LinYu_HW5

May 12, 2024

1 HGEN HW5

[1]:
```
#pip install cvxpy
import cvxpy as cp
import numpy as np
#pip install cvxopt
from cvxopt import matrix, solvers
from sklearn.linear_model import Lasso
```

1.1 Problem 1 Dantzig Selector

1.1.1 PART 1

Recast the Dantzig Selector into Linear Programming

Recall from the lecture notes, A LP problem is an optimization problem of the form:

Minimize:

$$c^T x$$

Subject to:

$$Ax = b$$

$$x \geq 0$$

Linear Programming Formulation Using Auxiliary Variables Use auxiliary variables (z_j), where:

$$z_j \geq \beta_j \quad \text{and} \quad z_j \geq -\beta_j$$

Objective: Minimize the sum of the auxiliary variables:

$$\sum_j z_j$$

Subject to: For each (j):

$$z_j \geq \beta_j \quad \text{and} \quad z_j \geq -\beta_j$$

For each (i):

$$-\lambda \leq \sum_j X_{ij}^T (X\beta - y)_j \leq \lambda$$

[2]:
```python
# Simulate data (so we can test all our methods)
np.random.seed(1)
n = 200  # number of observations
p = 600  # number of predictors
X = np.random.randn(n, p)
beta_true = np.random.randn(p) * (np.random.rand(p) < 0.1)  # sparse true␣
↪coefficients
y = X @ beta_true + np.random.randn(n)  # generate response
```

[3]:
```python
# Use package cvxpy to solve this convex problem
# Dimensions
lambda_val =1
# Define variables
beta = cp.Variable(p)
z = cp.Variable(p, nonneg=True)  # Auxiliary variable for the L1 norm

# Objective function
objective = cp.Minimize(cp.sum(z))

# Constraints
constraints = [z >= beta, z >= -beta]  # Enforces z_j >= |beta_j|

# Constructing constraint for ||X^T(X*beta - y)||_infty <= lambda
# (Python has the package for this norm constraints so we don't need to␣
↪translate it into inequality as
# show in part a)
r = X @ beta - y
XTr = X.T @ r
constraints += [cp.norm(XTr, 'inf') <= lambda_val]

# Problem setup for Dantzig Selector
problem_Dantzig = cp.Problem(objective, constraints)

# Solve the problem
problem_Dantzig.solve()

beta_Dantzig=beta.value
# Print the results
print("Status:", problem_Dantzig.status)
print("Optimal value:", problem_Dantzig.value)
print("Optimal beta:", beta_Dantzig)
```

```
/Users/linyu/opt/anaconda3/envs/pycourse/lib/python3.8/site-
packages/cvxpy/reductions/solvers/solving_chain.py:336: FutureWarning:
    Your problem is being solved with the ECOS solver by default. Starting in
    CVXPY 1.5.0, Clarabel will be used as the default solver instead. To
continue
    using ECOS, specify the ECOS solver explicitly using the ``solver=cp.ECOS``
```

```
    argument to the ``problem.solve`` method.

  warnings.warn(ECOS_DEPRECATION_MSG, FutureWarning)
Status: optimal
Optimal value: 59.073695270978945
Optimal beta: [-9.79236430e-13  1.06120654e-01  2.81761710e-14  2.97815762e-12
