Introduction and rationale

Prepared by Artem Sotnikov, initial draft 2023-03-18

For the current design cycle, the descion was made to construct a new parachute. The reasons for this descision were as follows:

1. Legacy knowledge about how parachutes are constructed was not passed one during the COVID era, and some of it was at risk of being lost as the last people who sewed the parachute (Roman, Zach) graduated. Sewing the new parachute would serve as knowledge transfer.
2. The rocket underwent significant modifications and as a result, became much lighter during this design cycle. This meant that the parachute could be made smaller. As this eases packing, volume constraints, tangling risks, and mass, this would be a worthwhile pursuit for the recovery system.
3. The old parachute has begun to deteriorate due to the overwhelmingly large amount of testing conducted on it. If it was chosen to fly this year, it would have possibly needed significant maintenance and cleaning.

The format of this document is a jupyter notebook in the github simulink workspace due to the necessity of carrying out calculations and also since the calculations may involve the creation of simulink models (although this is admittedly unlikely).

Revision 1: 2023-03-21, Artem Sotnikov

A mistake in the initial draft was caught, which was the lack of a square root term in the nominal diameter scaling equation. After this was corrected, it became clear that the size of the parachute would not descrese significantly, despite a significant reduction in the weight of the rocket. It was decided at this point that the design will be performed to the nominal mass case. No additional safety factors or analysis will be performed on the maximum mass case, with only a check to ensure that it still conform to the DTEG requirements. Some notes on vent sizing have also been summarized from slack threads and ongoing discussion.

In [141…

```python
# Python initialization

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

import math
import matplotlib.pyplot as plt
from scipy.optimize import fsolve
from scipy import stats

%load_ext ipydex.displaytools
import ipydex # For better display

%load_ext autoreload
%autoreload 2
%matplotlib widget

from rocketpy.EnvironmentAnalysis import EnvironmentAnalysis
from datetime import datetime

FT_TO_M = 0.3048 ;
LB_TO_KG = 0.453592 ;
```

```
The ipydex.displaytools extension is already loaded. To reload it, use:
  %reload_ext ipydex.displaytools
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

Sizing approach

The parachute from the previous design cycle was relatively well characterized. It had an unreefed drag area of rouhgly $14m^2$, against a constructed diameter of 3.8 meters. This value is based on experimental data collected during Truck Test 4 and analyzed during the 2022 competition cycle using the 'truck-test-rapid-analyzer' script, code for which is available on github (https://github.com/waterloo-rocketry/truck-test-rapid-analyzer). The specific analysis file package is also uploaded to the team drive at Recovery/2023/TruckTest4AnalysisRedone. It must be noted that there is uncertainty in this unreefed area due to the necessity of data filtering via hand-picked constants. This is explored to some extent in the podium presentation and can be observed directly by changing the threshold values in the script-generated excel spreadsheet. The $14m^2$ (corresponding to the 150 ft^2 in the original units of the calculator) is actually a conservative underestimate based on the range of values given by the data analysis, with values going up to 160 -

170 for the higher windspeed cases. Therefore, this lower value can be used with some degree of reliability. This also gives a useful value of the drag coefficient with respect to the constructed-diameter-area (which should stay the same for the new parachute, assuming that their characteristics are the same):

In [142…

```
cd_const = 14/(0.25*math.pi*3.8**2) ##:
```

```
cd_const := 1.2344427719038975
---
```

Given that the parachute to be sewn will employ a very similar construction, the best way to size it would be a scaling of the old parachute. In the manual (Parachute Recovery Systems Design Manual, T.W. Knacke, 1992), almost all drag area calculations are ultimately tied to the 'nomial diameter' of the parachute, which is a function of the total fabric surface area. Therefore, the selected approach for sizing the parachute would be to calculate a new nominal diamater assuming the same relative drag coefficient.

Sizing parachutes could be done liberally to place a large factor of safety on the landing loads. In the author's opinion, such a design philosophy is appropriate for a dual-separation system, which by its nature allows large parachutes and simple parachute bays, where tangling is not a significant concern. It also lends itself well to well-chaterized legacy systems where the risk of main at apogee even is low. However, in a single-bay reefing system, a larger parachute carries a significant penalty to system development and reliability.

In a newly developed system, a 'main at apogee' is a possible scenario. A larger main would exarcerbate the effects of this scenario and could lead to system loss should the parachute drift into the white sands missile range. It also carries the disadvantage of larger drift in the last 1500 ft, although this is smaller concern.

The mass that the new parachute would need to target would be 84.2 lbs (which is the expected load as predicted in the cycle 2.2 load analysis). Additionally, a check will need to performed to ensure that the parachute still falls within the allowable envelope even at the maximum load case, which is 92.9 lbs as per the cycle 2.2 load analysis. However, for this check, no additional factors of safety or climactic considerations will be performed, since doing so will necessistate a larger parachute and lead back to the risks of large parachutes as discussed in the previous paragraph

The laod case is contrasted with the old parachute that was to be used on the projected 100 lb KotS rocket (which ultimately grew to 105.51 lbs), and was sized for a 113 lbs SotS rocket, which was ultimately 104.2 lbs (values taken from the respective reports, available on the drive or the website).

Due to the nature of the reefing system, only one parachute needs to be sized, with the reefed stage being fully controlable by the length of the reefing line, which is done after the construction of the parachute itself. Thus, the driving factor of the sizing is the ground impact velocity. As per the DTEG, it is recommended at lower than 30 ft/s. Using a 5% margin, ths speed becomes:

In [143…

