Missile_1feat_fuel_isp

January 16, 2023

[1]:
```
write_images = False


wirte_output_txt = False
# Specify everytime Simulation is called
# WARNING --> Set to False when running more then 10 simulations
#             (otherwise it will be super slow and might crash)
```

[2]:
```
import numpy as np
```

[3]:
```
from emukit.core import ContinuousParameter, ParameterSpace
from emukit.core.initial_designs import RandomDesign

import GPy
from GPy.models import GPRegression
from emukit.model_wrappers import GPyModelWrapper
from emukit.sensitivity.monte_carlo import MonteCarloSensitivity

import matplotlib.pyplot as plt
import mlai.plot as plot
```

[4]:
```
%run Missile_utils.ipynb
```

[ ]:
```

```

[5]:
```
simulation_output = 'range'
# We divide by 1000 to avoid dealing with too large numbers
```

[ ]:
```

```

[ ]:
```

```

We consider missiles with only 1 stage

[6]:
```
basic_param_spaces = {
    'payload':  [10, 2410],
    'missilediam':  [0.1, 9.9],
    'rvdiam':  [0.1, 9.9],
    'estrange': [100, 4900],
    'fuelmass': [500, 6000], # [500, 7000],
    'drymass':  [1000, 3000],
    'Isp0':  [100, 800],# [100, 800],
    'thrust0':  [10000, 69000],
```

```
}
```

[7]:
```
from sklearn.metrics import mean_squared_error
import math

def compute_rmse(y_actual, y_predicted):
    MSE = mean_squared_error(y_actual, y_predicted)
    RMSE = math.sqrt(MSE)

    return RMSE

def evaluate_prediction(y_actual, y_predicted):
    return compute_rmse(y_actual, y_predicted)
```

[ ]:
```

```

1 0. Only one param - m0

[8]:
```
m0_param_1 = 'fuelmass'
m0_domain_param_1 = basic_param_spaces[m0_param_1]

m0_space = ParameterSpace(
          [ContinuousParameter(m0_param_1, *m0_domain_param_1),
          ])

custom_param_names = [m0_param_1]
```

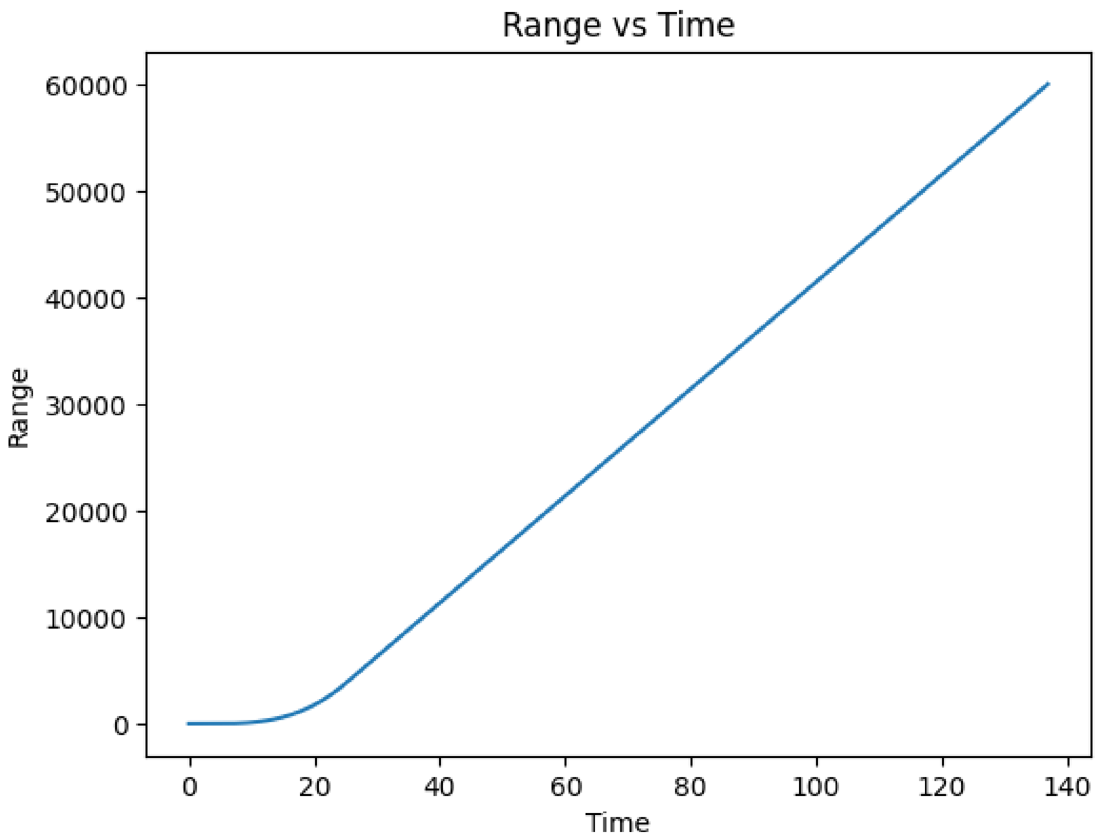
[9]:
```
def run_missile_sim(custom_params):
    """
    Recives in input an array of custom parameters.
    Each row represents a set of different parameters
    Each column is a different parameter (#cols = len(custom_param_names))
    """
    default_params_IRAQ = {
        'payload':500,
        'missilediam':0.88,
        'rvdiam':0,
        'estrange':600,
        'numstages':1,
        'fuelmass':[0,5600],
        'drymass':[0,1200],
        'Isp0':[0,226],
        'thrust0':[0,9177.4]
    }


    y = np.zeros((custom_params.shape[0], 1))
    for i in range(custom_params.shape[0]):
        params_to_use = default_params_IRAQ
        # Overwrite default param variables
```

```
        for j in range(custom_params.shape[1]):
            param_name = custom_param_names[j]
            if param_name in ['fuelmass', 'drymass', 'Isp0', 'thrust0']:
                params_to_use[param_name][1] = custom_params[i,j]
            else:
                params_to_use[param_name] = custom_params[i, j]

            if j==0:
                print('\nNew simulation \n')
            str_to_print = param_name + ': ' + str(custom_params[i,j])
            print(str_to_print)


        # Run simulation
        output_path = 'results/results_' + str(i) + '.txt'
        sim_output = run_one_sim(
            numstages=params_to_use["numstages"],
            fuelmass=params_to_use["fuelmass"],
            drymass=params_to_use["drymass"],
            thrust0=params_to_use["thrust0"],
            Isp0=params_to_use["Isp0"],
            payload=params_to_use["payload"],
            missilediam=params_to_use["missilediam"],
            rvdiam=params_to_use["rvdiam"],
            est_range=params_to_use["estrange"],
            output_path=output_path,
            simulation_output=simulation_output,
        )

        y[i, 0] = sim_output
    return y
```

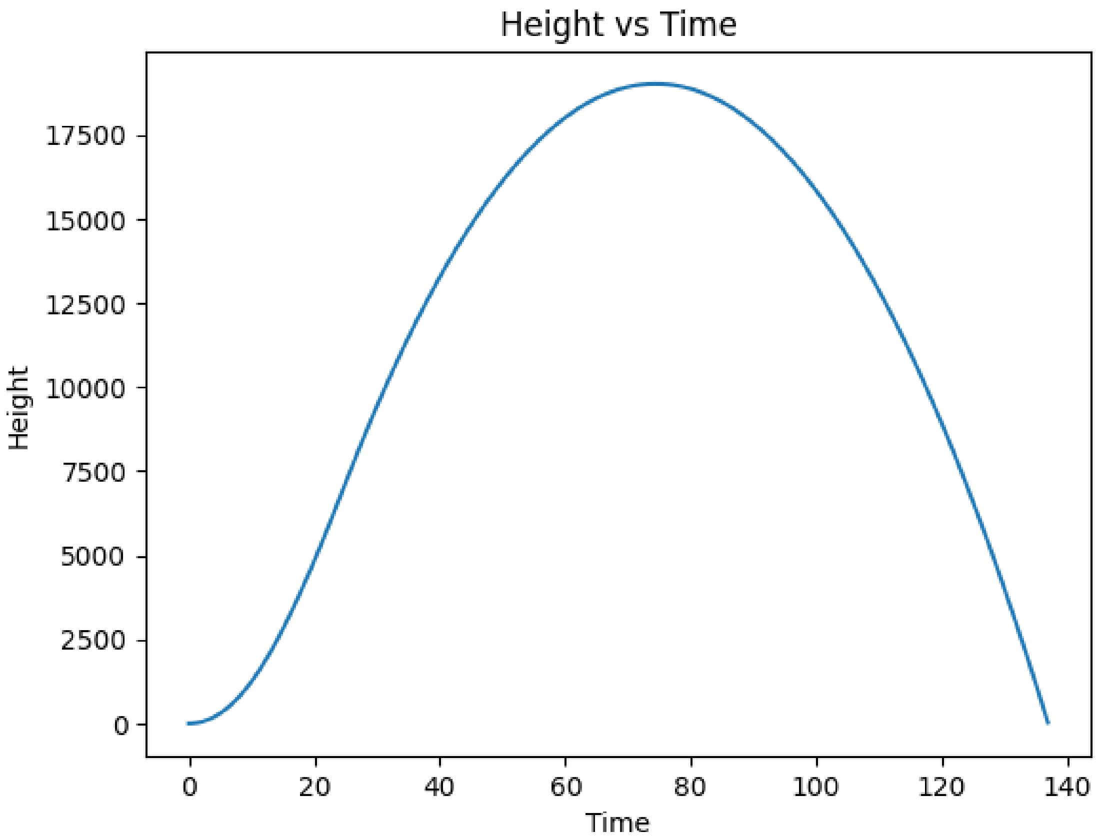
[10]:

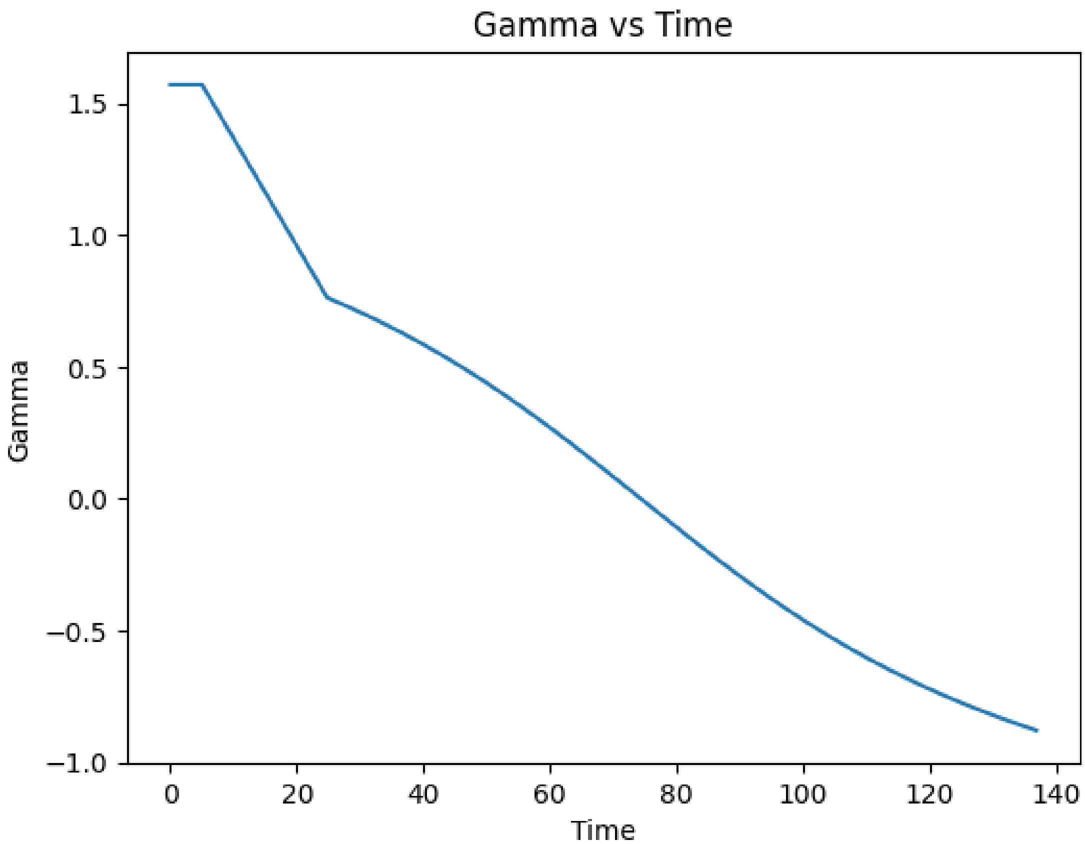
```
# Get true points (to build model)
wirte_output_txt = True

m0_design = RandomDesign(m0_space)
m0_x = m0_design.get_samples(3)
m0_y = run_missile_sim(m0_x)
```

```
New simulation

fuelmass: 1007.8840088887675


Stage 1 burnout
Velocity (km/s):  0.6978016112431643
Angle (deg h):  43.69724960661012
Range (km):  3.6927765908978962
Time (sec):  24.800000000000086
Final results:
Range (km):  60.11879127124692
Apogee (km):  19.01539535261587
Time to target (sec):  136.9999999999965
```

