Missile_2feats_plusMaxRange

January 17, 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
```

[6]:
```
run_grid_simulation = True # If true takes much longer and does 3D plots and so on for␣
↪MODEL with 2 FEATS
```

[ ]:
```

```

We consider missiles with only 1 stage

[7]:
```
basic_param_spaces = {
    'payload':  [10, 2410],
    'missilediam':  [0.1, 9.9],
    'rvdiam':  [0.1, 9.9],
    'estrange': [100, 4900],
    'fuelmass': [500, 6000], # [500, 6000],
    'drymass':  [1000, 3000],
    'Isp0':  [100, 800],# [100, 800],
```

```
    'thrust0':  [10000, 69000],
}
```

[8]:
```
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 1. Two params

[9]:
```
m2_param_1 = 'fuelmass'
m2_domain_param_1 = basic_param_spaces[m2_param_1]
m2_param_2 = 'Isp0'
m2_domain_param_2 = basic_param_spaces[m2_param_2]

m2_space = ParameterSpace(
          [ContinuousParameter(m2_param_1, *m2_domain_param_1),
           ContinuousParameter(m2_param_2, *m2_domain_param_2),
          ])

custom_param_names = [m2_param_1, m2_param_2]
```

[10]:
```
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


def neg_run_missile_sim(custom_params):
    return -run_missile_sim(custom_params)
```

1.1 1. Experimental design

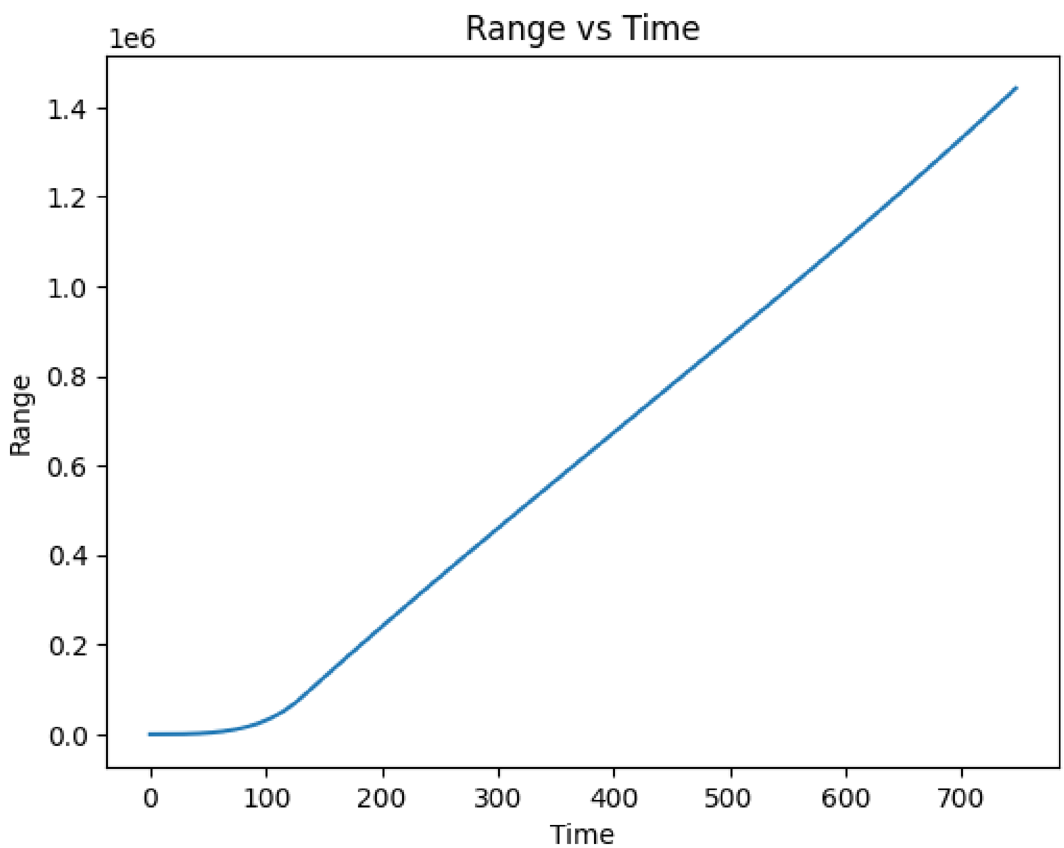
1.1.1 Use model-free experimental design to start

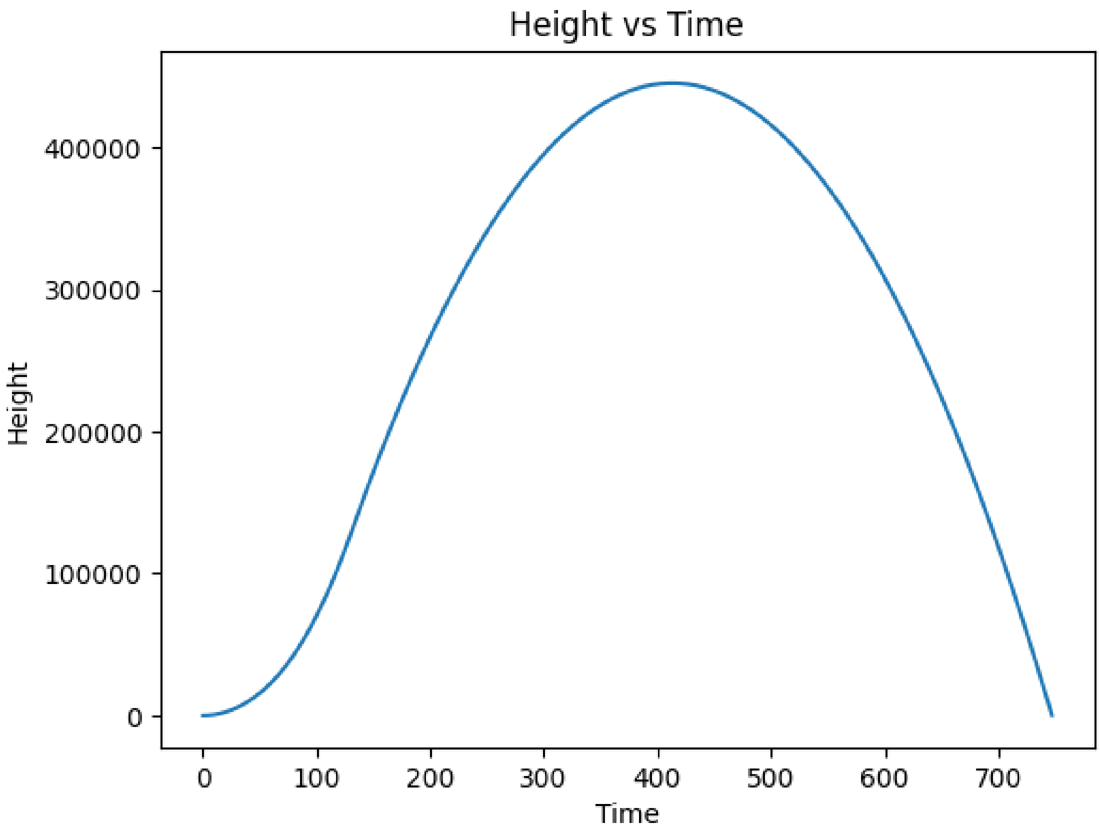
(RandomDesign or Latin Design)

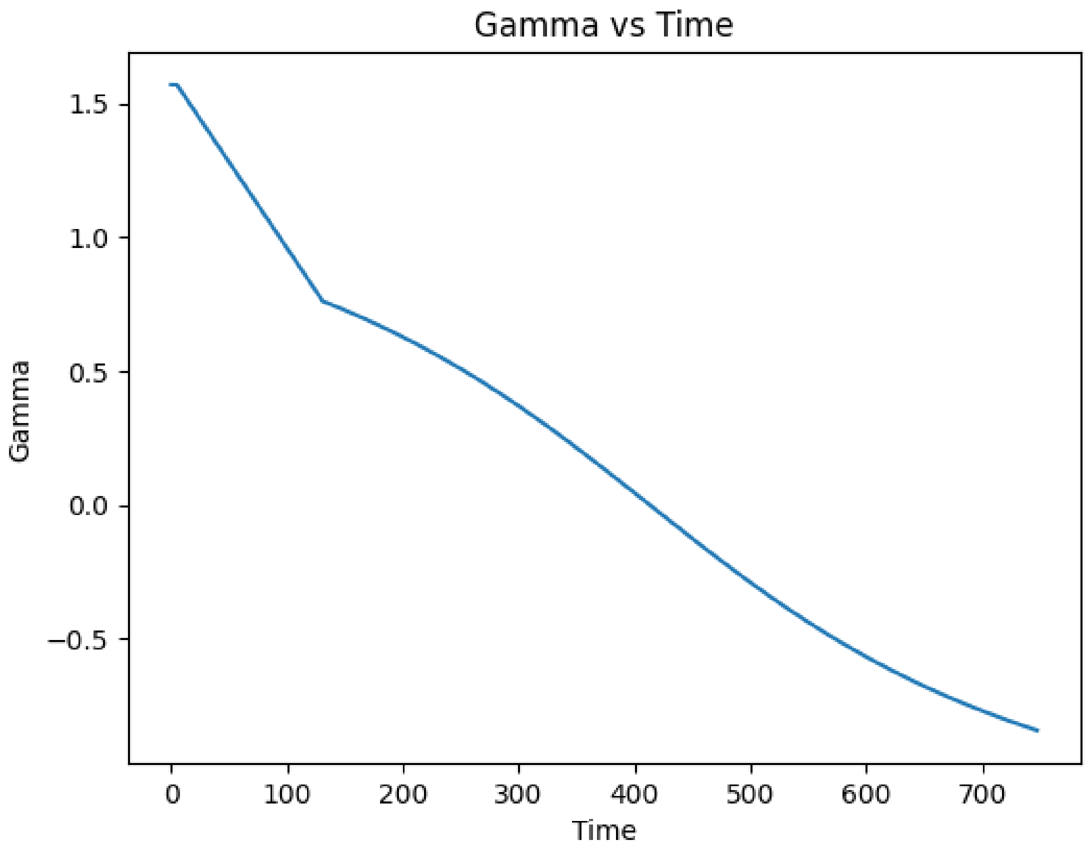
[11]:
```
wirte_output_txt = True


# from emukit.core.initial_designs.latin_design import LatinDesign
# design = LatinDesign(parameter_space)

m2_design = RandomDesign(m2_space)
m2_x = m2_design.get_samples(3*2)
```

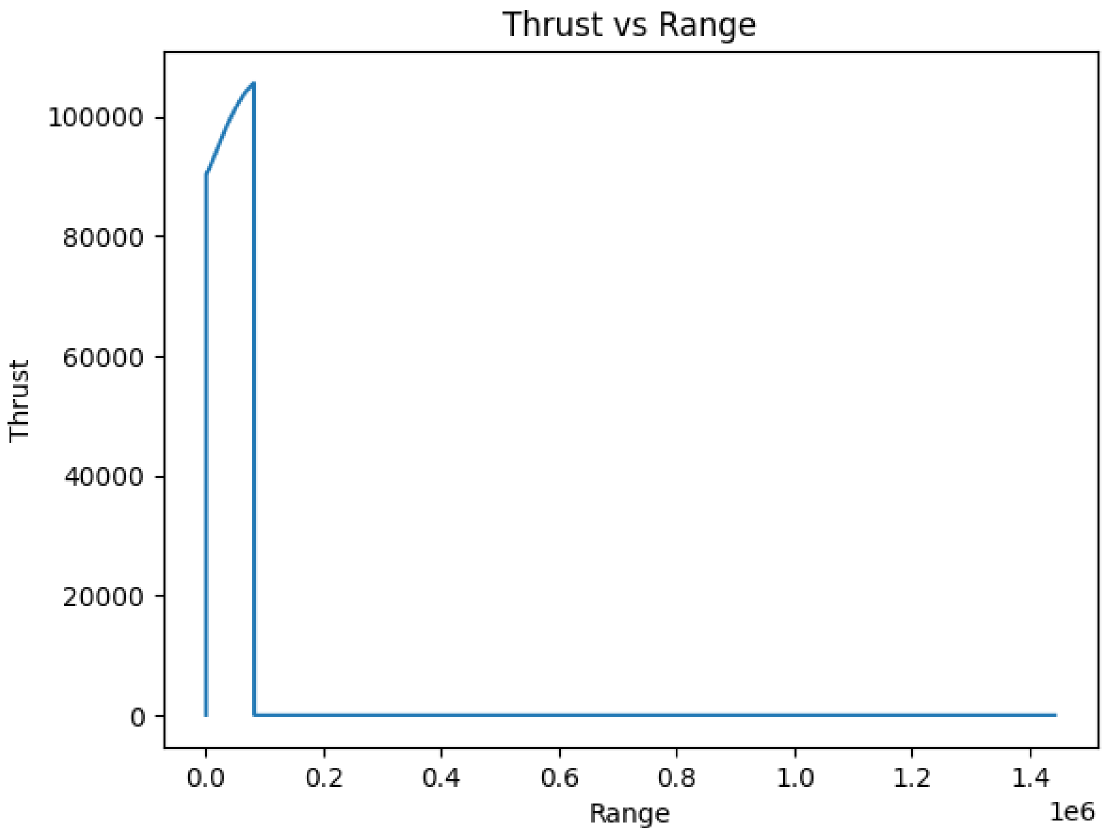
```
m2_y = neg_run_missile_sim(m2_x)
```

```
New simulation

fuelmass: 2461.699830729682
Isp0: 486.9259736726181


Stage 1 burnout
Velocity (km/s):  3.2928421666750873
Angle (deg h):  43.654711759668814
Range (km):  81.70654746630755
Time (sec):  130.59999999999687
Final results:
Range (km):  1443.036540282888
Apogee (km):  445.40587370665315
Time to target (sec):  747.3000000001014
```

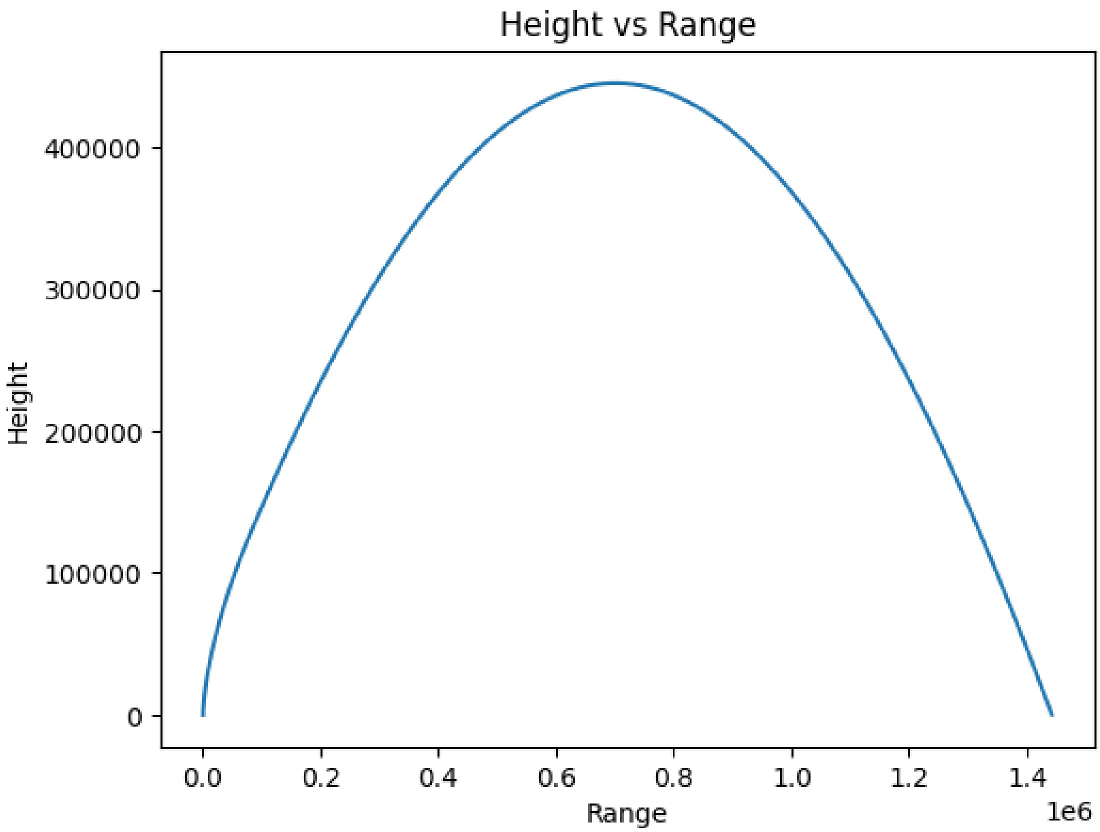



Height vs Time
400000
300000
200000
100000
0
Height
0
100
200
300
400
500
600
700
Time

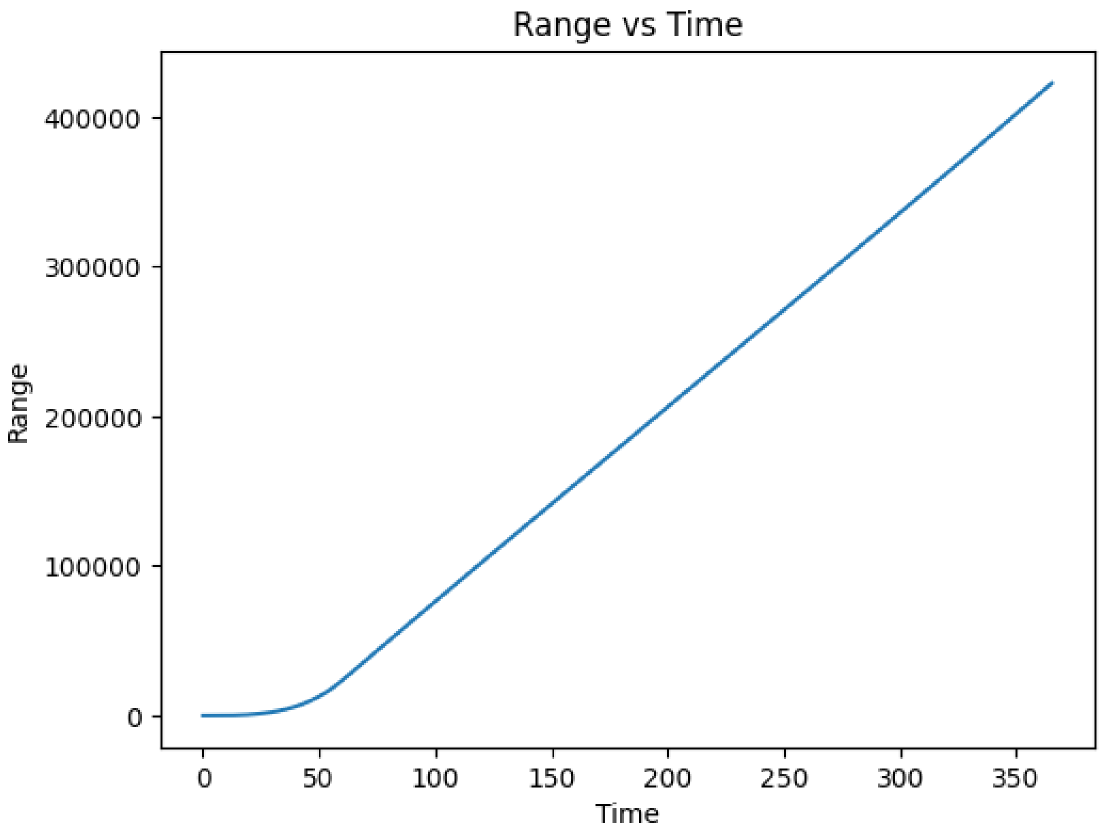

Gamma vs Time
1.5
1.0
0.5
0.0
−0.5
Gamma
0
100
200
300
400
500
600
700
Time

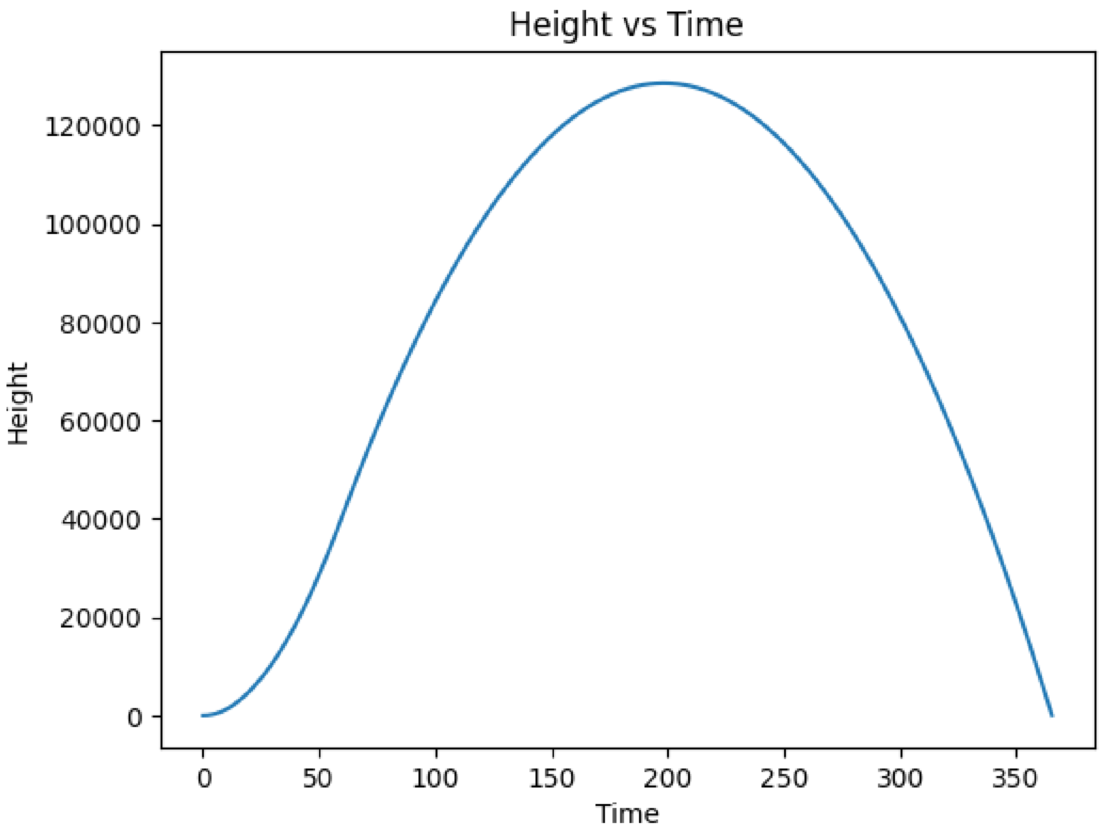


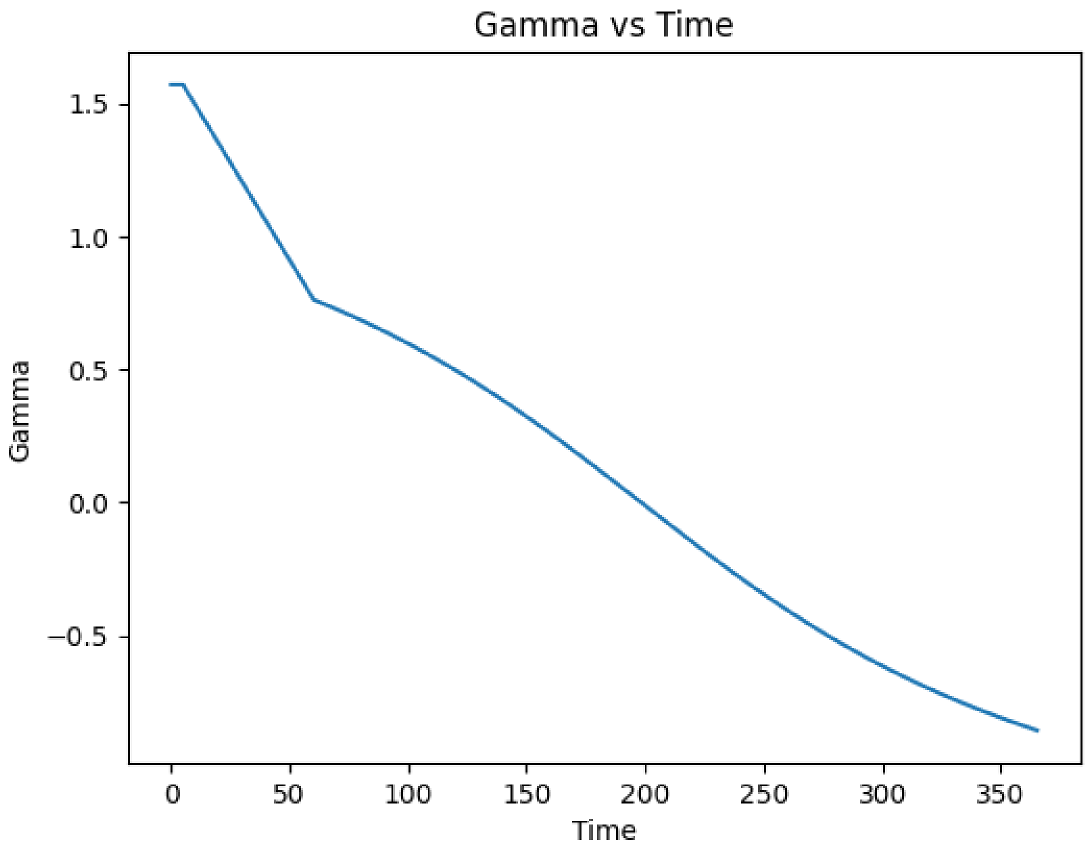
Thrust vs Range
Thrust
100000
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1e6
Range