```
target_descent_speed = 30*(1 - 0.05) ##:
```

```
target_descent_speed := 28.5
---
```

As noted in many texts on descent speeds, the descent speed of the parachute is given by:

$$V_{desc} = \sqrt{\frac{2m}{C_d S_o \rho}}$$

Written in a form that allows for easier corrections due to altitude effects, it is also stated in the manual as: $\therefore V_{desc} = \sqrt{\frac{2m}{C_d S_o \rho_o}} \cdot \frac{1}{\sqrt{\rho / \rho_o}}$

Where (as per the original source):

> m is the mass of the vehicle, in lbs
> $C_d S_o$ is the drag area, in ft^2
> ρ is the air density, (in slugs/ft^3 !)
> ρ_o is the air density at sea level (in slugs/ft^3)

The slug to lbs conversion is given by the acceleration due to gravity, or lbs can be used and the mass must be changed to newtons (i.e. the gravitational force on the system). This makes for easier operations with metric units:

$$V_{desc} = \sqrt{\frac{2mg}{C_d S_o \rho}}$$

This can also be trivially re-arranged to solve for CdSo:

$$C_d S_o = \frac{2mg}{\rho V_{desc}^2}$$

In [144…

```
def v_desc(m, CdSo, rho):
    return math.sqrt(2*m*(9.81)/(CdSo * rho))

def cdso(m, v, rho):
    return 2 * m * (9.81) / (rho * v**2)
```

Air density exists as a function of altitude and can be found in many tables. Since this is a code-capable document, all desired pressures can be calculated directly using the metric earth atmosphere model provided by NASA: https://www.grc.nasa.gov/www/k-12/airplane/atmosmet.html.

Which states that pressure in (in kPa) can be calculated from temperature (in °C):

$$p = 101.29 \cdot (\frac{T + 273.1}{288.08})^{5.256}$$

Where temperature varies directly with altitude as a 'lapse rate', given by $T = 15.04 - 0.00649 \cdot h$

And from this, density is $\rho = p \cdot (T + 273.1))^{-1}$

In last year's report, variations in pressure and temperature were considered to model the uncertainty in these values. The methodology used (which was somewhat crude but nontheless valid) revealed that temperature varied by 10 degrees, and pressure itself could vary by about 0.7 kPa due to other weather effects.

This gives an upper and lower bound for pressure at any given altitude (ignoring the recursive variation of temperature on pressure):

$$\rho_{upper} = (101.29\,((\frac{(15.04 - 0.00649 \cdot h) + 273.1}{288.08})^{5.256}) + 0.7\ ) \cdot ((15.04 - 0.00649 \cdot h) + 273.1 - 10\ ))^{-1}$$

$$\rho_{lower} = (101.29\,((\frac{(15.04 - 0.00649 \cdot h) + 273.1}{288.08})^{5.256}) - 0.7\ ) \cdot ((15.04 - 0.00649 \cdot h) + 273.1 + 10\ ))^{-1}$$

Another less conservative and more accurate approach is to use the historical values from new mexico directly. This website lists historical averages for June directly: https://www.timeanddate.com/weather/@5492576/climate. However, it clearly lists them adjusted for altitude, which must be converted back using a conversion factor taken from https://www.engineeringtoolbox.com/air-altitude-pressure-d_462.html

Then temperature values can be taken from the extremes directly, adding 2.5 degrees to account for additional peak temperature variation over the course of a day. For pressure, the average value can be modulated by adding or subtracting values. A value of 1 ihHg (3.39 Kpa) is given by this source as a very extreme local pressure variation, and 0.5 inHg (1.695) as the average fluctuation. https://www.infoplease.com/math-science/weather/weather-feeling-the-pressure#:~:text=Generally%2C%20at%20ground%20level%2C%20the,the%20pressure%20will%20be%20lower.

However, this would seem to conflict with the rocketpy environmental modelling of the launch site. Perfoming an analysis of this sort is out of scope for this year, but it provides some interesting numbers in the examples. An even more accurate modelling capability is provided by rocketpy's environmental analysis package, but will not be considered within scope for this analysis. https://docs.rocketpy.org/en/latest/notebooks/environment_analysis_class_usage.html

It gives the average daytime surface pressure as 25.16 inHg (85.2 kPa), with a variation of 0.08 inHg (0.27 kPa). This is much less varied than some of the analytical techniques described above. The daily minimum and maximum temperatures roughly align with the historical data, at 35 and 18 degrees celsius respectively. Due to the nature of this dataset being specifically for launch modelling, it will ultimately be used for the historical value - derived air density function.

In [145…

