Lecture 7: MPT Applications

Douglas Chung

National Chengchi University

29 October 2021

1 Numerical solver for optimal portfolios

[1]:
```python
import numpy as np
import pandas as pd
import pandas_datareader as pdr
from numpy.linalg import inv
import matplotlib.pyplot as plt
from scipy.optimize import minimize
import seaborn as sns
cmap = sns.color_palette()

factor = pdr.get_data_famafrench('F-F_Research_Data_Factors', start='1-1-1926')
asset = pdr.get_data_famafrench('10_Industry_Portfolios', start='1-1-1926')
print(asset['DESCR'])
```

```
10 Industry Portfolios
----------------------

This file was created by CMPT_IND_RETS using the 202107 CRSP database. It
contains value- and equal-weighted returns for 10 industry portfolios. The
portfolios are constructed at the end of June. The annual returns are from
January to December. Missing data are indicated by -99.99 or -999. Copyright
2021 Kenneth R. French

  0 : Average Value Weighted Returns -- Monthly (1141 rows x 10 cols)
  1 : Average Equal Weighted Returns -- Monthly (1141 rows x 10 cols)
  2 : Average Value Weighted Returns -- Annual (94 rows x 10 cols)
  3 : Average Equal Weighted Returns -- Annual (94 rows x 10 cols)
  4 : Number of Firms in Portfolios (1141 rows x 10 cols)
  5 : Average Firm Size (1141 rows x 10 cols)
  6 : Sum of BE / Sum of ME (96 rows x 10 cols)
  7 : Value-Weighted Average of BE/ME (96 rows x 10 cols)
```