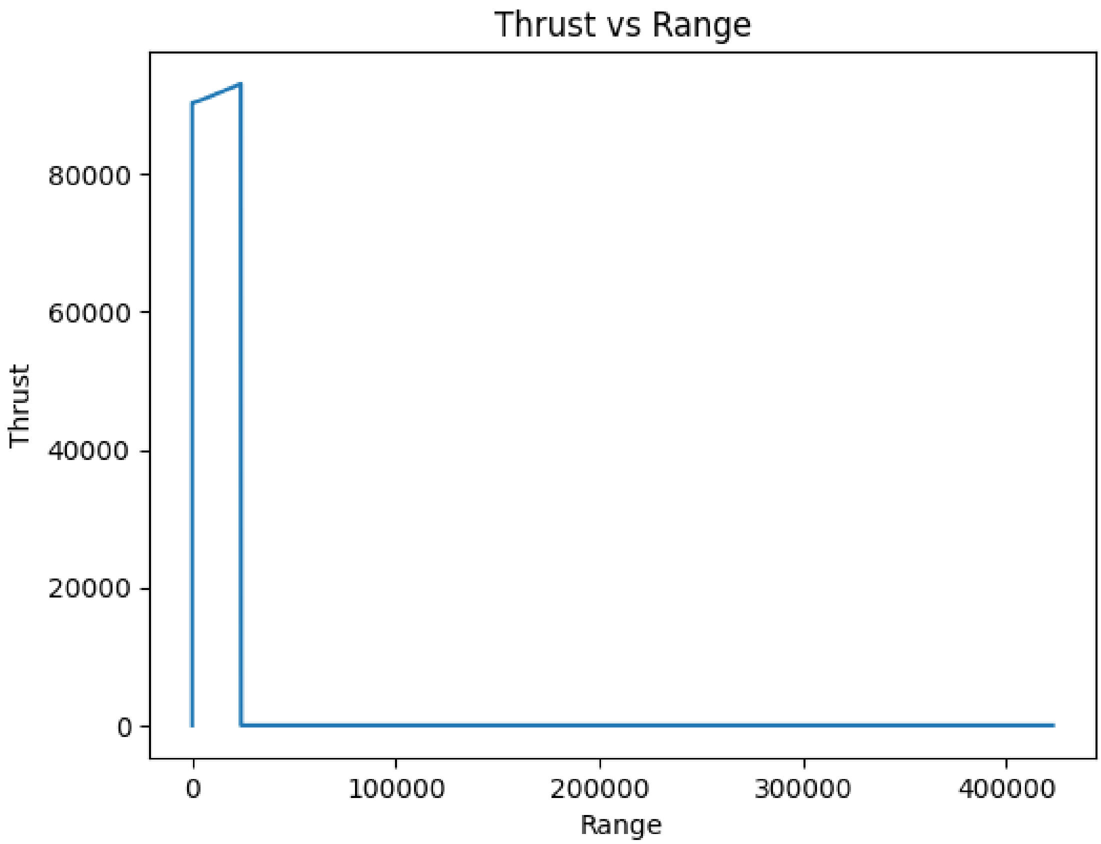



```
Data written to 'results/results_0.txt'

New simulation

fuelmass: 920.1477703768564
Isp0: 599.8626891636919


Stage 1 burnout
Velocity (km/s):  1.8475465441910104
Angle (deg h):  43.666929694041904
Range (km):  23.859429394146524
Time (sec):  60.200000000000585
Final results:
Range (km):  422.7984243339626
Apogee (km):  128.53208806192185
Time to target (sec):  365.4000000000146
```

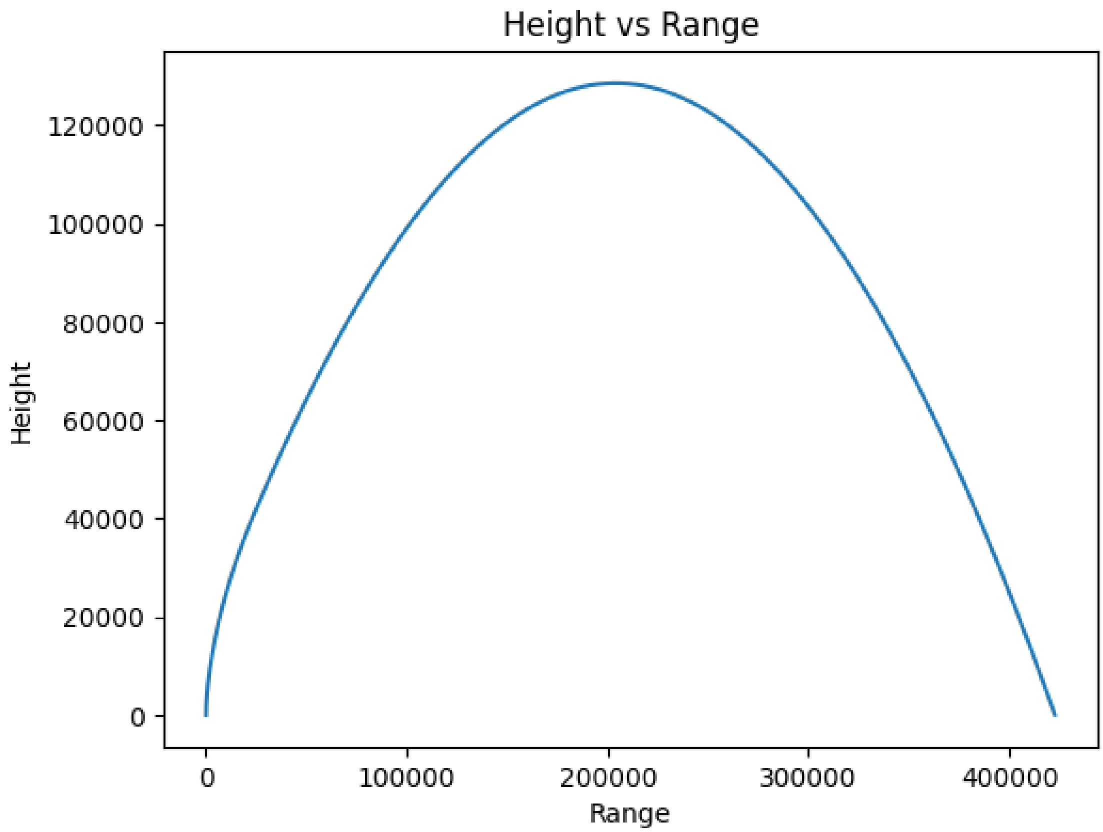

Range vs Time
Range
400000
300000
200000
100000
0
Time
0
50
100
150
200
250
300
350


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

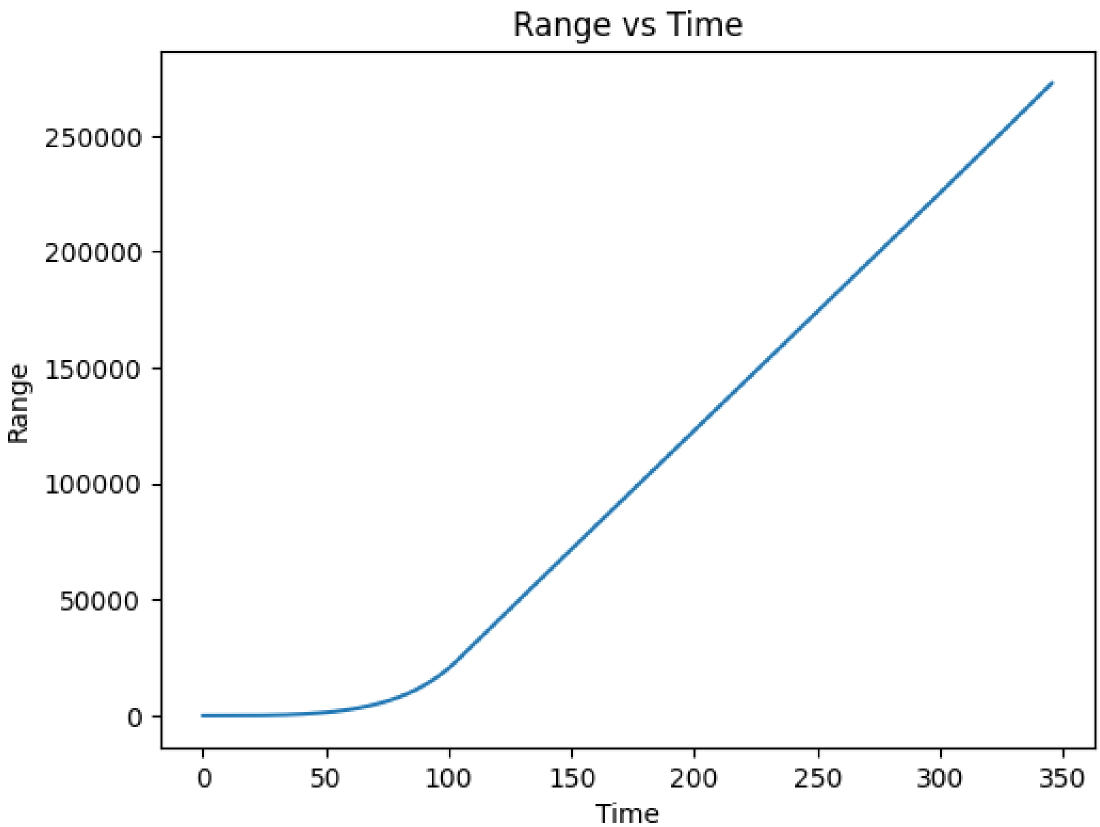

Gamma vs Time
Gamma
Time
1.5
1.0
0.5
0.0
−0.5
0
50
100
150
200
250
300
350

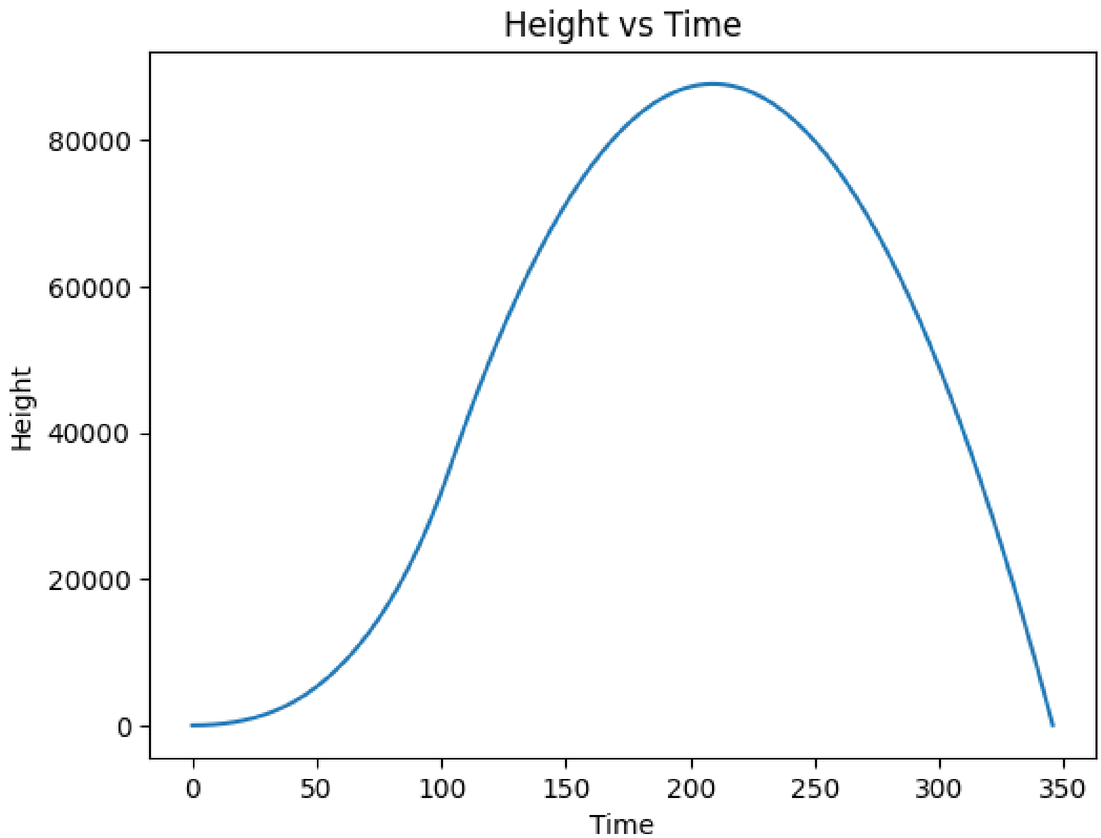


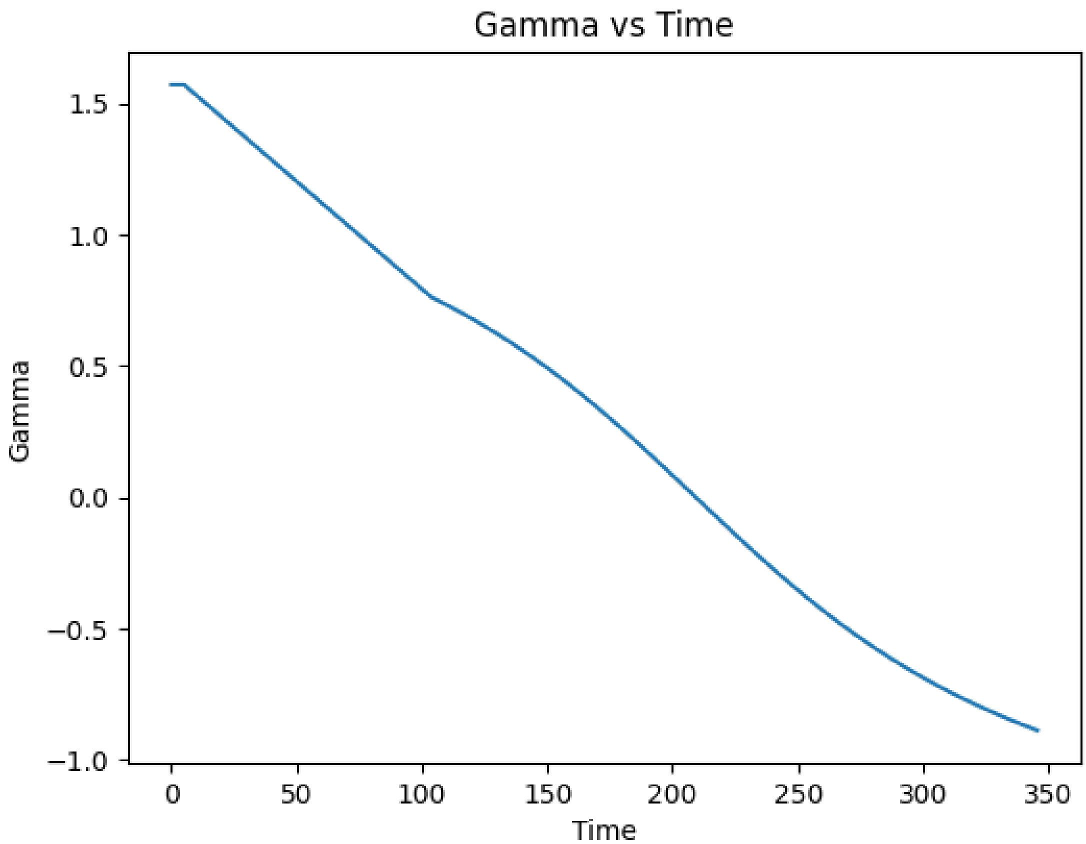
Thrust vs Range
Thrust
80000
60000
40000
20000
0
0
100000
200000
300000
400000
Range

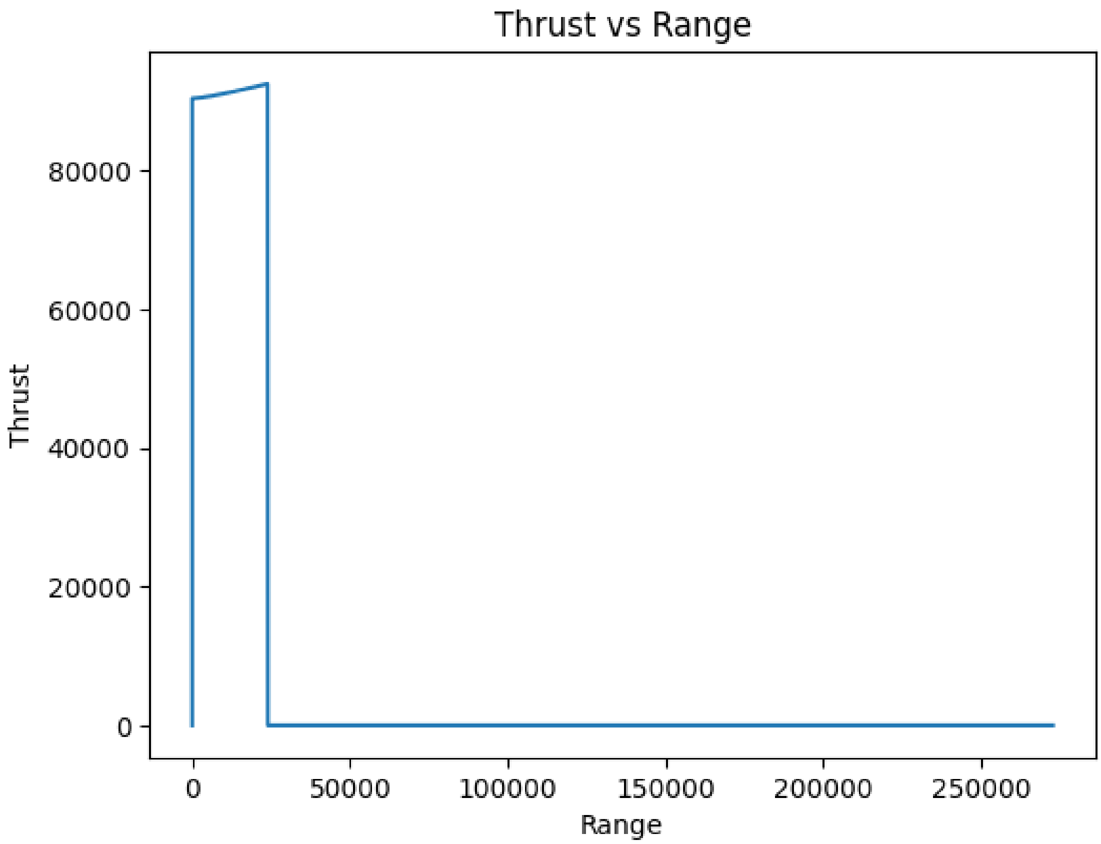


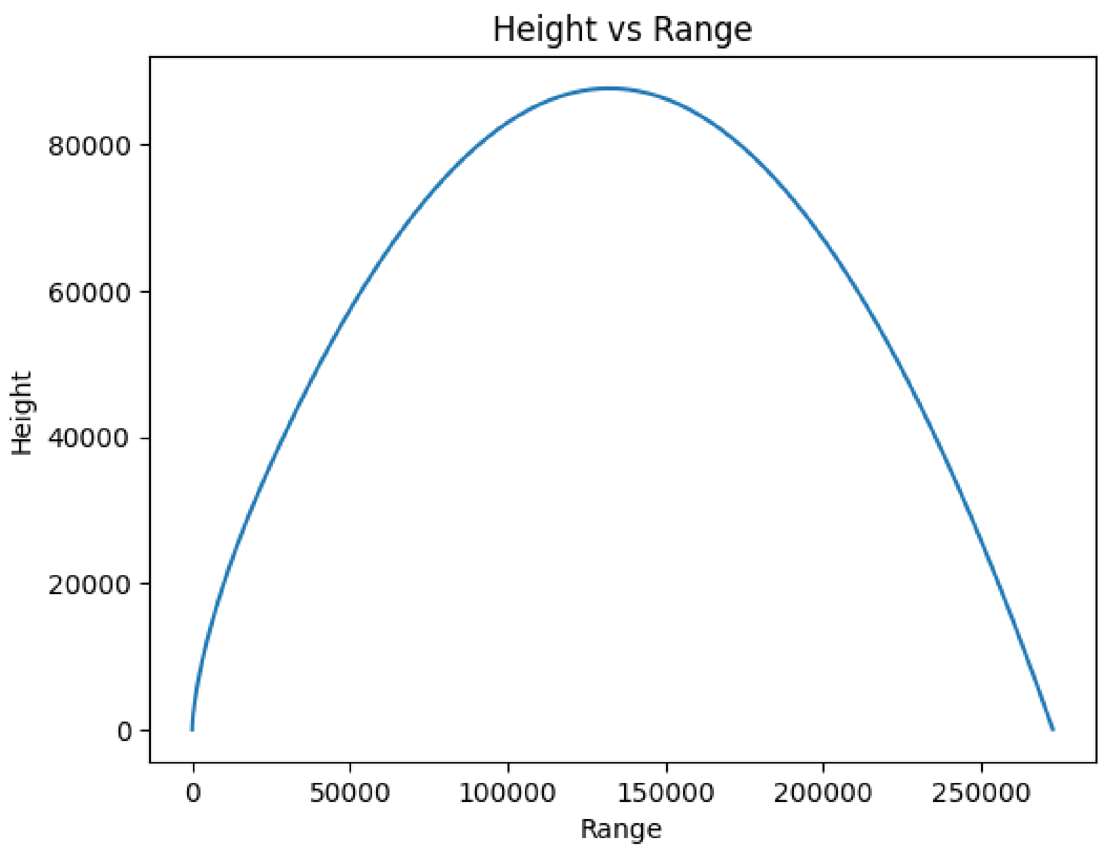

```
Data written to 'results/results_1.txt'

New simulation

fuelmass: 5672.14884132695
Isp0: 167.5983593012405


Stage 1 burnout
Velocity (km/s):  1.4417597535653883
Angle (deg h):  43.66386706640104
Range (km):  23.8802344333796
Time (sec):  103.59999999999839
Final results:
Range (km):  272.7963038431324
Apogee (km):  87.67103398367681
Time to target (sec):  345.7000000000101
```


