

RobustOptFifthTrial

June 23, 2024

0.0.1 Packages Import

```
[1]: import numpy as np
import pandas as pd
import gurobipy as grb
import matplotlib.pyplot as plt
import statsmodels.api as sm
from scipy import stats as st
```

Generate both samples from linear demand functions but with unlike parameters, they represent sold quantities of different products made by the same company (production costs are not equal indeed).

```
[2]: # PRODUCT 1
prod_cost = 10
# demand function's real slope and intercept
alpha_real_1 = 200
beta_real_1 = -5
# lower and upper bounds
lower_p1 = prod_cost
upper_p1 = -alpha_real_1 / beta_real_1 # upper bound to maintain the demand
↳ greater or equal to zero, below this value it will be negative
```

```
[3]: # PRODUCT 2
prod_cost_2 = 15
# demand function's real slope and intercept
alpha_real_2 = 350
beta_real_2 = -7
# lower and upper bounds
lower_p2 = prod_cost_2
upper_p2 = -alpha_real_2 / beta_real_2 # upper bound to maintain the demand
↳ greater or equal to zero, below this value it will be negative
```

Robust solution nominal problem The max-min optimization is actually the simple max because there is no uncertainty in the nominal case, all the coefficients are known. The optimal objective value obtained will be the benchmark to which compare the other results since it would be the best revenue possible. Since this time the price has to be selected within a discrete set of values, we set parameter vtype equal to "I" which stands for Integer.

```
[4]: try:
# create the model
    model = grb.Model()
# create decision variables
    p = model.addVars(2, ub=[upper_p1, upper_p2], lb=[lower_p1, lower_p2],
↳vtype='I', name='p')
# set objective function
    expr = (alpha_real_1 + beta_real_1*p[0])*p[0]+(alpha_real_2 +
↳beta_real_2*p[1])*p[1]
    model.setObjective(expr, sense=grb.GRB.MAXIMIZE)
# solve the problem
    model.optimize()
    for v in model.getVars():
        print('%s %g' % (v.varName, v.x))
    print('Obj: %g' % model.objVal)
except grb.GurobiError as e:
    print('Error code' + str(e))
except AttributeError:
    print('Encountered an attribute error')
obj_nominal = model.objVal
```

Restricted license - for non-production use only - expires 2025-11-24
Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (win64 - Windows 11.0
(22631.2))

CPU model: 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz, instruction set
[SSE2|AVX|AVX2|AVX512]
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 0 rows, 2 columns and 0 nonzeros
Model fingerprint: 0x7ec0ea4a
Model has 2 quadratic objective terms
Variable types: 0 continuous, 2 integer (0 binary)
Coefficient statistics:

Matrix range	[0e+00, 0e+00]
Objective range	[2e+02, 4e+02]
QObjective range	[1e+01, 1e+01]
Bounds range	[1e+01, 5e+01]
RHS range	[0e+00, 0e+00]

Found heuristic solution: objective 5175.0000000
Presolve removed 0 rows and 2 columns
Presolve time: 0.00s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.02 seconds (0.00 work units)
Thread count was 1 (of 8 available processors)

Solution count 2: 6375 5175

Optimal solution found (tolerance 1.00e-04)

Best objective 6.375000000000e+03, best bound 6.375000000000e+03, gap 0.0000%

p[0] 20

p[1] 25

Obj: 6375

To perform online learning there should be few historical available data since we want to learn step by step, we choose a low value of sample cardinality indeed. We also set the value of iteration number that means the steps we will complete to obtain a satisfying result (convergence).

```
[5]: np.random.seed(42)
n = 3
n_iterations = 10
var = 1
object_val = []
eps = np.random.normal(0, var, size=(n_iterations+n+1, 2)) # noise component
# DATA PRODUCT 1
price_1 = np.random.uniform(lower_p1, upper_p1, n)
demand_1 = alpha_real_1 + beta_real_1 * price_1 + eps[:n, 0]
# DATA PRODUCT 2
price_2 = np.random.uniform(lower_p2, upper_p2, n)
demand_2 = alpha_real_2 + beta_real_2 * price_2 + eps[:n, 1]
conf_level = 0.05 #significance level
```

```
[6]: data_1 = pd.DataFrame({'price': price_1, 'demand': demand_1})
data_2 = pd.DataFrame({'price': price_2, 'demand': demand_2})
X_1 = pd.DataFrame({'price': data_1['price'].copy()})
y_1 = pd.DataFrame({'demand': data_1['demand'].copy()})
X_1['constant'] = 1
# LINEAR REGRESSION 1
np.random.seed(42)
returns_1 = np.linalg.lstsq(X_1, y_1, rcond=None)
params_1 = returns_1[0]
alpha_1 = params_1[1]
beta_1 = params_1[0]
sum_sq_residuals_1 = returns_1[1] # sum of squared residuals
dof_1 = X_1.shape[0] - len(params_1) # degrees of freedom
mse_1 = sum_sq_residuals_1 / dof_1 # mean squared error
cov_1 = mse_1 * np.diagonal(np.linalg.inv(X_1.T @ X_1)) # covariance matrix
se_1 = np.sqrt(cov_1) # standard errors
t_1 = st.t.ppf(1 - (conf_level / 2), dof_1) # t-distribution quantile function
delta_1 = t_1 * se_1 # margin of error
# confidence interval 95%
alpha_up_1 = alpha_1 + delta_1[0]
alpha_low_1 = alpha_1 - delta_1[0]
beta_up_1 = beta_1 + delta_1[1]
```

```

beta_low_1 = beta_1 - delta_1[1]
X_2 = pd.DataFrame({'price': data_2['price'].copy()})
y_2 = pd.DataFrame({'demand': data_2['demand'].copy()})
X_2['constant'] = 1
# LINEAR REGRESSION 2
returns_2 = np.linalg.lstsq(X_2, y_2, rcond=None)
params_2 = returns_2[0]
alpha_2 = params_2[1]
beta_2 = params_2[0]
sum_sq_residuals_2 = returns_2[1] # sum of squared residuals
dof_2 = X_2.shape[0] - len(params_2) # degrees of freedom
mse_2 = sum_sq_residuals_2 / dof_2 # mean squared error
cov_2 = mse_2 * np.diagonal(np.linalg.inv(X_2.T @ X_2)) # covariance matrix
se_2 = np.sqrt(cov_2) # standard errors
t_2 = st.t.ppf(1 - (conf_level / 2), dof_2) # t-distribution quantile function
delta_2 = t_2 * se_2 # margin of error
# confidence interval 95%
alpha_up_2 = alpha_2 + delta_2[0]
alpha_low_2 = alpha_2 - delta_2[0]
beta_up_2 = beta_2 + delta_2[1]
beta_low_2 = beta_2 - delta_2[1]

