In [1]:
```
# QP solvers installation

%pip install qpsolvers
%pip install qpsolvers[open_source_solvers]
```

In [2]:
```
import numpy as np
from qpsolvers import solve_qp
import matplotlib.pyplot as plt
import warnings
import pandas as pd

warnings.filterwarnings("ignore")
np.set_printoptions(precision=3)
pd.options.display.float_format = '{:.4f}'.format
```

Input data

In [3]:
```
n_companies = 8
index = list(range(1, n_companies + 1))

carbon_emissions = { # in ktCO2e
    'scope_1': np.array([75, 5000, 720, 50, 2500, 25, 30000, 5]),
    'scope_2': np.array([75, 5000, 1030, 350, 4500, 5, 2000, 64]),
    'scope_3': np.array([200, 500, 520, 850, 8000, 50, 200, 146])
}

revenue = np.array([300, 328, 125, 100, 200, 102, 107, 25]) # in $ bn

sector_1 = np.array([1, 0, 1, 1, 0, 1, 0, 0]) # indicator of sector 1
sector_2 = np.array([0, 1, 0, 0, 1, 0, 1, 1]) # indicator of sector 2
```

In [4]:
```
vol_array = 0.01 * np.array([12, 21, 23, 19, 20, 33, 43, 19]) # volatility of returns

correl_mat = 0.01 * np.array([
    [100, 0, 0, 0, 0, 0, 0, 0],
    [80, 100, 0, 0, 0, 0, 0, 0],
    [70, 75, 100, 0, 0, 0, 0, 0],
    [60, 65, 80, 100, 0, 0 ,0, 0],
    [70, 50, 70, 85, 100, 0, 0, 0],
    [50, 60, 70, 80, 60, 100, 0, 0],
    [70, 50, 70, 75, 80, 50, 100, 0],
    [70, 75, 80, 85, 75, 80, 70, 100]
])
correl_mat = (correl_mat + correl_mat.T) - np.eye(n_companies)

weights_ref = 0.01 * np.array([20, 17, 17, 13, 11, 10, 6, 6])

print("Correlation matrix =\n")
pd.DataFrame(correl_mat, columns=index, index=index)
```

```
Correlation matrix =
```

Out[4]:

	1	2	3	4	5	6	7	8
1	1.0000	0.8000	0.7000	0.6000	0.7000	0.5000	0.7000	0.7000
2	0.8000	1.0000	0.7500	0.6500	0.5000	0.6000	0.5000	0.7500
3	0.7000	0.7500	1.0000	0.8000	0.7000	0.7000	0.7000	0.8000
4	0.6000	0.6500	0.8000	1.0000	0.8500	0.8000	0.7500	0.8500
5	0.7000	0.5000	0.7000	0.8500	1.0000	0.6000	0.8000	0.7500
6	0.5000	0.6000	0.7000	0.8000	0.6000	1.0000	0.5000	0.8000

7	0.7000	0.5000	0.7000	0.7500	0.8000	0.5000	1.0000	0.7000
8	0.7000	0.7500	0.8000	0.8500	0.7500	0.8000	0.7000	1.0000

Question 1

Calculation of carbon intencities for reference portfolio b.

(a) Carbon intensities $\mathcal{CI}_{1+2}$ for scopes $1+2$ decribe the company's carbon emissions per revenue (ktCO2e per $ 1 mn revenue):

$$\mathcal{CI}_{i,1+2} = \frac{\mathcal{CE}_{i,1} + \mathcal{CE}_{i,2}}{Y}$$

In [5]:
```
carbon_intensity_12 = (carbon_emissions['scope_1'] + carbon_emissions['scope_2']) / r
print('Carbon intensities (scope 1 + scope 2):')
pd.DataFrame(carbon_intensity_12, index=index, columns=[r'CI(1+2)'])
```

```
Carbon intensities (scope 1 + scope 2):
```

Out[5]:

	CI(1+2)
1	0.5000
2	30.4878
3	14.0000
4	4.0000
5	35.0000
6	0.2941
7	299.0654
8	2.7600

(b) Weighed average carbon intensity of bechmark is given by

$$\mathcal{CI}_{1+2}(b) = \sum_i b_i \mathcal{CI}_{i,1+2}$$

In [6]:
```
waci_ref_12 = weights_ref @ carbon_intensity_12
print('Benchmark WACI =', np.round(waci_ref_12, 5))
```

```
Benchmark WACI = 30.17186
```