Range vs Time
Range
250000
200000
150000
100000
50000
0
0
50
100
150
200
250
300
350
Time

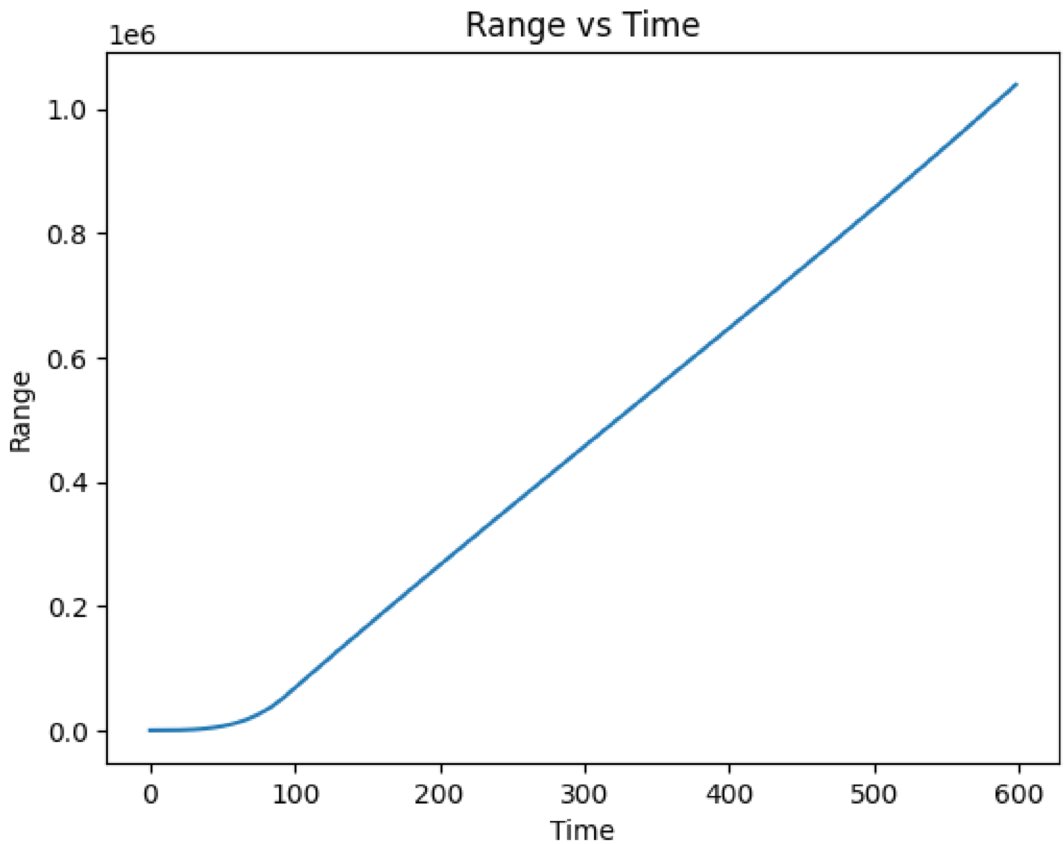

Height vs Time
Height
Time
0
20000
40000
60000
80000
0
50
100
150
200
250
300
350

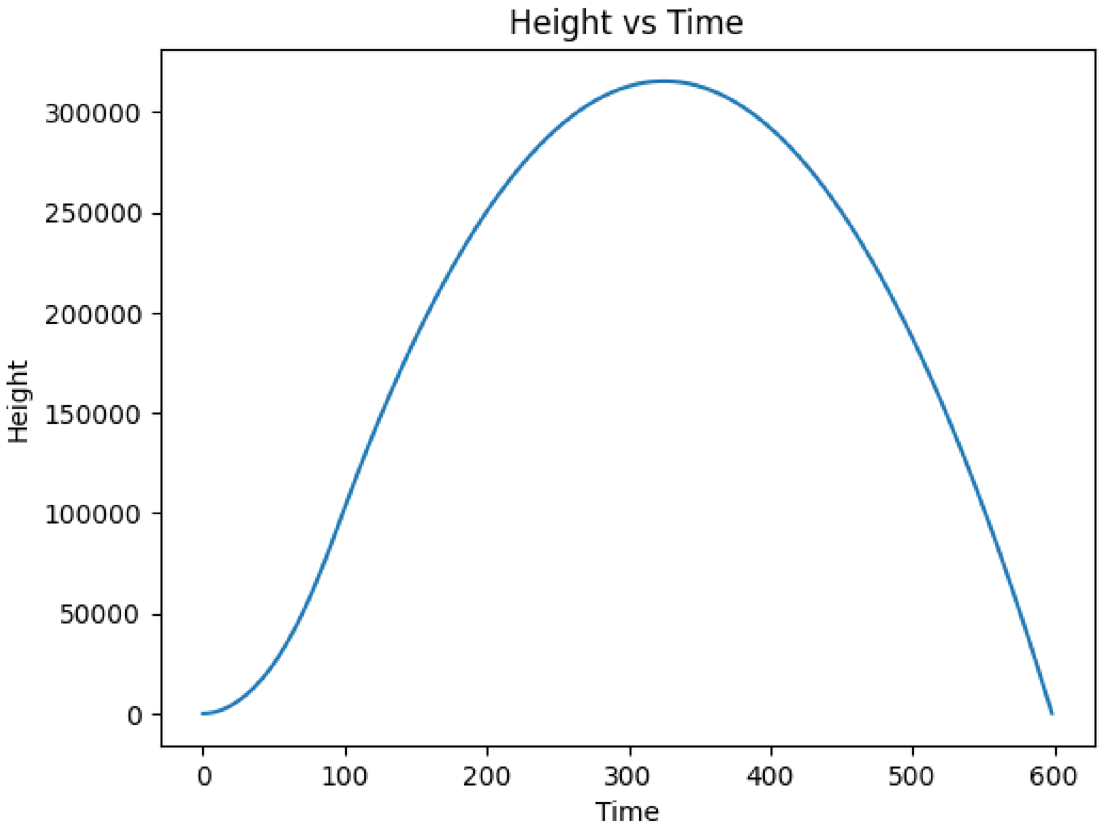

Gamma vs Time
1.5
1.0
0.5
0.0
−0.5
−1.0
Gamma
0
50
100
150
200
250
300
350
Time

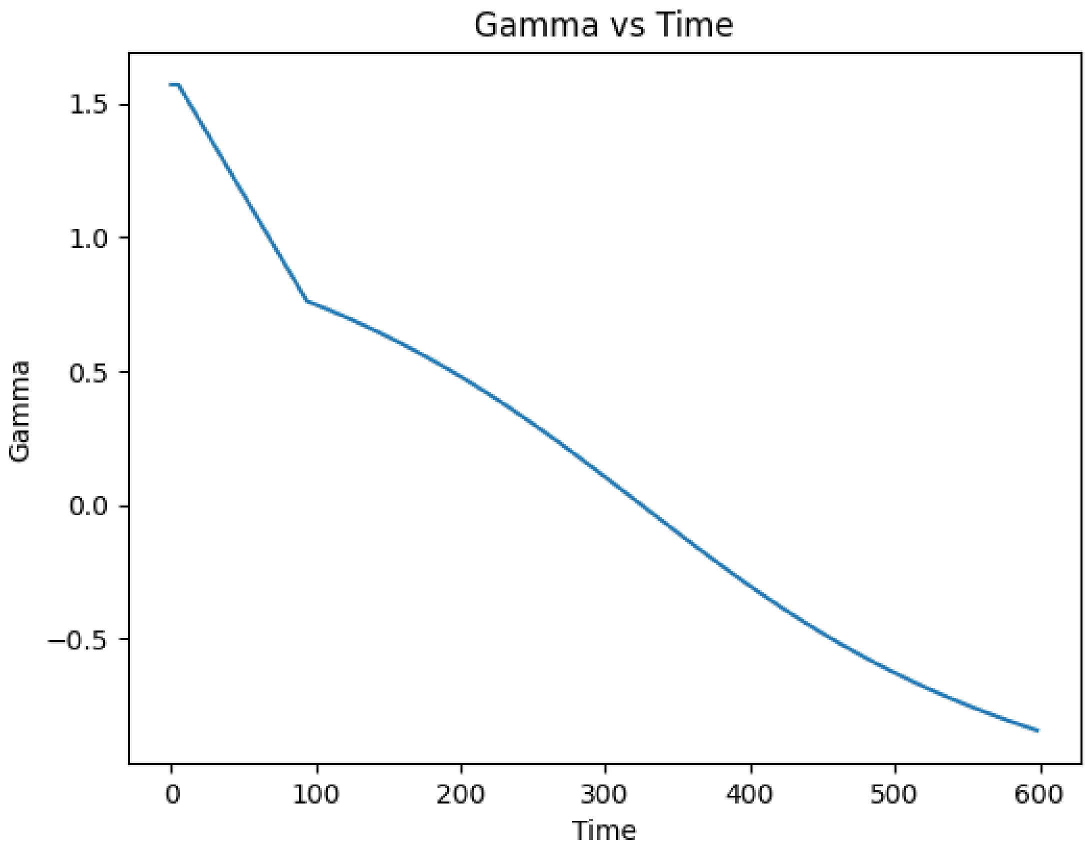

Thrust vs Range
Thrust
80000
60000
40000
20000
0
0
50000
100000
150000
200000
250000
Range

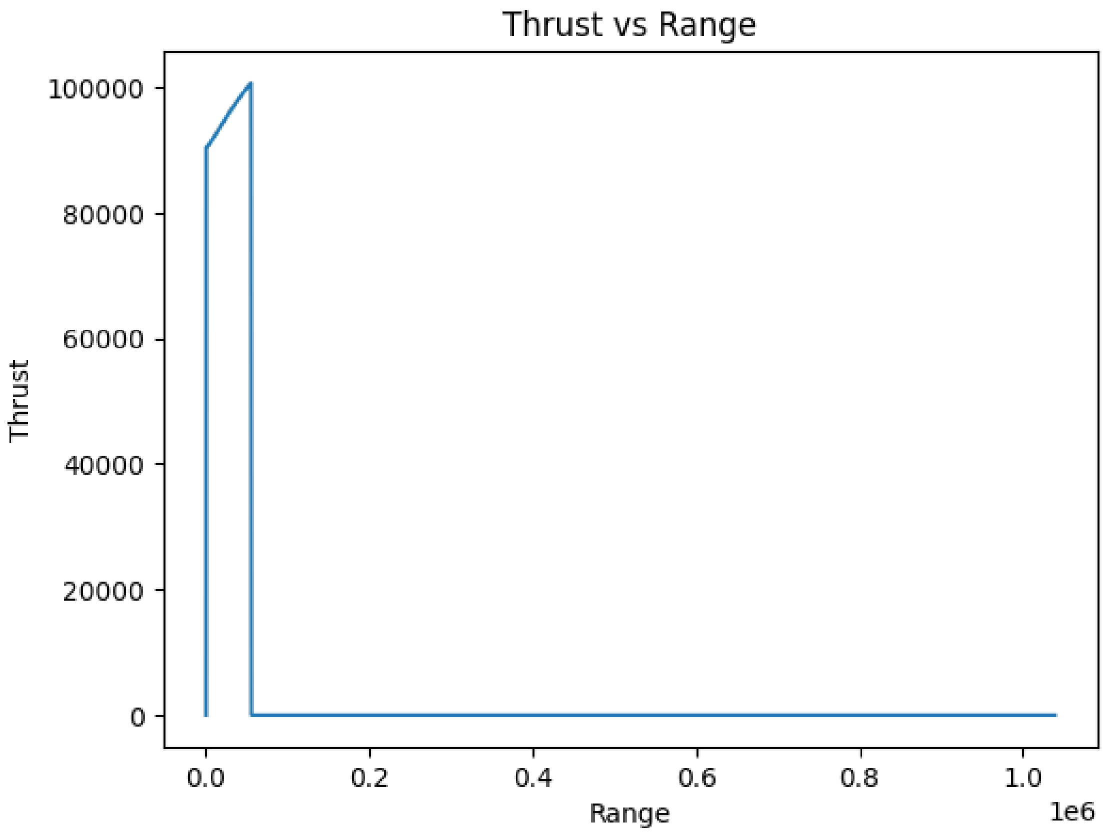


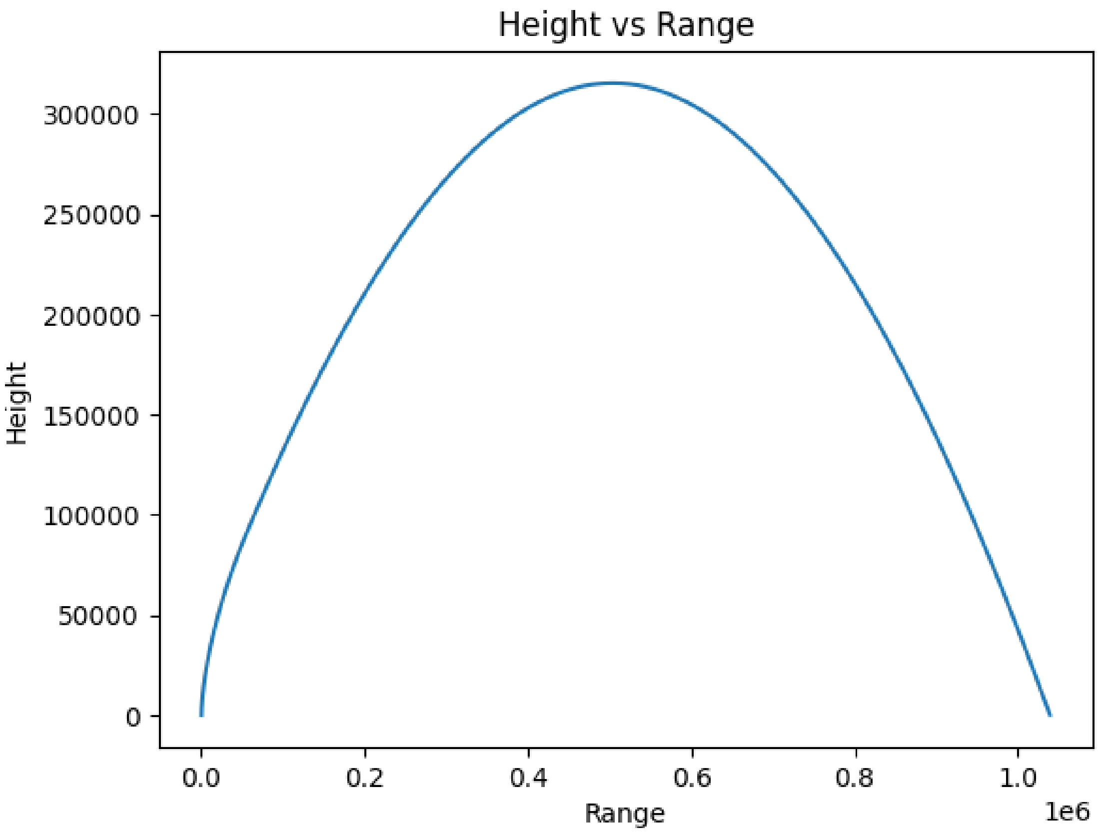

```
Data written to 'results/results_2.txt'

New simulation

fuelmass: 1289.4787352316134
Isp0: 666.2904956492962


Stage 1 burnout
Velocity (km/s):  2.846157313086886
Angle (deg h):  43.660083041406125
Range (km):  55.14622505917165
Time (sec):  93.59999999999896
Final results:
Range (km):  1039.241717666173
Apogee (km):  315.47440889796735
Time to target (sec):  598.4000000000675
```


Range vs Time
1e6
1.0
0.8
0.6
0.4
0.2
0.0
Range
0
100
200
300
400
500
600
Time

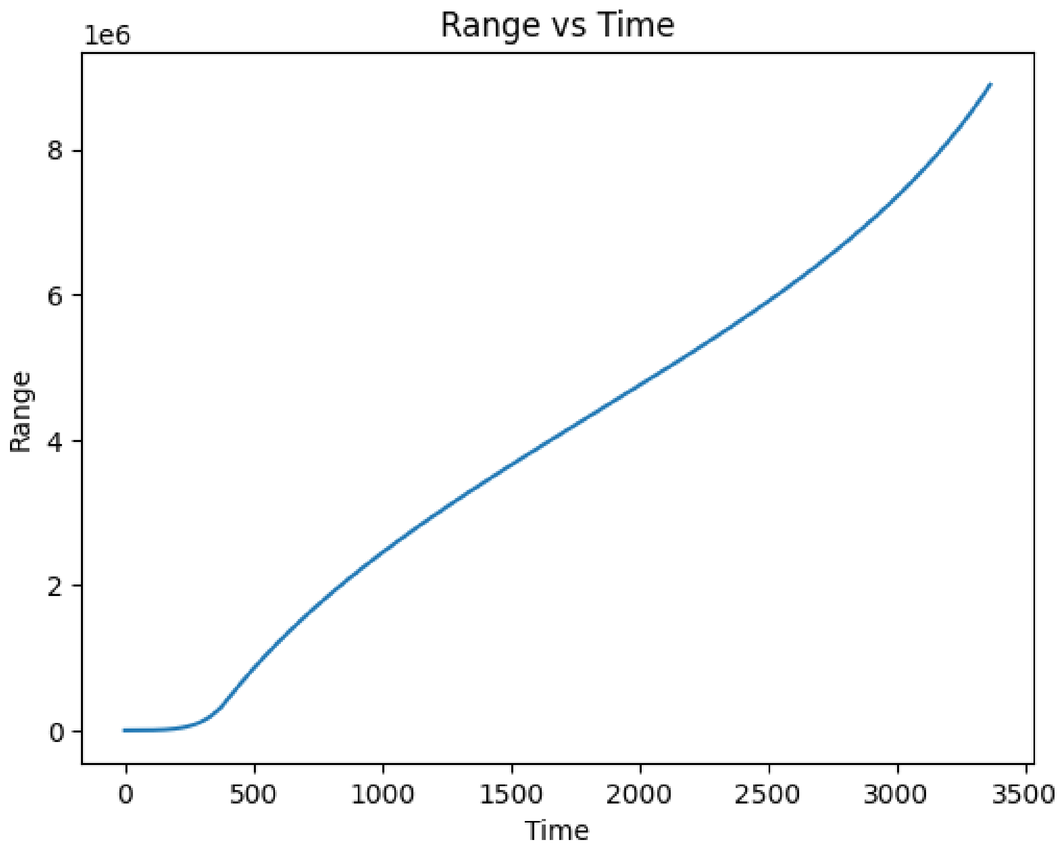

Height vs Time
Height
300000
250000
200000
150000
100000
50000
0
0
100
200
300
400
500
600
Time