```

We define the function `model Updating` to apply at each iteration the update of the linear regression coefficients since we have added new data record.

```

[7]: def model Updating(p1, p2, d1, d2):
    global price_1, price_2, demand_1, demand_2
    price_1 = np.append(price_1, p1)
    price_2 = np.append(price_2, p2)
    demand_1 = np.append(demand_1, d1)
    demand_2 = np.append(demand_2, d2)
    data_1 = pd.DataFrame({'price': price_1, 'demand': demand_1})
    data_2 = pd.DataFrame({'price': price_2, 'demand': demand_2})
    X_1 = pd.DataFrame({'price': data_1['price'].copy()})
    y_1 = pd.DataFrame({'demand': data_1['demand'].copy()})
    X_1['constant'] = 1
    # LINEAR REGRESSION 1
    np.random.seed(42)
    returns_1 = np.linalg.lstsq(X_1, y_1, rcond=None)
    params_1 = returns_1[0]
    alpha_1 = params_1[1]
    beta_1 = params_1[0]
    sum_sq_residuals_1 = returns_1[1] # sum of squared residuals
    dof_1 = X_1.shape[0] - len(params_1) # degrees of freedom
    mse_1 = sum_sq_residuals_1 / dof_1 # mean squared error
    cov_1 = mse_1 * np.diagonal(np.linalg.inv(X_1.T @ X_1)) # covariance matrix
    se_1 = np.sqrt(cov_1) # standard errors

```

```

    t_1 = st.t.ppf(1 - (conf_level / 2), dof_1) # t-distribution quantile
↪function
    delta_1 = t_1 * se_1 # margin of error
    # confidence interval 95%
    alpha_up_1 = alpha_1 + delta_1[0]
    alpha_low_1 = alpha_1 - delta_1[0]
    beta_up_1 = beta_1 + delta_1[1]
    beta_low_1 = beta_1 - delta_1[1]
    X_2 = pd.DataFrame({'price': data_2['price'].copy()})
    y_2 = pd.DataFrame({'demand': data_2['demand'].copy()})
    X_2['constant'] = 1
    # LINEAR REGRESSION 2
    np.random.seed(42)
    returns_2 = np.linalg.lstsq(X_2, y_2, rcond=None)
    params_2 = returns_2[0]
    alpha_2 = params_2[1]
    beta_2 = params_2[0]
    sum_sq_residuals_2 = returns_2[1] # sum of squared residuals
    dof_2 = X_2.shape[0] - len(params_2) # degrees of freedom
    mse_2 = sum_sq_residuals_2 / dof_2 # mean squared error
    cov_2 = mse_2 * np.diagonal(np.linalg.inv(X_2.T @ X_2)) # covariance matrix
    se_2 = np.sqrt(cov_2) # standard errors
    t_2 = st.t.ppf(1 - (conf_level / 2), dof_2) # t-distribution quantile
↪function
    delta_2 = t_2 * se_2 # margin of error
    # confidence interval 95%
    alpha_up_2 = alpha_2 + delta_2[0]
    alpha_low_2 = alpha_2 - delta_2[0]
    beta_up_2 = beta_2 + delta_2[1]
    beta_low_2 = beta_2 - delta_2[1]
    return alpha_low_1, alpha_up_1, beta_low_1, beta_up_1, alpha_low_2,
↪alpha_up_2, beta_low_2, beta_up_2

```

Initialization of parameters: slope and intercept on the basis of available historical data through linear regression model (ordinary least squares), then we solve the first robust optimization problem to obtain the current price and the corresponding demand observed, both are appended in the respective lists.

```

[8]: #uncertainty set sampling
np.random.seed(42)
n_samples = 20
alpha_sim_1 = np.random.uniform(alpha_low_1, alpha_up_1, size=n_samples)
alpha_sim_2 = np.random.uniform(alpha_low_2, alpha_up_2, size=n_samples)
beta_sim_1 = np.random.uniform(beta_low_1, beta_up_1, size=n_samples)
beta_sim_2 = np.random.uniform(beta_low_2, beta_up_2, size=n_samples)
# basic statistics
alpha_min_1 = min(alpha_sim_1)

```

```

alpha_max_1 = max(alpha_sim_1)
beta_min_1 = min(beta_sim_1)
beta_max_1 = max(beta_sim_1)
alpha_min_2 = min(alpha_sim_2)
alpha_max_2 = max(alpha_sim_2)
beta_min_2 = min(beta_sim_2)
beta_max_2 = max(beta_sim_2)
alpha_mean_1 = sum(alpha_sim_1)/n_samples
alpha_mean_2 = sum(alpha_sim_2)/n_samples
beta_mean_1 = sum(beta_sim_1)/n_samples
beta_mean_2 = sum(beta_sim_2)/n_samples
alpha_std_1 = np.sqrt(sum((alpha_sim_1-alpha_mean_1)**2)/(n_samples-1))
alpha_std_2 = np.sqrt(sum((alpha_sim_2-alpha_mean_2)**2)/(n_samples-1))
beta_std_1 = np.sqrt(sum((beta_sim_1-beta_mean_1)**2)/(n_samples-1))
beta_std_2 = np.sqrt(sum((beta_sim_2-beta_mean_2)**2)/(n_samples-1))
# budget of uncertainty delta
delta_test = 0