  1.89021635e-13  9.29757957e-12 -7.15677921e-04  2.90091530e-01
  5.89827434e-11  7.30840515e-13 -9.36744778e-02  3.63614869e-12
 -1.08110297e-12 -1.13556065e-12  1.64967363e-11 -7.38431161e-13
 -1.03792108e-11  1.08475191e-12 -6.51350219e-02  2.59412757e-01
 -6.28947233e-12  1.28766663e-11  3.64014800e-12 -3.80939429e-12
 -7.01912726e-13  1.28814187e-12  1.60288430e-12 -4.10233200e-14
  1.46811680e-02 -1.11011343e-01  2.31269603e-12 -8.80244061e-14
 -3.46466452e-11 -5.98967225e-13 -1.60152320e-01 -1.26946247e-12
  1.62671422e-13 -1.21222249e-12 -2.35587204e-12  2.99872583e-12
 -3.30903423e-12  2.39442090e-01 -2.38153790e-11 -3.84108792e-12
 -5.03968885e-02 -4.88432317e-13  7.89258445e-01  3.38708512e-12
  2.95404424e-11 -3.02302148e-12 -6.14176719e-12 -1.09358589e-11
  3.37104066e-01  1.94247687e-03  3.54779340e-12 -1.11895153e-01
  6.78105022e-12 -3.62207410e-13 -1.91908336e-12  1.14214095e-02
 -1.34385783e-01 -3.40083546e-13 -1.76006608e-12 -6.03462806e-02
 -5.58821173e-12  3.74166907e-12  1.17956149e-12 -2.51492570e-12
 -1.01048401e-11 -2.65340390e-02  5.31675444e-13  1.62379585e-12
 -9.96581444e-12  4.61466779e-01 -4.01723201e-13 -5.31262530e-01
  6.23298643e-02  8.41643069e-02  1.91765275e-01  2.18658631e-13
 -4.40096112e-13  2.12019834e+00 -2.28654688e-12 -4.11044757e-13
  1.52925412e+00 -1.36768459e-12  2.40983564e-13  6.19828592e-02
 -2.98818038e-02  4.17153790e-01  3.17935194e-12 -2.38631267e-10
  4.76160211e-11  2.65007830e-12  1.32536065e+00 -3.88818955e-12
 -8.13496805e-03 -7.17692526e-02  4.20341960e-10  4.09242902e-12
  1.89227175e-12  1.53034737e-12 -1.91853983e-11 -4.31887734e-03
 -1.15266885e-12  1.09639373e-01  7.35787135e-14  5.26539159e-12
 -2.55830842e-01  1.19821872e-10 -6.13158045e-11 -6.32031890e-13
 -8.11034289e-01 -9.94486334e-13 -1.28697653e-01 -1.48170111e-12
  6.09699659e-13  1.01192901e-12  2.17926649e-12  4.88204155e-02
  9.23083521e-12  1.25917007e-13 -5.99921086e-12  2.31795723e-02
  1.47964461e-10  2.20196539e-13 -1.79236182e-01 -3.08857959e-12
  2.92366733e-12 -5.68137902e-12  4.52139362e-12  1.50967336e-01
 -2.80145952e-13 -1.81326098e-12 -2.38639381e-12  1.05988910e-11
  6.77730158e-02 -5.09760471e-12  2.02511035e-01 -3.30581568e-12
  1.36679391e-12 -5.13030715e-02  1.16362313e-11  1.19512548e-01
 -1.40032584e-04 -3.42819190e-01 -2.31305507e-12  5.40180154e-13
  3.98781051e-10  9.94115432e-12  2.60511968e-01 -1.39485077e-02
 -7.48942892e-02  1.11231554e-11 -1.33569189e-01  4.29964087e-01
 -2.36956234e-12 -8.02675821e-02  2.87477405e-01  8.56268903e-01
  1.14754617e-12  9.04573887e-12 -3.15207388e-12 -1.83273943e-12
  5.68375931e-02  1.71867478e-11 -2.98265412e-12  1.35451871e+00
```