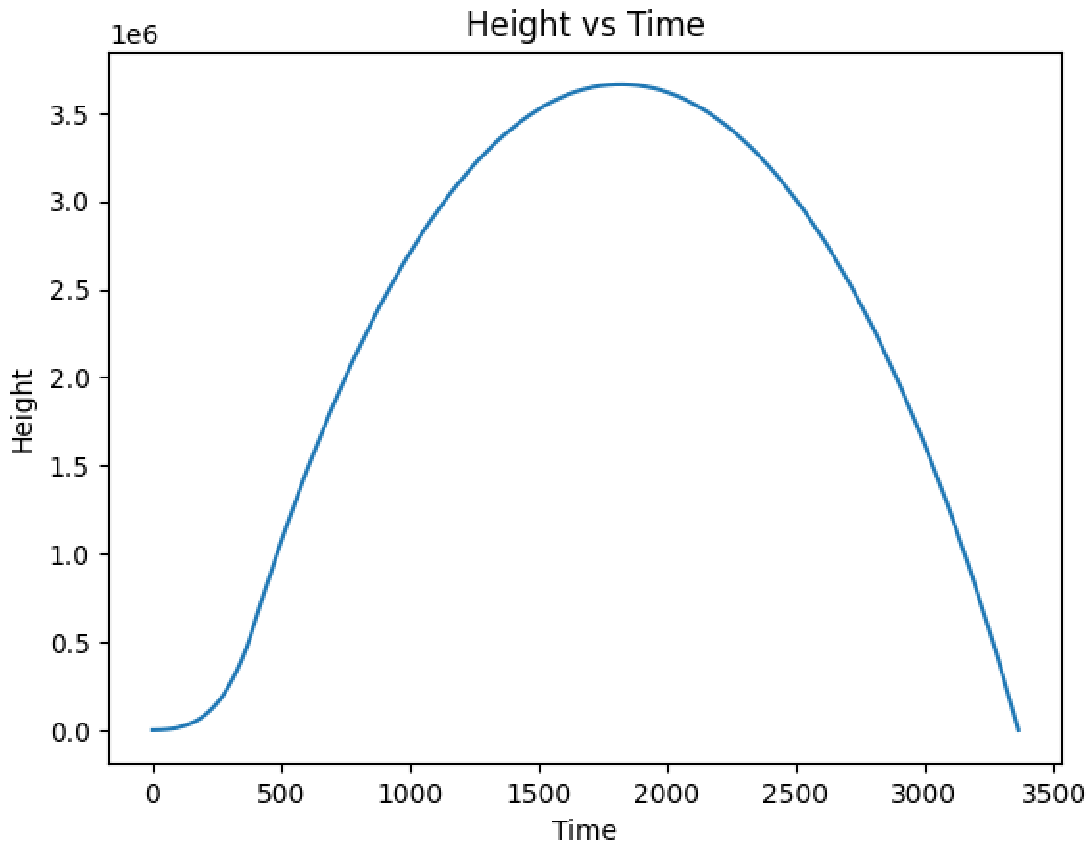

Gamma vs Time
1.5
1.0
0.5
0.0
−0.5
Gamma
0
100
200
300
400
500
600
Time

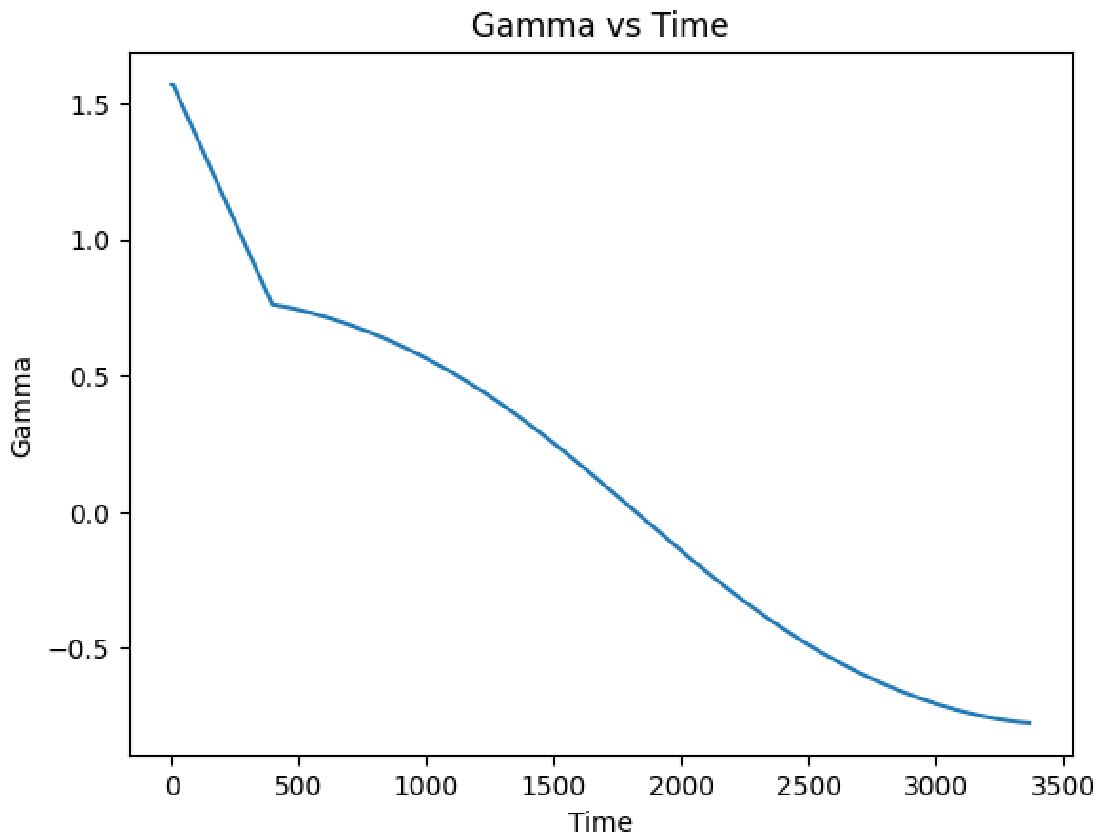

Thrust vs Range
100000
80000
60000
40000
20000
0
Thrust
0.0
0.2
0.4
0.6
0.8
1.0
1e6
Range

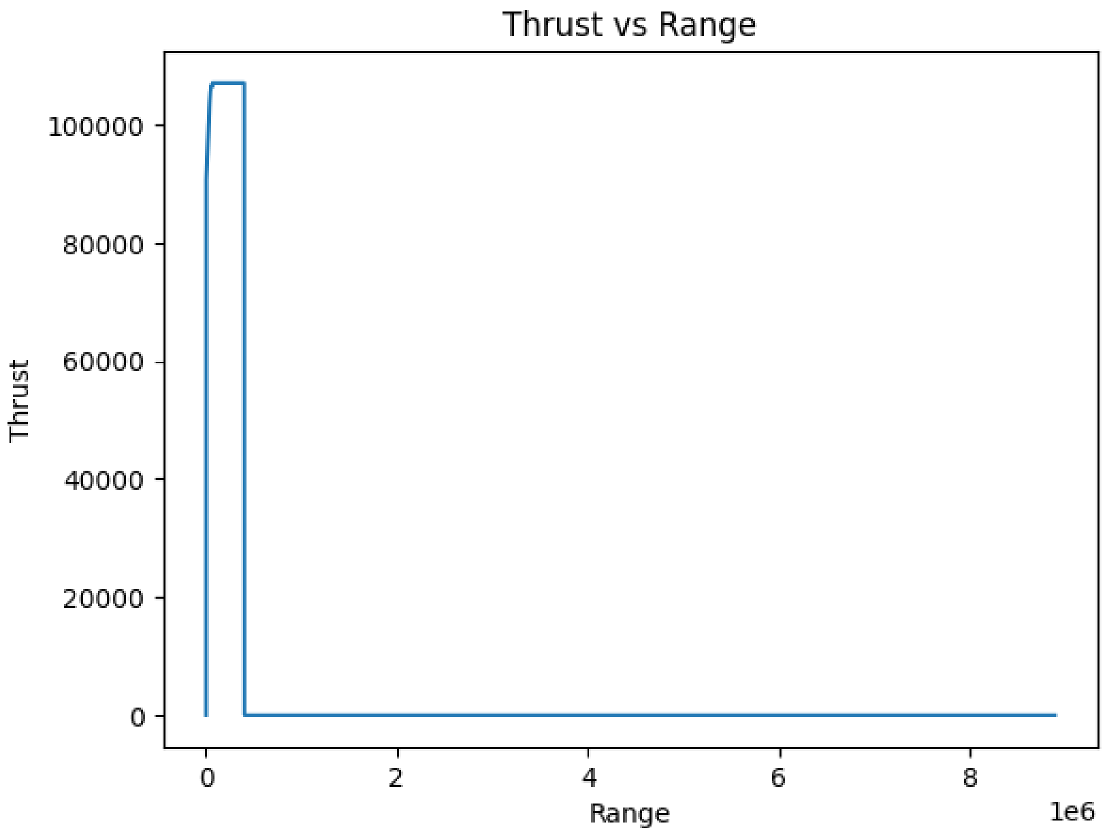


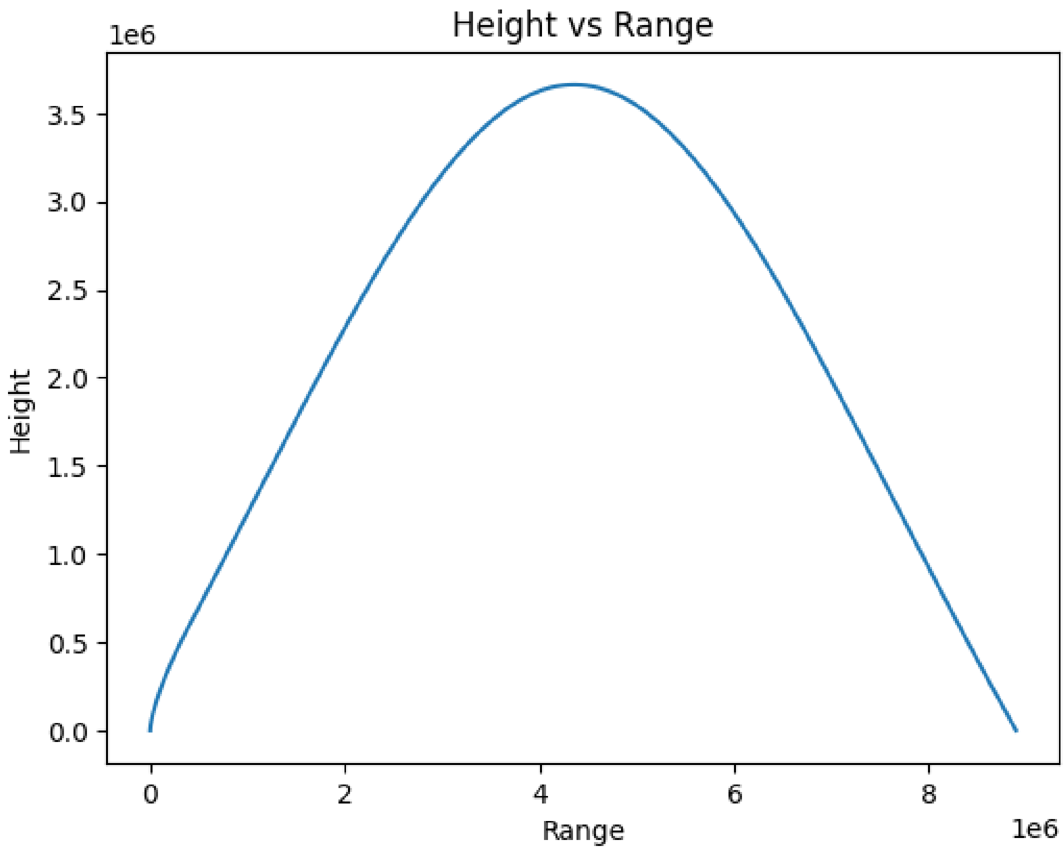

```
Data written to 'results/results_3.txt'

New simulation

fuelmass: 5884.749310520711
Isp0: 614.8107619645466


Stage 1 burnout
Velocity (km/s):  6.840928406755898
Angle (deg h):  43.65348191761386
Range (km):  400.33202594915537
Time (sec):  394.30000000002116
Final results:
Range (km):  8896.112873729066
Apogee (km):  3662.845779804255
Time to target (sec):  3364.1999999980358
```


Range vs Time
1e6
8
6
4
2
0
Range
0
500
1000
1500
2000
2500
3000
3500
Time

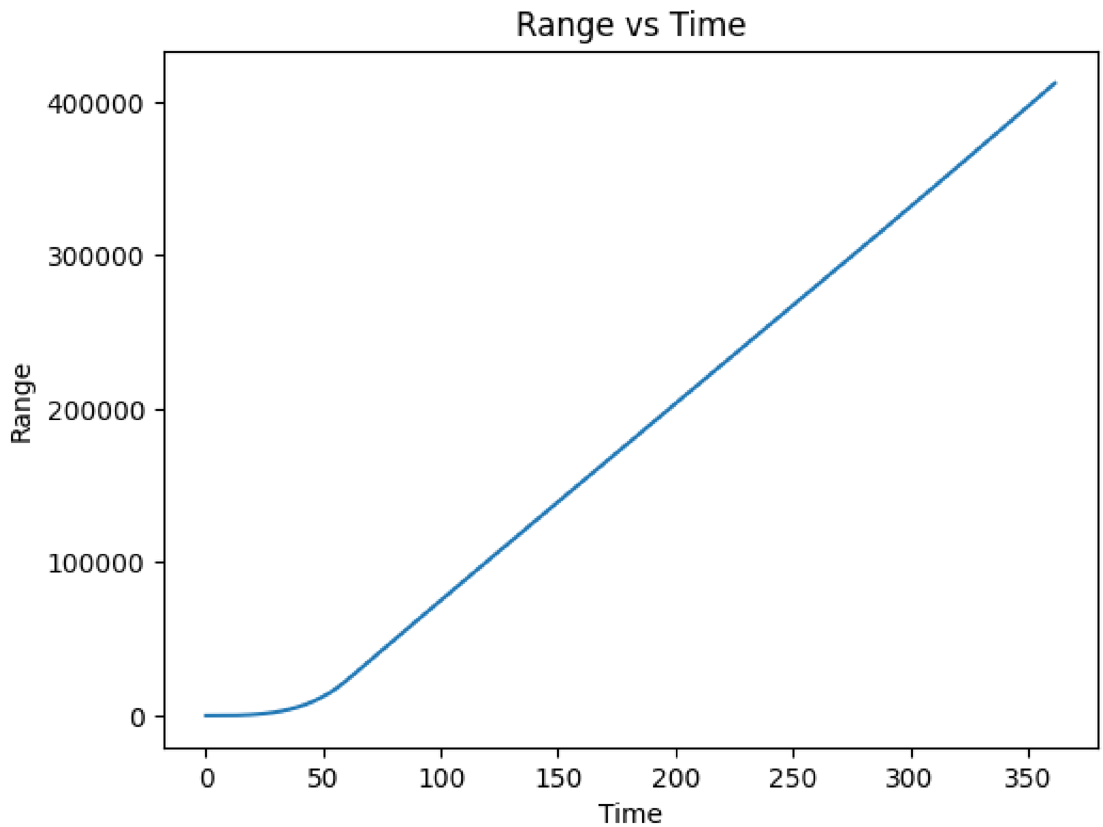

Height vs Time
1e6
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Height
0
500
1000
1500
2000
2500
3000
3500
Time

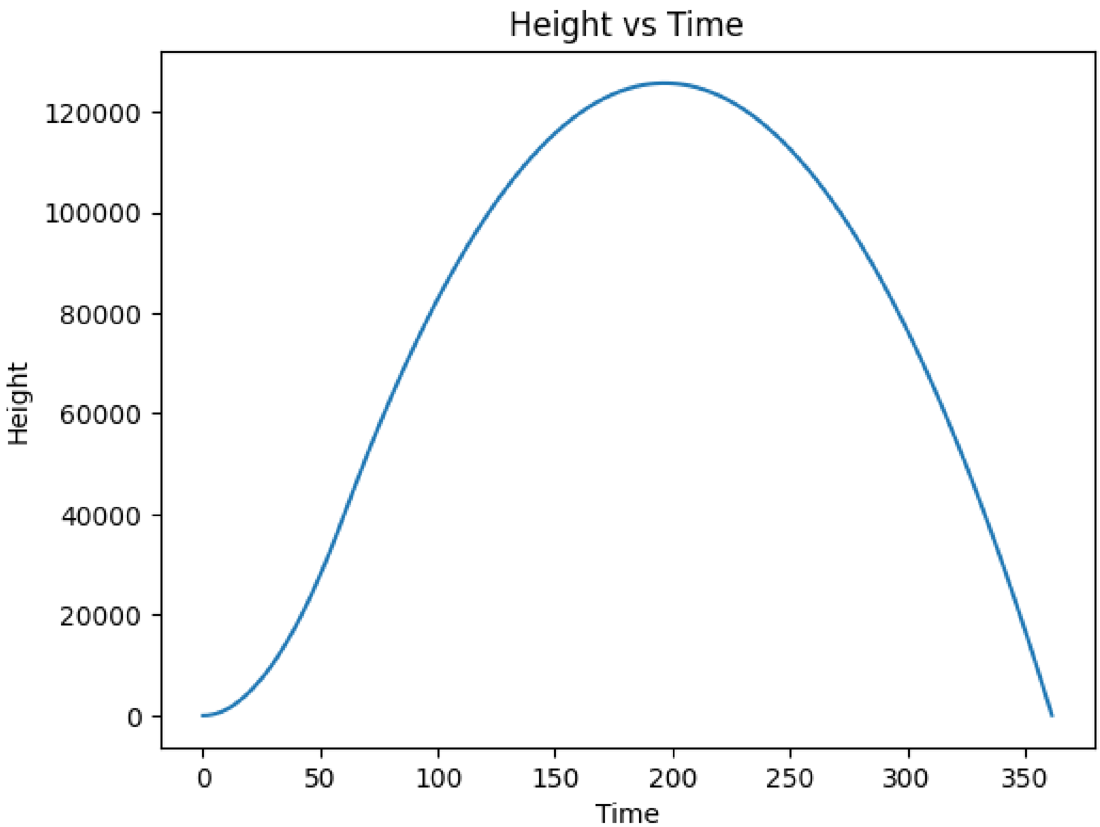

Gamma vs Time
1.5
1.0
0.5
0.0
−0.5
Gamma
0
500
1000
1500
2000
2500
3000
3500
Time

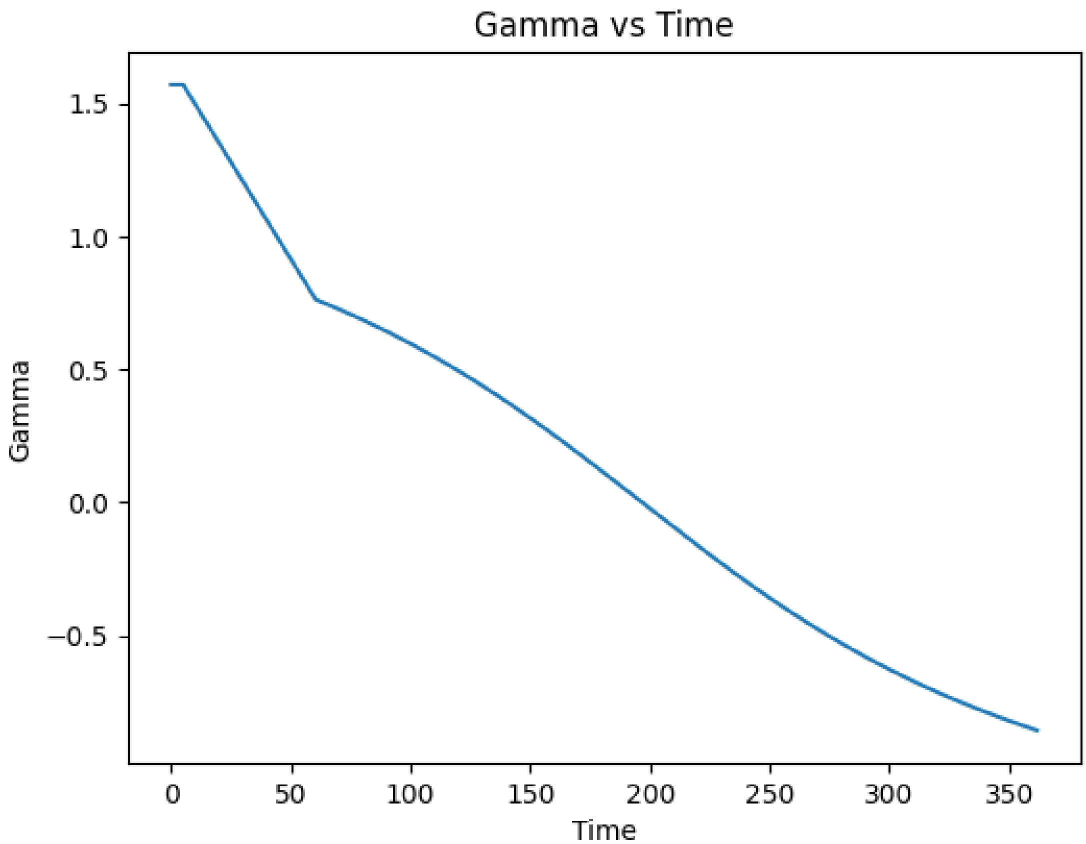

Thrust vs Range
Thrust
100000
80000
60000
40000
20000
0
0
2
4
6
8
Range
1e6

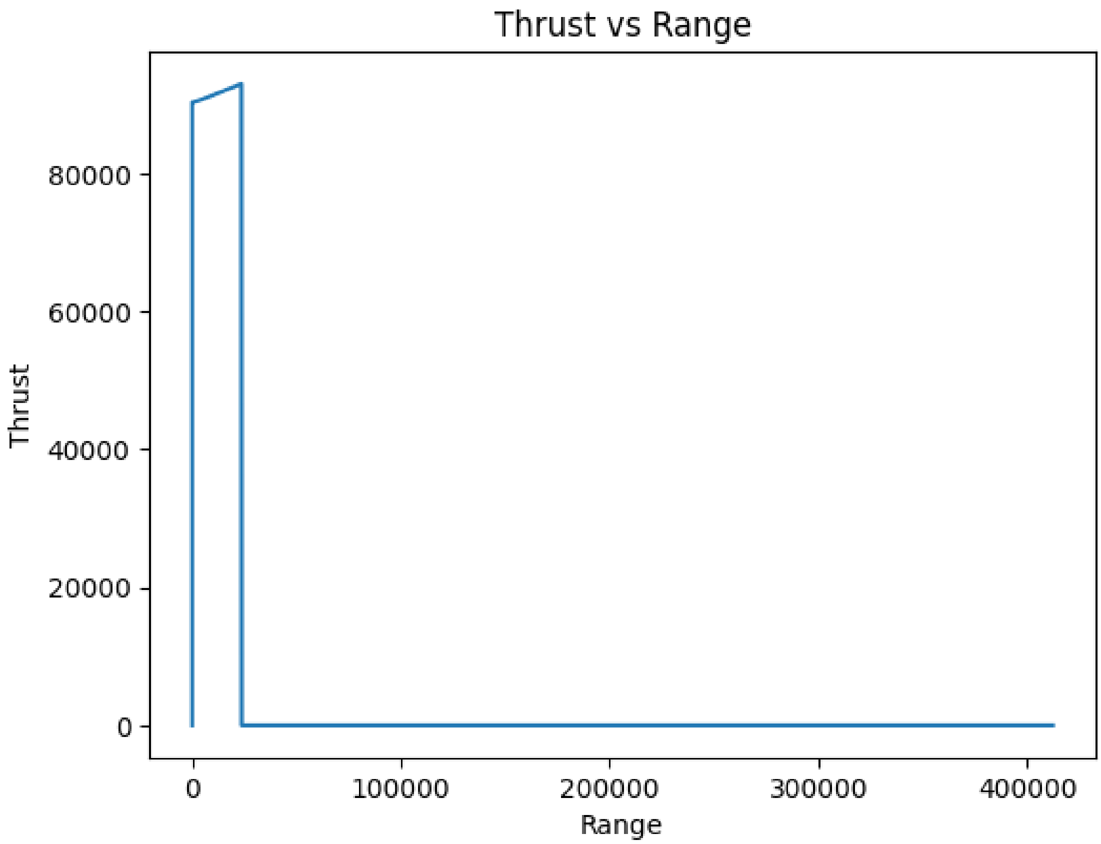



```
Data written to 'results/results_4.txt'

New simulation

fuelmass: 984.2424643743145
Isp0: 562.9484052008906


Stage 1 burnout
Velocity (km/s):  1.8239142293971027
Angle (deg h):  43.69201511792158
Range (km):  23.533022260837498
Time (sec):  60.40000000000059
Final results:
Range (km):  412.37779861564456
Apogee (km):  125.66449326051323
Time to target (sec):  361.70000000001374
```