```

```

[9]: #first optimization problem resolution to obtain the current prices to test
try:
    # create the robust model
    robust_model = grb.Model()
    # create decision variables
    z_r = robust_model.addVar(lb=0, vtype='C', name='z_r')
    price_r = robust_model.addVars(2, lb=[lower_p1, lower_p2], ub=[upper_p1,
↪upper_p2], vtype='I', name='price_r')
    mu_ra = robust_model.addVars(4, lb=0, vtype='C', name='mu_ra')
    mu_rb = robust_model.addVars(4, lb=0, vtype='C', name='mu_rb')
    gamma_ra = robust_model.addVars(4, lb=0, vtype='C', name='gamma_ra')
    gamma_rb = robust_model.addVars(4, lb=0, vtype='C', name='gamma_rb')
    # set objective function
    robust_model.setObjective(z_r, sense=grb.GRB.MAXIMIZE)
    # set constraints
    robust_model.addConstr(mu_ra[0]*alpha_min_1 - mu_ra[1]*alpha_max_1 +
↪mu_ra[2]*alpha_min_2 - mu_ra[3]*alpha_max_2 +
        mu_rb[0]*beta_min_1 - mu_rb[1]*beta_max_1 +
↪mu_rb[2]*beta_min_2 - mu_rb[3]*beta_max_2 -
        gamma_ra[0]*(alpha_mean_1 + delta_test*alpha_std_1) +
↪gamma_ra[1]*(alpha_mean_1 - delta_test*alpha_std_1) -
        gamma_ra[2]*(alpha_mean_2 + delta_test*alpha_std_2) +
↪gamma_ra[3]*(alpha_mean_2 - delta_test*alpha_std_2) -
        gamma_rb[0]*(beta_mean_1 + delta_test*beta_std_1) +
↪gamma_rb[1]*(beta_mean_1 - delta_test*beta_std_1) -
        gamma_rb[2]*(beta_mean_2 + delta_test*beta_std_2) +
↪gamma_rb[3]*(beta_mean_2 - delta_test*beta_std_2) >= z_r, name='c1')

```

```

    robust_model.addConstr(price_r[0] - mu_ra[0] + mu_ra[1] + gamma_ra[0] -
↳gamma_ra[1] == 0, name='c2')
    robust_model.addConstr(price_r[1] - mu_ra[2] + mu_ra[3] + gamma_ra[2] -
↳gamma_ra[3] == 0, name='c3')
    robust_model.addConstr(price_r[0]**2 - mu_rb[0] + mu_rb[1] + gamma_rb[0] -
↳gamma_rb[1] == 0, name='c4')
    robust_model.addConstr(price_r[1]**2 - mu_rb[2] + mu_rb[3] + gamma_rb[2] -
↳gamma_rb[3] == 0, name='c5')
    # solve the problem
    robust_model.params.NonConvex = 2
    robust_model.optimize()
except grb.GurobiError as e:
    print('Error code' + str(e))
except AttributeError:
    print('Encountered an attribute error')
object_val.append(robust_model.objVal)
current_price1 = robust_model.getVars()[1].x
current_price2 = robust_model.getVars()[2].x
obs_demand1 = alpha_real_1 + beta_real_1 * current_price1 + eps[n, 0]
obs_demand2 = alpha_real_2 + beta_real_2 * current_price2 + eps[n, 1]

```

Set parameter NonConvex to value 2

Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (win64 - Windows 11.0 (22631.2))

CPU model: 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz, instruction set [SSE2|AVX|AVX2|AVX512]

Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 3 rows, 19 columns and 27 nonzeros

Model fingerprint: 0xfff3ceb1

Model has 2 quadratic constraints

Variable types: 17 continuous, 2 integer (0 binary)

Coefficient statistics:

Matrix range	[5e-01, 4e+02]
QMatrix range	[1e+00, 1e+00]
QLMatrix range	[1e+00, 1e+00]
Objective range	[1e+00, 1e+00]
Bounds range	[1e+01, 5e+01]
RHS range	[0e+00, 0e+00]

Presolve removed 0 rows and 2 columns

Presolve time: 0.00s

Presolved: 9 rows, 17 columns, 53 nonzeros

Presolved model has 2 bilinear constraint(s)

Solving non-convex MIQCP

Variable types: 15 continuous, 2 integer (0 binary)

Root relaxation: objective 8.218786e+03, 8 iterations, 0.00 seconds (0.00 work units)

Nodes		Current Node			Objective Bounds			Work		
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time	
0	0	8218.78563	0	3	- 8218.78563	-	-	-	0s	
0	0	6563.50797	0	4	- 6563.50797	-	-	-	0s	
0	0	5893.48831	0	4	- 5893.48831	-	-	-	0s	
0	0	5771.72162	0	4	- 5771.72162	-	-	-	0s	
0	0	5747.36828	0	2	- 5747.36828	-	-	-	0s	
0	0	5724.28993	0	4	- 5724.28993	-	-	-	0s	
0	0	5719.45724	0	4	- 5719.45724	-	-	-	0s	
0	0	5718.41084	0	4	- 5718.41084	-	-	-	0s	
0	0	5716.51770	0	4	- 5716.51770	-	-	-	0s	
0	0	5715.57113	0	4	- 5715.57113	-	-	-	0s	
H	0	0			5712.2442450	5714.21080	0.03%	-	0s	
	0	2	5714.21080	0	4	5712.24425	5714.21080	0.03%	-	0s

Explored 3 nodes (32 simplex iterations) in 0.06 seconds (0.00 work units)
Thread count was 8 (of 8 available processors)

Solution count 1: 5712.24

Optimal solution found (tolerance 1.00e-04)