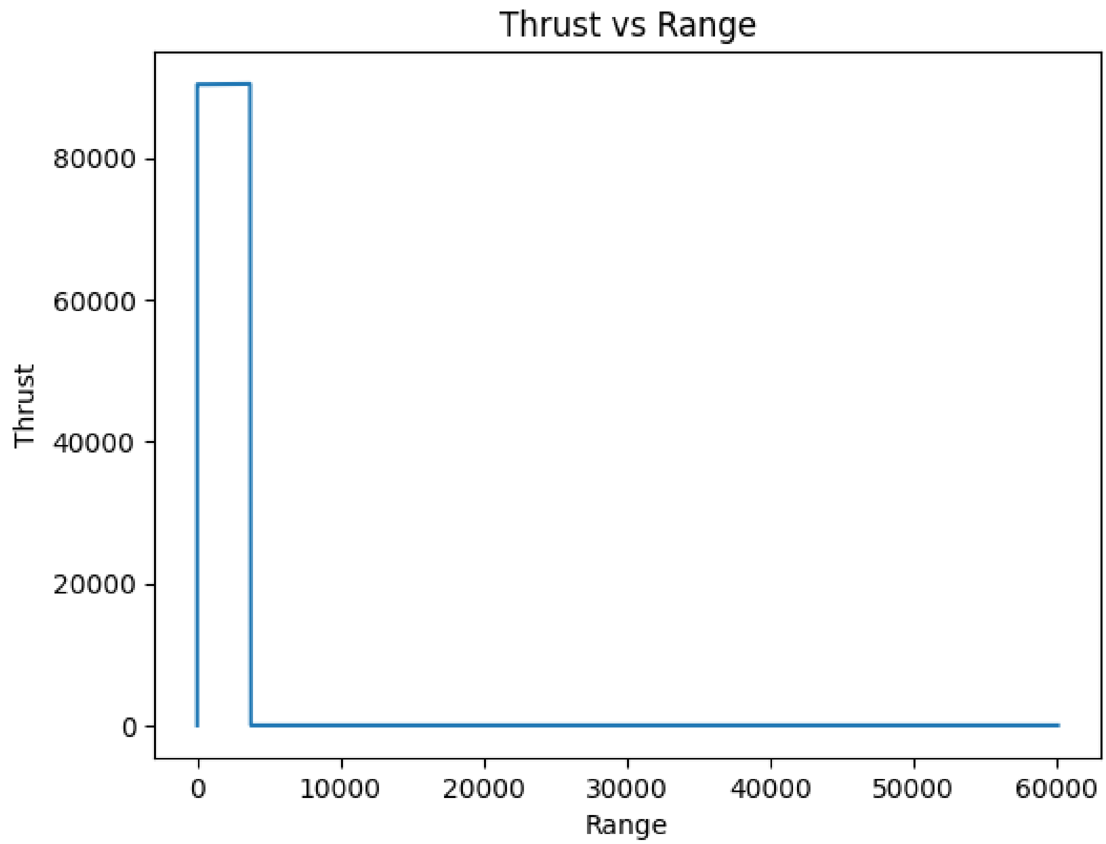


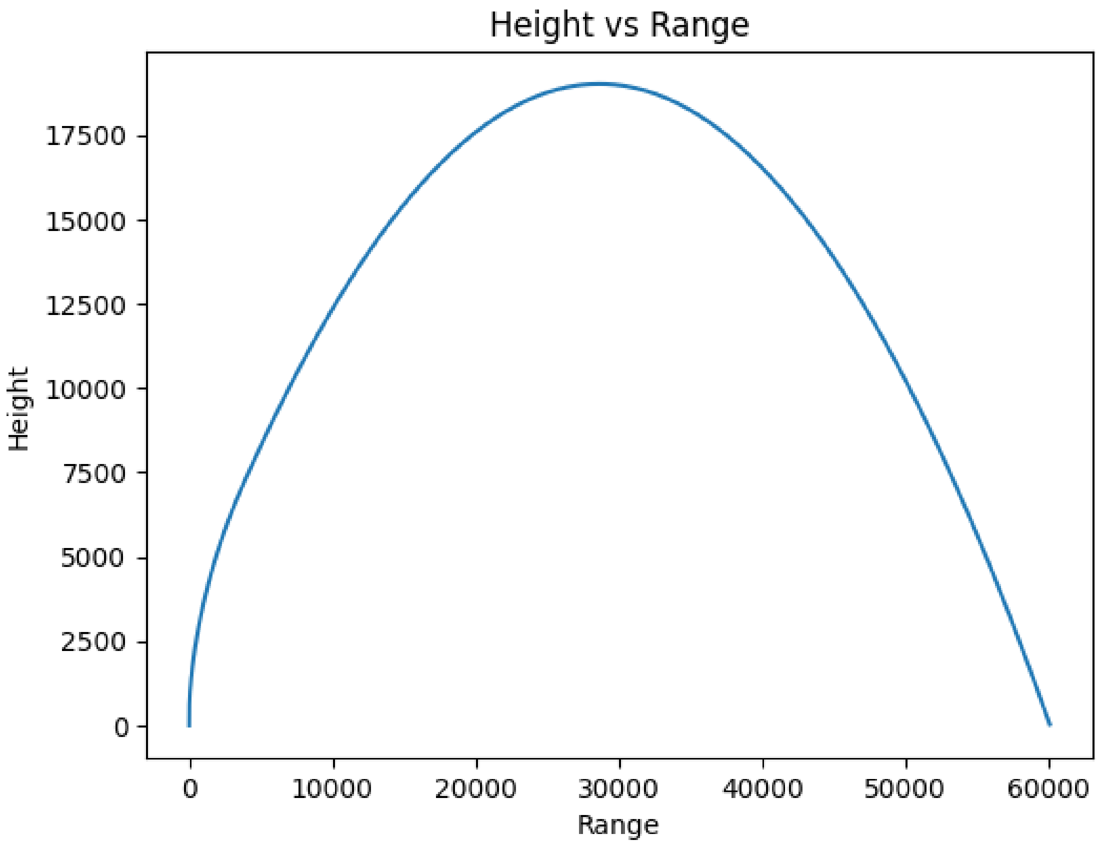
Range vs Time
60000
50000
40000
30000
20000
10000
0
Range
0
20
40
60
80
100
120
140
Time


Height vs Time
17500
15000
12500
10000
7500
5000
2500
0
Height
0
20
40
60
80
100
120
140
Time

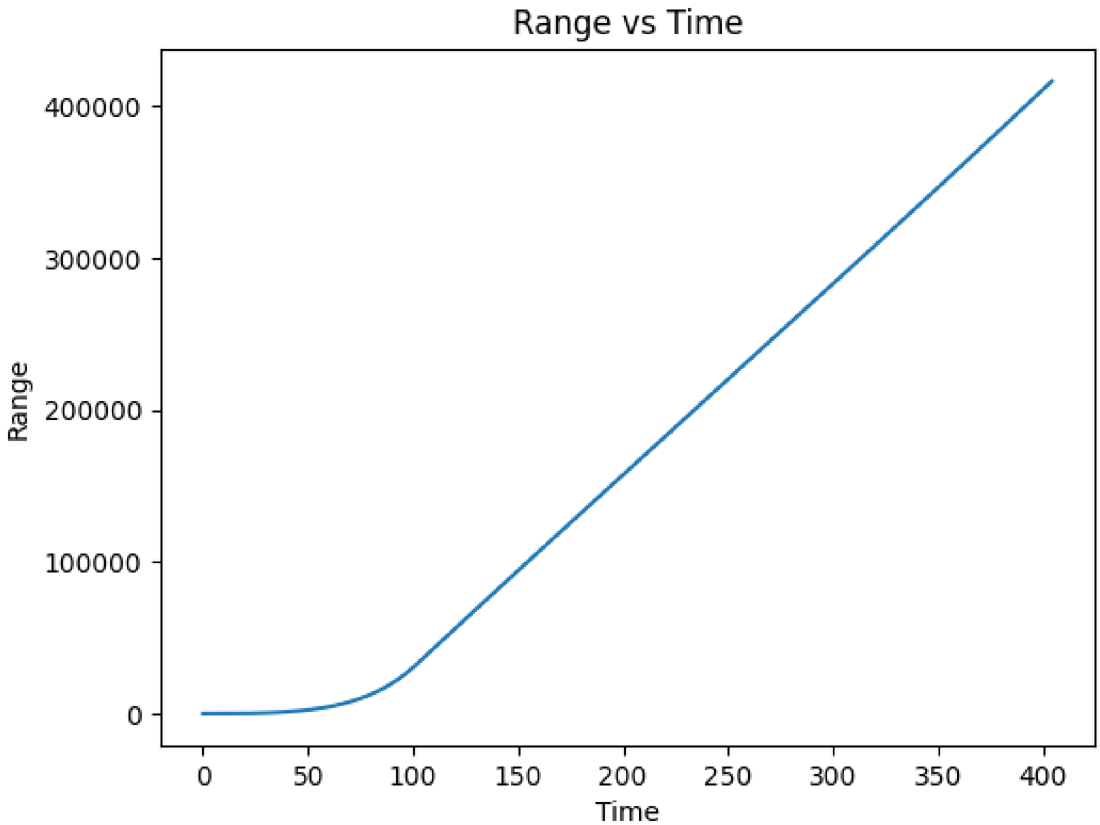


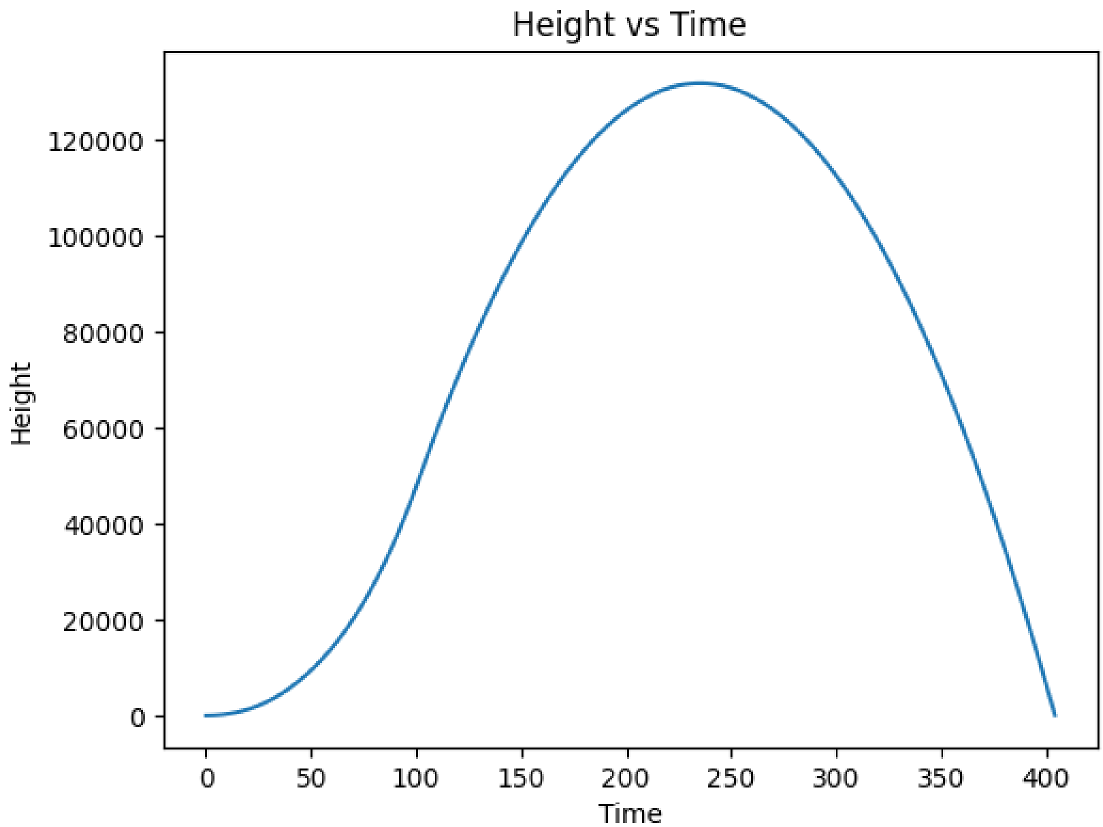
Gamma vs Time
Gamma
Time
1.5
1.0
0.5
0.0
−0.5
−1.0
0
20
40
60
80
100
120
140

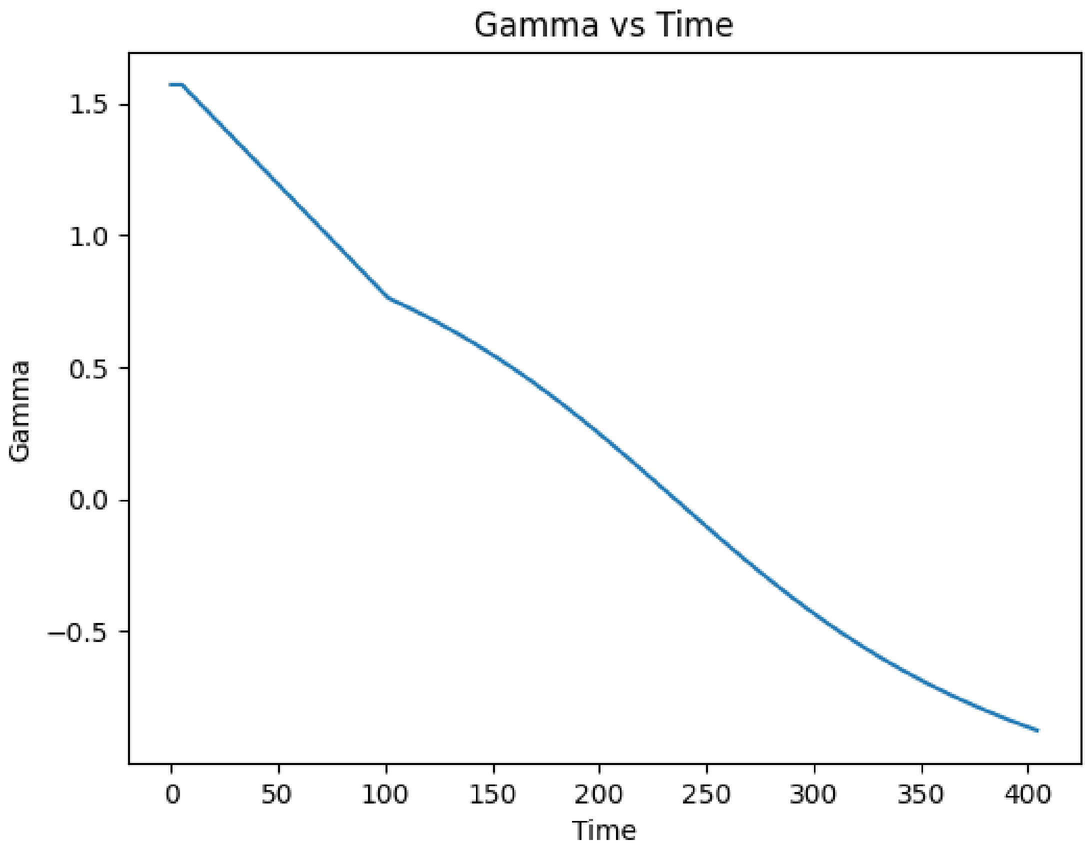

Thrust vs Range
80000
60000
40000
20000
0
Thrust
0
10000
20000
30000
40000
50000
60000
Range

