rpm_mean, rpm_max = np.min(solution[5]), np.mean(solution[5]), np.
  ↪max(solution[5])
```

```
# Print optimized results with \n after each value
print(f"slope_min: {slope_min:.2f}\nslope_mean: {slope_min:.2f}\nslope_max:␣
↪{slope_max:.2f}\n"
      f"drawbar_pull_min: {drawbar_pull_min:.2f}\ndrawbar_pull_mean:␣
↪{drawbar_pull_mean:.2f}\ndrawbar_pull_max: {drawbar_pull_max:.2f}\n"
      f"v_min: {v_min:.4f}\nv_mean: {v_mean:.4f}\nv_max: {v_max:.4f}\n"
      f"torque_min: {torque_min:.2f}\ntorque_mean: {torque_mean:.
↪2f}\ntorque_max: {torque_max:.2f}\n"
      f"P_min: {P_min:.2f}\nP_mean: {P_mean:.2f}\nP_max: {P_max:.2f}\n"
      f"rpm_min: {rpm_min:.4f}\nrpm_mean: {rpm_mean:.4f}\nrpm_max: {rpm_max:.
↪4f}")
```

```
[ ]: from scipy.interpolate import griddata
data = pd.read_csv('slope.csv')

distance = data['position']
longitude = -data['lon']
latitude = -data['lat']
slope = data['TerrainSlope']

# Create a grid spherical
xi_sphere = np.linspace(min(longitude), max(longitude), len(longitude))
yi_sphere = np.linspace(min(latitude), max(latitude), len(latitude))
xi_sphere, yi_sphere = np.meshgrid(xi_sphere, yi_sphere)
zi_sphere = griddata((longitude, latitude), slope, (xi_sphere, yi_sphere),␣
↪method='nearest')

solution = func(slope, v_rover_constant, 1800, 0.8, 150*0.7, 0.35, 1.62, 0.03,␣
↪8)
v_mean = np.mean(solution[0])
end_distance = distance.iloc[-1]
total_time = end_distance/v_mean
v_min = np.min(solution[0])
v_max = np.max(solution[0])

fig, (ax1) = plt.subplots(1, 1, figsize=(6, 6))
ax1.grid(True)
contour1 = ax1.contourf(xi_sphere, yi_sphere, zi_sphere, cmap='viridis')
fig.colorbar(contour1, ax=ax1, label='Terrain Slope [°]')
scatter1 = ax1.scatter(longitude, latitude, c=solution[0], linewidths=0.1,␣
↪marker='o', s=5, cmap='autumn')
fig.colorbar(scatter1, ax=ax1, label='Speed [km/h]')
ax1.axis('auto')
margin = 0.2
ax1.set_xlim(min(longitude) - margin, max(longitude) + margin)
ax1.set_ylim(min(latitude) - margin/5, max(latitude) + margin/5)
```

```
# Add markers for every k hours
k = 50  # hours
tolerance = 0.1  # Adjust this value based on your desired tolerance
marker_added = False  # Reset the marker flag at the start

for i in range(len(distance)):
    if i > 0:
        v_mean_up_to = np.cumsum(solution[0])[i] / i
        time_passed = distance[i] / v_mean_up_to
        # Check if the current time_passed is within a new k-hour interval
        if abs((time_passed % k) / k - 1) < tolerance and not marker_added:
            label_text = '{}h'.format(round(time_passed / k) * k)
            #ax1.plot(longitude[i], latitude[i], 'o', color='white',␣
 ↪markersize=6, markeredgecolor='black')
            ax1.text(longitude[i], latitude[i], label_text, ha='center',␣
 ↪va='center', color='white', fontsize=8, alpha=1)
            marker_added = True  # Set the flag to True after adding the marker␣
 ↪and label
        elif abs((time_passed % k) / k - 1) >= tolerance:
            marker_added = False  # Reset the flag if outside the tolerance

ax1.set_xlabel('Longitude [°]')
ax1.set_ylabel('Latitude [°]')
ax1.set_title('Constant Power Variable Velocity\n'
              'Mean Velocity = {:.2f} km/h\n'
              'Min Velocity = {:.2f} km/h\n'
              'Max Velocity = {:.2f} km/h\n'
              'Total time = {:.2f} h\n'
              'Total Gearbox and Motor efficiencies $\\eta$ = {:.2f}\n'
              'Constant Electrical Power = {:.2f} W\n'
              'Mass = {:.2f} kg\n'
              'Radius = {:.2f} cm\n'
              'Width = {:.2f} cm'
              .format(v_mean, v_min, v_max, total_time, eta, power_constant/
 ↪eta, mass, radius, b_width))
plt.tight_layout()
plt.savefig("tire_sim_constant_power_variable_velocity.png")
plt.show()
```