Best objective 5.712244245002e+03, best bound 5.712438864114e+03, gap 0.0034%

After the first iteration, we perform the for loop in which we shrink step by step budget of uncertainty value since we add information and consequently the coefficient estimations will become more and more reliable.

```
[10]: for i in range(n_iterations):
    alpha_low_1, alpha_up_1, beta_low_1, beta_up_1, alpha_low_2, alpha_up_2,
    ↪ beta_low_2, beta_up_2 = model Updating(current_price1,
    ↪
    ↪ current_price2,
    ↪
    ↪ obs_demand1,
    ↪
    ↪ obs_demand2)
    np.random.seed(42)
    n_samples = 20
    alpha_sim_1 = np.random.uniform(alpha_low_1, alpha_up_1, size=n_samples)
    alpha_sim_2 = np.random.uniform(alpha_low_2, alpha_up_2, size=n_samples)
    beta_sim_1 = np.random.uniform(beta_low_1, beta_up_1, size=n_samples)
    beta_sim_2 = np.random.uniform(beta_low_2, beta_up_2, size=n_samples)
```

```

# basic statistics
alpha_min_1 = min(alpha_sim_1)
alpha_max_1 = max(alpha_sim_1)
beta_min_1 = min(beta_sim_1)
beta_max_1 = max(beta_sim_1)
alpha_min_2 = min(alpha_sim_2)
alpha_max_2 = max(alpha_sim_2)
beta_min_2 = min(beta_sim_2)
beta_max_2 = max(beta_sim_2)
alpha_mean_1 = sum(alpha_sim_1)/n_samples
alpha_mean_2 = sum(alpha_sim_2)/n_samples
beta_mean_1 = sum(beta_sim_1)/n_samples
beta_mean_2 = sum(beta_sim_2)/n_samples
alpha_std_1 = np.sqrt(sum((alpha_sim_1-alpha_mean_1)**2)/(n_samples-1))
alpha_std_2 = np.sqrt(sum((alpha_sim_2-alpha_mean_2)**2)/(n_samples-1))
beta_std_1 = np.sqrt(sum((beta_sim_1-beta_mean_1)**2)/(n_samples-1))
beta_std_2 = np.sqrt(sum((beta_sim_2-beta_mean_2)**2)/(n_samples-1))
delta_test = 0
try:
    # create the robust model
    robust_model = grb.Model()
    # create decision variables
    z_r = robust_model.addVar(lb=0, vtype='C', name='z_r')
    price_r = robust_model.addVars(2, lb=[lower_p1, lower_p2], ub=[upper_p1,
↪upper_p2], vtype='I', name='price_r')
    mu_ra = robust_model.addVars(4, lb=0, vtype='C', name='mu_ra')
    mu_rb = robust_model.addVars(4, lb=0, vtype='C', name='mu_rb')
    gamma_ra = robust_model.addVars(4, lb=0, vtype='C', name='gamma_ra')
    gamma_rb = robust_model.addVars(4, lb=0, vtype='C', name='gamma_rb')
    # set objective function
    robust_model.setObjective(z_r, sense=grb.GRB.MAXIMIZE)
    # set constraints
    robust_model.addConstr(mu_ra[0]*alpha_min_1 - mu_ra[1]*alpha_max_1 +
↪mu_ra[2]*alpha_min_2 - mu_ra[3]*alpha_max_2 +
        mu_rb[0]*beta_min_1 - mu_rb[1]*beta_max_1 +
↪mu_rb[2]*beta_min_2 - mu_rb[3]*beta_max_2 -
        gamma_ra[0]*(alpha_mean_1 +
↪delta_test*alpha_std_1) + gamma_ra[1]*(alpha_mean_1 - delta_test*alpha_std_1) -
        gamma_ra[2]*(alpha_mean_2 +
↪delta_test*alpha_std_2) + gamma_ra[3]*(alpha_mean_2 - delta_test*alpha_std_2) -
        gamma_rb[0]*(beta_mean_1 + delta_test*beta_std_1)
↪+ gamma_rb[1]*(beta_mean_1 - delta_test*beta_std_1) -
        gamma_rb[2]*(beta_mean_2 + delta_test*beta_std_2)
↪+
        gamma_rb[3]*(beta_mean_2 - delta_test*beta_std_2)
↪>= z_r, name='c1')

```

```

robust_model.addConstr(price_r[0] - mu_ra[0] + mu_ra[1] + gamma_ra[0] -
↳gamma_ra[1] == 0, name='c2')
robust_model.addConstr(price_r[1] - mu_ra[2] + mu_ra[3] + gamma_ra[2] -
↳gamma_ra[3] == 0, name='c3')
robust_model.addConstr(price_r[0]**2 - mu_rb[0] + mu_rb[1] + gamma_rb[0] -
↳gamma_rb[1] == 0, name='c4')
robust_model.addConstr(price_r[1]**2 - mu_rb[2] + mu_rb[3] + gamma_rb[2] -
↳gamma_rb[3] == 0, name='c5')
# solve the problem
robust_model.params.NonConvex = 2
robust_model.optimize()
except grb.GurobiError as e:
    print('Error code' + str(e))
except AttributeError:
    print('Encountered an attribute error')
object_val.append(robust_model.objVal)
current_price1 = robust_model.getVars()[1].x
current_price2 = robust_model.getVars()[2].x
obs_demand1 = alpha_real_1 + beta_real_1 * current_price1 + eps[i+n+1, 0]
obs_demand2 = alpha_real_2 + beta_real_2 * current_price2 + eps[i+n+1, 1]

```

Set parameter NonConvex to value 2

Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (win64 - Windows 11.0 (22631.2))

CPU model: 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz, instruction set [SSE2|AVX|AVX2|AVX512]

Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 3 rows, 19 columns and 27 nonzeros

Model fingerprint: 0xa23a2fe7

Model has 2 quadratic constraints

Variable types: 17 continuous, 2 integer (0 binary)

Coefficient statistics:

Matrix range	[1e+00, 4e+02]
QMatrix range	[1e+00, 1e+00]
QLMatrix range	[1e+00, 1e+00]
Objective range	[1e+00, 1e+00]
Bounds range	[1e+01, 5e+01]
RHS range	[0e+00, 0e+00]