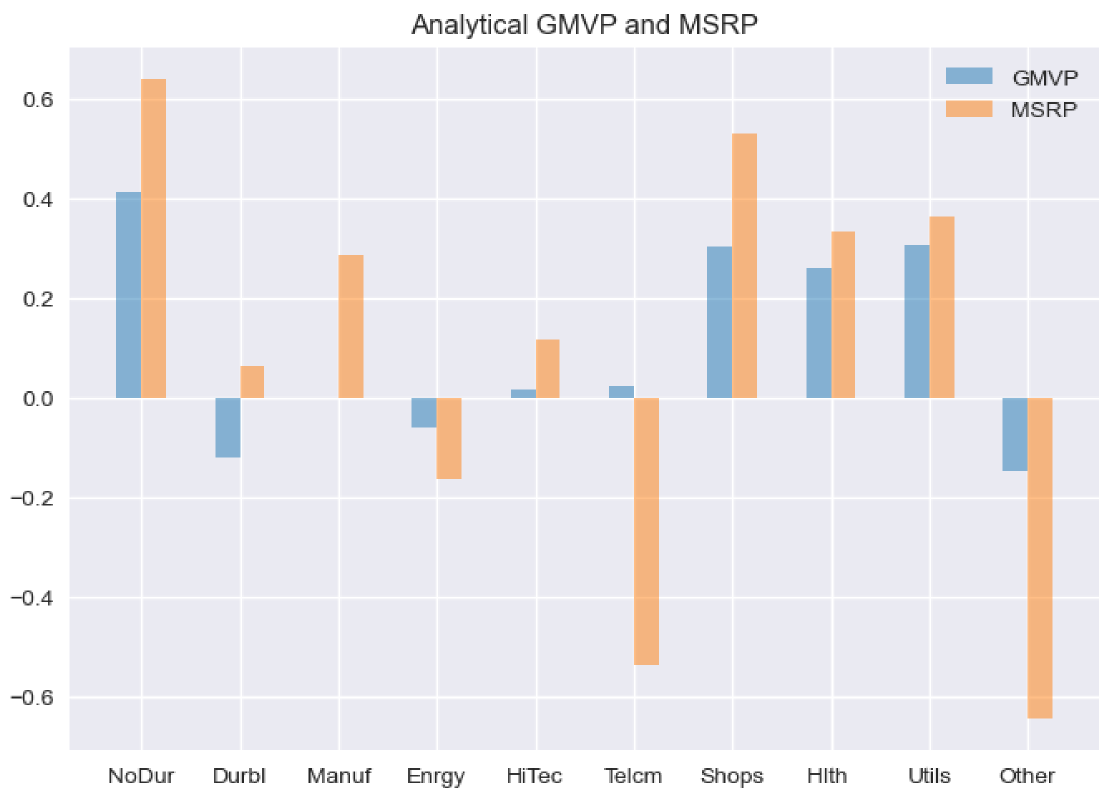
1.1 Subsetting the data

[2]:
```
df_FF = factor[0].loc['2000':'2020']
df_R = asset[0].loc['2000':'2020']
df_ER = df_R.subtract(df_FF.RF,axis=0)
```

1.2 Analytical solutions

[3]:
```
S = df_R.cov()
R = df_R.mean(axis=0)
ER = df_ER.mean(axis=0)
N = len(ER)
ONE = np.ones(len(ER))
GMVP = (inv(S) @ ONE) / (ONE.T @ inv(S) @ ONE)
MSRP = (inv(S) @ ER) / (ONE.T @ inv(S) @ ER)
```

[4]:
```
plt.style.use('seaborn')
ind = np.arange(N)
plt.bar(ind, GMVP, width=0.25, alpha=0.5, color=cmap[0])
plt.bar(ind + 0.25, MSRP, width=0.25, alpha=0.5, color=cmap[1])
plt.xticks(ind + 0.25 / 2, ER.index)
plt.legend(['GMVP','MSRP'])
plt.title('Analytical GMVP and MSRP')
plt.show()
```



1.3 Define objective functions

For GMVP, we want to minimize the portfolio variance:

$\sigma_p^2 = \mathbf{w}^T \Sigma \mathbf{w}$

subject to:

$\mathbf{w}^T \mathbf{1} = 1$

For MSRP, we want to maximize the portfolio Sharpe ratio:

$\mathbf{SR}_p = \frac{\mathbf{w}^T \mathbf{R}}{\sqrt{\mathbf{w}^T \Sigma \mathbf{w}}}$

subject to:

$\mathbf{w}^T \mathbf{1} = 1$

Since numerical solvers find parameters that minimize the objective function, we have to define the objective function as the negative of portfolio Sharpe ratio.

[5]:
```
def pvar(w, S):
    return (w @ S @ w)

def pret(w, ER):
    return (w @ ER)

def sharpe(w, ER, S):
    return -(w @ ER)/np.sqrt(w @ S @ w)
```

1.3.1 Numerical GMVP

Our initial guess is simply:

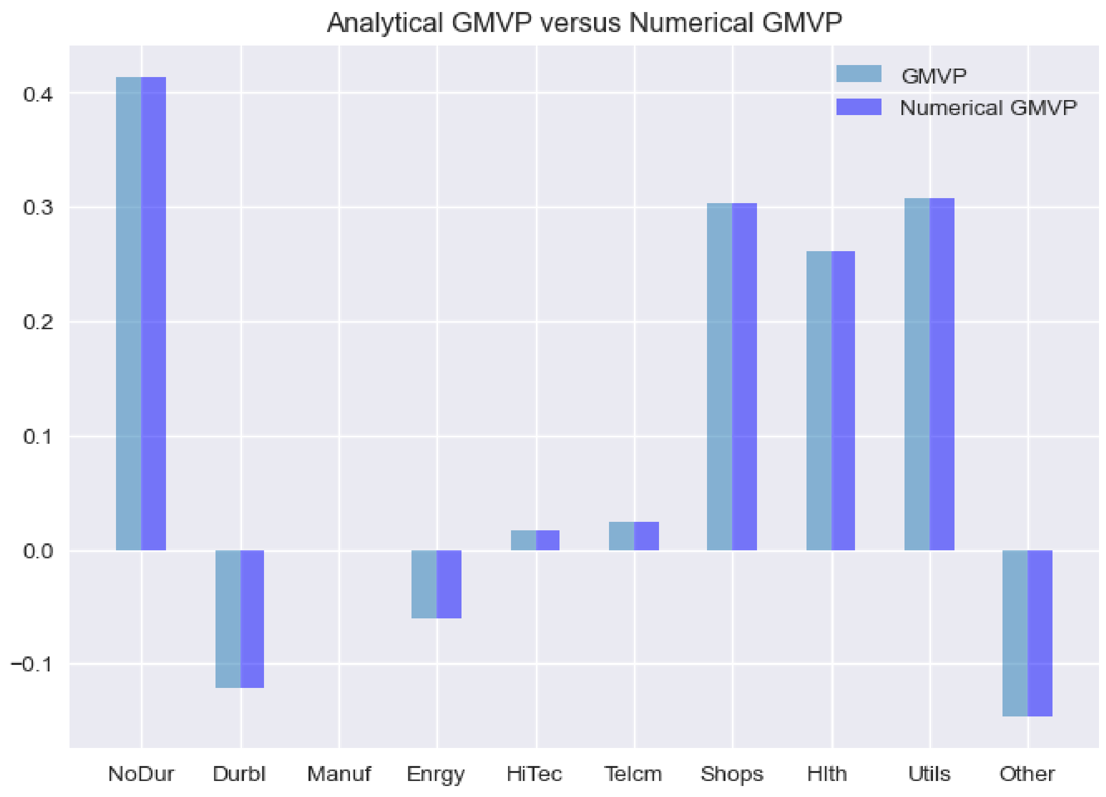
$w_i = \frac{1}{N}$

[6]:
```
x0 = np.ones(N)/N
cons = ({'type': 'eq', 'fun' : lambda x: np.sum(x) - 1}),
NGMVP = minimize(pvar, x0, method='SLSQP', args=S, constraints=cons,␣
 ↪options={'disp': True, 'ftol': 1e-9})
NGMVP
```

```
Optimization terminated successfully    (Exit mode 0)
            Current function value: 9.390276191622382
            Iterations: 11
            Function evaluations: 135
            Gradient evaluations: 11
```

[6]:
```
      fun: 9.390276191622382
      jac: array([18.78055251, 18.78055251, 18.78055251, 18.78055263,
18.78055251,
       18.78055251, 18.78055263, 18.78055251, 18.78055274, 18.78055251])
 message: 'Optimization terminated successfully'
    nfev: 135
     nit: 11
    njev: 11
  status: 0
 success: True
       x: array([ 4.13753735e-01, -1.20903693e-01,  8.54418565e-05,
-5.99156115e-02,
        1.70379759e-02,  2.34358013e-02,  3.03497904e-01,  2.61004190e-01,
        3.08068716e-01, -1.46064460e-01])
```

[7]:
```python
ind = np.arange(N)
plt.bar(ind, GMVP, width=0.25, alpha=0.5, color=cmap[0])
plt.bar(ind + 0.25, NGMVP.x, width=0.25, alpha=0.5, color='blue')
plt.xticks(ind + 0.25 / 2, ER.index)
plt.legend(['GMVP','Numerical GMVP'])
plt.title('Analytical GMVP versus Numerical GMVP')
plt.show()
```