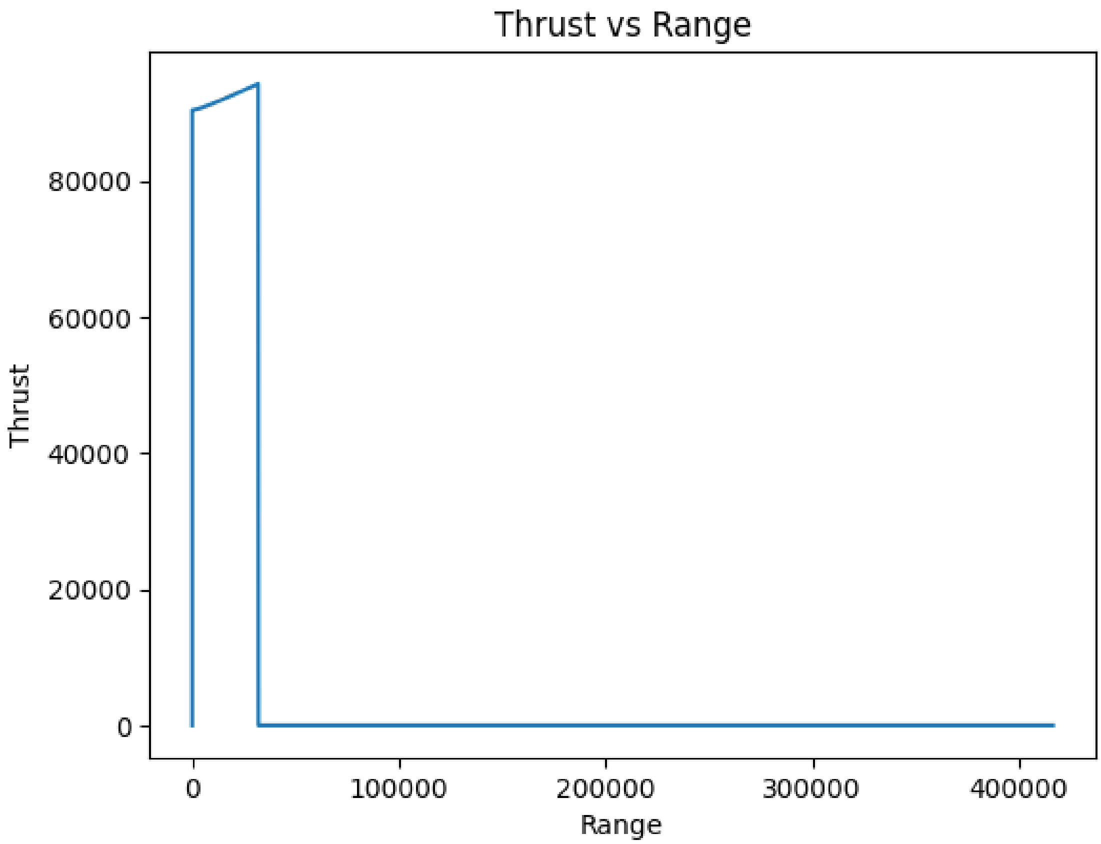


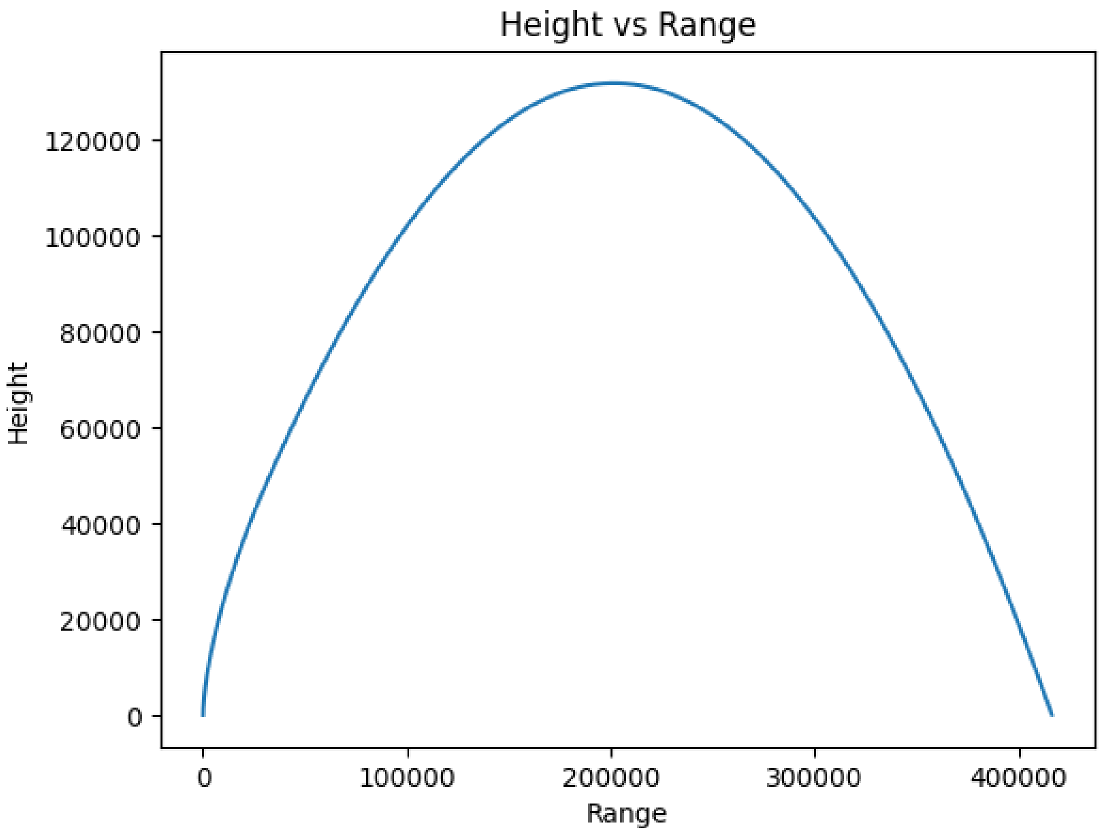

```
Data written to 'results/results_0.txt'

New simulation

fuelmass: 4113.2114793652045


Stage 1 burnout
Velocity (km/s):  1.7930023621327982
Angle (deg h):  43.673294728018924
Range (km):  31.900601741935937
Time (sec):  101.29999999999852
Final results:
Range (km):  416.4717608333348
Apogee (km):  131.7242674622094
Time to target (sec):  404.40000000002345
```


Range vs Time
400000
300000
200000
100000
0
Range
0
50
100
150
200
250
300
350
400
Time

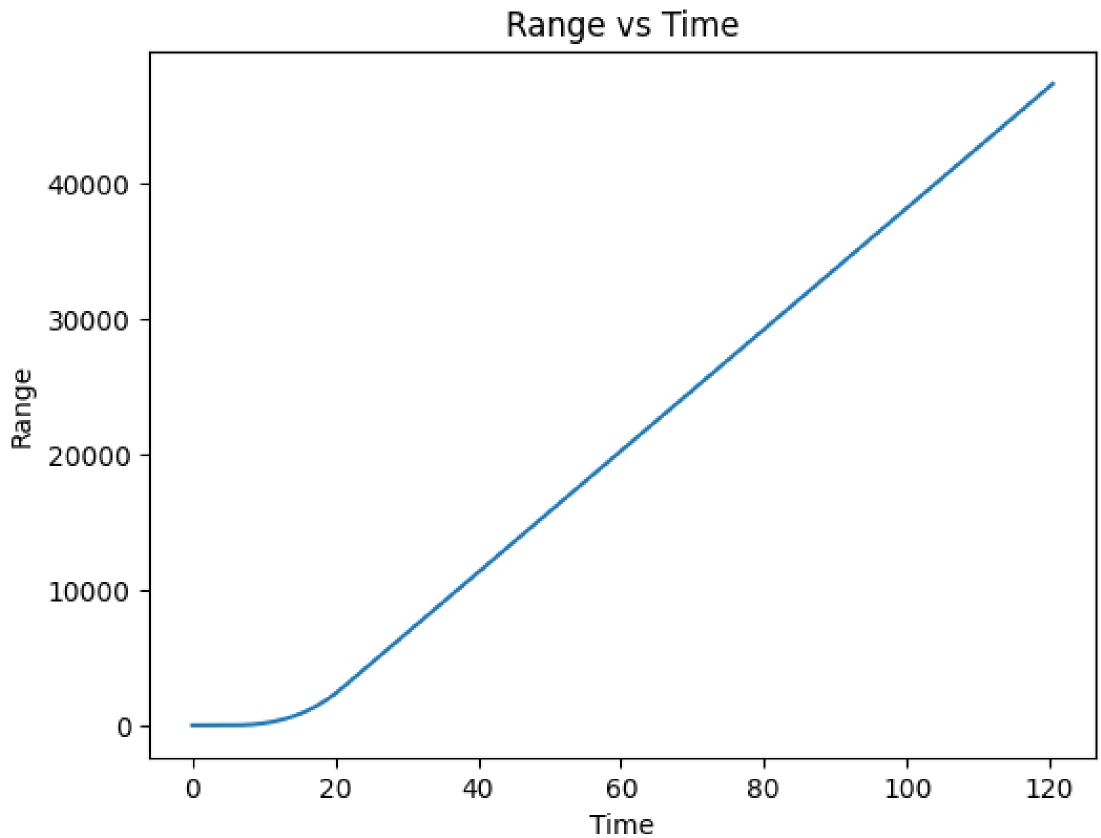

Height vs Time
Height
120000
100000
80000
60000
40000
20000
0
0
50
100
150
200
250
300
350
400
Time

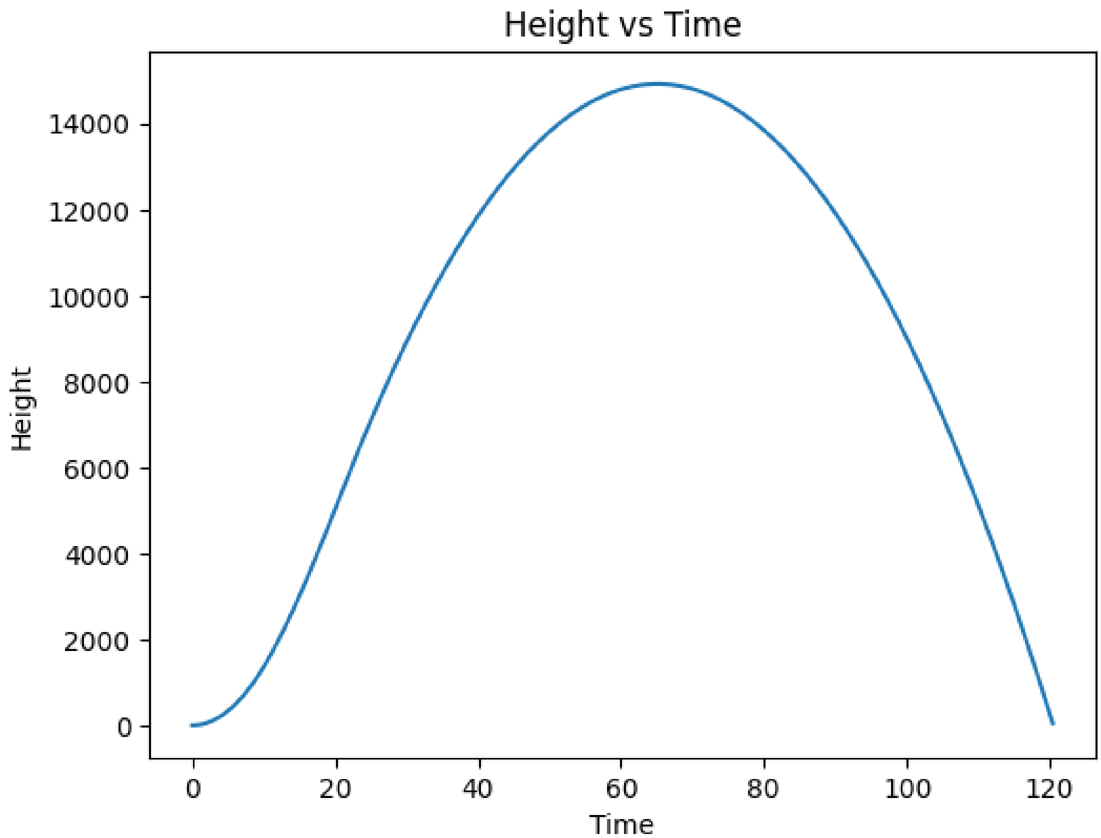

Gamma vs Time
1.5
1.0
0.5
0.0
−0.5
Gamma
0
50
100
150
200
250
300
350
400
Time

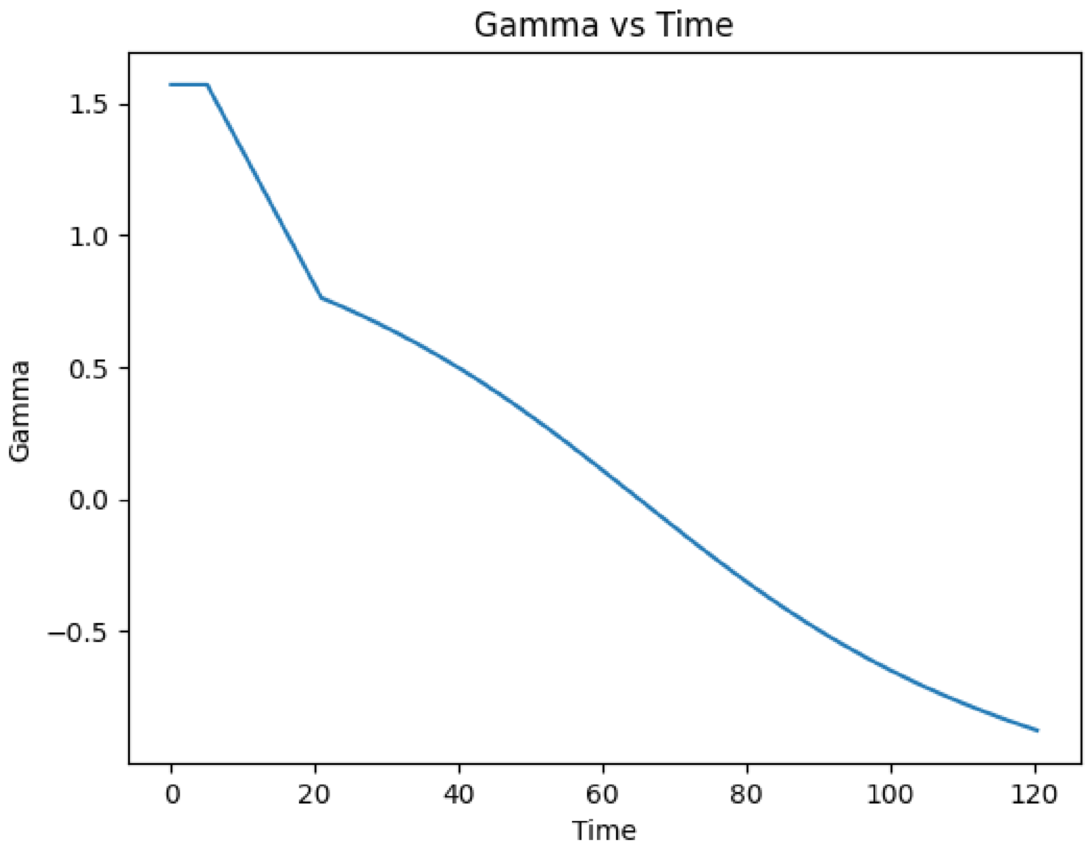

Thrust vs Range
Thrust
80000
60000
40000
20000
0
0
100000
200000
300000
400000
Range