```
def calculate_densities_lapse(h):
    T = 15.04 - 0.00649*h

    rho_u = ((101.29*(((T + 273.1)/288.08)**(5.256))) + 0.7) * (0.2869*(T + 273.1 - 10))**(-1)
    rho_avg = ((101.29*(((T + 273.1)/288.08)**(5.256)))) * (0.2869*(T + 273.1))**(-1)
    rho_l = ((101.29*(((T + 273.1)/288.08)**(5.256))) - 0.7) * (0.2869*(T + 273.1 + 10))**(-1)
    return {"rho_u" : rho_u, "rho_avg" : rho_avg , "rho_l" : rho_l}

def calculate_densities_hist():

    # avg_p = 101.14 * (1 - 2.25577 * (10**(-5)) * (1401))**5.25588
    avg_p = 85.2
    p_var = 0.27

    rho_u = (avg_p + p_var) * (0.2869*(273.15 + 18 - 2.5))**(-1)
    rho_avg = (avg_p) * (0.2869*(273.15 + 26))**(-1)
    rho_l = (avg_p - p_var) * (0.2869*(273.15 + 35 + 2.5))**(-1)

    return {"rho_u" : rho_u, "rho_avg" : rho_avg, "rho_l" : rho_l}
```

Given the ground elevation at spaceport america, which is 1,401 m, the densities can be calculated:

In [146…
```
rho_lapse = calculate_densities_lapse(1401) ##:
rho = calculate_densities_hist() ##:
```

```
rho_lapse := {'rho_u': 1.1189855884049338,
 'rho_avg': 1.070141825546099,
 'rho_l': 1.0246776982951848}
---
rho := {'rho_u': 1.0320757976172659,
 'rho_avg': 0.9927046114799415,
 'rho_l': 0.9529260906686793}
---
```

This reveals that the historical data for the spaceport has a lower average air density than would otherwise be predicted. Due to this, the historical data will be used.

These are the desired densities, since the final descent occurs at the ground. Now executing the calculation for drag areas (in m^2):

In [147…
```
target_descent_speed ##: For reference
cdso_l = cdso(84.2*LB_TO_KG, target_descent_speed*FT_TO_M, rho['rho_u']) ##:
cdso_h = cdso(84.2*LB_TO_KG, target_descent_speed*FT_TO_M, rho['rho_l']) ##:
```

```
(target_descent_speed) := 28.5
---
cdso_l := 9.621554442669723
---
cdso_h := 10.420717380891716
---
```

For reference (as stated earlier) the old parachute had a drag area of $14m^2$, and with it the descent speed value (under the low density) would have been:

In [148…
```
v_desc(93*LB_TO_KG, 14, rho['rho_l'])/FT_TO_M # in ft/s
```

Out[148…
```
25.841342933238806
```

Sizing the parachute for the lower density case (higher drag area case) can now be done. The last parachute possessed a nominal diamater of 4.84, and this is the value being scaled in the new parachute. Therefore, the new nominal diameter must be:

In [149…
```
D_o_new = 4.84 * (cdso_h/14)**(1/2) ##:
```