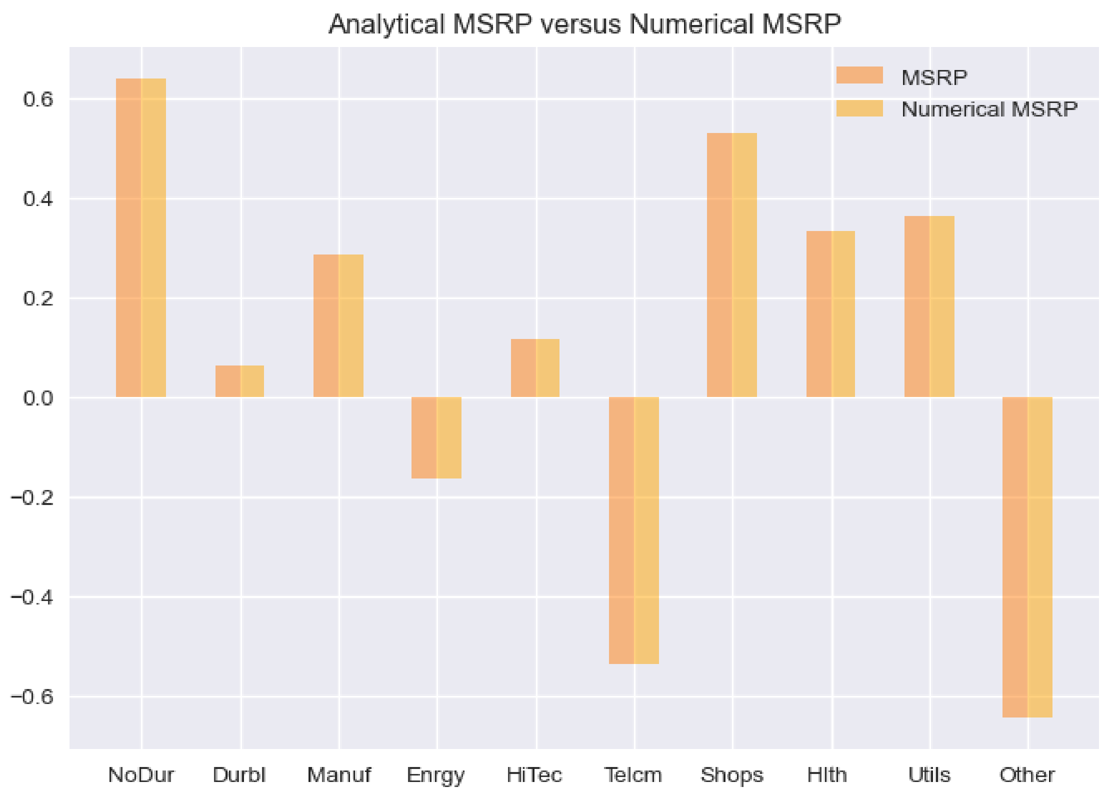
1.3.2 Numerical MSRP

[8]:
```
x0 = np.arange(N)/N
cons = ({'type': 'eq', 'fun' : lambda x: np.sum(x) - 1}),
NMSRP = minimize(sharpe, x0, method='SLSQP', args=(ER, S), constraints=cons,␣
 ↪options={'disp': True, 'ftol': 1e-9})
NMSRP
```

```
Optimization terminated successfully    (Exit mode 0)
            Current function value: -0.2966267564788634
            Iterations: 30
            Function evaluations: 331
            Gradient evaluations: 30
```

[8]:
```
     fun: -0.2966267564788634
     jac: array([ 4.78327274e-06,  2.13831663e-06,  1.87382102e-06,
1.30087137e-05,
       -6.10947609e-06, -7.71135092e-07,  8.25524330e-06, -1.62422657e-06,
        7.89016485e-06,  1.25765800e-05])
 message: 'Optimization terminated successfully'
    nfev: 331
     nit: 30
    njev: 30
  status: 0
 success: True
       x: array([ 0.64003671,  0.06497472,  0.28757172, -0.16099811,
0.11635957,
       -0.53696856,  0.53157973,  0.33495086,  0.36472336, -0.64223001])
```

[9]:
```
ind = np.arange(N)
plt.bar(ind, MSRP, width=0.25, alpha=0.5, color=cmap[1])
plt.bar(ind + 0.25, NMSRP.x, width=0.25, alpha=0.5, color='orange')
plt.xticks(ind + 0.25 / 2, ER.index)
plt.legend(['MSRP','Numerical MSRP'])
plt.title('Analytical MSRP versus Numerical MSRP')
plt.show()
```