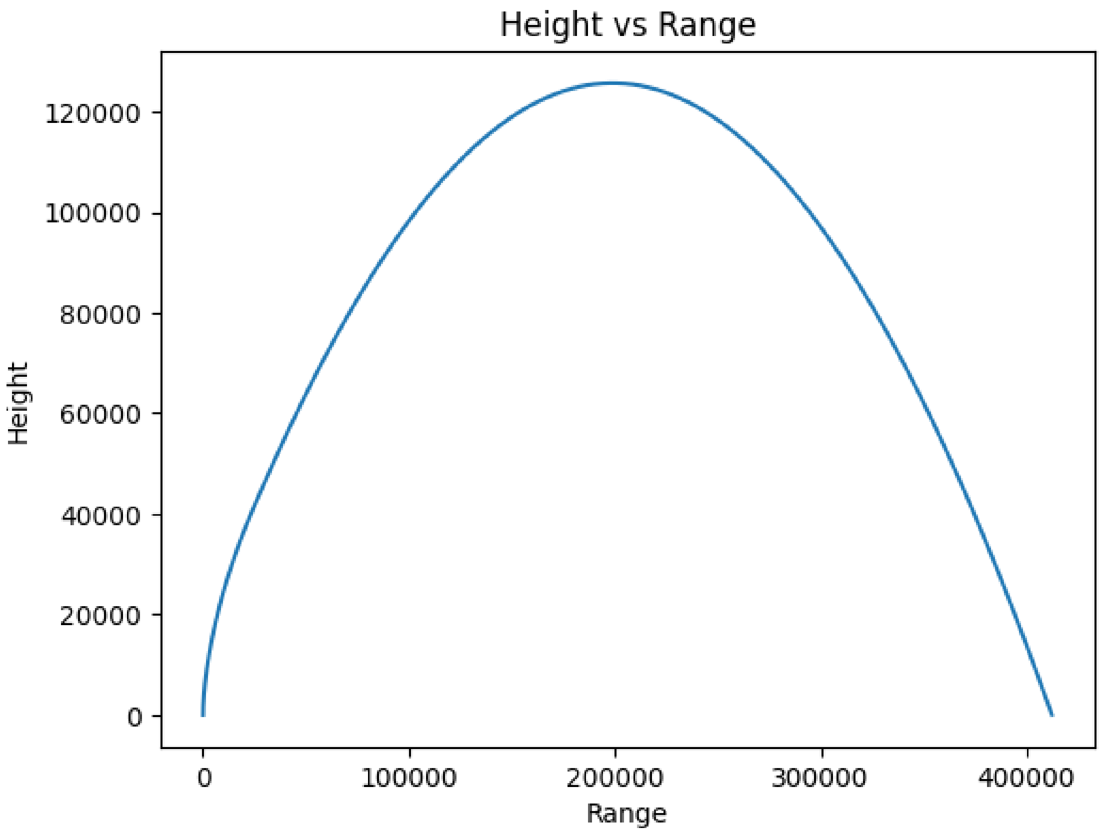

Range vs Time
Range
Time
400000
300000
200000
100000
0
0
50
100
150
200
250
300
350


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


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



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



Data written to 'results/results_5.txt'

[12]:
```
# Build model
m2_var_kernel = (100)**2
m2_lengthscale = 100 # 100 # 1
m2_var_noise = 1e-5 # small value

#kern = GPy.kern.RBF(input_dim=2, lengthscale=lengthscale, variance =var_kernel)  # ,␣
↪lengthscale=0.08, variance=20
# kern = GPy.kern.Matern32(input_dim=1)
# kern = GPy.kern.Linear(input_dim=1)

constrain_lengthscale = True

m2_rbf_kern = GPy.kern.RBF(input_dim=2, lengthscale=m2_lengthscale)
if constrain_lengthscale:
    m2_rbf_kern.lengthscale.constrain_bounded(m2_lengthscale, m2_lengthscale*1e12)

# m2_kern = m2_rbf_kern + \
#     GPy.kern.Linear(input_dim=2)
m2_kern = (GPy.kern.RBF(input_dim=2, lengthscale=500) * \
           GPy.kern.RBF(input_dim=2, lengthscale=100)) * \
    GPy.kern.Linear(input_dim=2)
# m2_kern = m2_rbf_kern
```

```
m2_model_gpy = GPRegression(m2_x,m2_y, kernel=m2_kern)
m2_model_gpy.kern.variance =  m2_var_kernel
m2_model_gpy.likelihood.variance.fix(m2_var_noise)

display(m2_model_gpy)
```

```
reconstraining parameters rbf.lengthscale
```

```
<GPy.models.gp_regression.GPRegression at 0x7fba29a8d2d0>
```

[13]:
```
# m2_model_gpy_opt = m2_model_gpy
# m2_model_gpy_opt.optimize()
# m2_model_gpy_opt.plot()
```

[14]:
```
m2_model_emukit = GPyModelWrapper(m2_model_gpy)
m2_model_emukit.optimize()
```

[15]:
```
display(m2_model_gpy)
```

```
<GPy.models.gp_regression.GPRegression at 0x7fba29a8d2d0>
```

[16]:
```
# Create data for plot
wirte_output_txt = False
nr_points_plot = 101
m2_param_1_x_plot = np.linspace(m2_space.parameters[0].min, m2_space.parameters[0].
 ↪max, nr_points_plot)[:, None]
m2_param_2_x_plot = np.linspace(m2_space.parameters[1].min, m2_space.parameters[1].
 ↪max, nr_points_plot)[:, None]
m2_x_plot_mesh, m2_y_plot_mesh = np.meshgrid(m2_param_1_x_plot, m2_param_2_x_plot)
m2_x_plot = np.array([m2_x_plot_mesh, m2_y_plot_mesh]).T.reshape(-1,2)


# TEMP read data from txt
# np.savetxt('test1.txt', a, fmt='%f')
# m2_y_plot = np.loadtxt('m2_y_plot_neg.txt', dtype=float)[:,None]

if run_grid_simulation:
    m2_y_plot = neg_run_missile_sim(m2_x_plot) # TAKES LONG TIME
    m2_Z = m2_y_plot.reshape(m2_x_plot_mesh.shape)
```

[17]:
```
m2_x
```

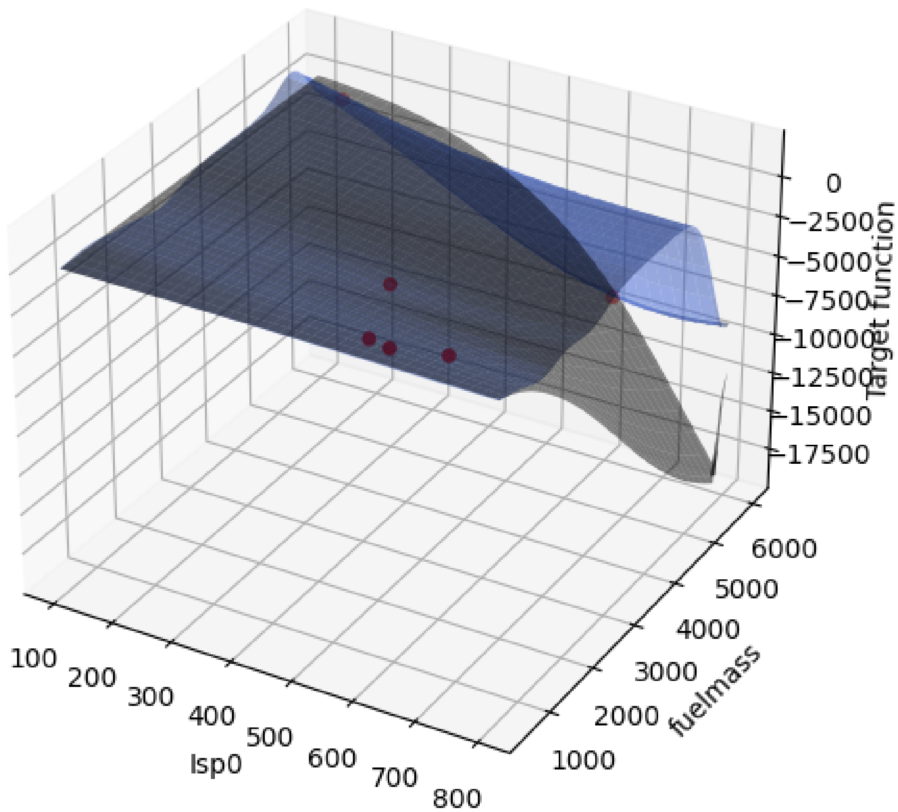
[17]:
```
array([[2461.69983073,  486.92597367],
       [ 920.14777038,  599.86268916],
       [5672.14884133,  167.5983593 ],
       [1289.47873523,  666.29049565],
       [5884.74931052,  614.81076196],
       [ 984.24246437,  562.9484052 ]])
```

[18]:
```
# Compute current prediction
m2_mu_plot_grid_pred1, var_plot_grid_pred1 = m2_model_emukit.predict(m2_x_plot)
m2_mu_plot_pred1 = m2_mu_plot_grid_pred1.reshape(m2_x_plot_mesh.shape)
m2_var_plot_pred1 = var_plot_grid_pred1.reshape(m2_x_plot_mesh.shape)
```

```
if run_grid_simulation:
    m2_rmse = evaluate_prediction(y_actual=m2_y_plot,␣
↪y_predicted=m2_mu_plot_grid_pred1)
    print("RMSE m2 (before experiment design loop): ", m2_rmse)
```

```
RMSE m2 (before experiment design loop):  3903.7185215433874
```

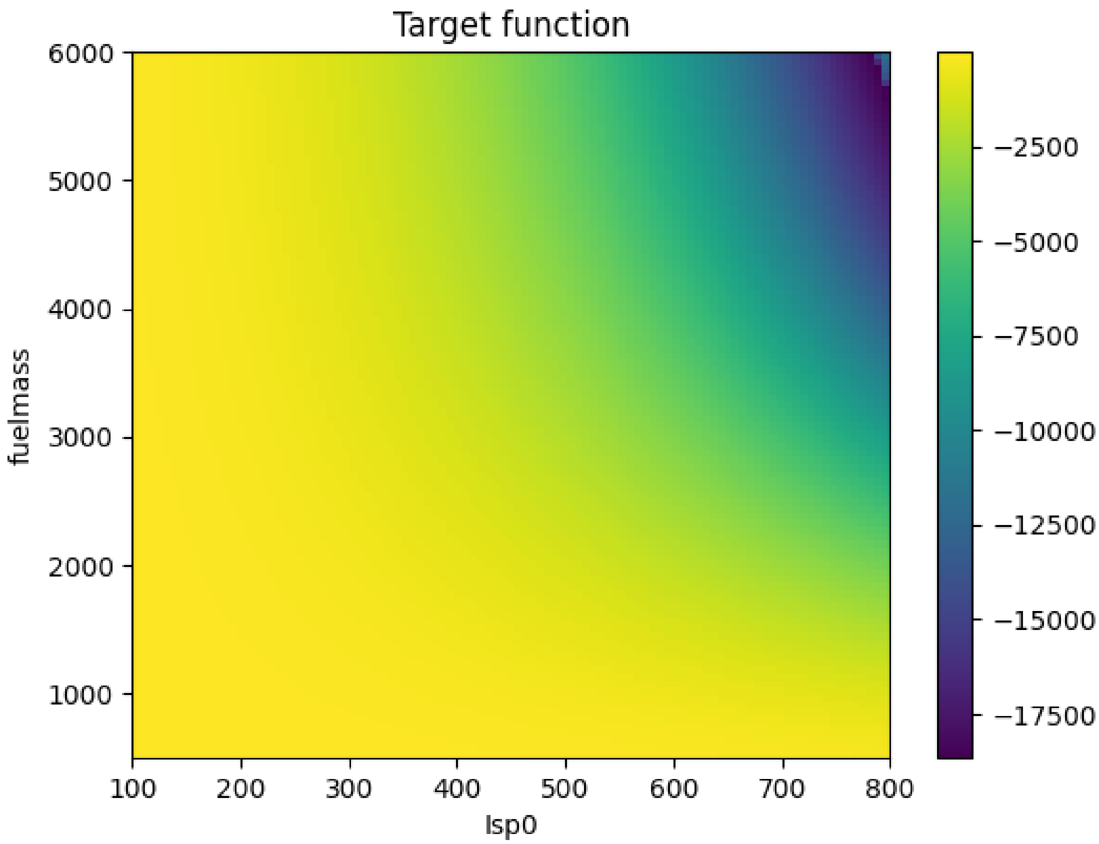
[19]:
```
if run_grid_simulation:
    # 3D Plot
    add_bands = False

    # REVERSE
    fig = plt.figure()
    ax = fig.add_subplot(projection='3d')

    # True surface
    surf = ax.plot_surface(m2_y_plot_mesh, m2_x_plot_mesh, (m2_Z).transpose(),
                           alpha = .5,
                           label='target function',
                           color='black'
                          )
    # Mean predicted
    surf = ax.plot_surface(m2_y_plot_mesh, m2_x_plot_mesh, (m2_mu_plot_pred1).
↪transpose(),
                           alpha = .5,
                           label='model', # Mean
                           color='royalblue'
                          )
    # True points observed
    ax.scatter(m2_x[:,1], m2_x[:,0], m2_y, marker='o', color='red')

    ax.set_xlabel(m2_param_2)
    ax.set_ylabel(m2_param_1)
    ax.set_zlabel('Target function')
    plt.tight_layout()
```



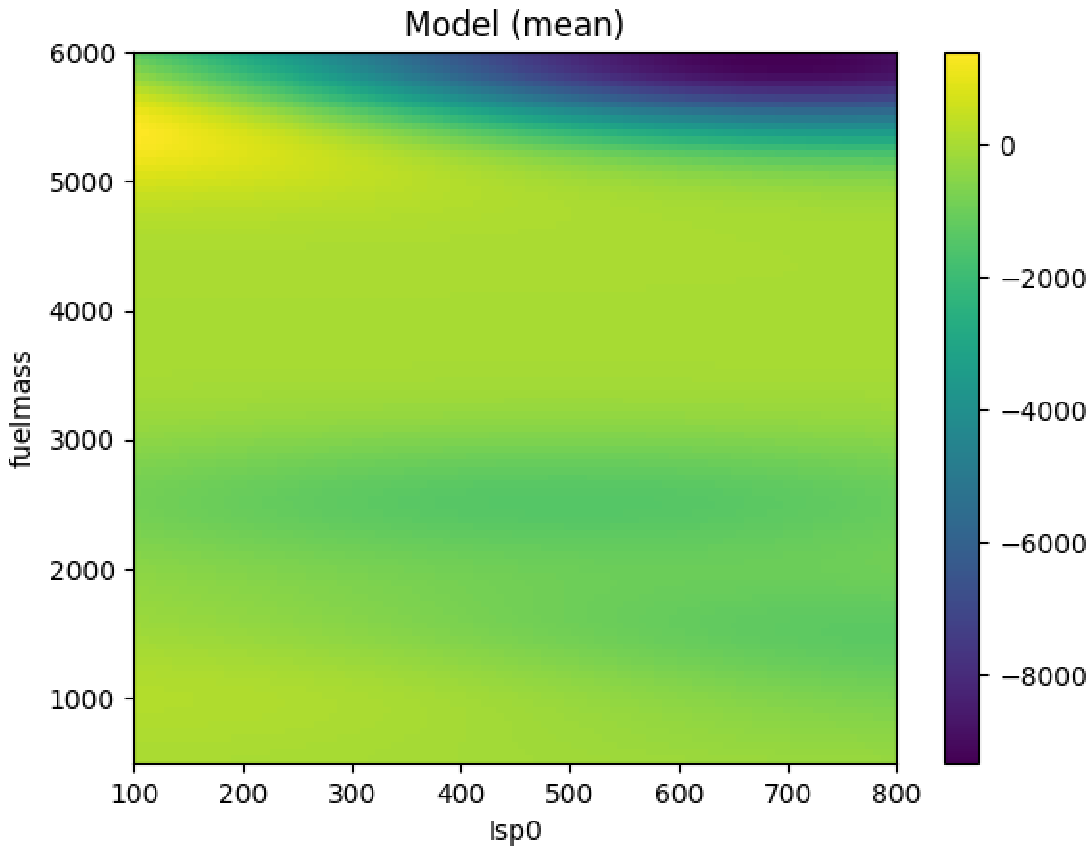
[20]:
```
if run_grid_simulation:
    from matplotlib import colors
    divnorm=colors.TwoSlopeNorm(vcenter=0.) # vmin=-5., vcenter=0., vmax=10

    ## Heatmaps
    extents = [m2_space.parameters[1].min, m2_space.parameters[1].max,
               m2_space.parameters[0].min, m2_space.parameters[0].max]

    # True values
    fig, ax = plt.subplots()
    im = ax.imshow(m2_Z, extent=extents, aspect='auto',  origin='lower')
    ax.set_title('Target function')
    ax.set_xlabel(m2_param_2)
    ax.set_ylabel(m2_param_1)
    fig.colorbar(im, ax=ax)
    fig.show()

    # Model
    fig, ax = plt.subplots()
    im = ax.imshow(m2_mu_plot_pred1, extent=extents, aspect='auto', origin='lower')
    ax.set_title('Model (mean)')
```

```
    ax.set_xlabel(m2_param_2)
    ax.set_ylabel(m2_param_1)
    fig.colorbar(im, ax=ax)
    fig.show()


    # Difference
    fig, ax = plt.subplots()
    vmin = (m2_mu_plot_pred1-m2_Z).min()
    vmax = (m2_mu_plot_pred1-m2_Z).max()
    vmin_max = max(abs(vmin), abs(vmax))
    divnorm=colors.TwoSlopeNorm(vcenter=0., vmin=-vmin_max, vmax=vmin_max) # vmin=-5.,␣
↪vcenter=0., vmax=10
    im = ax.imshow(m2_mu_plot_pred1-m2_Z, extent=extents, aspect='auto', cmap="bwr",␣
↪norm=divnorm, origin='lower')
    ax.set_title('Difference between model and target function')
    ax.set_xlabel(m2_param_2)
    ax.set_ylabel(m2_param_1)
    # Add points where simulation evaluated
    ax.plot(m2_x[:,1], m2_x[:,0], 'ro')
    fig.colorbar(im, ax=ax)
    fig.show()
```

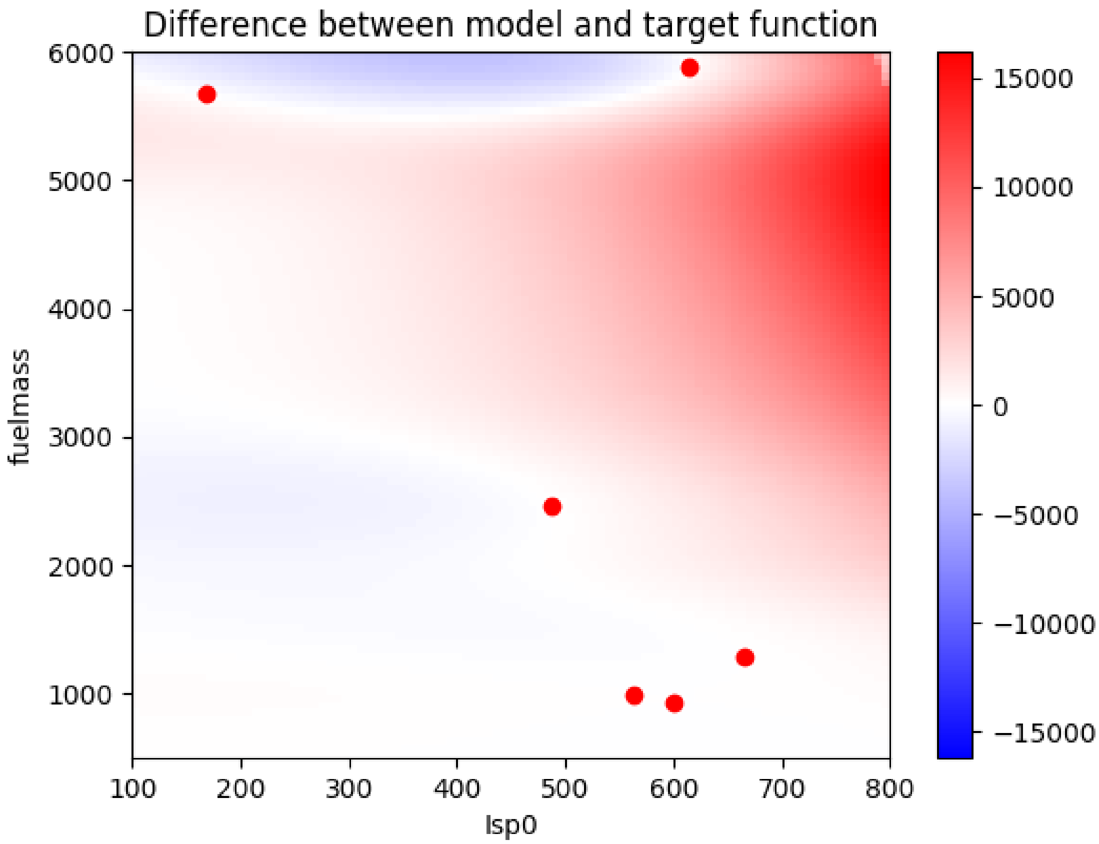
```
/var/folders/98/fv11ygzs4p51c21s8jln0jzc0000gn/T/ipykernel_12821/375354680.py:1
6: UserWarning:Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
/var/folders/98/fv11ygzs4p51c21s8jln0jzc0000gn/T/ipykernel_12821/375354680.py:2
5: UserWarning:Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
/var/folders/98/fv11ygzs4p51c21s8jln0jzc0000gn/T/ipykernel_12821/375354680.py:4
1: UserWarning:Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
```



Target function
6000
5000
4000
3000
2000
1000
fuelmass
100
200
300
400
500
600
700
800
Isp0
−2500
−5000
−7500
−10000
−12500
−15000
−17500



Model (mean)
6000
5000
4000
3000
2000
1000
fuelmass
100
200
300
400
500
600
700
800
Isp0
0
−2000
−4000
−6000
−8000



[21]:
```
m2_x
```

[21]:
```
array([[2461.69983073,  486.92597367],
       [ 920.14777038,  599.86268916],
       [5672.14884133,  167.5983593 ],
       [1289.47873523,  666.29049565],
       [5884.74931052,  614.81076196],
       [ 984.24246437,  562.9484052 ]])
```

1.1.2 Use the model created for model-based experimental design

use the model to decide which are the best points to collect using some data collection criteria (acquisition function).