2.61766709e-01	1.36016931e-11	3.58678603e-15	-4.38563329e-12
5.85278034e-12	-8.77331804e-12	-4.10830149e-02	-1.15509895e+00
1.28829746e-12	-1.61265665e-12	-6.05741814e-12	-1.75614754e-02
-5.55901994e-12	3.06909828e-12	-9.67313223e-12	2.94988539e-12
1.82125762e-12	2.74658496e-12	1.18045526e-12	7.80194536e-13
-2.22731628e-11	2.76455399e-12	-1.18296561e-12	9.14009854e-13
-9.64259282e-13	-6.94161700e-12	1.78748843e-12	4.11394150e-11
-1.59208671e-11	-7.58714945e-13	-8.01886910e-03	9.51134465e-02
-3.82802475e-12	1.09371861e-10	-1.11286347e-12	-6.88426804e-02
-2.19310021e-01	-8.40337580e-13	-2.18972862e-12	6.39611727e-12
9.49693486e-02	-1.08893059e-12	-5.96420602e-13	-2.70260696e-12
-1.44567283e-12	7.26315894e-13	-1.87239347e-12	-1.53279132e-01
-6.42623359e-13	-1.43184345e-02	9.90506584e-13	-1.86788330e-10
-9.12434279e-12	-1.98606439e-01	-2.14767716e-12	6.53529363e-12
-1.20052997e-12	-3.96742477e-13	-4.41010436e-12	-3.54131001e-02
3.26399922e-12	-3.32653530e-02	-2.89772541e-12	-1.90093930e-01
-3.54517864e-13	-4.10324790e-13	-5.41003874e-01	-1.84397595e-13
-2.92880929e-01	-5.74083315e-13	-7.27207151e-02	6.59082748e-13
-6.92005839e-14	-6.41668627e-12	-8.51853288e-13	3.59291609e-02
-2.67290724e-12	-2.10011137e-12	-5.53776031e-12	-3.39594981e-12
9.76476728e-02	9.68251128e-02	1.11209645e-01	8.34753616e-13
-1.51662584e-12	-3.20868890e-13	-5.31030635e-01	-7.81715109e-12
-1.31479648e-11	-1.31254225e-01	-1.85804690e-12	6.31141898e-02
-9.00255845e-11	1.75977666e-12	1.94616898e-11	2.09914904e-12
4.55265188e-12	-3.34790572e-01	1.22215024e-12	-1.69759318e-12
-1.88174338e+00	2.83345626e-01	6.80065328e-12	6.17634989e-13
1.73019618e-12	-3.75903907e-01	2.60600089e-12	1.68412511e-01
2.93870305e-12	7.31020576e-12	-4.12661923e-12	-8.17282119e-02
-8.42238201e-12	-1.77792506e-02	1.61131125e-01	-7.91654401e-02
-2.87631746e-12	-1.27647049e-01	8.52527760e-13	-1.18723734e-11
-6.25060244e-12	-2.70909226e-12	2.31529026e-01	-4.03168620e-02
4.04875723e-01	-1.51530262e-01	2.09039150e-12	6.26283778e-12
1.62521199e-12	1.55907249e-12	-1.85132682e-01	1.22975208e-01
-1.53973207e-12	6.61551673e-12	6.75924262e-13	-1.21793473e-01
3.92992856e-12	-7.04734553e-12	-8.36241791e-14	5.05828834e-01
-4.30751747e-12	-1.02619182e-12	-1.86511776e+00	-1.88795749e-12
1.03039083e-01	-6.75653413e-12	-2.17498417e-02	6.79877517e-01
5.46580004e-11	1.23723087e-10	-1.17922146e-12	2.16541198e-12
-9.91893071e-13	-1.52945824e-01	1.55155317e-12	7.69787407e-12
8.01670077e-02	-1.27230113e-02	-2.16157813e-02	4.03244230e-12
1.04478765e-12	2.16232171e-12	-1.19550734e-12	1.11952221e+00
-3.03333302e-12	1.34810024e-12	8.77724526e-13	-4.72958312e-12
7.48177389e-02	2.74366487e-12	-5.08824376e-12	-4.82054807e-02
1.16233496e-12	-4.47702313e-13	4.65950656e-11	-3.81469803e-12
3.69134542e-14	1.75469638e-11	-6.06113119e-13	-5.77818558e-12
-8.76314887e-01	-1.05339786e-12	1.66628153e-11	1.41673515e-02
-9.68848605e-03	1.57966351e-12	4.98446184e-12	1.60935722e-12
-1.60596938e-01	3.19468336e-02	1.74659177e-12	-3.14127303e-12