Presolve removed 0 rows and 2 columns

Presolve time: 0.00s

Presolved: 9 rows, 17 columns, 53 nonzeros

Presolved model has 2 bilinear constraint(s)

Solving non-convex MIQCP

Variable types: 15 continuous, 2 integer (0 binary)

Root relaxation: objective 8.944668e+03, 9 iterations, 0.00 seconds (0.00 work units)

Nodes		Current Node			Objective Bounds			Work	
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
0	0	8944.66828	0	3	- 8944.66828	-	-	-	0s
0	0	7035.46555	0	4	- 7035.46555	-	-	-	0s
0	0	6310.47251	0	4	- 6310.47251	-	-	-	0s
0	0	6303.81016	0	2	- 6303.81016	-	-	-	0s
0	0	6253.07043	0	4	- 6253.07043	-	-	-	0s
0	0	6240.59709	0	2	- 6240.59709	-	-	-	0s
0	0	6238.19656	0	4	- 6238.19656	-	-	-	0s
0	0	6232.23335	0	4	- 6232.23335	-	-	-	0s
0	0	6229.88482	0	4	- 6229.88482	-	-	-	0s
0	0	6225.97692	0	3	- 6225.97692	-	-	-	0s
0	0	6224.87262	0	3	- 6224.87262	-	-	-	0s
H	0	0			6224.2843732	6224.86117	0.01%	-	0s

Cutting planes:

MIR: 1

Explored 1 nodes (27 simplex iterations) in 0.06 seconds (0.00 work units)
Thread count was 8 (of 8 available processors)

Solution count 1: 6224.28

Optimal solution found (tolerance 1.00e-04)

Best objective 6.224284373228e+03, best bound 6.224861173633e+03, gap 0.0093%

Set parameter NonConvex to value 2

Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (win64 - Windows 11.0 (22631.2))

CPU model: 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz, instruction set [SSE2|AVX|AVX2|AVX512]

Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 3 rows, 19 columns and 27 nonzeros

Model fingerprint: 0x3f2acb22

Model has 2 quadratic constraints

Variable types: 17 continuous, 2 integer (0 binary)

Coefficient statistics:

Matrix range [1e+00, 4e+02]

QMatrix range [1e+00, 1e+00]

QLMatrix range [1e+00, 1e+00]

Objective range [1e+00, 1e+00]

Bounds range [1e+01, 5e+01]
 RHS range [0e+00, 0e+00]
 Presolve removed 0 rows and 2 columns
 Presolve time: 0.00s
 Presolved: 9 rows, 17 columns, 53 nonzeros
 Presolved model has 2 bilinear constraint(s)

Solving non-convex MIQCP

Variable types: 15 continuous, 2 integer (0 binary)

Root relaxation: objective 9.021027e+03, 9 iterations, 0.00 seconds (0.00 work units)

Nodes		Current Node			Objective Bounds			Work	
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
0	0	9021.02666	0	3	- 9021.02666	-	-	-	0s
0	0	7084.58089	0	4	- 7084.58089	-	-	-	0s
0	0	6390.76874	0	4	- 6390.76874	-	-	-	0s
0	0	6383.42622	0	2	- 6383.42622	-	-	-	0s
0	0	6322.16605	0	4	- 6322.16605	-	-	-	0s
0	0	6310.27695	0	2	- 6310.27695	-	-	-	0s
0	0	6297.79778	0	4	- 6297.79778	-	-	-	0s
0	0	6290.03056	0	4	- 6290.03056	-	-	-	0s
0	0	6288.93014	0	4	- 6288.93014	-	-	-	0s
0	0	6284.80612	0	3	- 6284.80612	-	-	-	0s
H	0	0			6284.2104669	6284.78045	0.01%	-	0s

Cutting planes:

MIR: 1

Explored 1 nodes (26 simplex iterations) in 0.04 seconds (0.00 work units)

Thread count was 8 (of 8 available processors)

Solution count 1: 6284.21

Optimal solution found (tolerance 1.00e-04)

Best objective 6.284210466880e+03, best bound 6.284780452899e+03, gap 0.0091%

Set parameter NonConvex to value 2

Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (win64 - Windows 11.0 (22631.2))

CPU model: 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz, instruction set [SSE2|AVX|AVX2|AVX512]

Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 3 rows, 19 columns and 27 nonzeros

Model fingerprint: 0xcb94b8f7
 Model has 2 quadratic constraints
 Variable types: 17 continuous, 2 integer (0 binary)

Coefficient statistics:
 Matrix range [1e+00, 4e+02]
 QMatrix range [1e+00, 1e+00]
 QLMatrix range [1e+00, 1e+00]
 Objective range [1e+00, 1e+00]
 Bounds range [1e+01, 5e+01]
 RHS range [0e+00, 0e+00]

Presolve removed 0 rows and 2 columns
 Presolve time: 0.00s
 Presolved: 9 rows, 17 columns, 53 nonzeros
 Presolved model has 2 bilinear constraint(s)

Solving non-convex MIQCP

Variable types: 15 continuous, 2 integer (0 binary)

Root relaxation: objective 9.030890e+03, 9 iterations, 0.00 seconds (0.00 work units)

Nodes		Current Node			Objective Bounds			Work	
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
0	0	9030.89048	0	3	- 9030.89048	-	-	-	0s
0	0	7090.27398	0	4	- 7090.27398	-	-	-	0s
0	0	6402.76834	0	4	- 6402.76834	-	-	-	0s
0	0	6395.27845	0	2	- 6395.27845	-	-	-	0s
0	0	6332.06769	0	4	- 6332.06769	-	-	-	0s
0	0	6320.40815	0	2	- 6320.40815	-	-	-	0s
0	0	6307.06471	0	4	- 6307.06471	-	-	-	0s
0	0	6298.56072	0	4	- 6298.56072	-	-	-	0s
0	0	6297.66511	0	4	- 6297.66511	-	-	-	0s
0	0	6294.07963	0	3	- 6294.07963	-	-	-	0s
H	0	0			6293.6849325	6294.03513	0.01%	-	0s