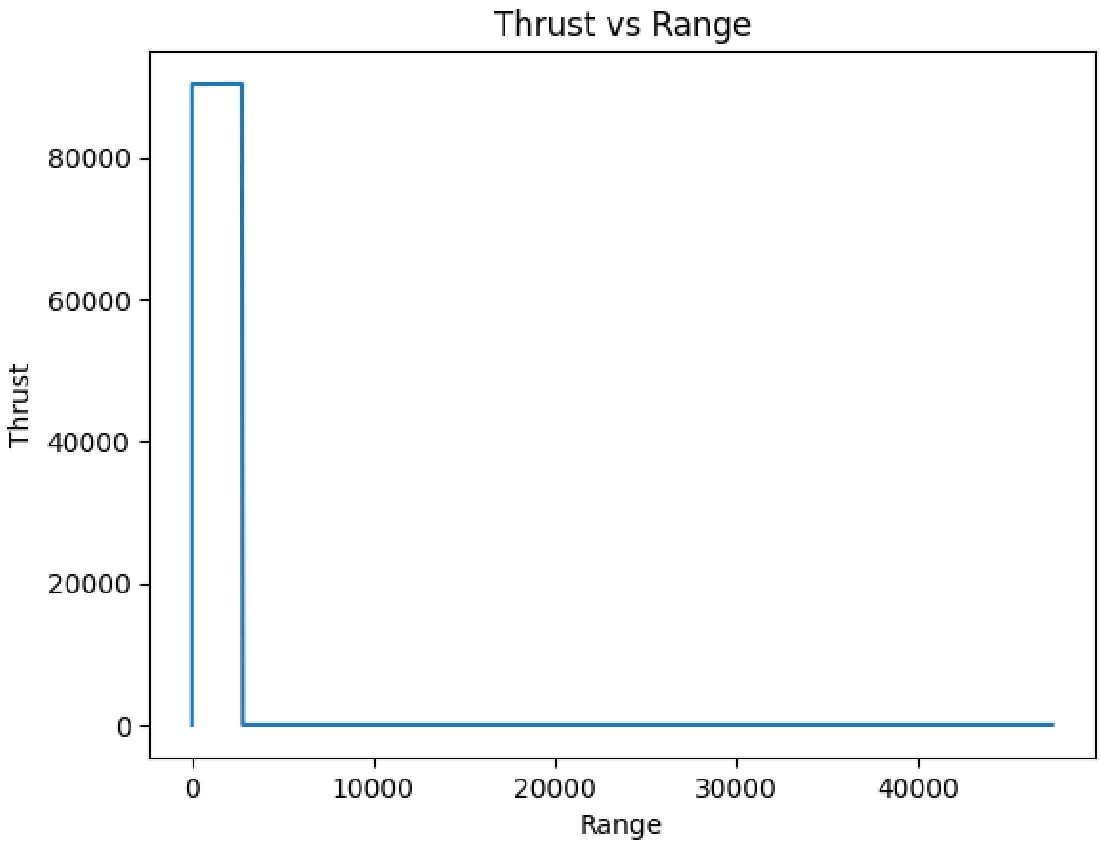


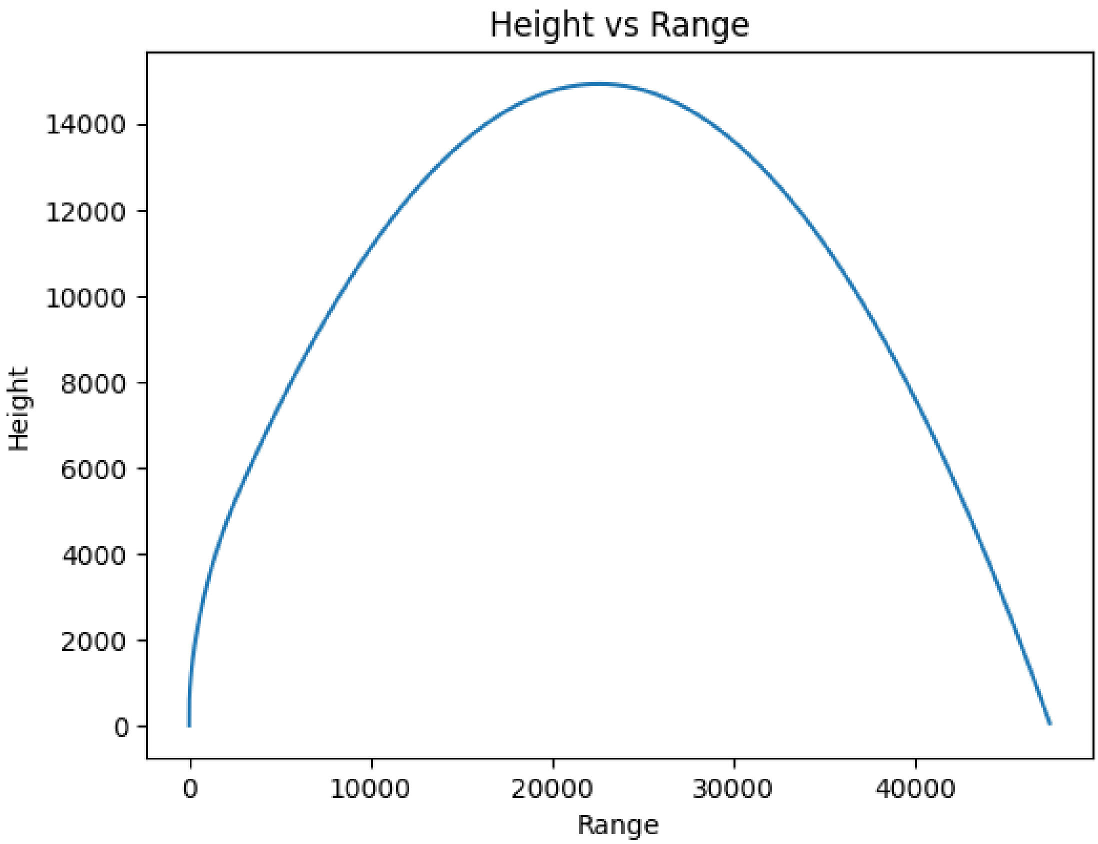

```
Data written to 'results/results_1.txt'

New simulation

fuelmass: 849.9309768862413


Stage 1 burnout
Velocity (km/s):  0.6221936524437466
Angle (deg h):  43.67346391307597
Range (km):  2.758683030930767
Time (sec):  21.000000000000032
Final results:
Range (km):  47.42350449217844
Apogee (km):  14.926926793285853
Time to target (sec):  120.49999999999743
```


Range vs Time
Range
Time
40000
30000
20000
10000
0
0
20
40
60
80
100
120


Height vs Time
14000
12000
10000
8000
6000
4000
2000
0
Height
0
20
40
60
80
100
120
Time


Gamma vs Time
1.5
1.0
0.5
0.0
−0.5
Gamma
0
20
40
60
80
100
120
Time


Thrust vs Range
80000
60000
40000
20000
0
Thrust
0
10000
20000
30000
40000
Range




```
Data written to 'results/results_2.txt'
```

[11]:
```
# Build model
m0_var_kernel = (100)**2
m0_lengthscale = 100 # 1
m0_var_linear_kernel = (100)**2
m0_var_noise = 1e-5 # small value

#kern = GPy.kern.RBF(input_dim=1, lengthscale=100, variance =var_kernel )  # ,␣
 ↪lengthscale=0.08, variance=20
# kern = GPy.kern.Matern32(input_dim=1)
# kern = GPy.kern.Linear(input_dim=1)


constrain_lengthscale = False

m0_rbf_kern = GPy.kern.RBF(input_dim=1, lengthscale=m0_lengthscale)
if constrain_lengthscale:
    m0_rbf_kern.lengthscale.constrain_bounded(m0_lengthscale, m0_lengthscale*1e12)

m0_kern = m0_rbf_kern + \
    GPy.kern.Linear(input_dim=1)
```

```
m0_model_gpy = GPRegression(m0_x,m0_y, kernel=m0_kern)
m0_model_gpy.kern.variance =  m0_var_kernel
m0_model_gpy.likelihood.variance.fix(m0_var_noise)

display(m0_model_gpy)
```

<GPy.models.gp_regression.GPRegression at 0x7ff43be81db0>

[12]:
```
# Fit emulator
m0_model_emukit = GPyModelWrapper(m0_model_gpy)
m0_model_emukit.optimize() # Optimize model hyperparameters
```

[13]:
```
display(m0_model_gpy)
```

<GPy.models.gp_regression.GPRegression at 0x7ff43be81db0>

[14]:
```
# Get true points corresponding to param_1_x_plot (for plot)
wirte_output_txt = False

nr_points_plot = 301
m0_param_1_x_plot = np.linspace(m0_space.parameters[0].min, m0_space.parameters[0].
 ↪max, nr_points_plot)[:, None]
m0_param_1_y_plot = run_missile_sim(m0_param_1_x_plot)
```

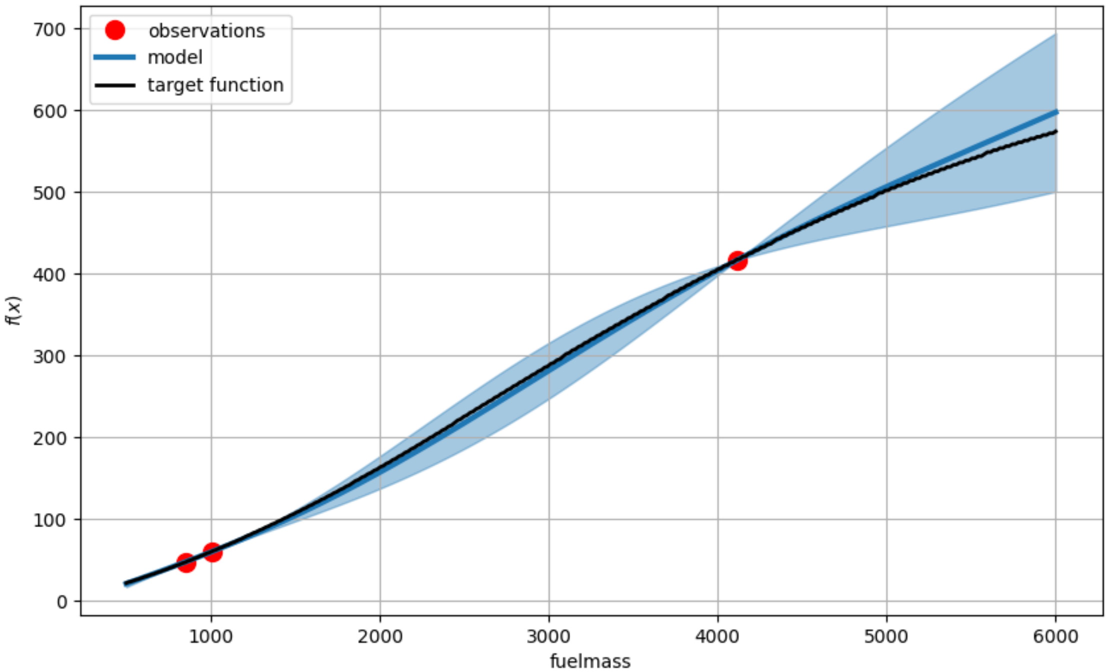
[15]:
```
# Get model prediction on param_1_x_plot
m0_mu_plot, m0_var_plot = m0_model_emukit.predict(m0_param_1_x_plot)
```

[ ]:
```

```

[16]:
```
# Plot
def helper_plot_emulator_errorbars(x_plot, y_plot, mu_plot, var_plot, model_emukit):
    """Helper function for plotting the emulator fit."""
    ax.plot(model_emukit.X[:, 0], model_emukit.Y, 'ro', markersize=10,␣
 ↪label='observations')
    ax.plot(x_plot[:, 0], mu_plot, 'C0', label='model', linewidth=3)
    ax.plot(x_plot[:, 0], y_plot, 'k', label='target function', linewidth=2)
#     ax.fill_between(x_plot[:, index],
#                     mu_plot[:, 0] + np.sqrt(var_plot)[:, 0],
#                     mu_plot[:, 0] - np.sqrt(var_plot)[:, 0], color='C0', alpha=0.6)
    ax.fill_between(x_plot[:, 0],
                    mu_plot[:, 0] + 2 * np.sqrt(var_plot)[:, 0],
                    mu_plot[:, 0] - 2 * np.sqrt(var_plot)[:, 0], color='C0', alpha=0.4)
#     ax.fill_between(x_plot[:, index],
#                     mu_plot[:, 0] + 3 * np.sqrt(var_plot)[:, 0],
#                     mu_plot[:, 0] - 3 * np.sqrt(var_plot)[:, 0], color='C0', alpha=0.2)
    ax.legend(loc=2)
    ax.set_xlabel(custom_param_names[0])
    ax.set_ylabel('$f(x)$')
    ax.grid(True)
    #ax.set_xlim(-0.01, 1)
    #ax.set_ylim([-20, 20])
```

[17]:
```
fig, ax = plt.subplots(figsize=plot.big_wide_figsize)
helper_plot_emulator_errorbars(x_plot=m0_param_1_x_plot, y_plot=m0_param_1_y_plot,
                               mu_plot=m0_mu_plot, var_plot=m0_var_plot,
                               model_emukit=m0_model_emukit)

m0_rmse = evaluate_prediction(y_actual=m0_param_1_y_plot, y_predicted=m0_mu_plot)
print("RMSE m0 (pre experiment design loop): ", m0_rmse)
```

```
RMSE m0 (pre experiment design loop):  7.175915276215401
```



1.0.1 Experiment design loop

[18]:
```
from emukit.experimental_design.experimental_design_loop import ExperimentalDesignLoop
from emukit.experimental_design.acquisitions import IntegratedVarianceReduction,␣
↪ModelVariance
```

[19]:
```
m0_2_model_emukit = m0_model_emukit
```

[20]:
```
wirte_output_txt = False

integrated_variance = IntegratedVarianceReduction(space=m0_space,
                                                  model=m0_2_model_emukit)
m0_ed = ExperimentalDesignLoop(space=m0_space,
                               model=m0_2_model_emukit,
                               acquisition = integrated_variance,
                               batch_size = 1)
# bach size is set to one in this example as we'll collect evaluations
# sequentially but parallel evaluations are allowed
```

```python
m0_ed.run_loop(user_function=run_missile_sim, stopping_condition=5)
```

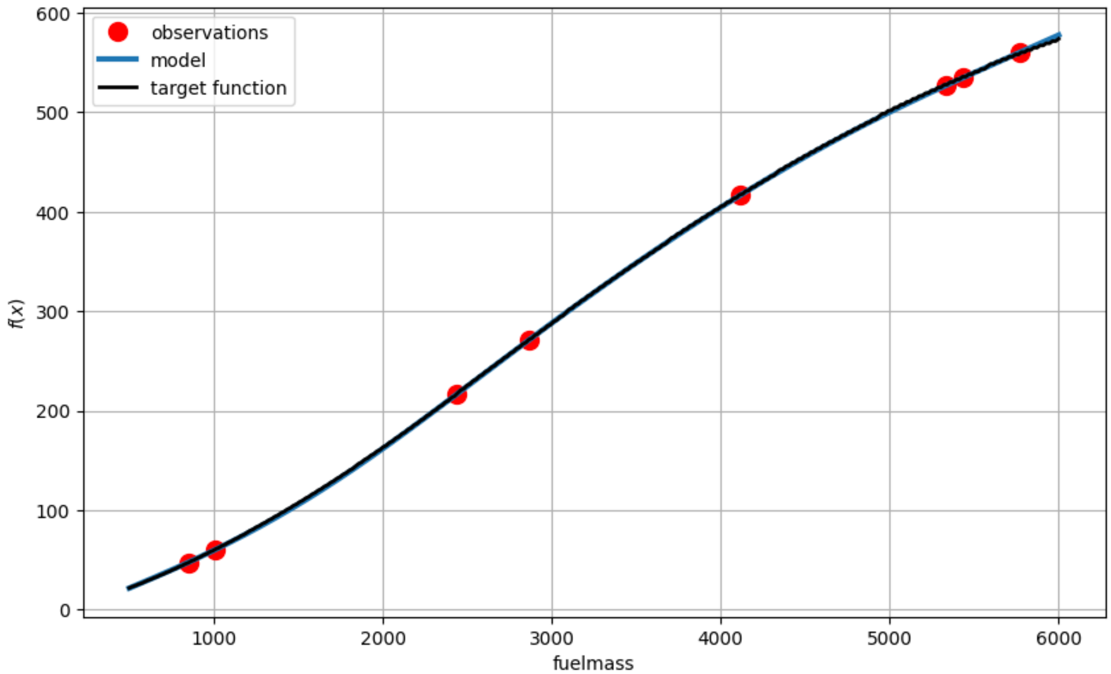
```
 /Users/ilariasartori/opt/anaconda3/envs/mlphysical/lib/python3.10/site-
packages/paramz/transformations.py:111: RuntimeWarning:overflow encountered in
expm1

New simulation

fuelmass: 5336.286633499005


Stage 1 burnout
Velocity (km/s):  2.0031498796355165
Angle (deg h):  43.65441827225012
Range (km):  42.40099812276978
Time (sec):  131.39999999999682
Final results:
Range (km):  527.3657480797298
Apogee (km):  167.1947200518729
Time to target (sec):  475.4000000000396



New simulation

fuelmass: 2870.056178551053


Stage 1 burnout
Velocity (km/s):  1.4603312433832933
Angle (deg h):  43.67861432087781
Range (km):  19.80304752155
Time (sec):  70.70000000000026
Final results:
Range (km):  271.76080359339636
Apogee (km):  85.81808126129849
Time to target (sec):  312.7000000000026



New simulation

fuelmass: 5437.440730935407


Stage 1 burnout
Velocity (km/s):  2.016959034014756
Angle (deg h):  43.65110927817635
Range (km):  43.185262500067815
Time (sec):  133.89999999999668
Final results:
Range (km):  535.1532002014948
Apogee (km):  169.6705625053785
```

```
Time to target (sec):  480.6000000000408



New simulation

fuelmass: 2436.318992076523


Stage 1 burnout
Velocity (km/s):  1.3059979248887084
Angle (deg h):  43.70171652329588
Range (km):  15.559733488322564
Time (sec):  60.00000000000058
Final results:
Range (km):  216.34529712094323
Apogee (km):  68.50093463288768
Time to target (sec):  275.09999999999405



New simulation

fuelmass: 5771.536834897998


Stage 1 burnout
Velocity (km/s):  2.0607572828510845
Angle (deg h):  43.642312875155234
Range (km):  45.71206092652399
Time (sec):  142.1999999999962
Final results:
Range (km):  560.3907944209011
Apogee (km):  177.55485806745324
Time to target (sec):  497.60000000004464
```