-7.31610392e-13	-1.63467793e-11	2.57949706e-12	-5.51580620e-12
1.68427898e-11	3.99287191e-12	-7.79081384e-02	1.35076048e-01
-7.30735485e-13	1.26975562e-12	1.07244122e-13	2.94899187e-01
-5.25873240e-01	2.70445729e-13	6.65478515e-13	-3.65414465e-12
-2.86899280e-12	-1.61175466e-12	-3.63116032e-02	6.62474925e-13
9.35972130e-02	1.59854425e-12	1.17011000e-01	4.50497527e-12
-1.97037923e+00	-1.35527147e-12	-9.38222650e-01	8.44894814e-13
6.89773974e-13	4.22439637e-12	1.24595966e-01	-6.58621887e-12
-3.30091837e-03	2.38585651e-12	-6.81754828e-13	7.71809716e-13
1.07394792e-12	-4.02158814e-12	-7.38789408e-14	-2.12662705e-11
5.34000676e-12	-1.13892479e+00	-8.52783648e-03	-2.04267847e-12
-3.10966336e-03	3.55498308e-12	9.96717518e-13	-2.60559786e-12
2.97617380e-12	-5.30876054e-12	-1.91200537e-01	-4.59427614e-12
-9.43342144e-01	8.56581183e-13	1.68610844e-01	-1.46925155e-12
-1.40638804e-02	1.33159211e-12	-9.52376365e-13	-4.44057462e-02
-1.00047690e-12	-1.59312768e+00	-5.76653953e-13	-1.28638266e-12
-2.03075815e-01	8.76866934e-03	-2.89886653e-13	-2.45612560e-13
1.30054261e-02	5.59821474e-02	5.44927915e-03	3.33569404e-12
-1.00321031e-11	2.68773624e-12	7.50461173e-13	-4.00660620e-02
2.00499731e-01	1.09438760e-12	-3.42441882e-11	-1.04947055e-12
-3.51323511e-13	2.30725730e-12	3.18757534e-12	-1.01981554e-11
-4.69332554e-13	-1.05299880e-01	7.75401445e-02	-1.65347814e-12
-3.12044782e-01	4.37727311e-13	-1.01707635e-11	1.68503008e-12
-4.15347206e-12	1.76400848e-01	1.00296991e-12	-1.87517812e-01
3.92855418e-12	1.02898474e-01	4.98300803e-13	2.89930313e-12
4.42923101e-12	4.61182451e-12	-1.27870508e-12	1.04010973e-11
-1.43838563e-12	1.18483301e-01	3.35094623e-13	-4.00118734e-12
3.30860427e-13	-7.24141042e-02	4.71578211e-12	7.34326235e-12
-1.92514528e-11	1.48190139e+00	3.82009064e-12	8.09770843e-02
-4.53703192e-13	2.78874982e-12	1.76886717e+00	-9.27730308e-01
7.98727818e-01	2.78658487e-13	-5.11027259e-12	-3.85422714e-01
3.10475629e-02	-3.81911774e-12	-2.54036236e-12	1.05195188e-12
-1.56971959e-12	4.83010651e-12	-1.14618213e-14	-2.74158219e-11
-1.35149283e-12	-7.61672018e-02	-2.05048487e-01	2.92759730e-11
1.22282147e-01	8.16252944e-13	1.02301556e-01	-1.36327769e-11
-4.08816942e-02	3.01010267e-12	9.61157393e-12	-2.30104079e-13
-4.03787378e-14	-7.97881102e-13	-1.11184292e-13	1.03024615e-11
2.20268417e-13	-3.63282690e-02	-9.25653571e-13	-1.28844737e+00
-6.10858754e-12	-1.52880675e+00	-2.17419907e-11	3.26500439e-02
5.30822421e-02	1.66377766e-12	1.90544512e-12	-1.05301866e-01
1.75958141e-01	-1.48362139e-01	-8.80375705e-12	-2.10162597e-01
-7.97254592e-01	-1.72754639e-12	-5.32249769e-12	2.14691560e-11
-2.88397231e-12	5.52543488e-13	-2.17034893e-01	3.17179861e-01
3.71714549e-12	9.82839103e-13	-1.36069339e-12	1.60468429e-12
5.03265737e-12	1.21400130e-12	-1.32908683e-11	-1.16933093e-01
-2.09940613e-12	-7.99128996e-12	-2.41641031e-11	-9.92784177e-12
-9.78136015e-02	-1.57789687e-12	-6.17861327e-02	7.79751367e-12
-2.60363998e-11	2.74628595e-01	4.15512828e-12	-1.15410831e-12

```
 -1.01678366e+00  3.99152237e-12  8.86864548e-02  3.50960821e-13
 -1.48665573e-12 -1.07697554e+00  4.66630480e-13 -6.38947504e-13
 -1.79921041e-12 -4.90638069e-12  1.92850818e-12 -1.38658709e-11
  4.16612390e-13 -1.32352016e+00  4.40657423e-01  2.45597242e-12
  5.29046302e-13 -4.81256535e-12  1.46117072e-12  1.19287549e-01
  2.56079698e-02 -3.72057662e-12  2.89552898e-13  9.28032670e-13
  2.06940200e-12 -2.54552451e-12 -7.12890146e-02 -2.27324431e-12
  4.54521861e-12 -1.46423092e-12 -1.76958739e-11 -4.84300922e-13
  1.02638764e-12  7.51761108e-12  3.28025759e-12 -1.38355181e-01
  1.36097325e+00 -3.06821868e-01  2.61642466e-02 -1.04477087e-01
 -2.40220577e-12  4.49632146e-02 -7.74529680e-02  2.52494327e-12
 -5.71842569e-13  1.79087543e-12  3.55689831e-01 -4.41104308e-12]
```