[22]:
```
from emukit.experimental_design.experimental_design_loop import ExperimentalDesignLoop
from emukit.experimental_design.acquisitions import IntegratedVarianceReduction,␣
↪ModelVariance
```

[23]:
```
# help(ExperimentalDesignLoop)
```

[24]:
```
m2_2_model_emukit = m2_model_emukit
```

[25]:
```python
wirte_output_txt = False

integrated_variance = IntegratedVarianceReduction(space=m2_space,
                                                   model=m2_2_model_emukit)
m2_ed = ExperimentalDesignLoop(space=m2_space,
                              model=m2_2_model_emukit,
                              acquisition = integrated_variance,
                              batch_size = 1)

m2_ed.run_loop(user_function=neg_run_missile_sim, stopping_condition=10*2)
```

```
New simulation

fuelmass: 4878.682066574241
Isp0: 461.4762727822375


Stage 1 burnout
Velocity (km/s):  4.60441860679277
Angle (deg h):  43.654584270916445
Range (km):  180.9857845303747
Time (sec):  245.29999999999035
Final results:
Range (km):  3158.6045364931383
Apogee (km):  1024.5547134484984
Time to target (sec):  1305.599999999908



New simulation

fuelmass: 4063.558783485642
Isp0: 453.93151760855153


Stage 1 burnout
Velocity (km/s):  4.148526147315459
Angle (deg h):  43.6658239525756
Range (km):  141.29851062716125
Time (sec):  200.99999999999287
Final results:
Range (km):  2455.4076192902166
Apogee (km):  780.1861276418522
Time to target (sec):  1080.5000000001128



New simulation

fuelmass: 3359.559699489315
Isp0: 454.40837486922806
```

```
Stage 1 burnout
Velocity (km/s):  3.7347111955387606
Angle (deg h):  43.65568417600361
Range (km):  110.61947219954926
Time (sec):  166.39999999999483
Final results:
Range (km):  1921.834343633677
Apogee (km):  601.3264485166641
Time to target (sec):  908.8000000001381



New simulation

fuelmass: 5294.398743759096
Isp0: 539.4919760902346


Stage 1 burnout
Velocity (km/s):  5.695852002695544
Angle (deg h):  43.652577078192486
Range (km):  275.74542902940306
Time (sec):  311.3000000000023
Final results:
Range (km):  5391.4508728134
Apogee (km):  1883.2069889190313
Time to target (sec):  1993.6999999992822



New simulation

fuelmass: 3408.998763550249
Isp0: 415.6613997100321


Stage 1 burnout
Velocity (km/s):  3.3727624303170067
Angle (deg h):  43.67239006023836
Range (km):  92.78530499827153
Time (sec):  154.39999999999552
Final results:
Range (km):  1536.594448949843
Apogee (km):  478.9712769531256
Time to target (sec):  796.4000000001125


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

New simulation
```

```
fuelmass: 2896.2379759805713
Isp0: 331.12115826598097


Stage 1 burnout
Velocity (km/s):  2.302811097766833
Angle (deg h):  43.67993464313428
Range (km):  45.242701936516895
Time (sec):  104.49999999999834
Final results:
Range (km):  683.8659833643062
Apogee (km):  212.4717865973708
Time to target (sec):  502.00000000004565



New simulation

fuelmass: 4302.336283094621
Isp0: 109.29918212473977


Stage 1 burnout
Velocity (km/s):  0.8040715537876274
Angle (deg h):  43.64002781579093
Range (km):  7.515094916182927
Time (sec):  51.30000000000046
Final results:
Range (km):  84.6779995910292
Apogee (km):  27.832012844593667
Time to target (sec):  184.4999999999938



New simulation

fuelmass: 5477.659772906554
Isp0: 572.3713751105622


Stage 1 burnout
Velocity (km/s):  6.165465906582221
Angle (deg h):  43.652665713593436
Range (km):  322.98278611907114
Time (sec):  341.7000000000092
Final results:
Range (km):  6665.66385236041
Apogee (km):  2454.426413741662
Time to target (sec):  2435.2999999988806


 /Users/ilariasartori/opt/anaconda3/envs/mlphysical/lib/python3.10/site-
packages/GPy/kern/src/stationary.py:168: RuntimeWarning:overflow encountered in
```

```
divide
 /Users/ilariasartori/opt/anaconda3/envs/mlphysical/lib/python3.10/site-
packages/GPy/kern/src/rbf.py:76: RuntimeWarning:invalid value encountered in
multiply

New simulation

fuelmass: 2260.443240465954
Isp0: 163.12095340257792


Stage 1 burnout
Velocity (km/s):  0.8696399980272381
Angle (deg h):  43.70693234408902
Range (km):  7.148694549839851
Time (sec):  40.2000000000003
Final results:
Range (km):  96.08810747152313
Apogee (km):  30.95470331963349
Time to target (sec):  182.39999999999392



New simulation

fuelmass: 4858.534711073726
Isp0: 690.8678746429392


Stage 1 burnout
Velocity (km/s):  7.282476943800215
Angle (deg h):  43.65652882926793
Range (km):  423.61641898494685
Time (sec):  365.8000000000147
Final results:
Range (km):  10508.38724670863
Apogee (km):  4779.527743628276
Time to target (sec):  4161.699999997609



New simulation

fuelmass: 5201.95186360266
Isp0: 182.18562042460977


Stage 1 burnout
Velocity (km/s):  1.543963902919711
Angle (deg h):  43.657267408173055
Range (km):  26.244727751061593
Time (sec):  103.2999999999984
Final results:
```

```
Range (km):  311.52260811939465
Apogee (km):  99.66079834232968
Time to target (sec):  362.900000000014



New simulation

fuelmass: 4382.367256166273
Isp0: 583.5270161047214


Stage 1 burnout
Velocity (km/s):  5.759430979309827
Angle (deg h):  43.65557522375997
Range (km):  265.62123014250244
Time (sec):  278.6999999999949
Final results:
Range (km):  5508.384487262774
Apogee (km):  1928.2628851721356
Time to target (sec):  1997.4999999992788



New simulation

fuelmass: 3798.4939118156003
Isp0: 669.0055434316113


Stage 1 burnout
Velocity (km/s):  6.269029188826729
Angle (deg h):  43.665853106098794
Range (km):  297.9843220169598
Time (sec):  276.89999999999446
Final results:
Range (km):  6867.107278116062
Apogee (km):  2542.7983191009175
Time to target (sec):  2442.699999998874



New simulation

fuelmass: 1663.9494898800779
Isp0: 321.15414775099043


Stage 1 burnout
Velocity (km/s):  1.4861658425338886
Angle (deg h):  43.649248537620174
Range (km):  17.901050240157193
Time (sec):  58.30000000000056
Final results:
```

```
Range (km):  276.50437057575783
Apogee (km):  85.86155110888325
Time to target (sec):  302.30000000000024



New simulation

fuelmass: 2829.5869516857406
Isp0: 769.6565317945232


Stage 1 burnout
Velocity (km/s):  6.215772159933899
Angle (deg h):  43.651089743905366
Range (km):  269.7946874779557
Time (sec):  237.2999999999908
Final results:
Range (km):  6631.344926032913
Apogee (km):  2416.697072218072
Time to target (sec):  2319.399999998986



New simulation

fuelmass: 1961.3797131591502
Isp0: 697.6850215516242


Stage 1 burnout
Velocity (km/s):  4.306158919270915
Angle (deg h):  43.65343099905379
Range (km):  124.9228096064081
Time (sec):  149.09999999999582
Final results:
Range (km):  2612.527347830653
Apogee (km):  818.6171338226619
Time to target (sec):  1067.8000000001243



New simulation

fuelmass: 5881.813300113165
Isp0: 503.30838012419605


Stage 1 burnout
Velocity (km/s):  5.448519541845869
Angle (deg h):  43.65790950484454
Range (km):  262.1433621509185
Time (sec):  322.60000000000485
Final results:
```

```
Range (km):  4823.478410005139
Apogee (km):  1651.7654218953978
Time to target (sec):  1836.2999999994254



New simulation

fuelmass: 1007.2053326384988
Isp0: 198.50752896674112


Stage 1 burnout
Velocity (km/s):  0.6209856645291464
Angle (deg h):  43.82293544970517
Range (km):  2.8409097063050073
Time (sec):  21.800000000000043
Final results:
Range (km):  47.441415598430595
Apogee (km):  15.043319913360616
Time to target (sec):  121.59999999999737



New simulation

fuelmass: 5830.769420256684
Isp0: 183.36895827670898


Stage 1 burnout
Velocity (km/s):  1.6151947273442484
Angle (deg h):  43.65148588966079
Range (km):  29.552128447297665
Time (sec):  116.49999999999766
Final results:
Range (km):  342.6933346377176
Apogee (km):  109.58406795928896
Time to target (sec):  389.30000000002



New simulation

fuelmass: 770.2349778825693
Isp0: 251.91851421042284


Stage 1 burnout
Velocity (km/s):  0.6388740947266406
Angle (deg h):  43.710547909155665
Range (km):  2.8761548020949204
Time (sec):  21.200000000000035
Final results:
```

```
Range (km):  49.94744211562406
Apogee (km):  15.707653805022598
Time to target (sec):  123.39999999999726
```

[26]:
```python
m2_2_model_emukit.X.shape
```

[26]:
```
(26, 2)
```

[27]:
```python
m2_2_model_emukit.__dict__
```

[27]:
```
{'model': <GPy.models.gp_regression.GPRegression at 0x7fba29a8d2d0>,
 'n_restarts': 1}
```

[28]:
```python
m2_ed.__dict__
```

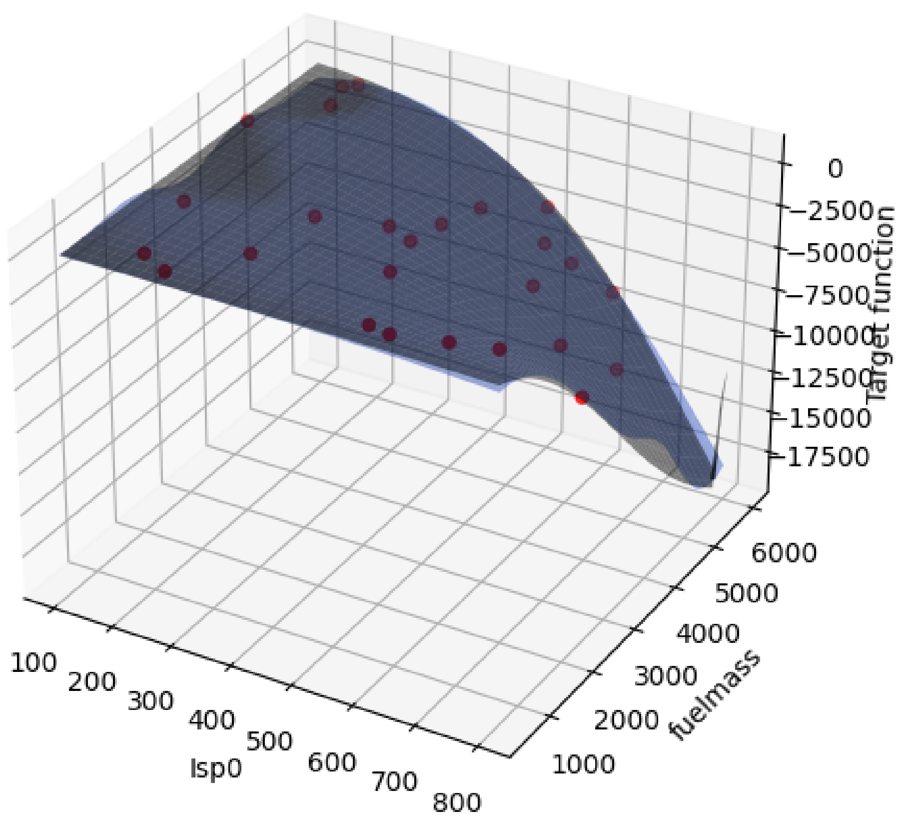
[28]:
```
{'candidate_point_calculator':
<emukit.core.loop.candidate_point_calculators.SequentialPointCalculator at
0x7fba588c0040>,
 'model_updaters': [<emukit.core.loop.model_updaters.FixedIntervalUpdater at
0x7fba48e45ea0>],
 'loop_state': <emukit.core.loop.loop_state.LoopState at 0x7fba48e45db0>,
 'loop_start_event': Event([]),
 'iteration_end_event': Event([]),
 'model': <emukit.model_wrappers.gpy_model_wrappers.GPyModelWrapper at
0x7fba29a8c3d0>}
```

[29]:
```python
if run_grid_simulation:
    # Compute new prediction
    m2_mu_plot_grid_pred2, var_plot_grid_pred2 = m2_2_model_emukit.predict(m2_x_plot)
    m2_mu_plot_pred2 = m2_mu_plot_grid_pred2.reshape(m2_x_plot_mesh.shape)
    m2_var_plot_pred2 = var_plot_grid_pred2.reshape(m2_x_plot_mesh.shape)

    m2_2_rmse = evaluate_prediction(y_actual=m2_y_plot,␣
 ↪y_predicted=m2_mu_plot_grid_pred2)
    print("RMSE m2 (post experiment design loop): ", m2_2_rmse)
```

```
RMSE m2 (post experiment design loop):  295.0875402186592
```

[30]:
```python
if run_grid_simulation:
    # 3D Plot
    add_bands = False


    # REVERSE
    fig = plt.figure()
    ax = fig.add_subplot(projection='3d')

    # True surface
    surf = ax.plot_surface(m2_y_plot_mesh, m2_x_plot_mesh, (m2_Z).transpose(),
                           alpha = .5,
                           label='target function',
                           color='black'
```

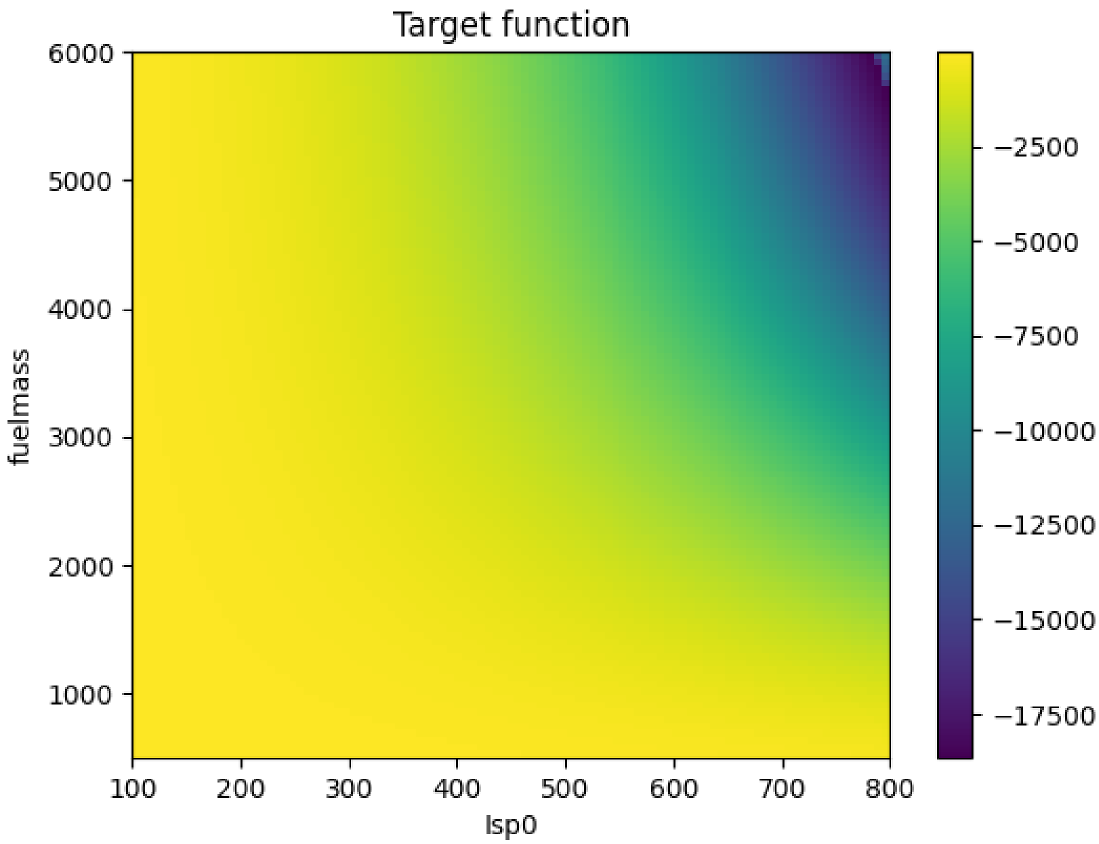
```
                          )
    # Mean predicted
    surf = ax.plot_surface(m2_y_plot_mesh, m2_x_plot_mesh, (m2_mu_plot_pred2).
↪transpose(),
                           alpha = .5,
                           label='model', # Mean
                           color='royalblue'
                          )
    # True points observed
    ax.scatter(np.array(m2_2_model_emukit.X)[:,1],
               np.array(m2_2_model_emukit.X)[:,0], m2_2_model_emukit.Y, marker='o',␣
↪color='red')

    ax.set_xlabel(m2_param_2)
    ax.set_ylabel(m2_param_1)
    ax.set_zlabel('Target function')
    plt.tight_layout()
```