```
D_o_new := 4.175708298154056
---
```

The nominal diameter is given as the diameter that would be obtained if all of the fabric was laid down in a flat circle. This is a function of the canopy's geometry:

$$S = (1/4)\pi D_o^2$$

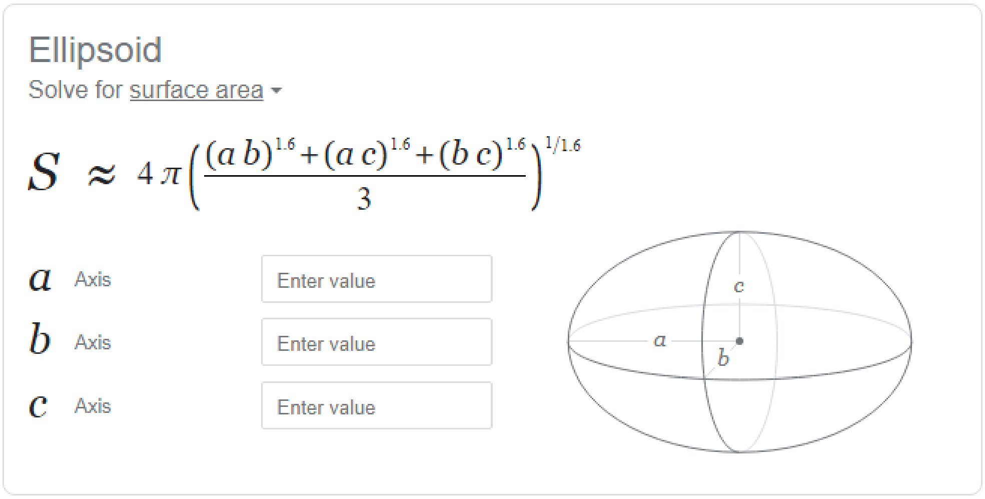
$$D_o = \sqrt{\frac{4 \cdot S}{\pi}}$$

This means that the new surface area must be:

In [150…
```
S_new = (1/4)*(math.pi)*D_o_new**2 ##:
```

```
S_new := 13.69462632807206
---
```

For the ellipsoidal shape that is used on the canopy (given by Nakka's website), the minor:major axis ratio is 0.707:1, due to claimed 'structural efficiency' that Nakka doesn't really elaborate on in his website. The major diameter can be referred ot as the 'constructed diameter' or D_c. Given the formula for the surface area of the ellipsoid:


Ellipsoid

Solve for surface area

$$S \approx 4\pi\left(\frac{(a\,b)^{1.6}+(a\,c)^{1.6}+(b\,c)^{1.6}}{3}\right)^{1/1.6}$$

In this specific case, (and with only half the ellipsoid) it would be:

$S = 2\pi((1/3)(D_c^{3.2} + 2(0.707D_c^2)^{1.6}))^{(1/1.6)}$

This is too cumbersome to re-arrange manually, so a numerical solver will be employed:

In [151…

```
def SA_ellipse(dc):
    rc = dc/2
    return 2*math.pi*((1/3)*(rc**(3.2) + 2*(0.707*rc**2)**(1.6)))**(1/1.6)
```

In [152…

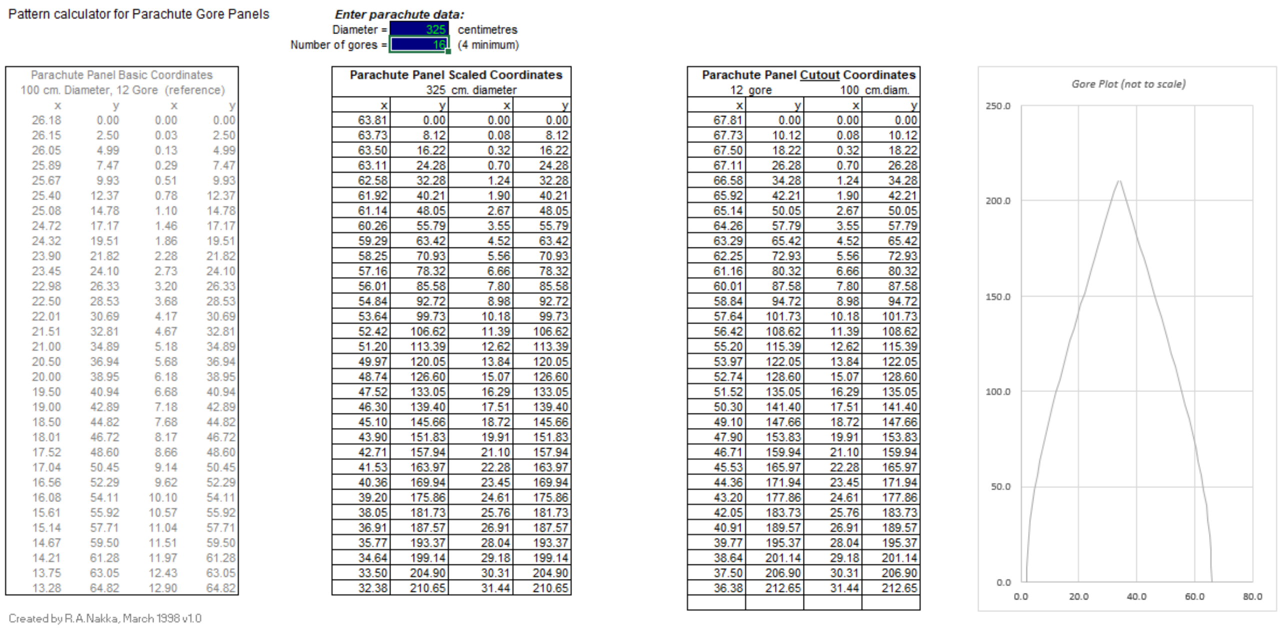
```
# fsolve works for roots of zero so I just stuck the thing in a lambda function
dc_new = fsolve(lambda dc: SA_ellipse(dc) - S_new, 4.84)[0] ##:

# Sanity check:
SA_ellipse(dc_new) ##:
```

```
dc_new := 3.2773909427578323
---
(SA_ellipse(dc_new)) := 13.694626328072113
---
```

This is the new diameter of the parachute, which is rounded down to 3.25m for simplicity (and admittedly, because that is the size of the gore template we already plotted at the time of writing). With this, Richard Nakka's parachute gore calculator script (shown in a link below) can be used to plot the specific gore pattens:

https://www.nakka-rocketry.net/softw.html#PARA

Aside from the diameter, the only other parameter required by the calculator is the number of gores. For simplicity and increased structural strength, this number will be retained at 16, same as the previous parachute (even though perhaps it could be reduced; but doing an analysis on it at this point is out of scope for this year's sizing due to how delayed it has been). Putting in the parameters into the calculator yields:



Pattern calculator for Parachute Gore Panels

Enter parachute data:
Diameter = 325 centimetres
Number of gores = 16 (4 minimum)

Parachute Panel Basic Coordinates
100 cm. Diameter, 12 Gore (reference)

x	y	x	y
26.18	0.00	0.00	0.00
26.15	2.50	0.03	2.50
26.05	4.99	0.13	4.99
25.89	7.47	0.29	7.47
25.67	9.93	0.51	9.93
25.40	12.37	0.78	12.37
25.08	14.78	1.10	14.78
24.72	17.17	1.46	17.17
24.32	19.51	1.86	19.51
23.90	21.82	2.28	21.82
23.45	24.10	2.73	24.10
22.98	26.33	3.20	26.33
22.50	28.53	3.68	28.53
22.01	30.69	4.17	30.69
21.51	32.81	4.67	32.81
21.00	34.89	5.18	34.89
20.50	36.94	5.68	36.94
20.00	38.95	6.18	38.95
19.50	40.94	6.68	40.94
19.00	42.89	7.18	42.89
18.50	44.82	7.68	44.82
18.01	46.72	8.17	46.72
17.52	48.60	8.66	48.60
17.04	50.45	9.14	50.45
16.56	52.29	9.62	52.29
16.08	54.11	10.10	54.11
15.61	55.92	10.57	55.92
15.14	57.71	11.04	57.71
14.67	59.50	11.51	59.50
14.21	61.28	11.97	61.28
13.75	63.05	12.43	63.05
13.28	64.82	12.90	64.82

Created by R.A.Nakka, March 1998 v1.0

Parachute Panel Scaled Coordinates
325 cm. diameter

x	y	x	y
63.81	0.00	0.00	0.00
63.73	8.12	0.08	8.12
63.50	16.22	0.32	16.22
63.11	24.28	0.70	24.28
62.58	32.28	1.24	32.28
61.92	40.21	1.90	40.21
61.14	48.05	2.67	48.05
60.26	55.79	3.55	55.79
59.29	63.42	4.52	63.42
58.25	70.93	5.56	70.93
57.16	78.32	6.66	78.32
56.01	85.58	7.80	85.58
54.84	92.72	8.98	92.72
53.64	99.73	10.18	99.73
52.42	106.62	11.39	106.62
51.20	113.39	12.62	113.39
49.97	120.05	13.84	120.05
48.74	126.60	15.07	126.60
47.52	133.05	16.29	133.05
46.30	139.40	17.51	139.40
45.10	145.66	18.72	145.66
43.90	151.83	19.91	151.83
42.71	157.94	21.10	157.94
41.53	163.97	22.28	163.97
40.36	169.94	23.45	169.94
39.20	175.86	24.61	175.86
38.05	181.73	25.76	181.73
36.91	187.57	26.91	187.57
35.77	193.37	28.04	193.37
34.64	199.14	29.18	199.14
33.50	204.90	30.31	204.90
32.38	210.65	31.44	210.65

Parachute Panel Cutout Coordinates
12 gore 100 cm.diam.

x	y	x	y
67.81	0.00	0.00	0.00
67.73	10.12	0.08	10.12
67.50	18.22	0.32	18.22
67.11	26.28	0.70	26.28
66.58	34.28	1.24	34.28
65.92	42.21	1.90	42.21
65.14	50.05	2.67	50.05
64.26	57.79	3.55	57.79
63.29	65.42	4.52	65.42
62.25	72.93	5.56	72.93
61.16	80.32	6.66	80.32
60.01	87.58	7.80	87.58
58.84	94.72	8.98	94.72
57.64	101.73	10.18	101.73
56.42	108.62	11.39	108.62
55.20	115.39	12.62	115.39
53.97	122.05	13.84	122.05
52.74	128.60	15.07	128.60
51.52	135.05	16.29	135.05
50.30	141.40	17.51	141.40
49.10	147.66	18.72	147.66
47.90	153.83	19.91	153.83
46.71	159.94	21.10	159.94
45.53	165.97	22.28	165.97
44.36	171.94	23.45	171.94
43.20	177.86	24.61	177.86
42.05	183.73	25.76	183.73
40.91	189.57	26.91	189.57
39.77	195.37	28.04	195.37
38.64	201.14	29.18	201.14
37.50	206.90	30.31	206.90
36.38	212.65	31.44	212.65

Which is the current proposed gore construction for the 2023 parachute. Furthermore, the top of each gore needs to be trimmed in order to produce the vent. Since the approach was to keep the geometry as similar as possible to the previous parachute. Measuring the gores on it, a gore height of about 195 cm was measured. For a 3.8 meter parachute, the full gore template possessed a height of 246 cm. This meant that 51 cm was taken, translating to roughly 20% of the gore. Online reasources such as the following mention the vent size as a function of diamater, so the approach to size it by diameter is followed.
https://fruitychutes.com/help_for_parachutes/parachute-help/how_to_make_a_parachute.htm#:~:text=Our%20standard%20design%20comes%20with,tape%20to%20ensure%20maximum%20strength
The same ratio is retained for this parachute, yielding the following trim length and new gore height:

In [153…