1.2 Question 2 Lasso

Solve Lasso with cvxpy package

[4]:
```
# Set up Lasso problem
lambda_lasso = 1.0
beta = cp.Variable(p)
loss = cp.sum_squares(y - X @ beta)
regularization = lambda_lasso * cp.norm1(beta)
objective = cp.Minimize(loss + regularization)
problem_lasso = cp.Problem(objective)

# Solve the problem
problem_lasso.solve()

# Results
lasso_beta_cvxpy = beta.value
```

[5]:
```
# Print the results for Lasso
print("Status:", problem_lasso.status)
print("Optimal value:", problem_lasso.value)
print("Optimal beta:", lasso_beta_cvxpy)
```

```
Status: optimal
Optimal value: 59.781247352119514
Optimal beta: [-4.21953950e-05  1.50399884e-01  2.66853853e-06  4.75087656e-05
 -1.46634518e-05 -3.83403400e-05 -1.57331635e-02  2.57258512e-01
 -1.16032092e-05  4.79353654e-05 -1.03724954e-01  3.76930465e-05
 -5.15254538e-05  1.59674444e-05 -2.51941772e-05  2.13687380e-05
  1.51527447e-05  5.08667096e-06 -3.37838937e-02  2.45847663e-01
 -7.10714835e-05 -3.93948035e-05 -1.31933571e-05  5.80698539e-06
  1.27814187e-05 -3.69511036e-05  5.86556954e-05  1.95747301e-05
  4.18814978e-02 -9.17713407e-02  1.41402023e-05 -2.86323380e-05
  5.06420340e-06  1.65076468e-05 -1.48428656e-01  3.01078126e-05
 -8.45666652e-05 -4.20996883e-05 -4.26249981e-06  1.62222482e-05
  4.44332672e-05  2.15400137e-01 -3.70371804e-02 -2.36826702e-05
```

-6.18584432e-02	-3.07562061e-05	8.24931360e-01	-4.61190133e-06
-1.92606633e-05	-7.98526534e-08	-3.93952447e-05	1.14541646e-04
3.44395920e-01	8.93424156e-03	-4.72344651e-05	-1.04403284e-01
-3.13042772e-05	4.57886423e-06	6.83158804e-05	1.27468326e-02
-1.65899627e-01	-2.35269385e-05	4.87874518e-05	-5.28804430e-02
4.67451734e-05	-7.54117930e-06	5.54173210e-05	-2.80665812e-05
-3.79477724e-03	-1.57901609e-02	-2.48065131e-05	5.98447566e-05
-1.58459930e-02	4.18636887e-01	1.52179617e-05	-5.22724697e-01
2.15589030e-02	7.84451744e-02	1.81414961e-01	4.86938887e-05
-4.12968991e-05	2.11150960e+00	-1.83602580e-05	-2.47064396e-06
1.52663745e+00	2.23282206e-05	-7.14660232e-05	3.21612357e-02
-5.77095354e-02	4.13346320e-01	3.63977054e-05	8.11212353e-05
-6.43181199e-06	5.98461822e-05	1.30987106e+00	-2.49476764e-05
-5.51306204e-03	-4.06413059e-02	8.01285369e-04	7.27109449e-06
-6.38700326e-05	-3.20026902e-05	-5.06220458e-05	3.66916302e-06
-2.19787364e-05	1.36622579e-01	3.23461220e-05	-4.90531412e-05
-3.44952581e-01	2.90402119e-03	-7.39743677e-03	5.67142527e-06
-8.22338913e-01	-7.30924323e-06	-9.45250153e-02	-3.64611231e-05
2.26970715e-06	-8.14982283e-06	3.59084033e-05	3.99880712e-02
-1.41061162e-05	-2.55284774e-05	-6.83025967e-05	6.10067200e-02
1.81987953e-02	4.33174716e-05	-2.01368840e-01	2.74090564e-05
2.67704336e-05	3.70026459e-05	1.61177979e-05	1.49775780e-01
3.09695319e-05	6.93314325e-05	1.43836060e-05	-3.62808675e-05
1.09107117e-01	4.91746934e-05	1.95789206e-01	-7.83622773e-05
4.09705630e-05	-8.52043002e-02	-4.72086516e-05	1.33140444e-01
-1.38198399e-02	-3.82581605e-01	-1.27287487e-05	-2.51076625e-05
-2.26410445e-05	-4.52832883e-06	2.54010697e-01	-1.07195342e-05
-2.90741689e-02	-1.20732175e-04	-1.03811115e-01	4.02324868e-01
-2.94251212e-05	-9.68726136e-02	3.04834485e-01	8.88310571e-01
-4.53454020e-05	-4.97535902e-05	-1.94692887e-05	7.90190719e-06
6.04498974e-02	-1.91064734e-05	-2.36272782e-06	1.37175592e+00
1.89725514e-01	-3.53114078e-05	-4.17137093e-05	3.10122343e-05
4.13176871e-06	2.43067680e-05	-7.71193292e-02	-1.18223862e+00
8.05767084e-05	1.64255645e-05	-2.14711799e-06	-8.78855082e-03
-4.17521677e-05	8.83082760e-06	7.45349236e-05	6.52522249e-06
-1.18227800e-05	-7.58493510e-05	3.02784202e-05	1.63402860e-06
4.35210367e-06	9.69728999e-06	-8.24275450e-06	5.56431964e-05
2.29101794e-05	5.41682862e-05	-3.62410701e-05	-2.11518201e-05
1.21482469e-05	6.00602008e-05	-3.43916033e-02	9.67451868e-02
-3.30783911e-05	5.16478045e-05	1.28194532e-06	-2.32042666e-02
-1.98086034e-01	-2.05449653e-05	5.01341589e-05	4.33929078e-05
7.11333228e-02	4.62550881e-05	-4.96170335e-05	-8.18112836e-05
1.74054484e-06	-2.67948376e-05	8.07815582e-05	-1.79558121e-01
8.62987532e-05	-2.73739503e-02	2.90141496e-05	-9.86724452e-05
-2.83208086e-05	-2.03698593e-01	-3.14257611e-05	8.11242674e-05
4.29849941e-05	-3.01901505e-05	-5.20192183e-05	-6.27481387e-02
-4.71939915e-05	-9.56322723e-03	-1.46389904e-05	-1.69493297e-01
-2.69611079e-05	5.96987505e-05	-5.33540912e-01	3.46624318e-05