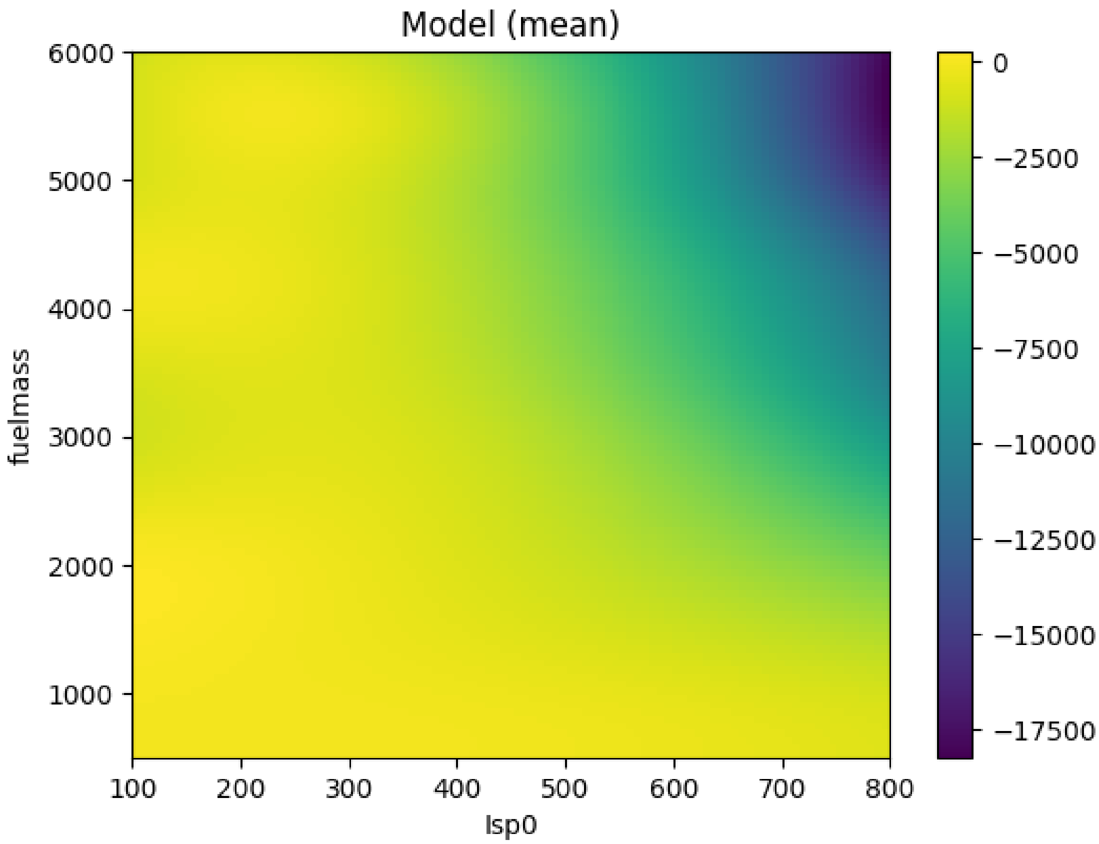


[31]:
```
if run_grid_simulation:

    # REVERSE
```

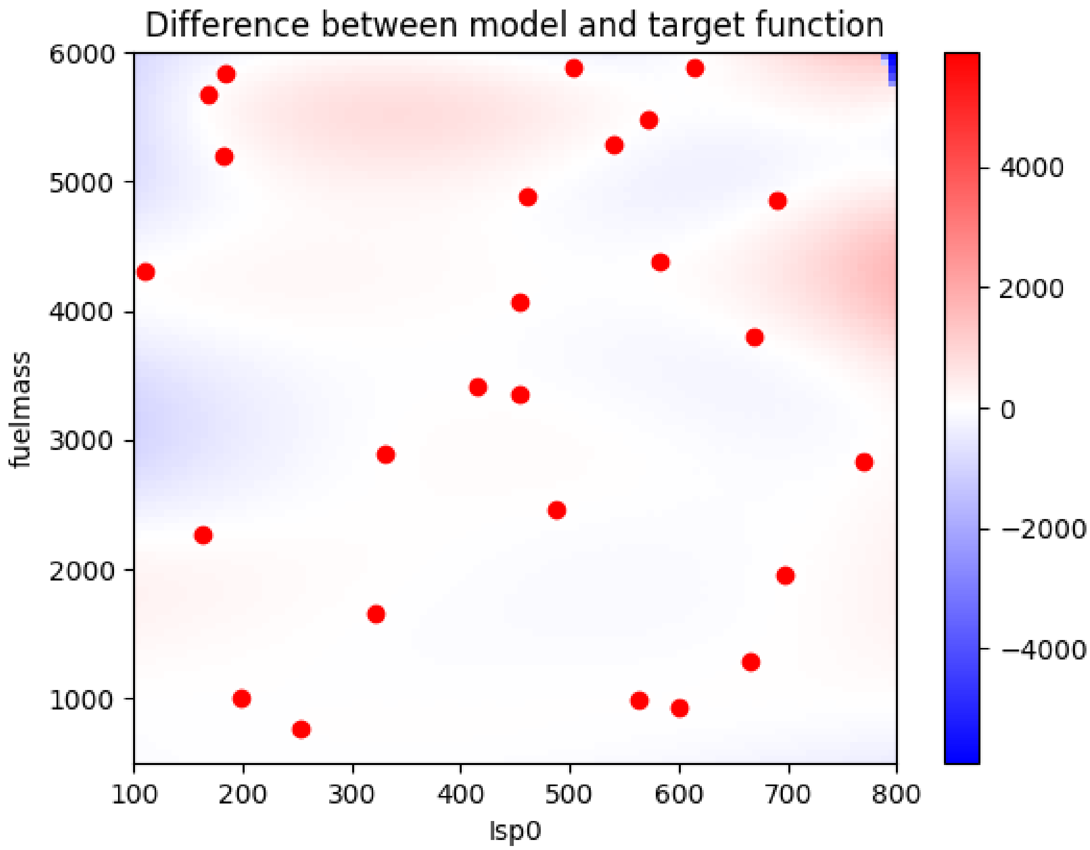
```python
## Heatmaps
extents = [m2_space.parameters[1].min, m2_space.parameters[1].max,
           m2_space.parameters[0].min, m2_space.parameters[0].max]

# True values
fig, ax = plt.subplots()
im = ax.imshow(m2_Z, extent=extents, aspect='auto', origin='lower')
ax.set_title('Target function')
ax.set_xlabel(m2_param_2)
ax.set_ylabel(m2_param_1)
fig.colorbar(im, ax=ax)
fig.show()

# Model
fig, ax = plt.subplots()
im = ax.imshow(m2_mu_plot_pred2, extent=extents, aspect='auto', origin='lower')
ax.set_title('Model (mean)')
ax.set_xlabel(m2_param_2)
ax.set_ylabel(m2_param_1)
fig.colorbar(im, ax=ax)
fig.show()


# Difference
fig, ax = plt.subplots()
vmin = (m2_mu_plot_pred2-m2_Z).min()
vmax = (m2_mu_plot_pred2-m2_Z).max()
vmin_max = max(abs(vmin), abs(vmax))
divnorm=colors.TwoSlopeNorm(vcenter=0., vmin=-vmin_max, vmax=vmin_max) # vmin=-5.,␣
↪vcenter=0., vmax=10
im = ax.imshow(m2_mu_plot_pred2-m2_Z, extent=extents, aspect='auto', cmap="bwr",␣
↪norm=divnorm, origin='lower')
ax.set_title('Difference between model and target function')
ax.set_xlabel(m2_param_2)
ax.set_ylabel(m2_param_1)
# Add points where simulation evaluated
ax.plot(np.array(m2_2_model_emukit.X)[:,1], np.array(m2_2_model_emukit.X)[:,0],␣
↪'ro')
fig.colorbar(im, ax=ax)
fig.show()
```

```
/var/folders/98/fv11ygzs4p51c21s8jln0jzc0000gn/T/ipykernel_12821/2861625379.py:
15: UserWarning:Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
/var/folders/98/fv11ygzs4p51c21s8jln0jzc0000gn/T/ipykernel_12821/2861625379.py:
24: UserWarning:Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
/var/folders/98/fv11ygzs4p51c21s8jln0jzc0000gn/T/ipykernel_12821/2861625379.py:
40: UserWarning:Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
```


Target function
6000
5000
4000
3000
2000
1000
fuelmass
100
200
300
400
500
600
700
800
Isp0
−2500
−5000
−7500
−10000
−12500
−15000
−17500



Model (mean)
6000
5000
4000
3000
2000
1000
fuelmass
100
200
300
400
500
600
700
800
Isp0
0
−2500
−5000
−7500
−10000
−12500
−15000
−17500



1.2 2. Maximization

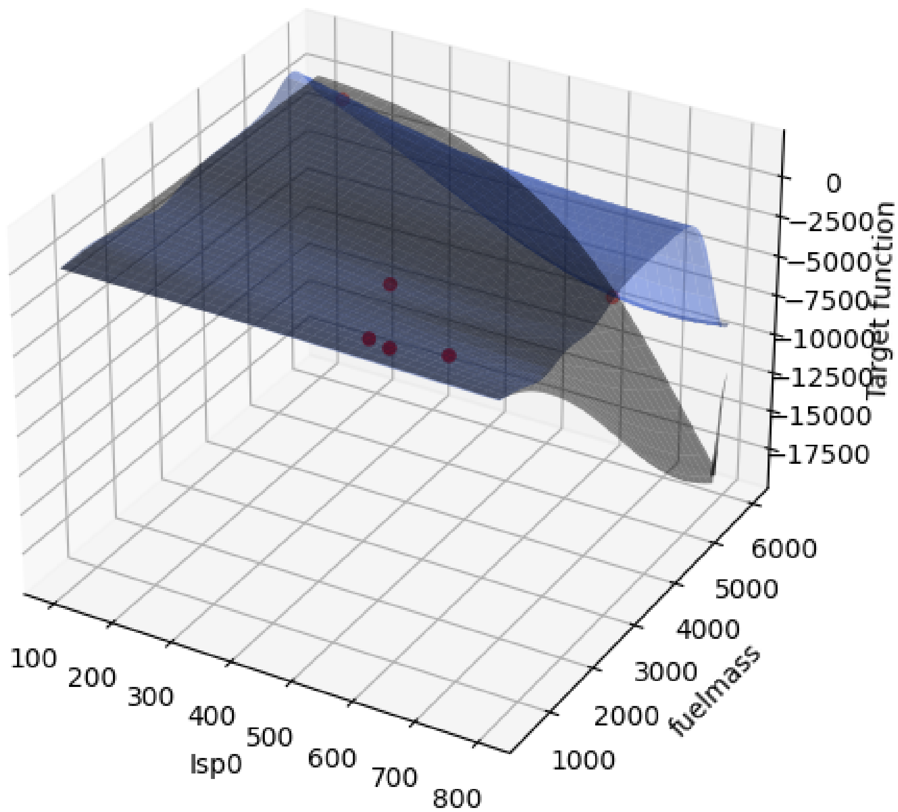
[32]:
```
m2_model_gpy = GPRegression(m2_x,m2_y, kernel=m2_kern)
m2_model_gpy.kern.variance =  m2_var_kernel
m2_model_gpy.likelihood.variance.fix(m2_var_noise)

m2_model_emukit = GPyModelWrapper(m2_model_gpy)
m2_model_emukit.optimize()
```

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

[33]:
```
# Compute current prediction
m2_mu_plot_grid_pred1, var_plot_grid_pred1 = m2_model_emukit.predict(m2_x_plot)
m2_mu_plot_pred1 = m2_mu_plot_grid_pred1.reshape(m2_x_plot_mesh.shape)
m2_var_plot_pred1 = var_plot_grid_pred1.reshape(m2_x_plot_mesh.shape)

if run_grid_simulation:
    m2_rmse = evaluate_prediction(y_actual=m2_y_plot,␣
 ↪y_predicted=m2_mu_plot_grid_pred1)
    print("RMSE m2 (before experiment design loop): ", m2_rmse)
```

```
RMSE m2 (before experiment design loop):  3903.716104090036
```

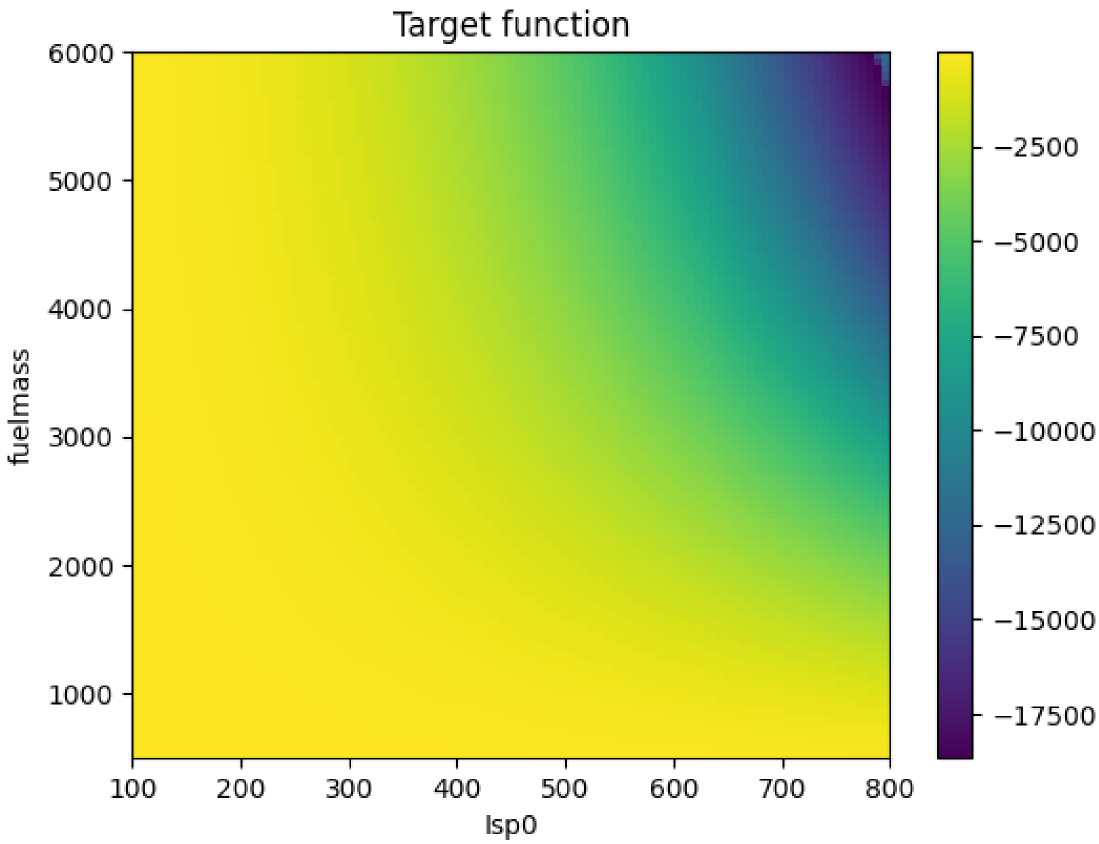
[34]:
```
if run_grid_simulation:
    # 3D Plot
    add_bands = False

    # REVERSE
    fig = plt.figure()
    ax = fig.add_subplot(projection='3d')

    # True surface
    surf = ax.plot_surface(m2_y_plot_mesh, m2_x_plot_mesh, (m2_Z).transpose(),
                           alpha = .5,
                           label='target function',
                           color='black'
                          )
    # Mean predicted
    surf = ax.plot_surface(m2_y_plot_mesh, m2_x_plot_mesh, (m2_mu_plot_pred1).
 ↪transpose(),
                           alpha = .5,
                           label='model', # Mean
                           color='royalblue'
                          )
    # True points observed
    ax.scatter(m2_x[:,1], m2_x[:,0], m2_y, marker='o', color='red')

    ax.set_xlabel(m2_param_2)
    ax.set_ylabel(m2_param_1)
    ax.set_zlabel('Target function')
    plt.tight_layout()
```

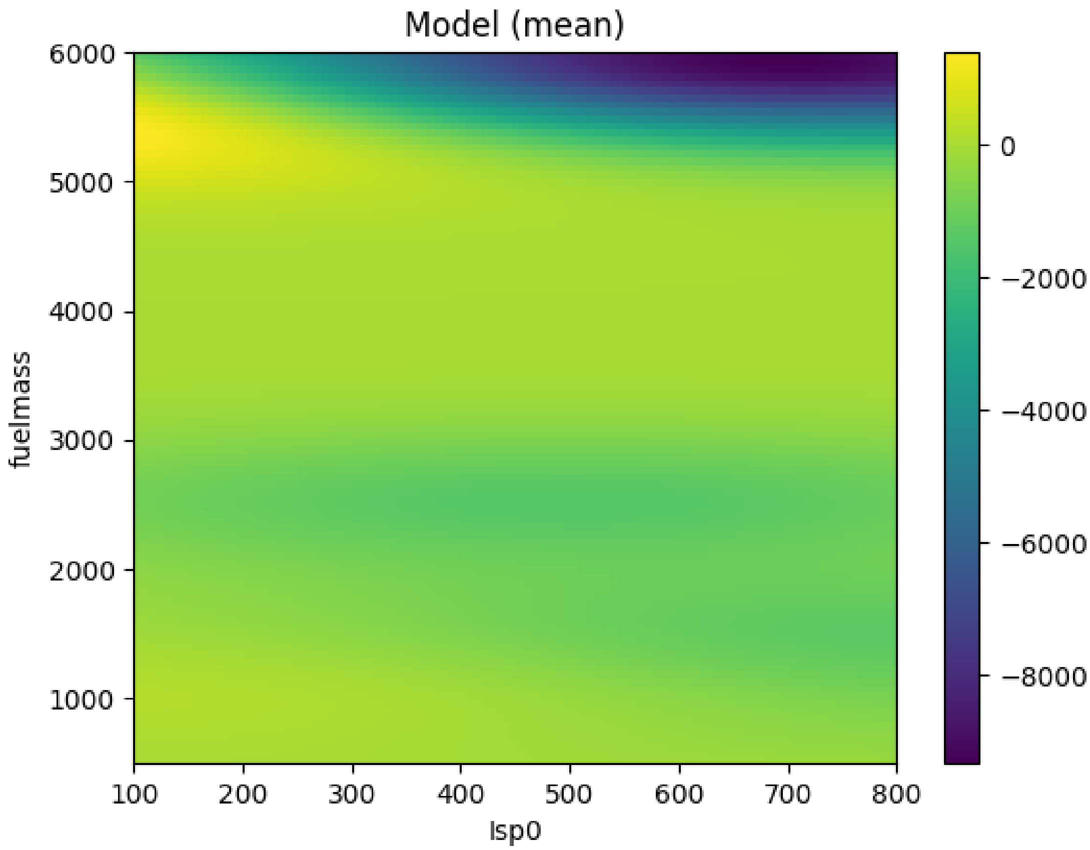


[35]:
```
if run_grid_simulation:
    from matplotlib import colors
#     divnorm=colors.TwoSlopeNorm(vcenter=0.) # vmin=-5., vcenter=0., vmax=10

    # REVERSE
    ## Heatmaps
    extents = [m2_space.parameters[1].min, m2_space.parameters[1].max,
               m2_space.parameters[0].min, m2_space.parameters[0].max]

    # True values
    fig, ax = plt.subplots()
    im = ax.imshow(m2_Z, extent=extents, aspect='auto', origin='lower')
    ax.set_title('Target function')
    ax.set_xlabel(m2_param_2)
    ax.set_ylabel(m2_param_1)
    fig.colorbar(im, ax=ax)
    fig.show()

    # Model
    fig, ax = plt.subplots()
    im = ax.imshow(m2_mu_plot_pred1, extent=extents, aspect='auto', origin='lower')
```

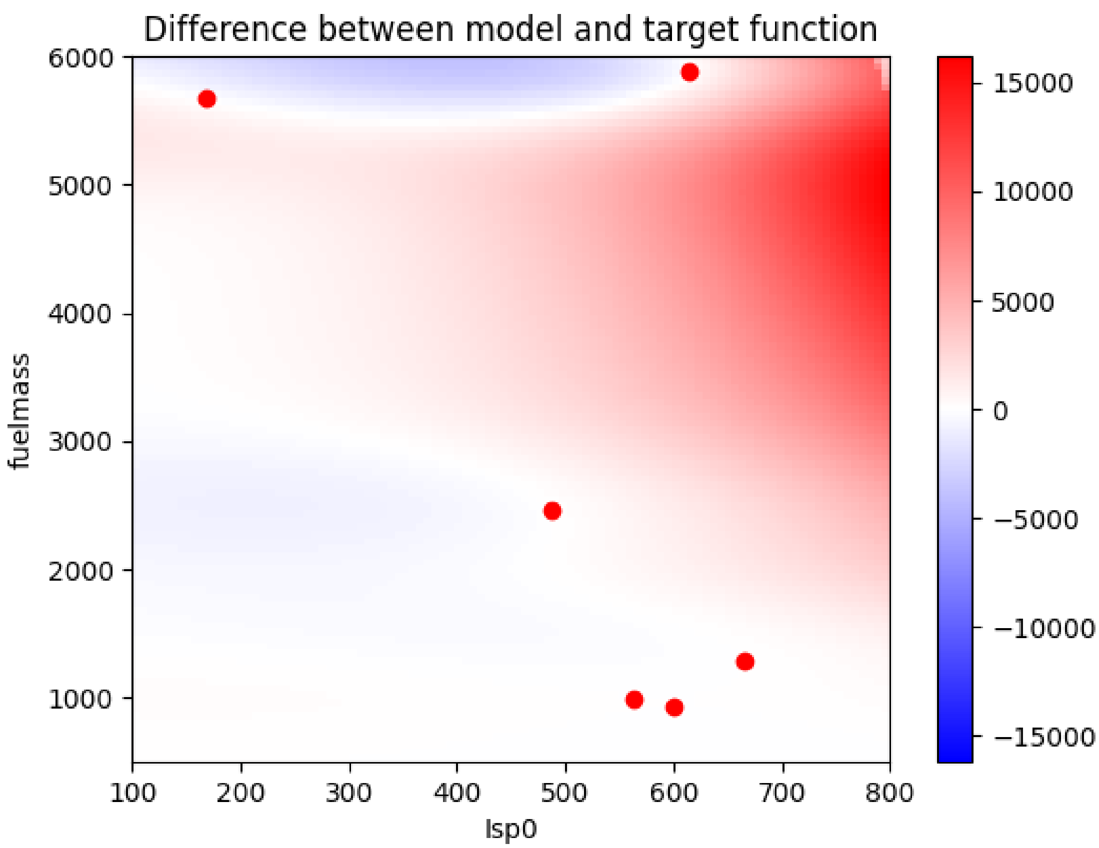
```
    ax.set_title('Model (mean)')
    ax.set_xlabel(m2_param_2)
    ax.set_ylabel(m2_param_1)
    fig.colorbar(im, ax=ax)
    fig.show()


    # Difference
    vmin = (m2_mu_plot_pred1-m2_Z).min()
    vmax = (m2_mu_plot_pred1-m2_Z).max()
    vmin_max = max(abs(vmin), abs(vmax))
    divnorm=colors.TwoSlopeNorm(vcenter=0., vmin=-vmin_max, vmax=vmin_max) # vmin=-5.,␣
 ↪vcenter=0., vmax=10
    fig, ax = plt.subplots()
    im = ax.imshow(m2_mu_plot_pred1-m2_Z, extent=extents, aspect='auto', cmap="bwr",␣
 ↪norm=divnorm, origin='lower')
    ax.set_title('Difference between model and target function')
    ax.set_xlabel(m2_param_2)
    ax.set_ylabel(m2_param_1)
    # Add points where simulation evaluated
    ax.plot(m2_x[:,1], m2_x[:,0], 'ro')
    fig.colorbar(im, ax=ax)
    fig.show()
```

```
/var/folders/98/fv11ygzs4p51c21s8jln0jzc0000gn/T/ipykernel_12821/2119566484.py:
17: UserWarning:Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
/var/folders/98/fv11ygzs4p51c21s8jln0jzc0000gn/T/ipykernel_12821/2119566484.py:
26: UserWarning:Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
/var/folders/98/fv11ygzs4p51c21s8jln0jzc0000gn/T/ipykernel_12821/2119566484.py:
42: UserWarning:Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
```


Target function
fuelmass
6000
5000
4000
3000
2000
1000
100
200
300
400
500
600
700
800
Isp0
−2500
−5000
−7500
−10000
−12500
−15000
−17500



Model (mean)
6000
5000
4000
3000
2000
1000
fuelmass
100
200
300
400
500
600
700
800
Isp0
0
−2000
−4000
−6000
−8000



1.2.1 Use the model created for model-based bayes optimization

use the model to decide which are the best points to collect using some data collection criteria (that we call acquisition).