(c) The same questions for scopes $1+2+3$.

$$\mathcal{CI}_{i,1+2+3} = \frac{\mathcal{CE}_{i,1} + \mathcal{CE}_{i,2} + \mathcal{CE}_{i,3}}{Y}$$

$$\mathcal{CI}_{1+2+3}(b) = \sum_i b_i \mathcal{CI}_{i,1+2+3}$$

In [7]:
```
carbon_intensity_123 = (carbon_emissions['scope_1'] + carbon_emissions['scope_2'] + c
print('Carbon intensities (scope 1 + scope 2 + scope 3):')
pd.DataFrame(carbon_intensity_123, index=index, columns=[r'CI(1+2+3)'])
```

```
Carbon intensities (scope 1 + scope 2 + scope 3):
```

Out[7]:

	CI(1+2+3)
1	1.1667
2	32.0122
3	18.1600

4	12.5000
5	75.0000
6	0.7843
7	300.9346
8	8.6000

In [8]:
```
waci_ref_123 = weights_ref @ carbon_intensity_123
print('Benchmark WACI =', waci_ref_123)
```

```
Benchmark WACI = 37.288112642969196
```

Question 2

The goal is to reduce the WACI of the benchmark portfolio by rate $\mathcal{R}$.

(a) The elements of covariance matrix Σ are equal

$$\Sigma_{ij} = \rho_{ij}\sigma_i\sigma_j$$

In [9]:
```
cov_mat = (vol_array[:, None] @ vol_array[None, :]) * correl_mat
print('Covariance matrix =')
pd.DataFrame(cov_mat, columns=index, index=index)
```

```
Covariance matrix =
```

Out[9]:

	1	2	3	4	5	6	7	8
1	0.0144	0.0202	0.0193	0.0137	0.0168	0.0198	0.0361	0.0160
2	0.0202	0.0441	0.0362	0.0259	0.0210	0.0416	0.0451	0.0299
3	0.0193	0.0362	0.0529	0.0350	0.0322	0.0531	0.0692	0.0350
4	0.0137	0.0259	0.0350	0.0361	0.0323	0.0502	0.0613	0.0307
5	0.0168	0.0210	0.0322	0.0323	0.0400	0.0396	0.0688	0.0285
6	0.0198	0.0416	0.0531	0.0502	0.0396	0.1089	0.0709	0.0502
7	0.0361	0.0451	0.0692	0.0613	0.0688	0.0709	0.1849	0.0572
8	0.0160	0.0299	0.0350	0.0307	0.0285	0.0502	0.0572	0.0361

(b) An optimization problem corresponding to the minimization of the tracking error under the intensity reduction constraint is given by:

$$\min \left\{ \frac{1}{2} w^T \Sigma w - w^T \Sigma b \right\}$$

$$\text{s.t.} \begin{cases} \mathbf{1}^T w = 1, \\ \mathbf{0} \leq w \leq \mathbf{1}, \\ \mathcal{CI}(w) \leq (1 - \mathcal{R})\mathcal{CI}(b). \end{cases}$$

(c) QP formulation of the considered problem:

$$\min \left\{ \frac{1}{2} w^T P w + w^T q \right\}$$

$$\text{s.t.} \begin{cases} Gx \leq h, \\ Ax = b, \\ w^- \leq w \leq w^+, \end{cases}$$

where for our problem

$$
\begin{gathered}
P = \Sigma, \quad q = -\Sigma b, \\
G = \mathcal{CI}^T, \quad h = (1 - \mathcal{R})\mathcal{CI}(b), \\
A = \mathbf{1}^T, \quad b = 1, \\
w^- = \mathbf{0}, \quad w^+ = \mathbf{1}.
\end{gathered}
$$

In [10]:

```python
def optimal_replicating_portfolio(
    reduction_rate: float,
    carbon_intensity: np.ndarray,
    reference_portfolio: np.ndarray,
    covariance_matrix: np.ndarray,
):
    optimal_portfolio = solve_qp(
        P=covariance_matrix,
        q=-covariance_matrix @ reference_portfolio,
        G=carbon_intensity,
        h=np.array([(1 - reduction_rate) * carbon_intensity @ reference_portfolio]),
        A=np.ones_like(reference_portfolio),
        b=np.array([1]),
        lb=np.zeros_like(reference_portfolio),
        ub=np.ones_like(reference_portfolio),
        solver="clarabel"
    )
    return optimal_portfolio
```