-2.79226377e-01	3.41387204e-06	-7.19845162e-02	1.36374554e-05
-6.83599901e-05	6.92846228e-05	9.03740306e-05	3.76332570e-02
4.70126487e-05	1.41853676e-06	4.11576099e-05	7.75192126e-06
1.11358882e-01	6.10459677e-02	7.89342941e-02	-1.57029952e-05
-2.56096158e-05	2.00930253e-05	-5.27214578e-01	-2.39344038e-05
1.14739950e-04	-1.53795433e-01	-4.18679929e-05	4.47445588e-02
6.21625708e-05	-2.43576502e-05	7.59296993e-03	1.51991053e-05
7.86978634e-05	-3.47173450e-01	6.62102401e-05	1.57365877e-06
-1.87368019e+00	3.13593639e-01	4.65532693e-05	6.55253764e-06
9.49919207e-05	-3.89279055e-01	7.12161987e-05	1.26884518e-01
8.87775568e-06	-2.70842185e-05	-7.80777546e-06	-1.29393731e-01
-3.83528360e-05	-4.85183036e-02	2.19970905e-01	-9.95862015e-02
-1.85169969e-05	-1.28637427e-01	-1.42253447e-05	1.71582112e-05
1.92499762e-05	2.56970298e-05	2.34830911e-01	-2.55174779e-05
4.24944705e-01	-1.60530328e-01	-3.72260341e-05	-3.60935893e-05
-1.40123169e-05	-7.94209560e-06	-1.90284582e-01	1.22301440e-01
-5.40107955e-05	9.79321871e-06	-3.86804201e-05	-9.07052110e-02
-1.73909652e-05	-1.39441270e-05	8.61875554e-06	5.47091381e-01
2.28646947e-05	-4.49008540e-05	-1.83979018e+00	3.36700618e-05
8.20379458e-02	4.18905929e-05	-7.17839019e-04	6.73442697e-01
2.61004975e-02	1.92926406e-02	-2.77864379e-05	-8.93143566e-07
-5.97632627e-05	-1.52712410e-01	4.50535124e-05	5.43252147e-06
7.01013351e-02	-2.43425342e-02	3.66118984e-05	-1.17379237e-05
2.01066724e-05	-1.25278550e-05	-3.62132782e-06	1.11571418e+00
-2.25572265e-05	1.73504068e-05	1.32081619e-05	2.20610730e-06
6.54914732e-02	-1.35941965e-05	3.75359279e-06	-7.18209239e-02
-2.09597492e-05	-5.49743312e-05	-3.73101465e-05	-3.28613363e-05
-6.72833277e-06	9.41187298e-06	1.99637859e-05	-1.23380429e-05
-8.57288515e-01	-6.23330549e-05	-3.02875036e-05	2.76437727e-03
-3.07338266e-02	-7.29267575e-06	-4.04050411e-05	-1.78443394e-05
-1.27079319e-01	7.51644999e-02	-7.56226794e-05	3.39741324e-05
-4.04408542e-05	-3.14210386e-05	2.13368766e-05	-9.91622910e-06
9.41114035e-06	-1.09060387e-05	-8.24091241e-02	1.67716674e-01
-4.16001902e-05	3.72102369e-05	-3.95198881e-05	3.15484425e-01
-5.53007857e-01	-4.69026602e-05	4.16057262e-05	-2.35338651e-05
-5.98101886e-05	-2.30894161e-05	-5.31914939e-02	-8.02730821e-05
8.80441206e-02	-1.10454425e-05	7.96563829e-02	1.68081919e-05
-1.95238132e+00	-1.08123262e-05	-9.65946999e-01	2.20721301e-05
-2.21761623e-05	-1.53430709e-05	8.52719774e-02	-5.28806062e-05
-1.17721939e-02	3.99505344e-05	6.91209914e-06	7.43298182e-06
5.50025687e-05	1.58665327e-05	2.39492067e-05	-6.47638654e-06
-1.87065701e-05	-1.15406681e+00	-1.40268904e-02	3.95945488e-05
-2.27507476e-05	-2.32084727e-05	-1.12130637e-04	-2.35463209e-05
5.48677158e-05	2.11369460e-05	-2.07204550e-01	-1.18879187e-05
-9.40306467e-01	7.66147461e-06	1.47107206e-01	2.07308141e-05
-4.48277236e-02	1.21446139e-05	-3.62412059e-05	-4.58924944e-05