Cutting planes:
 MIR: 1

Explored 1 nodes (26 simplex iterations) in 0.04 seconds (0.00 work units)
 Thread count was 8 (of 8 available processors)

Solution count 1: 6293.68

Optimal solution found (tolerance 1.00e-04)
 Best objective 6.293684932492e+03, best bound 6.294035127993e+03, gap 0.0056%
 Set parameter NonConvex to value 2

Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (win64 - Windows 11.0 (22631.2))

CPU model: 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz, instruction set [SSE2|AVX|AVX2|AVX512]
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 3 rows, 19 columns and 27 nonzeros

Model fingerprint: 0x5f147f11

Model has 2 quadratic constraints

Variable types: 17 continuous, 2 integer (0 binary)

Coefficient statistics:

Matrix range [1e+00, 4e+02]
QMatrix range [1e+00, 1e+00]
QLMatrix range [1e+00, 1e+00]
Objective range [1e+00, 1e+00]
Bounds range [1e+01, 5e+01]
RHS range [0e+00, 0e+00]

Presolve removed 0 rows and 2 columns

Presolve time: 0.00s

Presolved: 9 rows, 17 columns, 53 nonzeros

Presolved model has 2 bilinear constraint(s)

Solving non-convex MIQCP

Variable types: 15 continuous, 2 integer (0 binary)

Root relaxation: objective 8.998308e+03, 9 iterations, 0.00 seconds (0.00 work units)

Nodes		Current Node			Objective Bounds			Work	
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
0	0	8998.30837	0	3	- 8998.30837	-	-	-	0s
0	0	7068.42112	0	4	- 7068.42112	-	-	-	0s
0	0	6370.64924	0	4	- 6370.64924	-	-	-	0s
0	0	6363.35431	0	2	- 6363.35431	-	-	-	0s
0	0	6303.85631	0	4	- 6303.85631	-	-	-	0s
0	0	6292.20611	0	2	- 6292.20611	-	-	-	0s
0	0	6281.26766	0	4	- 6281.26766	-	-	-	0s
0	0	6274.20282	0	4	- 6274.20282	-	-	-	0s
0	0	6272.80930	0	4	- 6272.80930	-	-	-	0s
0	0	6269.13043	0	3	- 6269.13043	-	-	-	0s
H	0	0			6268.4976597	6269.06827	0.01%	-	0s

Cutting planes:

MIR: 1

Explored 1 nodes (26 simplex iterations) in 0.07 seconds (0.00 work units)
Thread count was 8 (of 8 available processors)

Solution count 1: 6268.5

Optimal solution found (tolerance 1.00e-04)
Best objective 6.268497659728e+03, best bound 6.269068273168e+03, gap 0.0091%
Set parameter NonConvex to value 2
Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (win64 - Windows 11.0
(22631.2))

CPU model: 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz, instruction set
[SSE2|AVX|AVX2|AVX512]
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 3 rows, 19 columns and 27 nonzeros
Model fingerprint: 0xabfb9a74
Model has 2 quadratic constraints
Variable types: 17 continuous, 2 integer (0 binary)

Coefficient statistics:
Matrix range [1e+00, 4e+02]
QMatrix range [1e+00, 1e+00]
QLMatrix range [1e+00, 1e+00]
Objective range [1e+00, 1e+00]
Bounds range [1e+01, 5e+01]
RHS range [0e+00, 0e+00]

Presolve removed 0 rows and 2 columns
Presolve time: 0.00s
Presolved: 9 rows, 17 columns, 53 nonzeros
Presolved model has 2 bilinear constraint(s)

Solving non-convex MIQCP

Variable types: 15 continuous, 2 integer (0 binary)

Root relaxation: objective 9.006724e+03, 9 iterations, 0.00 seconds (0.00 work units)

Nodes		Current Node			Objective Bounds			Work	
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
0	0	9006.72413	0	3	- 9006.72413	-	-	-	0s
0	0	7073.37571	0	4	- 7073.37571	-	-	-	0s
0	0	6380.86704	0	4	- 6380.86704	-	-	-	0s
0	0	6373.50277	0	2	- 6373.50277	-	-	-	0s
0	0	6312.32566	0	4	- 6312.32566	-	-	-	0s
0	0	6300.62124	0	2	- 6300.62124	-	-	-	0s
0	0	6288.44640	0	4	- 6288.44640	-	-	-	0s

	0	0	6280.70317	0	4	-	6280.70317	-	-	0s
	0	0	6279.56180	0	4	-	6279.56180	-	-	0s
	0	0	6275.55366	0	3	-	6275.55366	-	-	0s
H	0	0				6274.9385601	6275.50776	0.01%	-	0s

Cutting planes:

MIR: 1

Explored 1 nodes (26 simplex iterations) in 0.04 seconds (0.00 work units)
 Thread count was 8 (of 8 available processors)

Solution count 1: 6274.94

Optimal solution found (tolerance 1.00e-04)

Best objective 6.274938560079e+03, best bound 6.275507759308e+03, gap 0.0091%

Set parameter NonConvex to value 2

Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (win64 - Windows 11.0 (22631.2))

CPU model: 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz, instruction set [SSE2|AVX|AVX2|AVX512]

Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 3 rows, 19 columns and 27 nonzeros

Model fingerprint: 0xfb27fb96

Model has 2 quadratic constraints

Variable types: 17 continuous, 2 integer (0 binary)

Coefficient statistics:

Matrix range	[1e+00, 4e+02]
QMatrix range	[1e+00, 1e+00]
QLMatrix range	[1e+00, 1e+00]
Objective range	[1e+00, 1e+00]
Bounds range	[1e+01, 5e+01]
RHS range	[0e+00, 0e+00]

Presolve removed 0 rows and 2 columns

Presolve time: 0.00s

Presolved: 9 rows, 17 columns, 53 nonzeros

Presolved model has 2 bilinear constraint(s)