1.4 Numerical GMVP with short selling constraint

Each asset must have a weight great or equal to 0:

$w_i \geq 0$

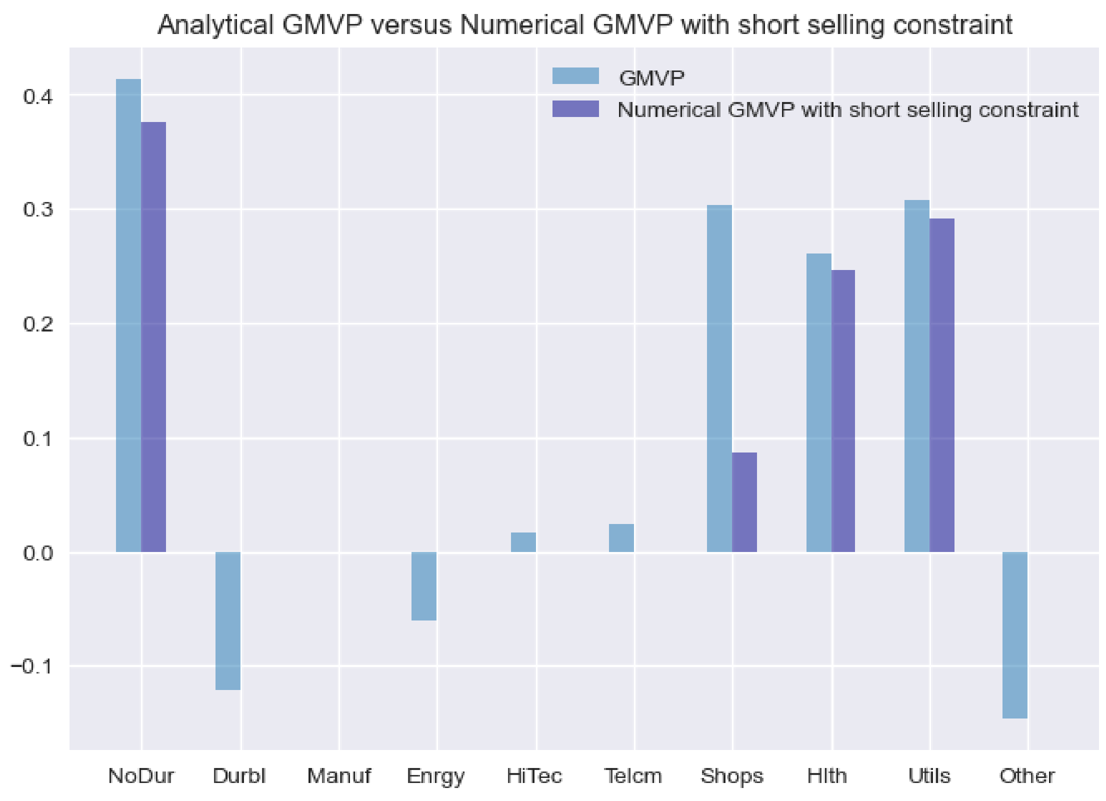
We have to define bounds by using the "Bounds()" method.

[10]:
```
from scipy.optimize import Bounds
bounds = Bounds(0, 1)

x0 = GMVP
cons = ({'type': 'eq', 'fun' : lambda x: np.sum(x) - 1})
NGMVPnoSS = minimize(pvar, x0, method='SLSQP', constraints=cons, args=S,
               options={'disp': True, 'ftol': 1e-9}, bounds=bounds)
NGMVPnoSS
```

```
Optimization terminated successfully    (Exit mode 0)
            Current function value: 10.677670244592571
            Iterations: 8
            Function evaluations: 91
            Gradient evaluations: 8
```

[10]:
```
      fun: 10.677670244592571
      jac: array([21.35534084, 33.41619241, 25.80424976, 28.39166665,
23.65987611,
       22.79438603, 21.35534108, 21.35534036, 21.35534036, 26.61583376])
  message: 'Optimization terminated successfully'
     nfev: 91
      nit: 8
     njev: 8
   status: 0
  success: True
        x: array([3.75368003e-01, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 1.62231095e-15, 8.67982811e-02, 2.46870695e-01,
       2.90963020e-01, 0.00000000e+00])
```