-4.73485185e-05	-1.59148473e+00	2.25539012e-05	1.20241706e-05
-2.11040373e-01	2.19120607e-02	6.96171368e-05	1.08509131e-05

```
-2.33049666e-05  4.51431027e-02  3.01139352e-02 -2.34383815e-06
 1.78141723e-06 -6.06317226e-06  3.61637736e-06 -4.50731558e-02
 2.08442290e-01 -6.53937344e-05 -1.19281808e-02 -3.58635181e-05
-3.67817377e-05  2.76277905e-06  7.95664380e-06 -3.00076944e-05
 3.63558499e-05 -1.33040986e-01  7.51932142e-02 -7.85706153e-06
-3.50774599e-01 -2.37204309e-05  6.87118129e-06  1.62787731e-06
-3.24827035e-05  1.84379612e-01 -3.94259203e-06 -2.25957405e-01
 2.26156549e-05  1.31319583e-01 -5.10259130e-05 -8.17525672e-05
 3.05145540e-05 -1.32345142e-05 -1.13086122e-05 -2.46270643e-05
-2.39335787e-05  1.29291319e-01 -5.16407176e-05 -7.01922044e-05
 7.46116286e-05 -6.95793724e-02 -6.01399148e-05  1.63481763e-05
-2.60097419e-05  1.51884504e+00  7.32301345e-05  5.79339552e-02
-6.38961239e-06  3.77136593e-06  1.78491224e+00 -9.39548715e-01
 8.30048303e-01  2.49409409e-05  4.82162182e-06 -3.92831169e-01
 2.06532486e-02  2.33288902e-05 -1.58803101e-05 -2.48741872e-05
 6.31697041e-05 -1.19292218e-05 -1.55731156e-05  2.41910324e-05
 3.65444483e-05 -7.85335864e-02 -1.87358133e-01  2.87148311e-02
 1.01138066e-01  2.57737014e-05  1.44648684e-01  2.77089829e-05
-3.24411614e-02 -7.11608503e-06  8.65232173e-05 -3.13885604e-05
 1.20296265e-05  5.66010659e-05  1.81101651e-05 -5.51150821e-05
 2.48275846e-05 -7.34327705e-05 -5.99750730e-05 -1.28554847e+00
-7.18650848e-05 -1.52107202e+00  2.66244162e-05  4.78306135e-02
 7.87496744e-02 -1.74236936e-05  2.60991088e-06 -8.01762139e-02
 1.90452222e-01 -1.76153075e-01 -4.40272596e-05 -2.28159119e-01
-7.69622715e-01  3.87780624e-05 -4.94341930e-06 -9.32844193e-06
-2.68327668e-05  6.00723783e-05 -1.61107393e-01  3.39884237e-01
-7.73896547e-05 -3.61883041e-05 -6.08911056e-05 -4.04612883e-05
 1.72228146e-05 -6.42858213e-06  1.52918343e-05 -8.37351424e-02
 8.15463938e-06 -6.02987726e-05  8.13510513e-05  5.21355162e-05
-8.89301927e-02  1.80842612e-05 -1.03469676e-01  2.25755077e-05
 8.82627036e-06  2.56398427e-01 -1.44190613e-05 -1.82314851e-05
-1.06997942e+00 -5.94751975e-06  6.80846390e-02  2.35165893e-05
-4.55466898e-05 -1.11317710e+00  8.09188530e-05  2.43694553e-05
 2.83797011e-05  2.48200713e-06 -2.02506520e-05  3.73920844e-05
 4.40048841e-05 -1.33300981e+00  4.02910434e-01 -4.05493338e-05
-4.02683769e-05 -6.50186541e-05  2.67977705e-05  1.16530457e-01
 3.89248167e-03 -8.86697218e-06  4.42109190e-05  7.27263832e-05
 4.54376845e-05 -4.08820056e-05 -6.97235443e-02 -1.75305463e-05
 1.30351102e-05  2.01051210e-05 -7.58556796e-05  4.36676262e-05
-3.52686254e-05 -2.29900539e-05  4.28090202e-05 -1.30444945e-01
 1.36272295e+00 -3.01988756e-01 -1.76050353e-05 -1.10772590e-01
-3.26770282e-05  5.39546138e-02 -1.03006280e-01 -2.14221470e-05
 5.15650463e-05  1.18313247e-05  3.54139481e-01 -4.38586876e-05]
```