[21]:
```
m0_2_mu_plot, m0_2_var_plot = m0_2_model_emukit.predict(m0_param_1_x_plot)
```

[22]:
```
fig, ax = plt.subplots(figsize=plot.big_wide_figsize)
helper_plot_emulator_errorbars(x_plot=m0_param_1_x_plot, y_plot=m0_param_1_y_plot,
                               mu_plot=m0_2_mu_plot, var_plot=m0_2_var_plot,
                               model_emukit=m0_2_model_emukit)

m0_2_rmse = evaluate_prediction(y_actual=m0_param_1_y_plot, y_predicted=m0_2_mu_plot)
print("RMSE m0 (post experiment design loop): ", m0_2_rmse)
```

```
RMSE m0 (post experiment design loop):  1.0654800808660763
```



2 0. Only one param - m1

[23]:
```
m1_param_1 = 'Isp0'
m1_domain_param_1 = basic_param_spaces[m1_param_1] # [500, 6000] # [5000,15000]

m1_space = ParameterSpace(
          [ContinuousParameter(m1_param_1, *m1_domain_param_1),
          ])

custom_param_names = [m1_param_1]
```

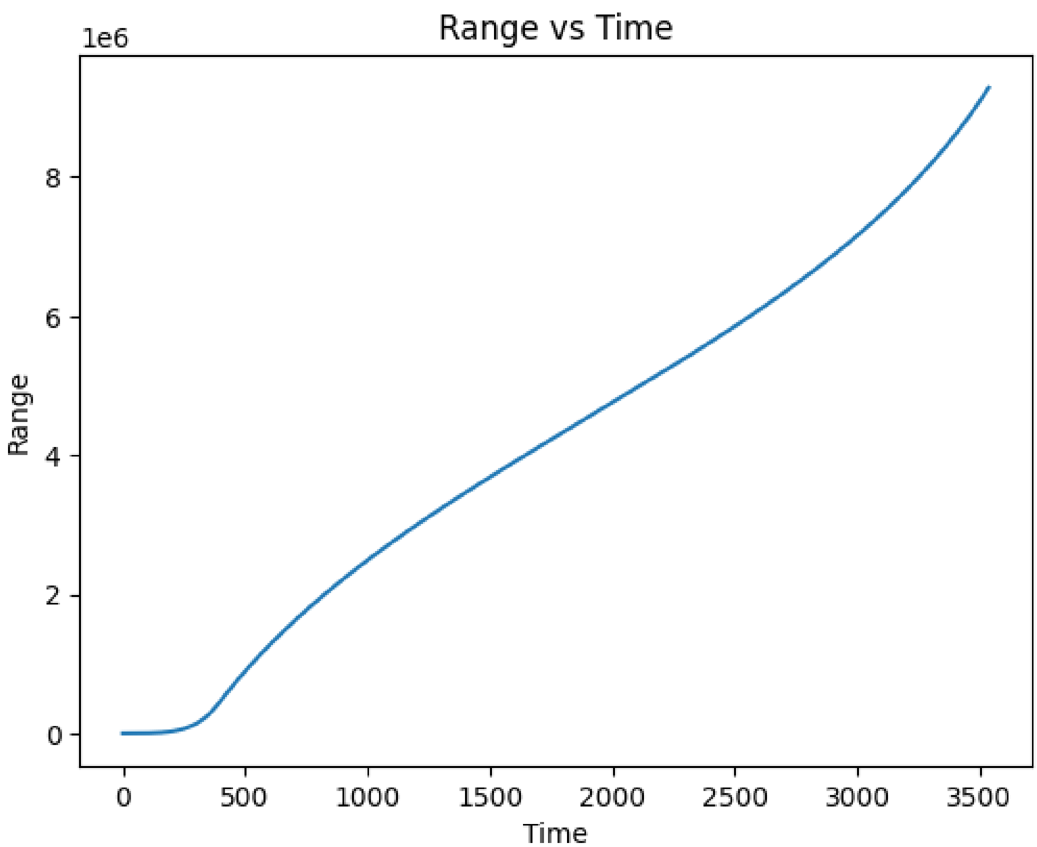
[24]:
```
def run_missile_sim(custom_params):
    """
    Recives in input an array of custom parameters.
    Each row represents a set of different parameters
    Each column is a different parameter (#cols = len(custom_param_names))
    """
    default_params_IRAQ = {
        'payload':500,
        'missilediam':0.88,
        'rvdiam':0,
        'estrange':600,
        'numstages':1,
        'fuelmass':[0,5600],
        'drymass':[0,1200],
        'Isp0':[0,226],
        'thrust0':[0,9177.4]
```

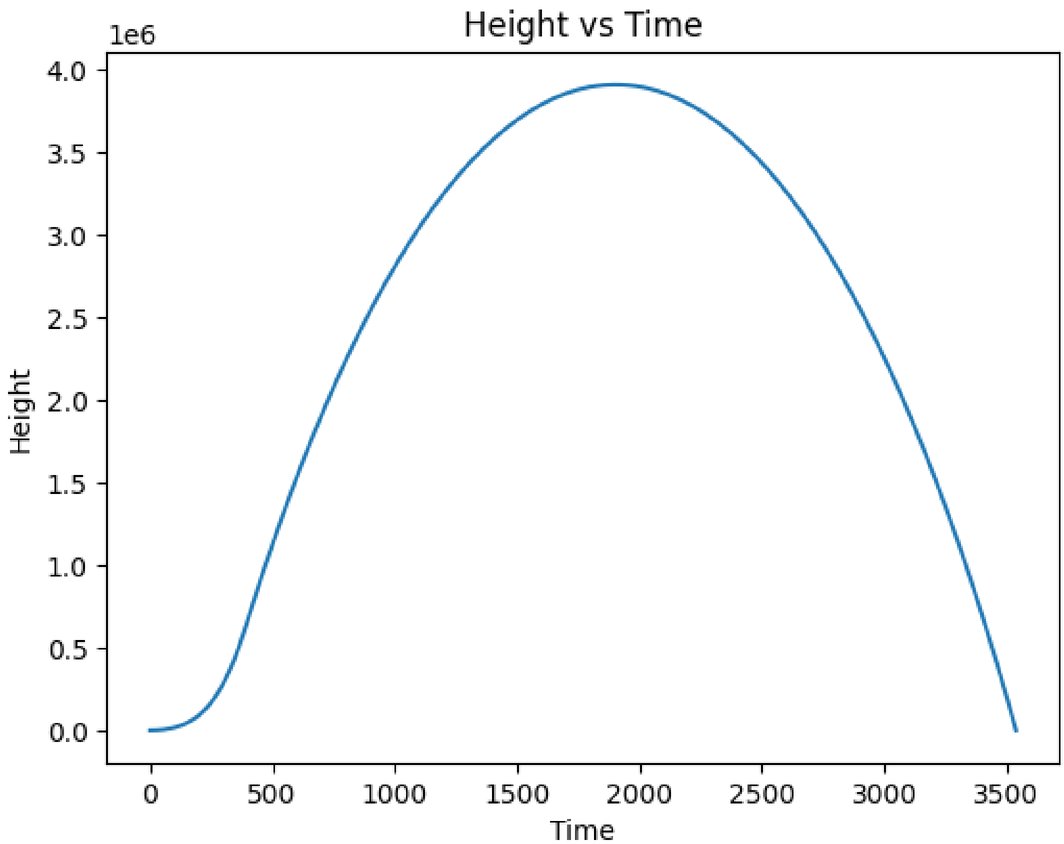
```
    }


    y = np.zeros((custom_params.shape[0], 1))
    for i in range(custom_params.shape[0]):
        params_to_use = default_params_IRAQ
        # Overwrite default param variables
        for j in range(custom_params.shape[1]):
            param_name = custom_param_names[j]
            if param_name in ['fuelmass', 'drymass', 'Isp0', 'thrust0']:
                params_to_use[param_name][1] = custom_params[i,j] # OK as long as we␣
↪are considering missiles with only 1 stage
            else:
                params_to_use[param_name] = custom_params[i, j]

            ## TEMP ## Better customise this
            if j==0:
                print('\nNew simulation \n')
            str_to_print = param_name + ': ' + str(custom_params[i,j])
            print(str_to_print)
            ##

        # Run simulation
        output_path = 'results/results_' + str(i) + '.txt' # TODO Define better␣
↪identifier
        sim_output = run_one_sim(
            numstages=params_to_use["numstages"],
            fuelmass=params_to_use["fuelmass"],
            drymass=params_to_use["drymass"],
            thrust0=params_to_use["thrust0"],
            Isp0=params_to_use["Isp0"],
            payload=params_to_use["payload"],
            missilediam=params_to_use["missilediam"],
            rvdiam=params_to_use["rvdiam"],
            est_range=params_to_use["estrange"],
            output_path=output_path,
            simulation_output=simulation_output,
        )

        y[i, 0] = sim_output
    return y
```

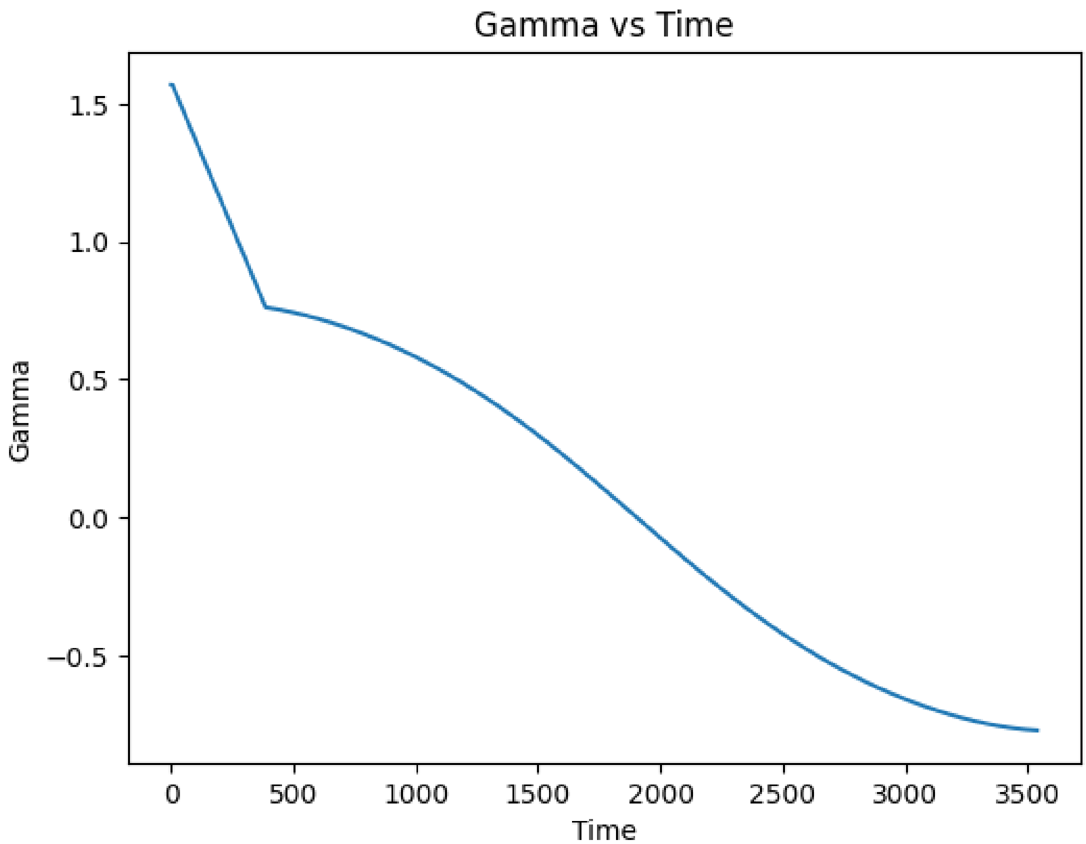
[25]:
```
# Get true points (to build model)
wirte_output_txt = True

m1_design = RandomDesign(m1_space)
m1_x = m1_design.get_samples(3)
m1_y = run_missile_sim(m1_x)
```

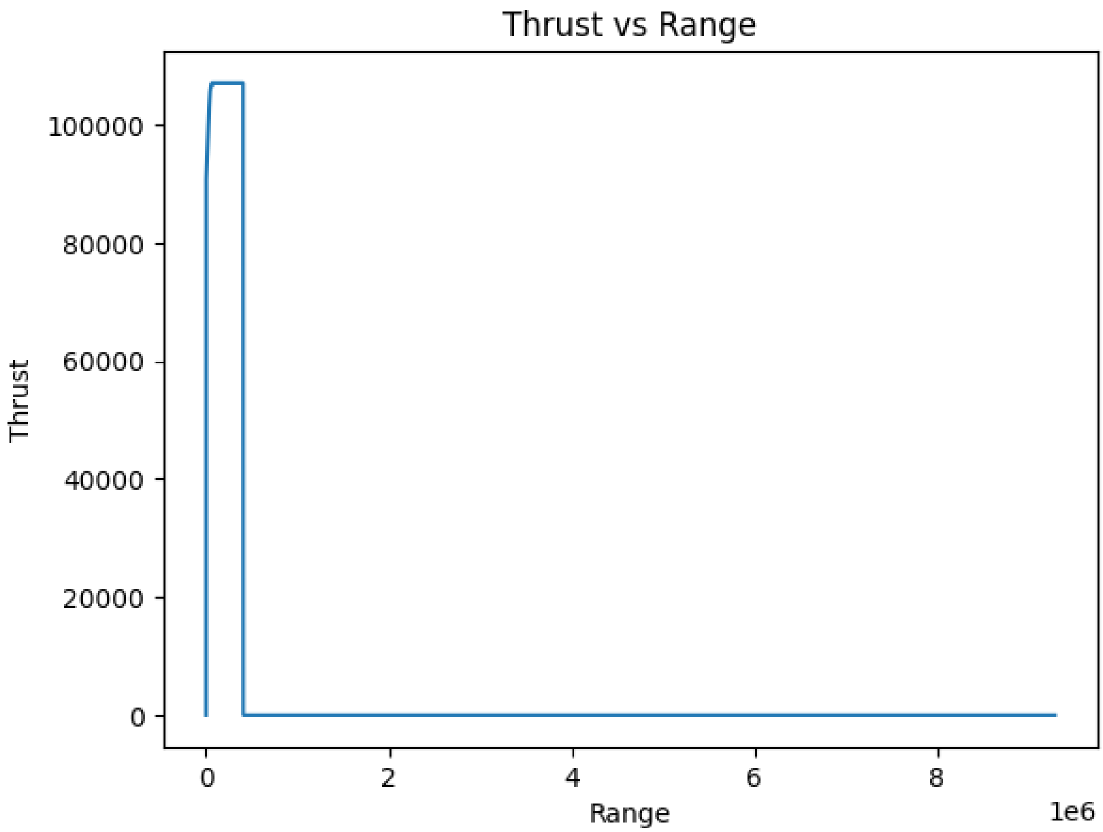
```
New simulation

Isp0: 632.4850588045783
```