[11]:
```python
ind = np.arange(N)
plt.bar(ind, GMVP, width=0.25, alpha=0.5, color=cmap[0])
plt.bar(ind + 0.25, NGMVPnoSS.x, width=0.25, alpha=0.5, color='darkblue')
plt.xticks(ind + 0.25 / 2, ER.index)
plt.legend(['GMVP','Numerical GMVP with short selling constraint'])
plt.title('Analytical GMVP versus Numerical GMVP with short selling constraint')
plt.show()
```



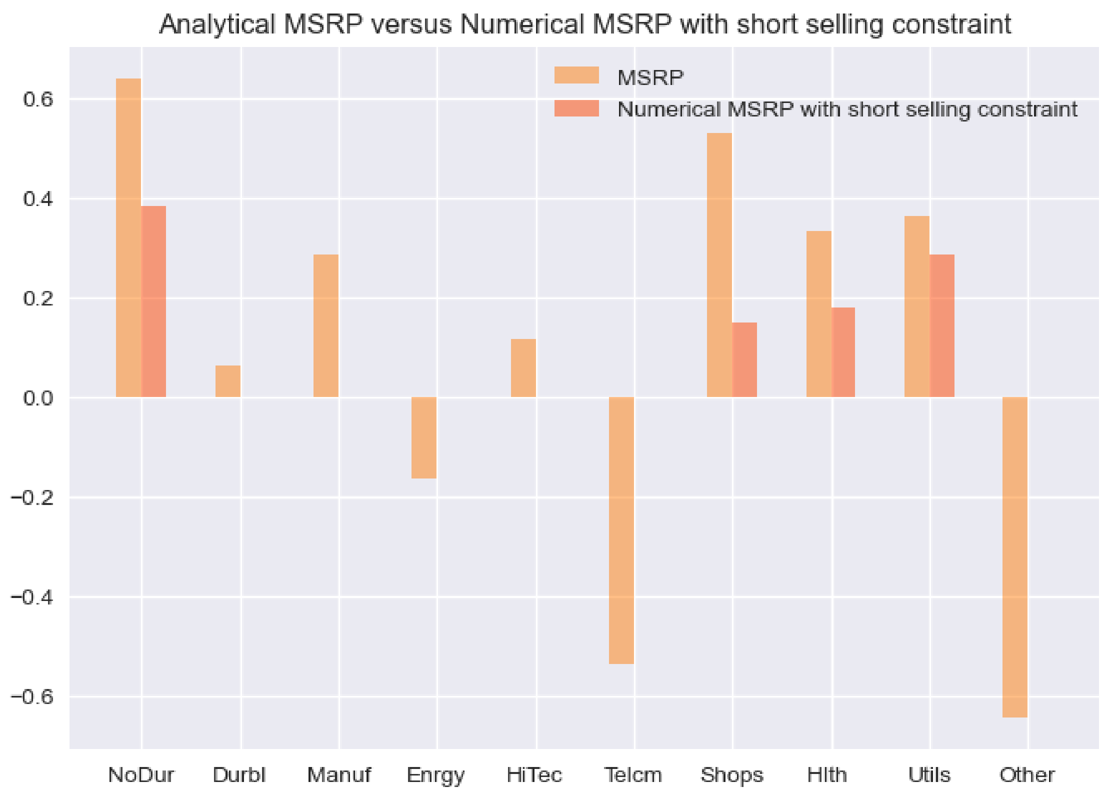
1.5 Numerical MSRP with short selling constraint