Solving non-convex MIQCP

Variable types: 15 continuous, 2 integer (0 binary)

Root relaxation: objective 9.018189e+03, 9 iterations, 0.00 seconds (0.00 work units)

Nodes		Current Node			Objective Bounds			Work	
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time

0	0	9018.18912	0	3	-	9018.18912	-	-	0s
0	0	7080.74389	0	4	-	7080.74389	-	-	0s
0	0	6393.08007	0	4	-	6393.08007	-	-	0s
0	0	6385.61135	0	2	-	6385.61135	-	-	0s
0	0	6322.81537	0	4	-	6322.81537	-	-	0s
0	0	6311.21297	0	2	-	6311.21297	-	-	0s
0	0	6298.16561	0	4	-	6298.16561	-	-	0s
0	0	6289.80173	0	4	-	6289.80173	-	-	0s
0	0	6288.85229	0	4	-	6288.85229	-	-	0s
0	0	6285.13107	0	3	-	6285.13107	-	-	0s
H	0	0			6284.7303991	6285.08029	0.01%	-	0s

Cutting planes:

MIR: 1

Explored 1 nodes (26 simplex iterations) in 0.03 seconds (0.00 work units)

Thread count was 8 (of 8 available processors)

Solution count 1: 6284.73

Optimal solution found (tolerance 1.00e-04)

Best objective 6.284730399124e+03, best bound 6.285080285963e+03, gap 0.0056%

Set parameter NonConvex to value 2

Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (win64 - Windows 11.0 (22631.2))

CPU model: 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz, instruction set [SSE2|AVX|AVX2|AVX512]

Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 3 rows, 19 columns and 27 nonzeros

Model fingerprint: 0x0e315820

Model has 2 quadratic constraints

Variable types: 17 continuous, 2 integer (0 binary)

Coefficient statistics:

Matrix range [1e+00, 4e+02]

QMatrix range [1e+00, 1e+00]

QLMatrix range [1e+00, 1e+00]

Objective range [1e+00, 1e+00]

Bounds range [1e+01, 5e+01]

RHS range [0e+00, 0e+00]

Presolve removed 0 rows and 2 columns

Presolve time: 0.00s

Presolved: 9 rows, 17 columns, 53 nonzeros

Presolved model has 2 bilinear constraint(s)

Solving non-convex MIQCP

Variable types: 15 continuous, 2 integer (0 binary)

Root relaxation: objective 9.014647e+03, 9 iterations, 0.00 seconds (0.00 work units)

Nodes		Current Node			Objective Bounds			Work	
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
0	0	9014.64742	0	3	- 9014.64742	-	-	-	0s
0	0	7078.07550	0	4	- 7078.07550	-	-	-	0s
0	0	6390.34650	0	4	- 6390.34650	-	-	-	0s
0	0	6382.87815	0	2	- 6382.87815	-	-	-	0s
0	0	6320.20726	0	4	- 6320.20726	-	-	-	0s
0	0	6308.64502	0	2	- 6308.64502	-	-	-	0s
0	0	6295.73667	0	4	- 6295.73667	-	-	-	0s
0	0	6287.42093	0	4	- 6287.42093	-	-	-	0s
0	0	6286.44737	0	4	- 6286.44737	-	-	-	0s
0	0	6282.67159	0	3	- 6282.67159	-	-	-	0s
H	0	0			6282.2659797	6282.61579	0.01%	-	0s

Cutting planes:

MIR: 1

Explored 1 nodes (26 simplex iterations) in 0.04 seconds (0.00 work units)
Thread count was 8 (of 8 available processors)

Solution count 1: 6282.27

Optimal solution found (tolerance 1.00e-04)

Best objective 6.282265979733e+03, best bound 6.282615785713e+03, gap 0.0056%
Set parameter NonConvex to value 2

Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (win64 - Windows 11.0
(22631.2))

CPU model: 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz, instruction set
[SSE2|AVX|AVX2|AVX512]

Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 3 rows, 19 columns and 27 nonzeros

Model fingerprint: 0x3248fff7

Model has 2 quadratic constraints

Variable types: 17 continuous, 2 integer (0 binary)

Coefficient statistics:

Matrix range [1e+00, 4e+02]
QMatrix range [1e+00, 1e+00]
QLMatrix range [1e+00, 1e+00]
Objective range [1e+00, 1e+00]

Bounds range [1e+01, 5e+01]
 RHS range [0e+00, 0e+00]
 Presolve removed 0 rows and 2 columns
 Presolve time: 0.00s
 Presolved: 9 rows, 17 columns, 53 nonzeros
 Presolved model has 2 bilinear constraint(s)

Solving non-convex MIQCP

Variable types: 15 continuous, 2 integer (0 binary)

Root relaxation: objective 9.024211e+03, 9 iterations, 0.00 seconds (0.00 work units)

Nodes		Current Node			Objective Bounds			Work	
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
0	0	9024.21063	0	3	- 9024.21063	-	-	-	0s
0	0	7084.51220	0	4	- 7084.51220	-	-	-	0s
0	0	6399.42460	0	4	- 6399.42460	-	-	-	0s
0	0	6391.89535	0	2	- 6391.89535	-	-	-	0s
0	0	6328.23420	0	4	- 6328.23420	-	-	-	0s
0	0	6316.69142	0	2	- 6316.69142	-	-	-	0s
0	0	6303.20827	0	4	- 6303.20827	-	-	-	0s
0	0	6294.52165	0	4	- 6294.52165	-	-	-	0s
0	0	6293.66975	0	4	- 6293.66975	-	-	-	0s
0	0	6290.24497	0	3	- 6290.24497	-	-	-	0s
H	0	0			6289.8406413	6290.19023	0.01%	-	0s

Cutting planes:

MIR: 1

Explored 1 nodes (26 simplex iterations) in 0.03 seconds (0.00 work units)

Thread count was 8 (of 8 available processors)

Solution count 1: 6289.84

Optimal solution found (tolerance 1.00e-04)