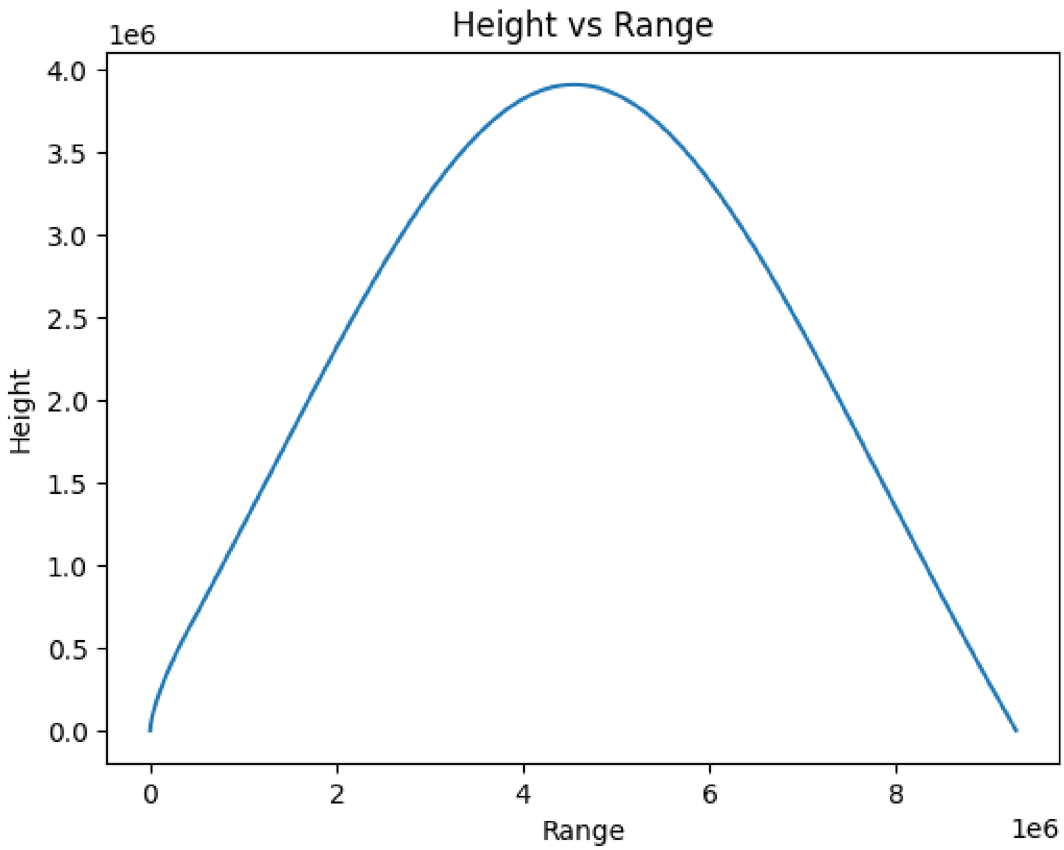
```
Stage 1 burnout
Velocity (km/s):  6.951986958952771
Angle (deg h):  43.65479851742443
Range (km):  406.2582827315758
Time (sec):  386.00000000001927
Final results:
Range (km):  9287.467919626113
Apogee (km):  3911.6038081153692
Time to target (sec):  3538.899999997877
```



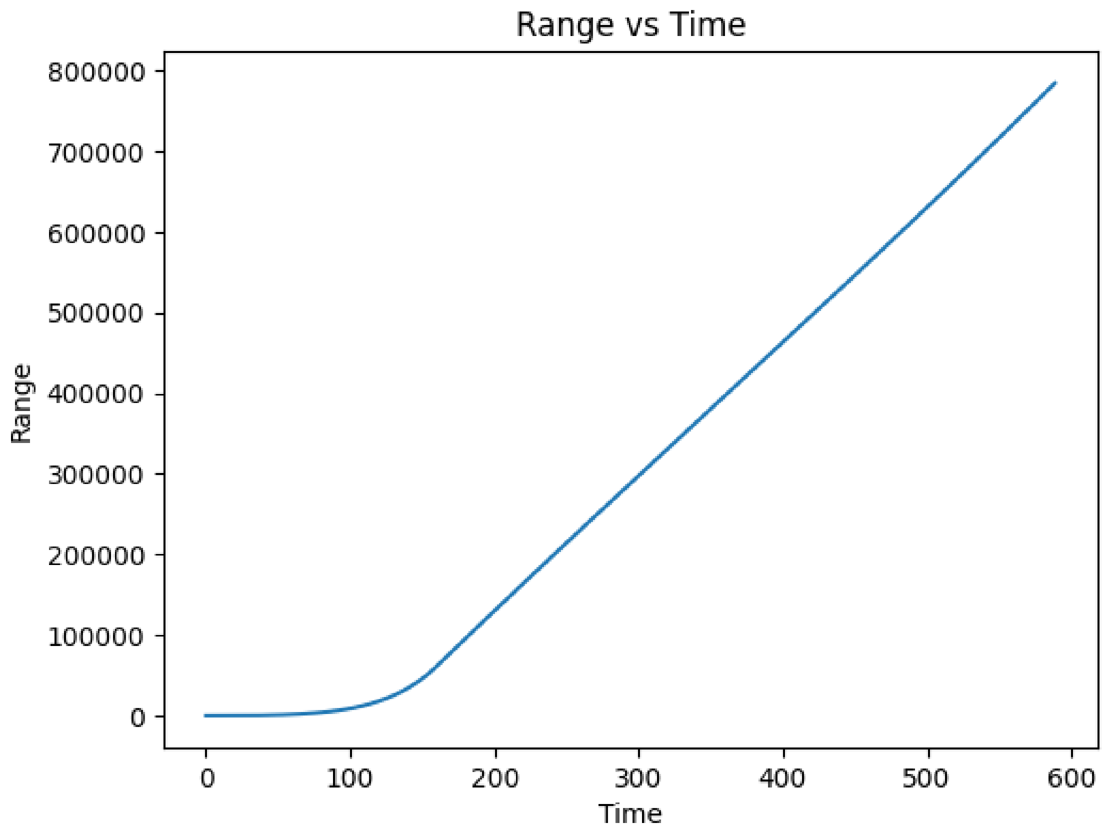

Height vs Time
1e6
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Height
0
500
1000
1500
2000
2500
3000
3500
Time

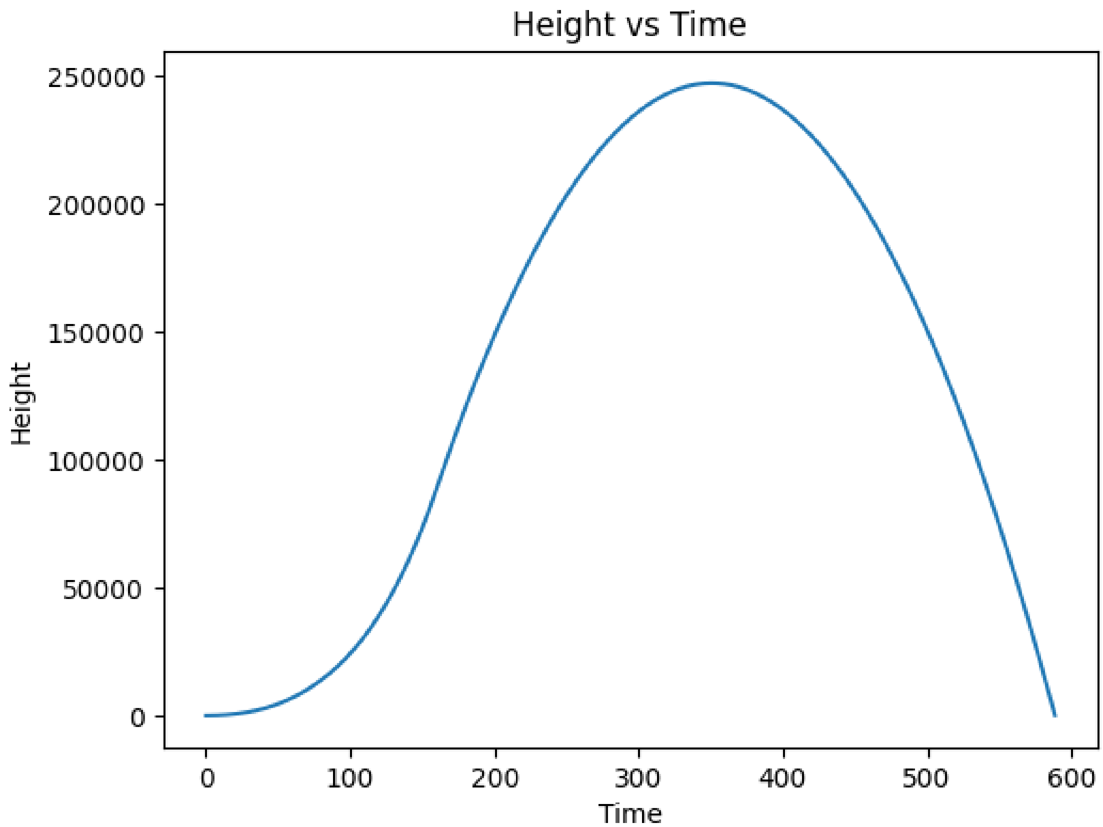

Gamma vs Time
1.5
1.0
0.5
0.0
−0.5
Gamma
0
500
1000
1500
2000
2500
3000
3500
Time