[12]:
```
from scipy.optimize import Bounds
bounds = Bounds(0, 1)

x0 = MSRP
cons = ({'type': 'eq', 'fun' : lambda x: np.sum(x) - 1})
NMSRPnoSS = minimize(sharpe, x0, method='SLSQP', constraints=cons, args=(ER, S),
               options={'disp': True, 'ftol': 1e-9}, bounds=bounds)
NMSRPnoSS
```

```
Optimization terminated successfully    (Exit mode 0)
            Current function value: -0.21830140101397755
            Iterations: 15
            Function evaluations: 165
            Gradient evaluations: 15
```

[12]:
```
     fun: -0.21830140101397755
     jac: array([-2.44192779e-06,  7.86497425e-02,  4.41473387e-02,
 1.43232191e-01,
        5.68958819e-02,  1.58268843e-01, -1.22934580e-07, -8.00937414e-07,
        3.85567546e-06,  1.17689932e-01])
 message: 'Optimization terminated successfully'
    nfev: 165
     nit: 15
    njev: 15
  status: 0
 success: True
       x: array([3.83523011e-01, 1.57886941e-18, 5.72560391e-17, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 1.51734348e-01, 1.78931849e-01,
       2.85810792e-01, 4.17417836e-18])
```

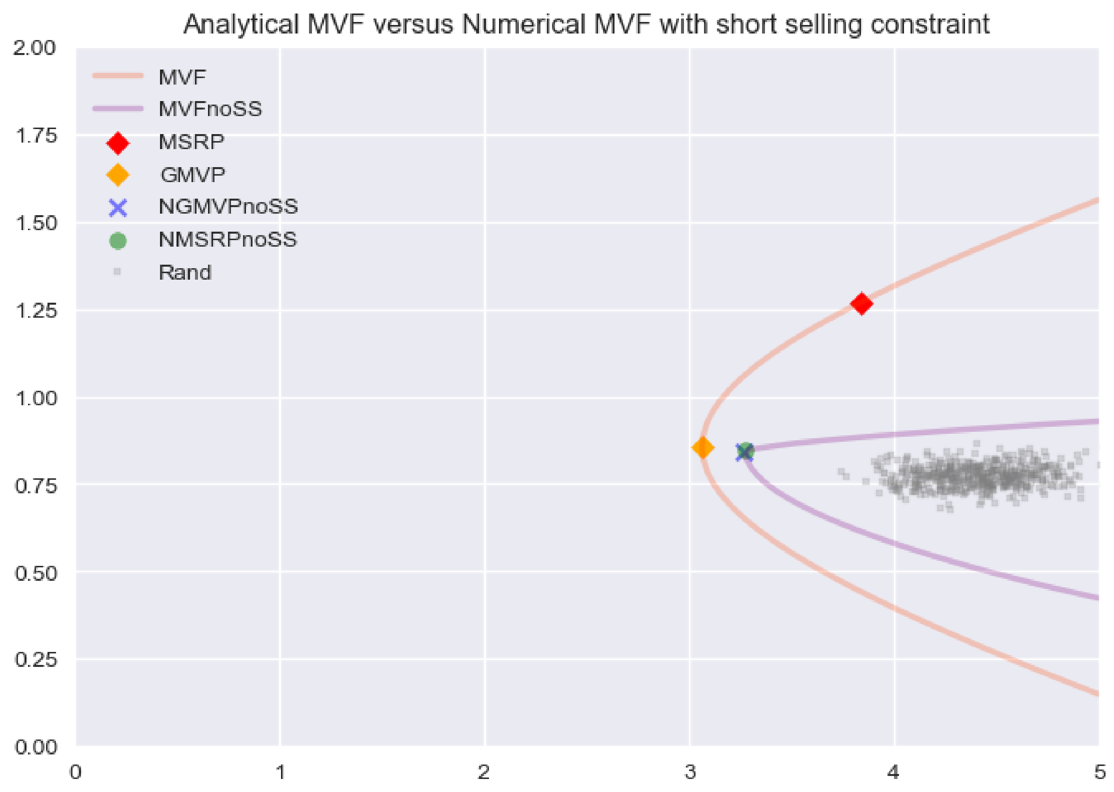
[13]:
```
ind = np.arange(N)
plt.bar(ind, MSRP, width=0.25, alpha=0.5, color=cmap[1])
plt.bar(ind + 0.25, NMSRPnoSS.x, width=0.25, alpha=0.5, color='orangered')
plt.xticks(ind + 0.25 / 2, ER.index)
plt.legend(['MSRP','Numerical MSRP with short selling constraint'])
plt.title('Analytical MSRP versus Numerical MSRP with short selling constraint')
plt.show()
```