```
trim_len = 2.10 * 0.20 ##:
new_height = 2.10 * (1 - 0.20) ##:
```

```
trim_len := 0.42000000000000004
---
new_height := 1.6800000000000002
---
```

That locks in all parameters for the 2023 parachute.

As a sanity check, the descent speed with this new constructed diameter and old coefficient of drag is:

In [154…

```
v_desc(84.2*LB_TO_KG, (0.25*math.pi*dc_new**2)*1.234,  rho['rho_l'])/FT_TO_M ##: feet per second
```

```
(v_desc(84.2*LB_TO_KG, (0.25*math.pi*dc_new**2)*1.234,  rho['rho_l'])/FT_TO_M) := 28.514312118812466
---
```

With the maximum mass case, the descent speed is (at both the minumum and average climactic conditions):

In [155…

```
v_desc(92.9*LB_TO_KG, (0.25*math.pi*dc_new**2)*1.234,  rho['rho_avg'])/FT_TO_M ##: feet per second
v_desc(92.9*LB_TO_KG, (0.25*math.pi*dc_new**2)*1.234,  rho['rho_l'])/FT_TO_M ##: feet per second
```

```
(v_desc(92.9*LB_TO_KG, (0.25*math.pi*dc_new**2)*1.234,  rho['rho_avg'])/FT_TO_M) := 29.345012552297966
---
(v_desc(92.9*LB_TO_KG, (0.25*math.pi*dc_new**2)*1.234,  rho['rho_l'])/FT_TO_M) := 29.95123332315653
---
```

Which is still within bounds of the allowable descent rate