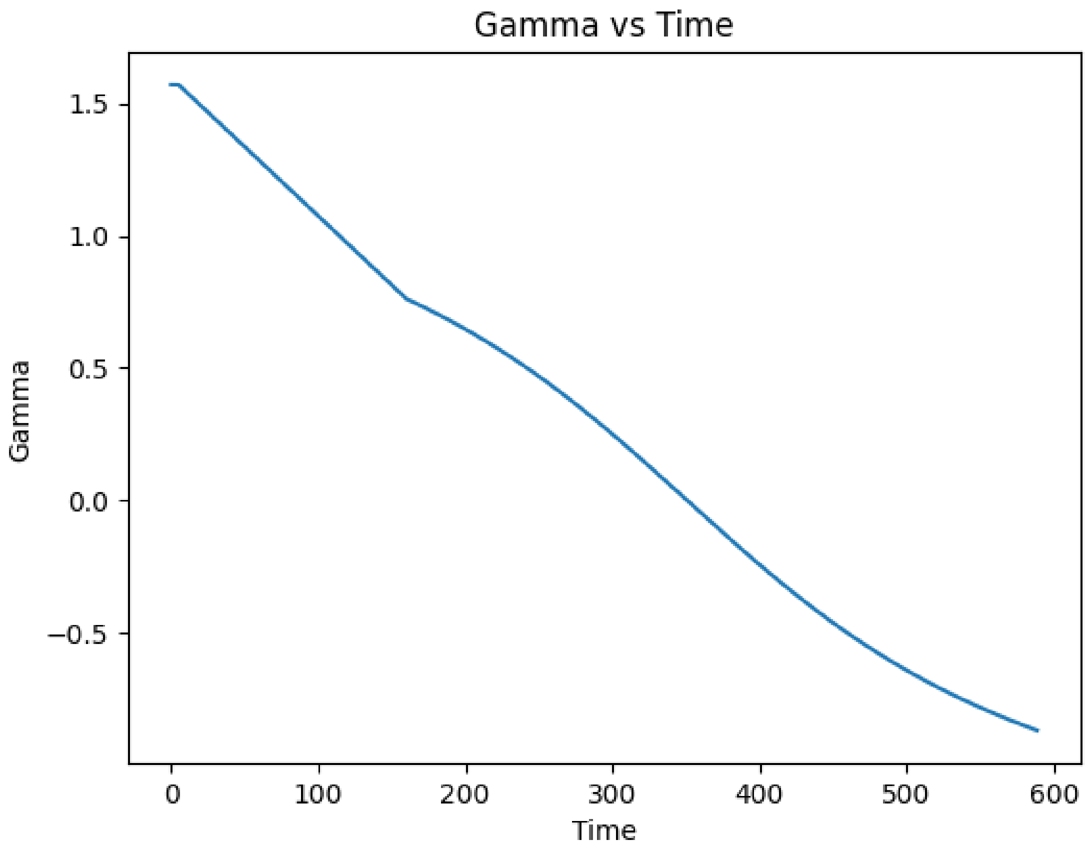

Thrust vs Range
Thrust
100000
80000
60000
40000
20000
0
0
2
4
6
8
Range
1e6

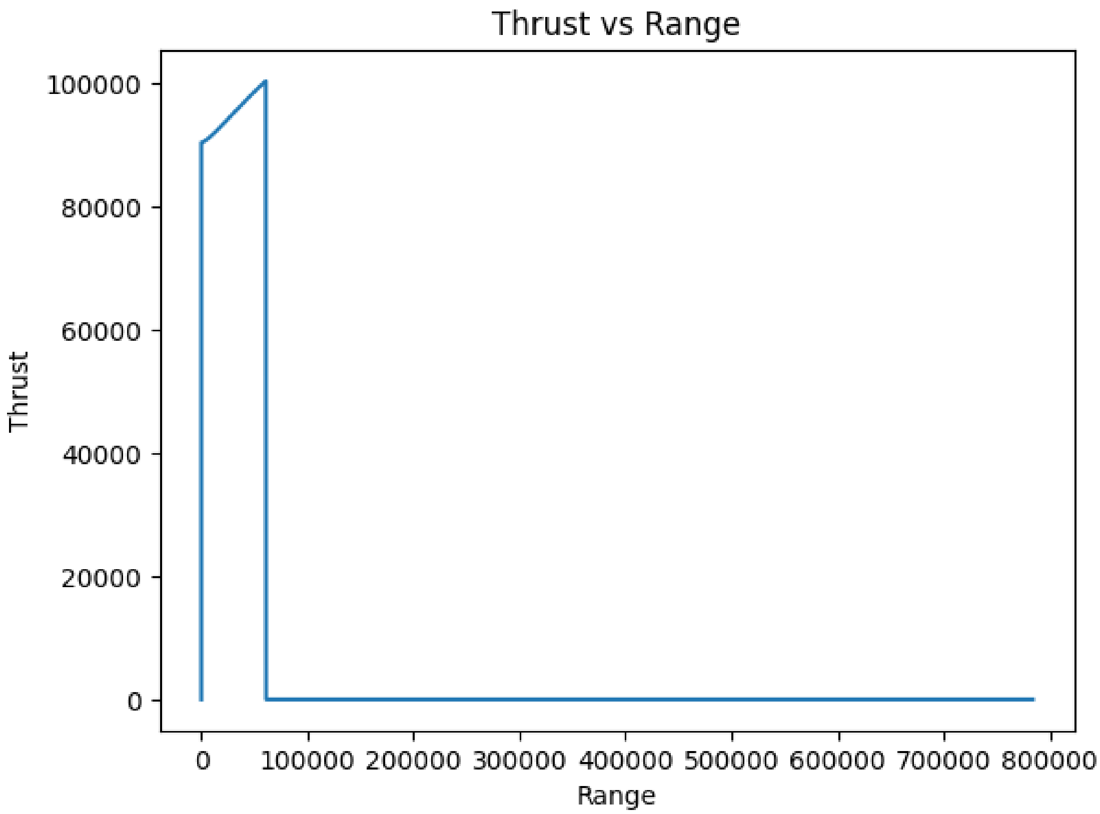


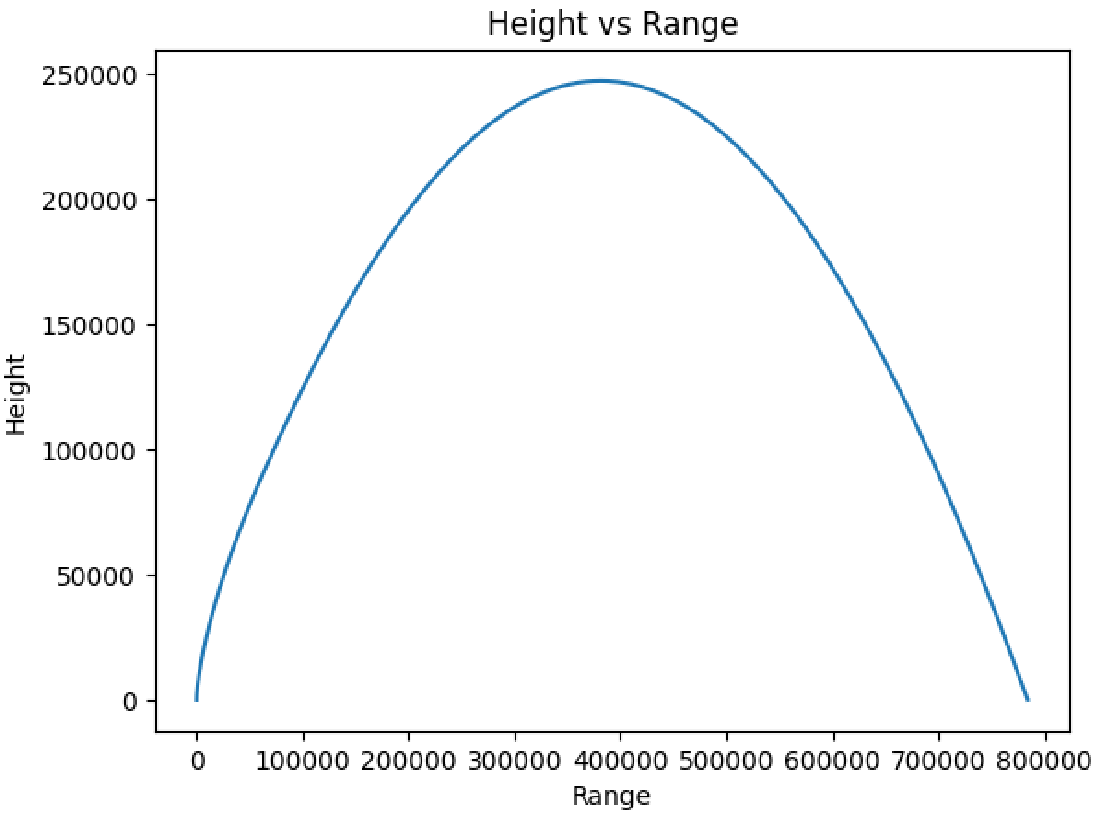

```
Data written to 'results/results_0.txt'

New simulation

Isp0: 261.652194637604


Stage 1 burnout
Velocity (km/s):  2.4293213942317604
Angle (deg h):  43.653675344881215
Range (km):  60.73458609084893
Time (sec):  159.6999999999952
Final results:
Range (km):  784.223400078734
Apogee (km):  247.42768830021498
Time to target (sec):  588.6000000000653
```


Range vs Time
Range
800000
700000
600000
500000
400000
300000
200000
100000
0
0
100
200
300
400
500
600
Time

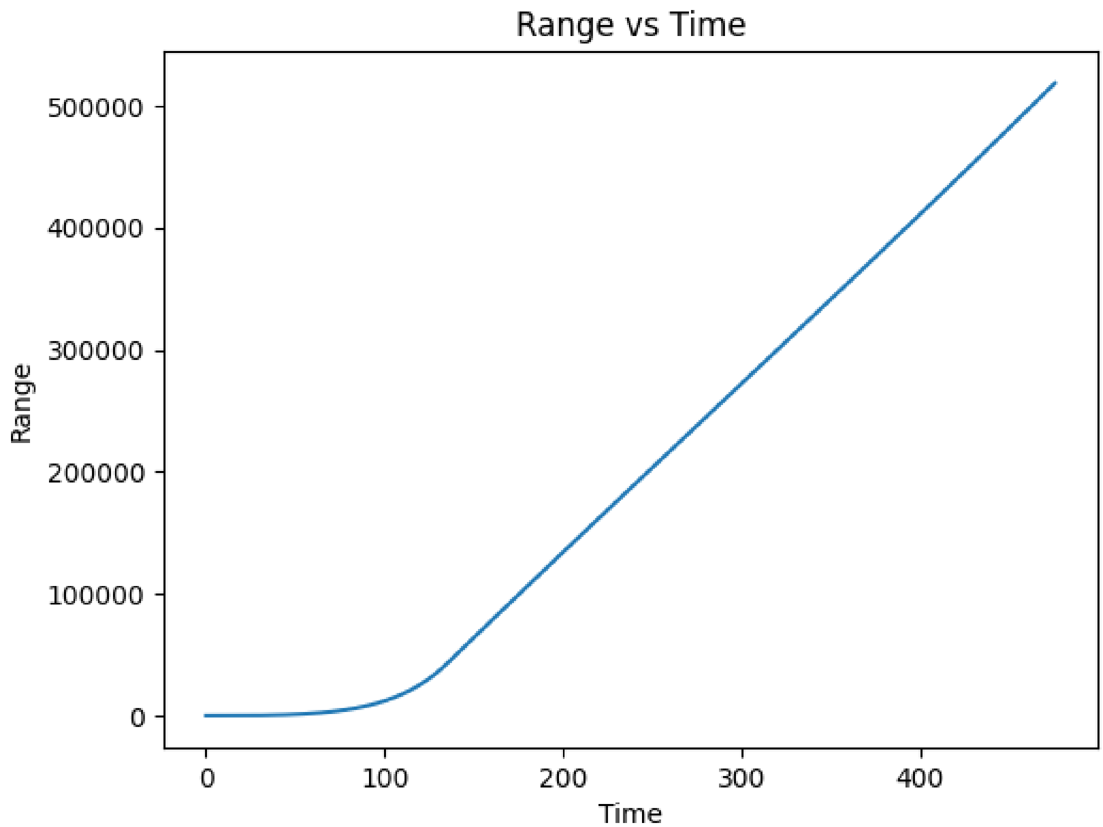

Height vs Time
Height
250000
200000
150000
100000
50000
0
0
100
200
300
400
500
600
Time

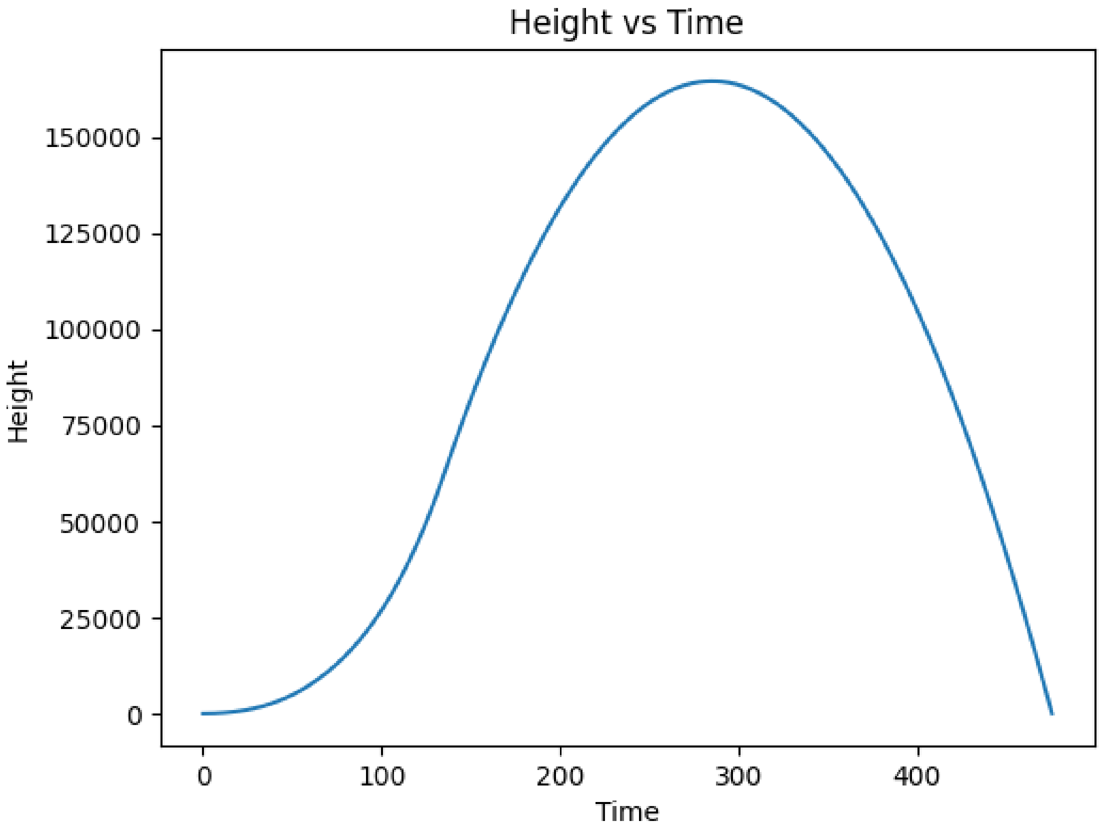

Gamma vs Time
1.5
1.0
0.5
0.0
−0.5
Gamma
0
100
200
300
400
500
600
Time

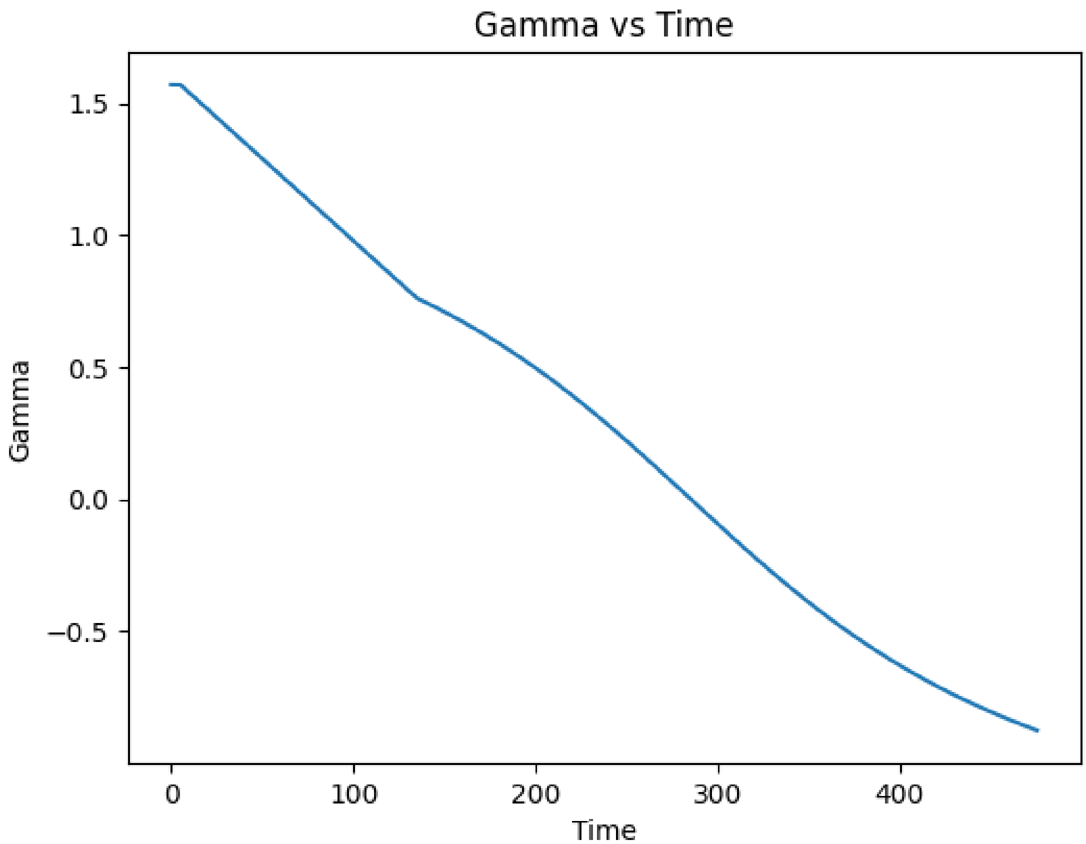

Thrust vs Range
Thrust
100000
80000
60000
40000
20000
0
0
100000
200000
300000
400000
500000
600000
700000
800000
Range