1.6 Numerical MVF with short selling constraint

[14]:
```
GMVPR = pret(GMVP, R)
GMVPSD = np.sqrt(pvar(GMVP, S))

MSRPR = pret(MSRP, R)
MSRPSD = np.sqrt(pvar(MSRP, S))

NGMVPnoSSR = pret(NGMVPnoSS.x, R)
NGMVPnoSSSD = np.sqrt(pvar(NGMVPnoSS.x, S))

NMSRPnoSSR = pret(NMSRPnoSS.x, R)
NMSRPnoSSSD = np.sqrt(pvar(NMSRPnoSS.x, S))

W = np.linspace(-4, 4, 100, endpoint=True)
MVFR = []
MVFSD = []
for u in W:
    MVFR.append(u*GMVPR + (1-u)*MSRPR)
    MVFSD.append(np.sqrt(u**2 * GMVPSD**2 + (1-u)**2 * MSRPSD**2 + 2*u*(1-u)*np.
 ↪dot(GMVP, np.dot(S, MSRP))))
```

```
NRand = 500
WRand = pd.DataFrame(np.random.uniform(0, 1, size=(NRand, N)))
WRand = WRand.divide(WRand.sum(axis=1), axis=0).to_numpy()

RandR = []
RandSD = []

for i in range(NRand):
    RandR.append(pret(WRand[i], R))
    RandSD.append(np.sqrt(pvar(WRand[i], S)))

MVFnoSSR = []
MVFnoSSSD = []

TR = np.linspace(0, 1.75, 100, endpoint=True)
for r in TR:
    x0 = np.arange(N)/N
    cons = ({'type': 'eq', 'fun' : lambda x: np.sum(x) - 1},
            {'type': 'eq', 'fun' : lambda x: pret(x, R) - r})
    TMP = minimize(pvar, x0, method='SLSQP', constraints=cons, args=S,
                   options={'disp': False, 'ftol': 1e-9}, bounds=bounds)
    MVFnoSSR.append(pret(TMP.x, R))
    MVFnoSSSD.append(np.sqrt(pvar(TMP.x, S)))

plt.plot(MVFSD, MVFR, color='orangered', linewidth=2.5, alpha=0.25)
plt.plot(MVFnoSSSD, MVFnoSSR, color='purple', linewidth=2.5, alpha=0.25)
plt.scatter(x=MSRPSD, y=MSRPR, c='red', marker='D', s=50, alpha=1)
plt.scatter(x=GMVPSD, y=GMVPR, c='orange', marker='D', s=50, alpha=1)
plt.scatter(x=NGMVPnoSSSD, y=NGMVPnoSSR, c='blue', marker='x', s=50, alpha=0.5)
plt.scatter(x=NMSRPnoSSSD, y=NMSRPnoSSR, c='green', marker='o', s=50, alpha=0.5)
plt.scatter(x=RandSD, y=RandR, c='gray', marker=',', s=10, alpha=0.25)
plt.xlim([0, 5])
plt.ylim([0, 2])
plt.legend(['MVF','MVFnoSS','MSRP','GMVP','NGMVPnoSS','NMSRPnoSS','Rand'])
plt.title('Analytical MVF versus Numerical MVF with short selling constraint')
plt.show()
```


Analytical MVF versus Numerical MVF with short selling constraint
MVF
MVFnoSS
MSRP
GMVP
NGMVPnoSS
NMSRPnoSS
Rand
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
1
2
3
4
5