Benchmark lasso solver from cvxpy package with Lasso from sklearn.linear_model

[6]:
```
from sklearn.linear_model import Lasso
```

```
lambda_lasso = 1.0
lasso_model = Lasso(alpha=lambda_lasso, fit_intercept=False)
lasso_model.fit(X, y)

# Retrieve the coefficients
lasso_beta_sklearn = lasso_model.coef_
```

[7]:
```
# Function to calculate MSE
def mse(y_true, y_pred):
    return ((y_true - y_pred) ** 2).mean()

# Calculate predictions from sklearn.linear_model
y_preds_sklearn = X @ lasso_beta_sklearn

# Calculate predictions from cvxpy package
y_preds_cvxpy = X @ lasso_beta_cvxpy

# Calculate MSE
print("MSE with cvxpy Lasso:", mse(y, y_preds_cvxpy))
print("MSE with sklearn Lasso:", mse(y, y_preds_sklearn))
```

```
MSE with cvxpy Lasso: 0.0010563906518943377
MSE with sklearn Lasso: 31.4434067353365
```

Benchmark Dantzig selector

[8]:
```
y_preds_Dantzig=X @ beta_Dantzig
print("MSE with cvxpy Dantzig:", mse(y,y_preds_Dantzig))
```

```
MSE with cvxpy Dantzig: 0.0057335071418471405
```

1.3 Question 3 Markowitz portfolio optimization

[9]:
```
from cvxopt import matrix, solvers

# Set up the problem parameters
mu_R = np.array([0.05, 0.06, 0.07])  # Expected returns of the assets
Sigma_R = np.array([[0.1, 0.01, 0.01],
                    [0.01, 0.1, 0.01],
                    [0.01, 0.01, 0.1]])  # Covariance matrix of asset returns
l = 0.06  # Minimum acceptable return

# Define the number of assets
n = len(mu_R)

# Quadratic programming matrices
P = matrix(Sigma_R)  # Covariance matrix as a CVXOPT matrix for quadratic␣
↪objective
```

```python
q = matrix(np.zeros(n))  # Coefficient vector for the linear term in the␣
↪objective (zero since we minimize variance)

# Inequality constraints Gx <= h to enforce minimum return and non-negative␣
↪weights
G = matrix(np.vstack((-mu_R, -np.eye(n))))  # Negative sign because we want␣
↪returns to be at least 'l' and weights non-negative
h = matrix(np.hstack((-l, np.zeros(n))))  # Right-hand side vector, first entry␣
↪for minimum return, others for non-negativity

# Equality constraint Ax = b to ensure that the sum of weights equals 1
A = matrix(1.0, (1, n))  # Row matrix of ones (length n)
b = matrix(1.0)  # Scalar value 1 in a CVXOPT matrix

# Solver settings
solvers.options['show_progress'] = False  # Disable solver progress output for␣
↪cleaner output

# Solve the quadratic programming problem
solution = solvers.qp(P, q, G, h, A, b)

# Output the results
if solution['status'] == 'optimal':
    weights = np.array(solution['x']).flatten()  # Extract and flatten the␣
↪weight vector
    print("Optimized weights:", weights)
else:
    print("No optimal solution found.")
```

```
Optimized weights: [0.33285245 0.33333333 0.33381422]
```