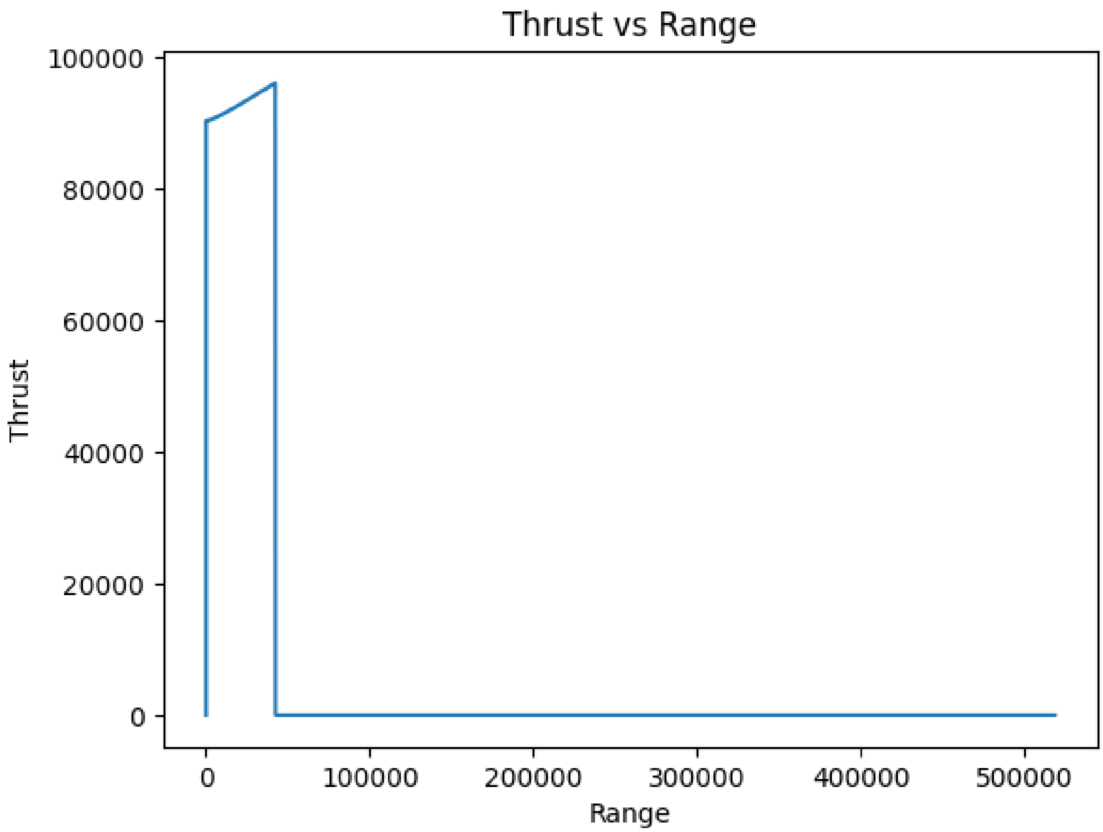


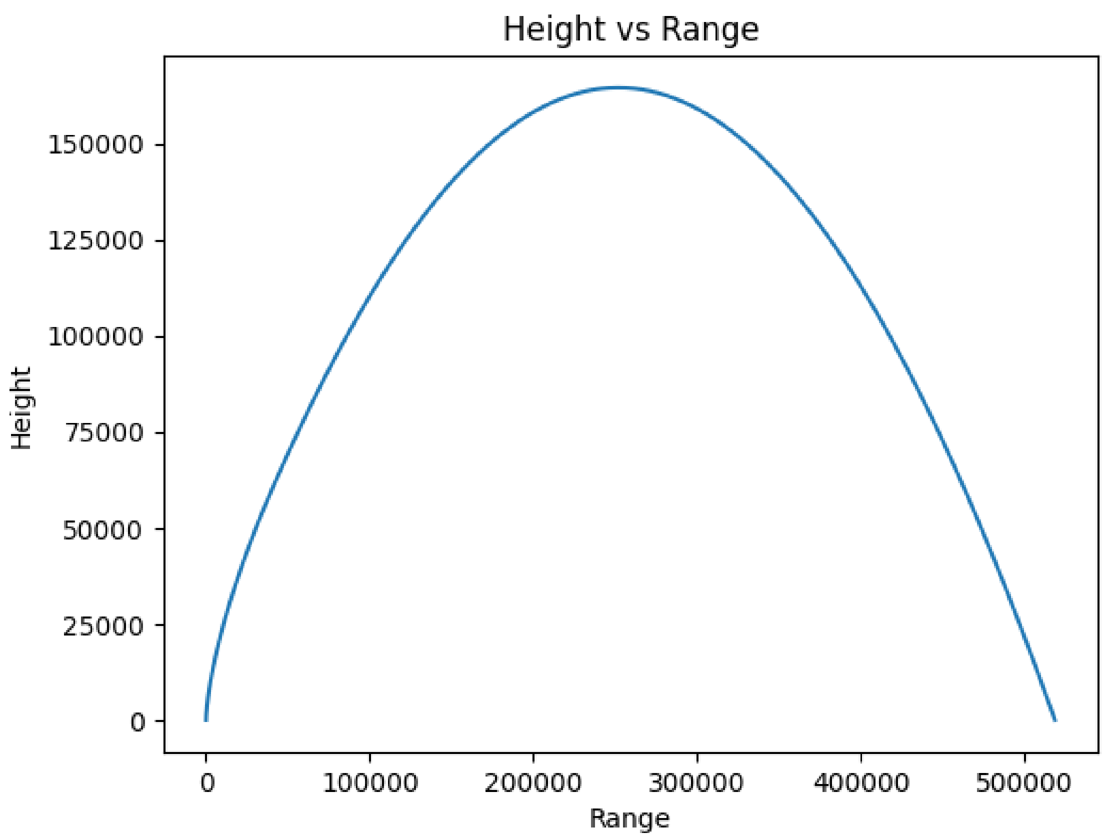

```
Data written to 'results/results_1.txt'

New simulation

Isp0: 220.8452126160107


Stage 1 burnout
Velocity (km/s):  1.98477240573621
Angle (deg h):  43.65288429473307
Range (km):  42.32475216111083
Time (sec):  134.79999999999663
Final results:
Range (km):  518.6145973012602
Apogee (km):  164.50094646411802
Time to target (sec):  475.60000000003964
```


Range vs Time
Range
Time
500000
400000
300000
200000
100000
0
0
100
200
300
400


Height vs Time
Height
150000
125000
100000
75000
50000
25000
0
0
100
200
300
400
Time


Gamma vs Time
Gamma
1.5
1.0
0.5
0.0
−0.5
0
100
200
300
400
Time


Thrust vs Range
100000
80000
60000
40000
20000
0
Thrust
0
100000
200000
300000
400000
500000
Range




```
Data written to 'results/results_2.txt'
```

[26]:
```
# Build model
m1_var_kernel = (100)**2
m1_lengthscale = 100 # 1
m1_var_linear_kernel = (100)**2
m1_var_noise = 1e-5 # small value

constrain_lengthscale = True

#kern = GPy.kern.RBF(input_dim=1, lengthscale=100, variance =var_kernel )  # ,␣
↪lengthscale=0.08, variance=20
# kern = GPy.kern.Matern32(input_dim=1)
# kern = GPy.kern.Linear(input_dim=1)
m1_rbf_kern = GPy.kern.RBF(input_dim=1, lengthscale=m1_lengthscale)
if constrain_lengthscale:
    m1_rbf_kern.lengthscale.constrain_bounded(m1_lengthscale, m1_lengthscale*1e12)

m1_kern = m1_rbf_kern + \
    GPy.kern.Linear(input_dim=1)
# m1_kern = m1_rbf_kern

m1_model_gpy = GPRegression(m1_x,m1_y, kernel=m1_kern)
m1_model_gpy.kern.variance =  m1_var_kernel
```

```
m1_model_gpy.likelihood.variance.fix(m1_var_noise)
display(m1_model_gpy)
```

```
reconstraining parameters rbf.lengthscale
```

```
<GPy.models.gp_regression.GPRegression at 0x7ff428841810>
```

[27]:
```
# Fit emulator
m1_model_emukit = GPyModelWrapper(m1_model_gpy)
m1_model_emukit.optimize() # Optimize model hyperparameters
```

[28]:
```
display(m1_model_gpy)
```

```
<GPy.models.gp_regression.GPRegression at 0x7ff428841810>
```

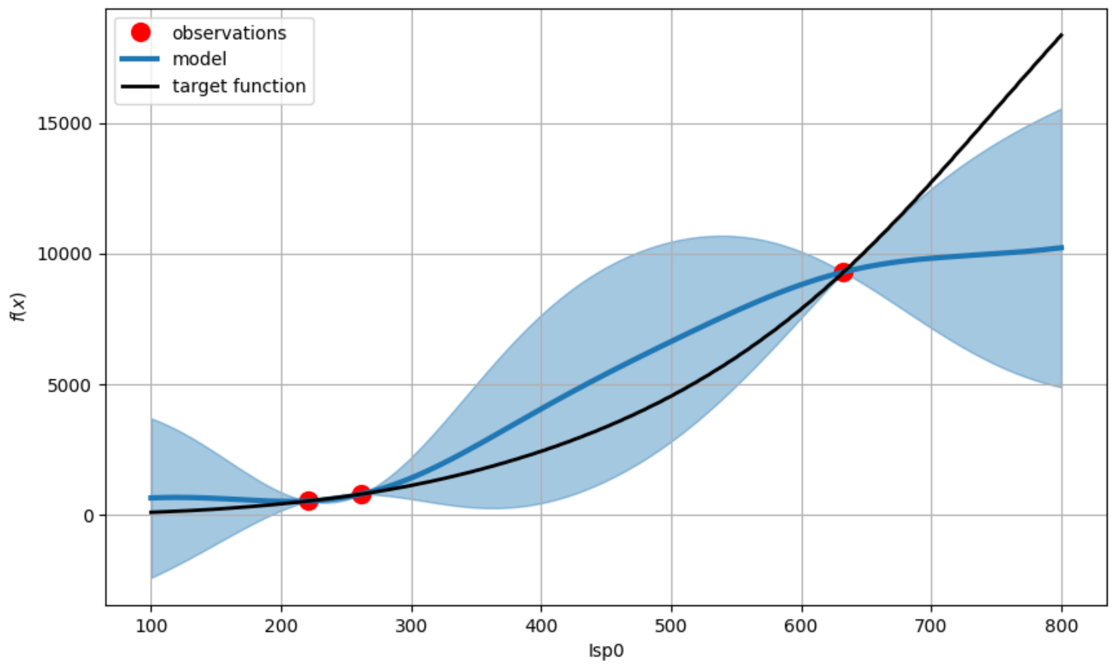
[29]:
```
# Get true points corresponding to param_1_x_plot (for plot)
wirte_output_txt = False

nr_points_plot = 301
m1_param_1_x_plot = np.linspace(m1_space.parameters[0].min, m1_space.parameters[0].
 ↪max, nr_points_plot)[:, None]
m1_param_1_y_plot = run_missile_sim(m1_param_1_x_plot)
```

[30]:
```
# Get model prediction on param_1_x_plot
m1_mu_plot, m1_var_plot = m1_model_emukit.predict(m1_param_1_x_plot)
```

[ ]:
```

```

[31]:
```
fig, ax = plt.subplots(figsize=plot.big_wide_figsize)
helper_plot_emulator_errorbars(x_plot=m1_param_1_x_plot, y_plot=m1_param_1_y_plot,
                               mu_plot=m1_mu_plot, var_plot=m1_var_plot,
                               model_emukit=m1_model_emukit)

m1_rmse = evaluate_prediction(y_actual=m1_param_1_y_plot, y_predicted=m1_mu_plot)
print("RMSE m1 (pre experiment design loop): ", m1_rmse)
```

```
RMSE m1 (pre experiment design loop):  2495.4252546081
```



[ ]:

2.0.1 Experiment design loop

[32]:
```
m1_2_model_emukit = m1_model_emukit
```

[33]:
```
wirte_output_txt = False

integrated_variance = IntegratedVarianceReduction(space=m1_space,
                                                  model=m1_2_model_emukit)
m1_ed = ExperimentalDesignLoop(space=m1_space,
                               model=m1_2_model_emukit,
                               acquisition = integrated_variance,
                               batch_size = 1)
# bach size is set to one in this example as we'll collect evaluations
# sequentially but parallel evaluations are allowed
m1_ed.run_loop(user_function=run_missile_sim, stopping_condition=5)
```

```
/Users/ilariasartori/opt/anaconda3/envs/mlphysical/lib/python3.10/site-
packages/paramz/transformations.py:111: RuntimeWarning:overflow encountered in
expm1


New simulation

Isp0: 453.8115854906686


Stage 1 burnout
Velocity (km/s):  4.757773066937232
```

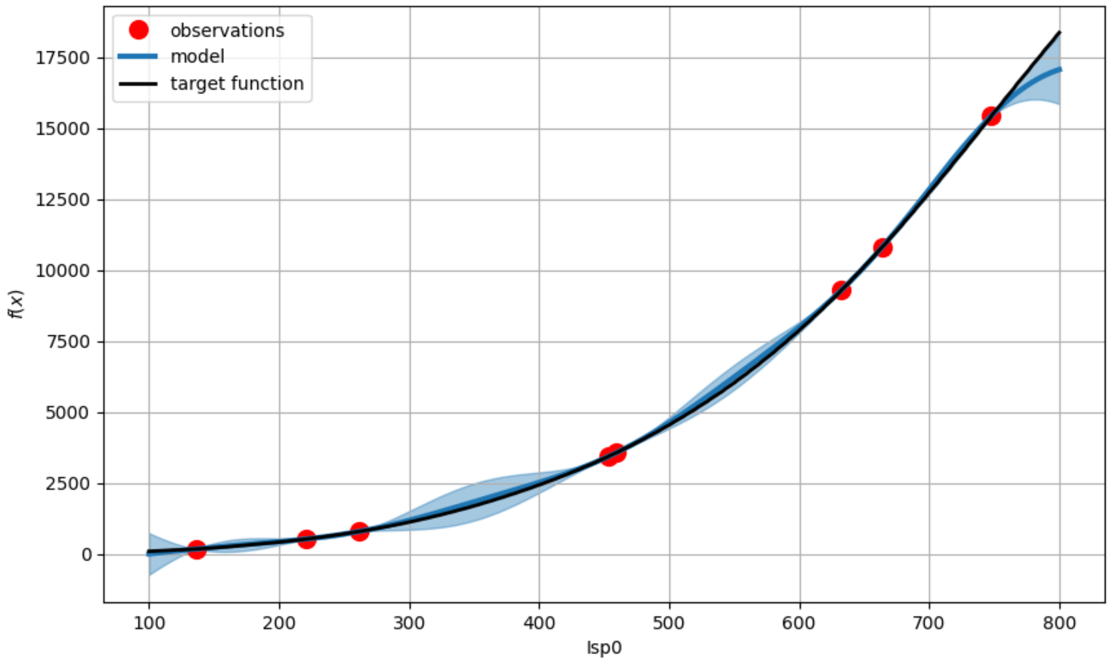
```
Angle (deg h):  43.65308698446824
Range (km):  200.62970617155497
Time (sec):  276.89999999999446
Final results:
Range (km):  3435.23616105572
Apogee (km):  1124.8530801502245
Time to target (sec):  1409.099999999814



New simulation

Isp0: 747.8325958719347


Stage 1 burnout
Velocity (km/s):  8.39211805728013
Angle (deg h):  43.65426424435811
Range (km):  572.8125394418782
Time (sec):  456.4000000000353
Final results:
Range (km):  15428.140370527834
Apogee (km):  10983.199112700311
Time to target (sec):  9072.500000015474



New simulation

Isp0: 459.0419390861636


Stage 1 burnout
Velocity (km/s):  4.822031152282082
Angle (deg h):  43.65163411011696
Range (km):  205.6886754418494
Time (sec):  280.0999999999952
Final results:
Range (km):  3548.962160892317
Apogee (km):  1165.8277895746091
Time to target (sec):  1442.2999999997837



New simulation

Isp0: 136.0755293944842


Stage 1 burnout
Velocity (km/s):  1.1243932391878897
Angle (deg h):  43.63489310225416
Range (km):  15.337397515705034
Time (sec):  83.09999999999955
```

```
Final results:
Range (km):  166.75202528661995
Apogee (km):  54.302467772155566
Time to target (sec):  270.7999999999931



New simulation

Isp0: 664.1460850222866


Stage 1 burnout
Velocity (km/s):  7.34343675835312
Angle (deg h):  43.657354331140674
Range (km):  449.3726221339817
Time (sec):  405.30000000002366
Final results:
Range (km):  10819.07080280896
Apogee (km):  5024.52619565074
Time to target (sec):  4380.199999998404
```

[34]:
```
m1_2_mu_plot, m1_2_var_plot = m1_2_model_emukit.predict(m1_param_1_x_plot)
```

[35]:
```
fig, ax = plt.subplots(figsize=plot.big_wide_figsize)
helper_plot_emulator_errorbars(x_plot=m1_param_1_x_plot, y_plot=m1_param_1_y_plot,
                               mu_plot=m1_2_mu_plot, var_plot=m1_2_var_plot,
                               model_emukit=m1_2_model_emukit)

m1_2_rmse = evaluate_prediction(y_actual=m1_param_1_y_plot, y_predicted=m1_2_mu_plot)
print("RMSE m1 (post experiment design loop): ", m1_2_rmse)
```

```
RMSE m1 (post experiment design loop):  202.81859322929324
```


observations
model
target function
17500
15000
12500
10000
7500
5000
2500
0
$f(x)$
100
200
300
400
500
600
700
800
Isp0

```
[ ]:
```

```
[ ]:
```