[36]:
```
m2_max_model_emukit = m2_model_emukit
```

[37]:
```
# Bayesian optimization using emulator
from emukit.bayesian_optimization.acquisitions import ExpectedImprovement
from emukit.bayesian_optimization.loops import BayesianOptimizationLoop

maxim_aquisition = ExpectedImprovement(model=m2_max_model_emukit)


bayesopt_loop = BayesianOptimizationLoop(model = m2_max_model_emukit,
                                         space = m2_space,
                                         acquisition = maxim_aquisition,
                                         batch_size = 1)

max_iterations = 5*2

bayesopt_loop.run_loop(neg_run_missile_sim, max_iterations)
```

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

New simulation

fuelmass: 6000.0
Isp0: 794.360549356994


Stage 1 burnout
Velocity (km/s):  9.178508685748767
Angle (deg h):  43.65600014577831
Range (km):  686.5357746028335
Time (sec):  519.4000000000495
Simulation exceeded time limit.
Final results:
Range (km):  13730.204805621252
Apogee (km):  24848.814173837214
Time to target (sec):  20000.099999989452



New simulation

fuelmass: 5725.99797866685
Isp0: 800.0


Stage 1 burnout
Velocity (km/s):  9.118048401086302
Angle (deg h):  43.65626668225193
Range (km):  669.0264340484053
Time (sec):  499.200000000045
Final results:
Range (km):  18662.00297949803
Apogee (km):  22913.42429568097
Time to target (sec):  19851.69999999161



New simulation

fuelmass: 5491.501957409965
Isp0: 800.0


Stage 1 burnout
Velocity (km/s):  8.986868397237318
Angle (deg h):  43.66205072403367
Range (km):  643.8666194042343
Time (sec):  478.70000000004035
Final results:
```

```
Range (km):  18068.70181252249
Apogee (km):  19560.869158892732
Time to target (sec):  16614.000000038726



New simulation

fuelmass: 5638.187402073278
Isp0: 800.0


Stage 1 burnout
Velocity (km/s):  9.069021152976038
Angle (deg h):  43.66061047352842
Range (km):  659.6287739181668
Time (sec):  491.50000000004326
Final results:
Range (km):  18441.599924881288
Apogee (km):  21567.716802165975
Time to target (sec):  18534.100000010785



New simulation

fuelmass: 5136.2218400038255
Isp0: 800.0


Stage 1 burnout
Velocity (km/s):  8.773306219479968
Angle (deg h):  43.655173745661216
Range (km):  604.2078966755495
Time (sec):  447.8000000000333
Final results:
Range (km):  17084.70004943095
Apogee (km):  15480.784296271606
Time to target (sec):  12878.20000002932



New simulation

fuelmass: 5214.108918923757
Isp0: 598.4149918308952


Stage 1 burnout
Velocity (km/s):  6.3664507172109435
Angle (deg h):  43.66125949674566
Range (km):  337.7559182400016
Time (sec):  340.0000000000088
Final results:
```

```
Range (km):  7259.152101533458
Apogee (km):  2747.0445220049146
Time to target (sec):  2645.899999998689



New simulation

fuelmass: 4580.8528879575
Isp0: 800.0


Stage 1 burnout
Velocity (km/s):  8.366900308288027
Angle (deg h):  43.6526820549504
Range (km):  535.8848511994823
Time (sec):  399.3000000000223
Final results:
Range (km):  15183.720438505185
Apogee (km):  10540.313747532786
Time to target (sec):  8656.10000001396



New simulation

fuelmass: 4423.650819361284
Isp0: 496.9281684185544


Stage 1 burnout
Velocity (km/s):  4.812367615463809
Angle (deg h):  43.65142941764638
Range (km):  190.566551832335
Time (sec):  239.59999999999067
Final results:
Range (km):  3499.8928831317667
Apogee (km):  1143.739479642611
Time to target (sec):  1389.3999999998318



New simulation

fuelmass: 4804.623926231683
Isp0: 800.0


Stage 1 burnout
Velocity (km/s):  8.5433286072883
Angle (deg h):  43.6545044250496
Range (km):  564.4400146052687
Time (sec):  418.90000000002675
Final results:
```

```
Range (km):  16009.045083093062
Apogee (km):  12354.506183455806
Time to target (sec):  10168.400000019461



New simulation

fuelmass: 3967.1975410829014
Isp0: 800.0


Stage 1 burnout
Velocity (km/s):  7.832664856369256
Angle (deg h):  43.65434767329817
Range (km):  455.13035058256884
Time (sec):  345.90000000001015
Final results:
Range (km):  12709.242416139505
Apogee (km):  6849.695397415779
Time to target (sec):  5702.800000003215
```

[38]:
```
results = bayesopt_loop.get_results()
results
```

[38]:
```
<emukit.bayesian_optimization.loops.bayesian_optimization_loop.BayesianOptimizat
ionResults at 0x7fba49a43ee0>
```

[39]:
```
m2_max_model_emukit.X.shape
```

[39]:
```
(16, 2)
```

[40]:
```
m2_max_model_emukit.__dict__
```

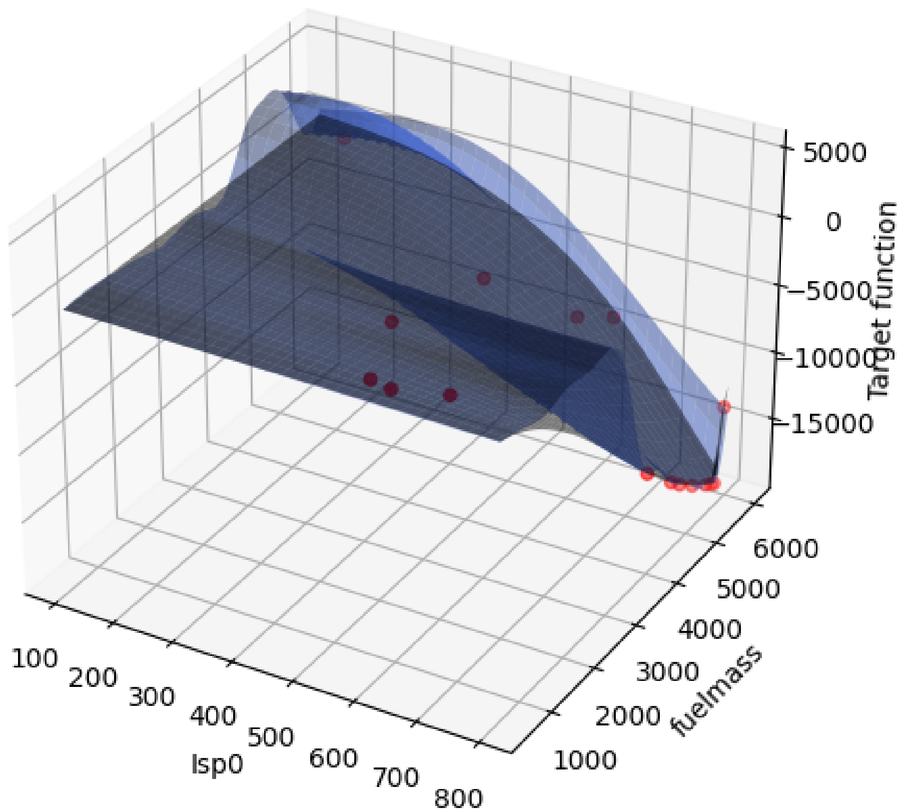
[40]:
```
{'model': <GPy.models.gp_regression.GPRegression at 0x7fba308d0ca0>,
 'n_restarts': 1}
```

[41]:
```
if run_grid_simulation:
    # Compute new prediction
    m2_mu_plot_grid_pred2, var_plot_grid_pred2 = m2_max_model_emukit.predict(m2_x_plot)
    m2_mu_plot_pred2 = m2_mu_plot_grid_pred2.reshape(m2_x_plot_mesh.shape)
    m2_var_plot_pred2 = var_plot_grid_pred2.reshape(m2_x_plot_mesh.shape)

    m2_max_rmse = evaluate_prediction(y_actual=m2_y_plot,␣
 ↪y_predicted=m2_mu_plot_grid_pred2)
    print("RMSE m2 (post bayes opt loop): ", m2_max_rmse)
```

```
RMSE m2 (post bayes opt loop):  1896.673493025905
```

[42]:
```
if run_grid_simulation:
    # 3D Plot
    add_bands = False
```

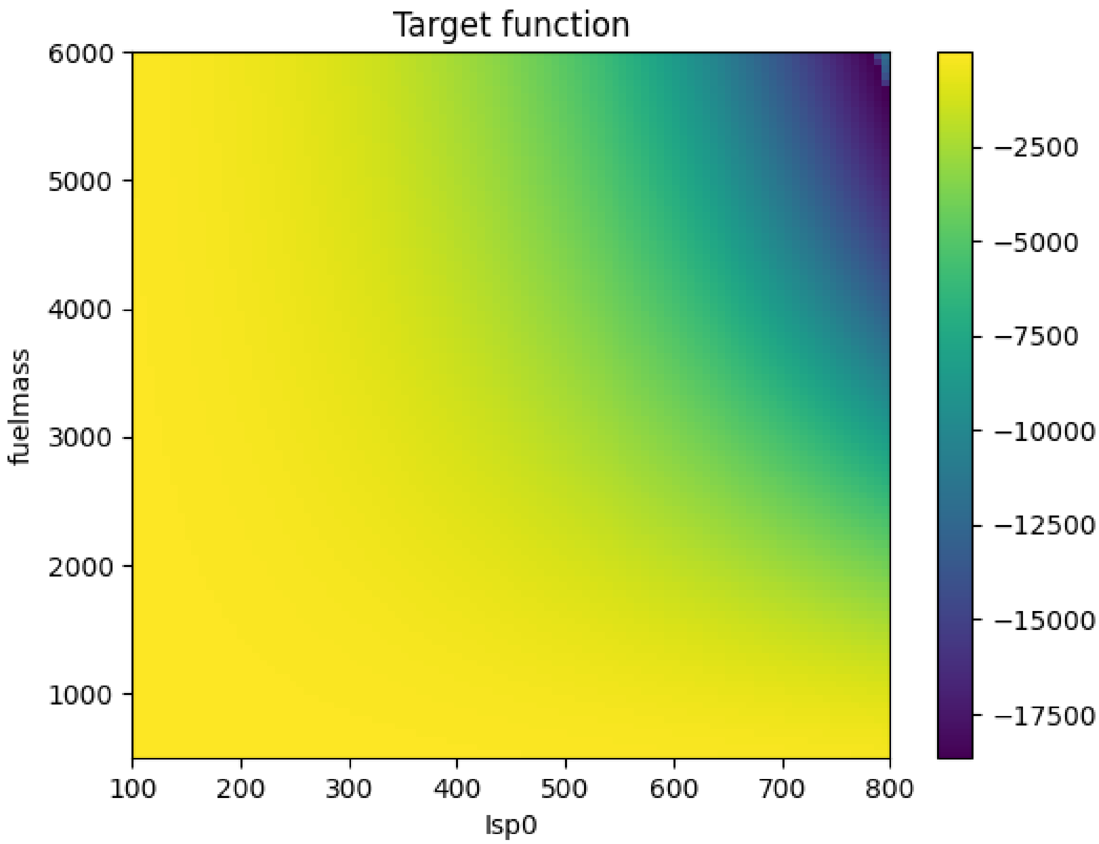
```
# REVERSE
fig = plt.figure()
ax = fig.add_subplot(projection='3d')

# True surface
surf = ax.plot_surface(m2_y_plot_mesh, m2_x_plot_mesh, (m2_Z).transpose(),
                        alpha = .5,
                        label='target function',
                        color='black'
                       )
# Mean predicted
surf = ax.plot_surface(m2_y_plot_mesh, m2_x_plot_mesh, (m2_mu_plot_pred2).
↪transpose(),
                        alpha = .5,
                        label='model', # Mean
                        color='royalblue'
                       )
# True points observed
ax.scatter(np.array(m2_max_model_emukit.X)[:,1],
           np.array(m2_max_model_emukit.X)[:,0], m2_max_model_emukit.Y,␣
↪marker='o', color='red')

ax.set_xlabel(m2_param_2)
ax.set_ylabel(m2_param_1)
ax.set_zlabel('Target function')
plt.tight_layout()
```

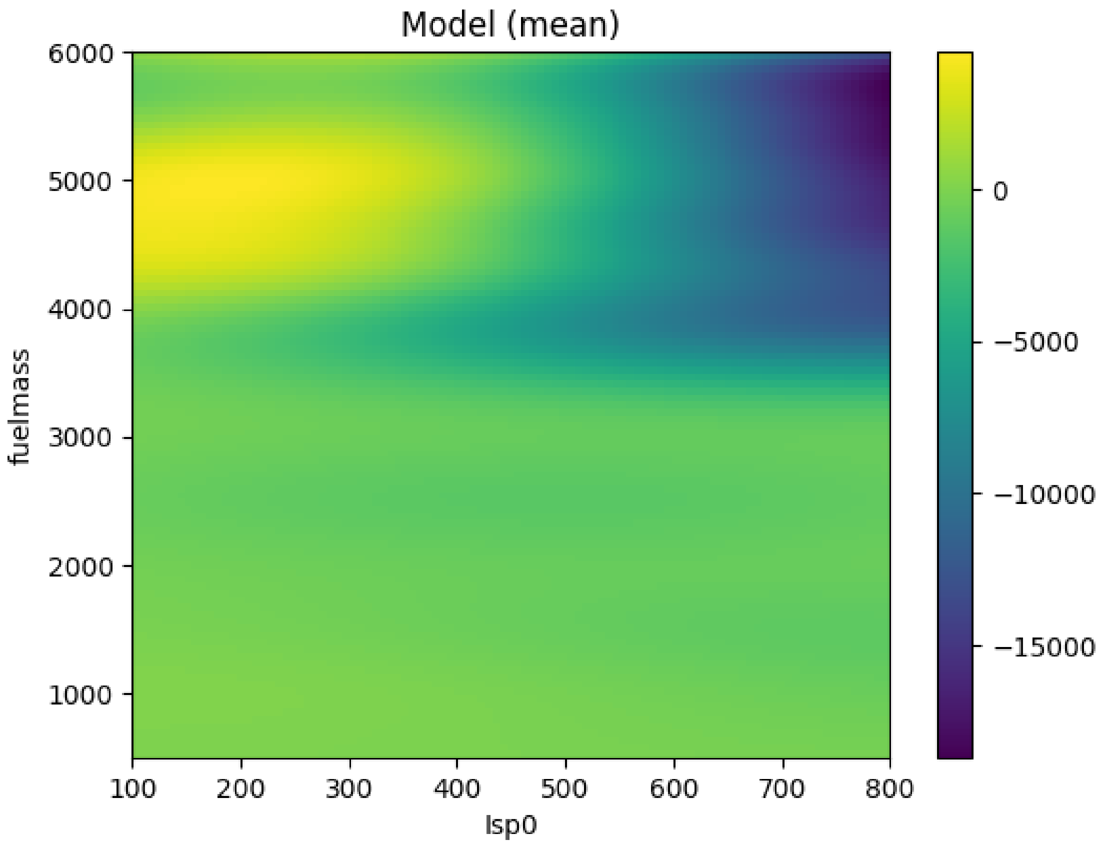


[43]:
```
if run_grid_simulation:

    ## Heatmaps
    extents = [m2_space.parameters[1].min, m2_space.parameters[1].max,
               m2_space.parameters[0].min, m2_space.parameters[0].max]

    # True values
    fig, ax = plt.subplots()
    im = ax.imshow(m2_Z, extent=extents, aspect='auto', origin='lower')
    ax.set_title('Target function')
    ax.set_xlabel(m2_param_2)
    ax.set_ylabel(m2_param_1)
    fig.colorbar(im, ax=ax)
    fig.show()

    # Model
    fig, ax = plt.subplots()
    im = ax.imshow(m2_mu_plot_pred2, extent=extents, aspect='auto', origin='lower')
    ax.set_title('Model (mean)')
    ax.set_xlabel(m2_param_2)
    ax.set_ylabel(m2_param_1)
```

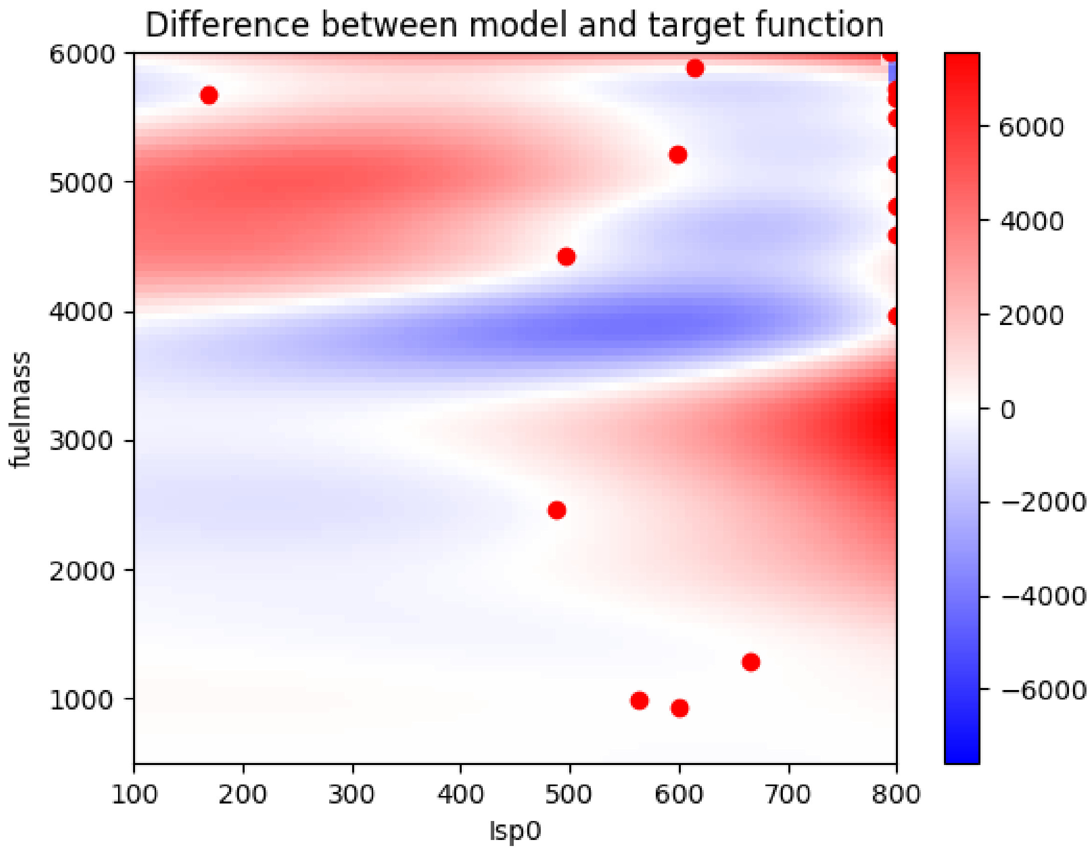
```
    fig.colorbar(im, ax=ax)
    fig.show()


    # Difference
    fig, ax = plt.subplots()
    vmin = (m2_mu_plot_pred2-m2_Z).min()
    vmax = (m2_mu_plot_pred2-m2_Z).max()
    vmin_max = max(abs(vmin), abs(vmax))
    divnorm=colors.TwoSlopeNorm(vcenter=0., vmin=-vmin_max, vmax=vmin_max) # vmin=-5.,␣
↪vcenter=0., vmax=10
    im = ax.imshow(m2_mu_plot_pred2-m2_Z, extent=extents, aspect='auto', cmap="bwr",␣
↪norm=divnorm, origin='lower')
    ax.set_title('Difference between model and target function')
    ax.set_xlabel(m2_param_2)
    ax.set_ylabel(m2_param_1)
    # Add points where simulation evaluated
#     ax.plot(m2_x[:,0], m2_x[:,1], 'ro')
#     ax.plot(m2_x[:,1], m2_x[:,0], 'ro')
    ax.plot(np.array(m2_max_model_emukit.X)[:,1], np.array(m2_max_model_emukit.X)[:
↪,0], 'ro')
    fig.colorbar(im, ax=ax)
    fig.show()
```

```
/var/folders/98/fv11ygzs4p51c21s8jln0jzc0000gn/T/ipykernel_12821/2525306052.py:
14: UserWarning:Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
/var/folders/98/fv11ygzs4p51c21s8jln0jzc0000gn/T/ipykernel_12821/2525306052.py:
23: UserWarning:Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
/var/folders/98/fv11ygzs4p51c21s8jln0jzc0000gn/T/ipykernel_12821/2525306052.py:
41: UserWarning:Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
```


Target function
6000
5000
4000
3000
2000
1000
fuelmass
100
200
300
400
500
600
700
800
Isp0
−2500
−5000
−7500
−10000
−12500
−15000
−17500



Model (mean)
6000
5000
4000
3000
2000
1000
fuelmass
100
200
300
400
500
600
700
800
Isp0
0
−5000
−10000
−15000



[44]:
```
## Greedy maximization using the simulator
# # opt 1
from collections import namedtuple
Min_val = namedtuple('Min_val', 'fun x')
min_idx = np.argmin(m2_y_plot)
true_minim = Min_val( m2_y_plot[min_idx], m2_x_plot[min_idx])

print("True min value: ", m2_y_plot[min_idx])
print("True min location: ", m2_x_plot[min_idx])

# # # opt2
# nr_custom_params = 2
# wirte_output_txt = False
# from scipy.optimize import minimize


# # func_to_minimize = lambda x: (x[0] - 1)**2 + (x[1] - 2.5)**2
# def func_to_minimize(x):
#     print(x)
#     return neg_run_missile_sim(np.array(x).reshape(1,nr_custom_params))

# bnds = [(m2_domain_param_1),
#          (m2_domain_param_2),
```

```
# #         (m3_domain_param_3),
# #         (m3_domain_param_4),
# #         (m3_domain_param_5),
# #         (m3_domain_param_6)
#         ]


# initial_guess = [np.mean(m2_domain_param_1),
#                   np.mean(m2_domain_param_2),
# #                   np.mean(m3_domain_param_3),
# #                   np.mean(m3_domain_param_4),
# #                   np.mean(m3_domain_param_5),
# #                   np.mean(m3_domain_param_6)
#                  ]
# true_minim = minimize(func_to_minimize, initial_guess, bounds=bnds) # ,␣
↪method='SLSQP'constraints=cons
```

```
True min value:  [-18655.87990491]
True min location:  [5725.  800.]
```

[ ]:

```

```

[45]:

```
min_val_from_sim = true_minim.fun
min_loc_from_sim = true_minim.x
min_val_from_emu = results.minimum_value
min_loc_from_emu = results.minimum_location

min_val_diff = min_val_from_sim - min_val_from_emu
min_loc_diff = min_loc_from_sim - min_loc_from_emu

print("Min val from sim - min val from em: \n", min_val_diff)
print('\n')
print("Min location from sim - min location from em: \n", min_loc_diff)
print('\n')
print('\n')
print("Min location from sim: \n", min_loc_from_sim)
print("Min location from emu: \n", min_loc_from_emu)
print('\n')
print('\n')
print("Min value from sim: \n", min_val_from_sim)
print("Min value from emu: \n", min_val_from_emu)
```

```
Min val from sim - min val from em:
 [5.35667777]


Min location from sim - min location from em:
 [-0.99797867  0.        ]




Min location from sim:
 [5725.  800.]
```

```
Min location from emu:
 [5725.99797867  800.        ]
```

```
Min value from sim:
 [-18655.87990491]
Min value from emu:
 -18661.23658268512
```

[ ]:

```

```

[ ]:

```

```