Best objective 6.289840641322e+03, best bound 6.290190226872e+03, gap 0.0056%

Set parameter NonConvex to value 2

Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (win64 - Windows 11.0 (22631.2))

CPU model: 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz, instruction set [SSE2|AVX|AVX2|AVX512]

Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 3 rows, 19 columns and 27 nonzeros

Model fingerprint: 0x7f09d36e
 Model has 2 quadratic constraints
 Variable types: 17 continuous, 2 integer (0 binary)

Coefficient statistics:
 Matrix range [1e+00, 3e+02]
 QMatrix range [1e+00, 1e+00]
 QLMatrix range [1e+00, 1e+00]
 Objective range [1e+00, 1e+00]
 Bounds range [1e+01, 5e+01]
 RHS range [0e+00, 0e+00]

Presolve removed 0 rows and 2 columns
 Presolve time: 0.00s
 Presolved: 9 rows, 17 columns, 53 nonzeros
 Presolved model has 2 bilinear constraint(s)

Solving non-convex MIQCP

Variable types: 15 continuous, 2 integer (0 binary)

Root relaxation: objective 9.023229e+03, 9 iterations, 0.00 seconds (0.00 work units)

Nodes		Current Node			Objective Bounds			Work	
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
0	0	9023.22879	0	3	- 9023.22879	-	-	-	0s
0	0	7083.69345	0	4	- 7083.69345	-	-	-	0s
0	0	6398.76981	0	4	- 6398.76981	-	-	-	0s
0	0	6391.22981	0	2	- 6391.22981	-	-	-	0s
0	0	6327.55880	0	4	- 6327.55880	-	-	-	0s
0	0	6316.06273	0	2	- 6316.06273	-	-	-	0s
0	0	6302.66060	0	4	- 6302.66060	-	-	-	0s
0	0	6293.97700	0	4	- 6293.97700	-	-	-	0s
0	0	6293.11358	0	4	- 6293.11358	-	-	-	0s
0	0	6289.67299	0	3	- 6289.67299	-	-	-	0s
H	0	0			6289.2628843	6289.61242	0.01%	-	0s

Cutting planes:
 MIR: 1

Explored 1 nodes (26 simplex iterations) in 0.05 seconds (0.00 work units)
 Thread count was 8 (of 8 available processors)

Solution count 1: 6289.26

Optimal solution found (tolerance 1.00e-04)
 Best objective 6.289262884324e+03, best bound 6.289612417146e+03, gap 0.0056%
 Set parameter NonConvex to value 2

Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (win64 - Windows 11.0 (22631.2))

CPU model: 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz, instruction set [SSE2|AVX|AVX2|AVX512]
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 3 rows, 19 columns and 27 nonzeros

Model fingerprint: 0xad705e80

Model has 2 quadratic constraints

Variable types: 17 continuous, 2 integer (0 binary)

Coefficient statistics:

Matrix range [1e+00, 3e+02]
QMatrix range [1e+00, 1e+00]
QLMatrix range [1e+00, 1e+00]
Objective range [1e+00, 1e+00]
Bounds range [1e+01, 5e+01]
RHS range [0e+00, 0e+00]

Presolve removed 0 rows and 2 columns

Presolve time: 0.00s

Presolved: 9 rows, 17 columns, 53 nonzeros

Presolved model has 2 bilinear constraint(s)

Solving non-convex MIQCP

Variable types: 15 continuous, 2 integer (0 binary)

Root relaxation: objective 9.028511e+03, 9 iterations, 0.00 seconds (0.00 work units)

Nodes		Current Node			Objective Bounds			Work	
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
0	0	9028.51063	0	3	- 9028.51063	-	-	-	0s
0	0	7087.10860	0	4	- 7087.10860	-	-	-	0s
0	0	6404.32643	0	4	- 6404.32643	-	-	-	0s
0	0	6396.73729	0	2	- 6396.73729	-	-	-	0s
0	0	6332.34672	0	4	- 6332.34672	-	-	-	0s
0	0	6320.90679	0	2	- 6320.90679	-	-	-	0s
0	0	6307.13958	0	4	- 6307.13958	-	-	-	0s
0	0	6298.18261	0	4	- 6298.18261	-	-	-	0s
0	0	6297.40087	0	4	- 6297.40087	-	-	-	0s
0	0	6294.20999	0	3	- 6294.20999	-	-	-	0s
H	0	0			6293.7965226	6294.14580	0.01%	-	0s

Cutting planes:

MIR: 1

Explored 1 nodes (26 simplex iterations) in 0.03 seconds (0.00 work units)
Thread count was 8 (of 8 available processors)

Solution count 1: 6293.8

Optimal solution found (tolerance 1.00e-04)
Best objective 6.293796522618e+03, best bound 6.294145801774e+03, gap 0.0055%

```
[11]: fig, ax = plt.subplots(figsize=(8,5))
ax.plot(np.arange(0, len(object_val)), object_val, color='red',
        ↳linestyle='dotted', label='Robust Objective Sequence')
plt.hlines(xmin=0, xmax=len(object_val), y=obj_nominal, linestyle='--',
        ↳color='black', label='Nominal Objective')
plt.grid()
plt.legend()
plt.savefig('OnlineLearningConvergence.png')
plt.show()
```

```
[12]: print(object_val)
```

```
[5712.244245002217, 6224.284373227671, 6284.210466879615, 6293.684932491939,
6268.497659728031, 6274.938560079021, 6284.730399124142, 6282.265979732711,
6289.840641321726, 6289.2628843242655, 6293.796522617704]
```

```
[13]: object_val = np.array(object_val)
perc_var = (object_val-obj_nominal)/obj_nominal
print(perc_var, sum(perc_var)/len(perc_var))
```

```
[-0.10396169 -0.02364167 -0.0142415 -0.0127553 -0.01670625 -0.01569591
-0.01415994 -0.01454651 -0.01335833 -0.01344896 -0.0127378 ]
-0.023204896049496715
```

```
[ ]:
```