(c) For $\mathcal{R} = 20$ and scope 1+2 carbon intensities we obtain the solution

In [11]:

```python
reduction_rate = 0.2

opt_portfolio_20_12 = optimal_replicating_portfolio(
    reduction_rate=reduction_rate,
    carbon_intensity=carbon_intensity_12,
    reference_portfolio=weights_ref,
    covariance_matrix=cov_mat
)
print('Optimized portfolio vs benchmark:')
pd.DataFrame(np.vstack([opt_portfolio_20_12, weights_ref]).T, index=index, columns=['
```

```
Optimized portfolio vs benchmark:
```

Out[11]:

	w20%	b
1	0.2103	0.2000
2	0.1449	0.1700
3	0.1844	0.1700
4	0.1484	0.1300
5	0.0993	0.1100
6	0.0969	0.1000
7	0.0426	0.0600
8	0.0731	0.0600

Sanity check

In [12]:

```python
print('Weights sum =', np.sum(opt_portfolio_20_12))
print('Portfolio CI =', carbon_intensity_12 @ opt_portfolio_20_12)
print('0.8 * CI(b) =', 0.8 * waci_ref_12)
```

```
Weights sum = 0.9999999999999999
Portfolio CI = 24.137424425955345
0.8 * CI(b) = 24.13749106209523
```

The volatility of the replication error is given by:

$$\sigma(w^*|b) = \sqrt{(w^* - b)^T \Sigma (w^* - b)}$$

In [13]:
```
def replication_vol(opt_pf, ref_pf, cov_mat):
    return np.sqrt((opt_pf - ref_pf).T @ cov_mat @ (opt_pf - ref_pf))
```

In [14]:
```
print('Replication volatility =', replication_vol(opt_portfolio_20_12, weights_ref, c
```

```
Replication volatility = 0.005636070383049719
```

(4) Same questions for $\mathcal{R} = 30\%, 50\%, 70\%$.

In [15]:
```
reduction_rate = 0.3

opt_portfolio_30_12 = optimal_replicating_portfolio(
    reduction_rate=reduction_rate,
    carbon_intensity=carbon_intensity_12,
    reference_portfolio=weights_ref,
    covariance_matrix=cov_mat
)

print('Replication volatility =', replication_vol(opt_portfolio_30_12, weights_ref, c
print('Optimized portfolio vs benchmark:')
pd.DataFrame(np.vstack([opt_portfolio_30_12, weights_ref]).T, index=index, columns=['
```

```
Replication volatility = 0.008454041635834691
Optimized portfolio vs benchmark:
```

Out[15]:

	w30%	b
1	0.2155	0.2000
2	0.1324	0.1700
3	0.1916	0.1700
4	0.1576	0.1300
5	0.0940	0.1100
6	0.0954	0.1000
7	0.0339	0.0600
8	0.0797	0.0600

In [16]:
```
reduction_rate = 0.5

opt_portfolio_50_12 = optimal_replicating_portfolio(
    reduction_rate=0.5,
    carbon_intensity=carbon_intensity_12,
    reference_portfolio=weights_ref,
    covariance_matrix=cov_mat
)
print('Replication volatility =', replication_vol(opt_portfolio_50_12, weights_ref, c
print('Optimized portfolio vs benchmark:')
pd.DataFrame(np.vstack([opt_portfolio_50_12, weights_ref]).T, index=index, columns=['
```

```
Replication volatility = 0.014090032982331201
Optimized portfolio vs benchmark:
```

Out[16]:

	w50%	b
1	0.2258	0.2000
2	0.1073	0.1700
3	0.2061	0.1700
4	0.1760	0.1300
5	0.0833	0.1100

6	0.0923	0.1000
7	0.0164	0.0600
8	0.0928	0.0600

In [17]:

```
reduction_rate = 0.7

opt_portfolio_70_12 = optimal_replicating_portfolio(
    reduction_rate=0.7,
    carbon_intensity=carbon_intensity_12,
    reference_portfolio=weights_ref,
    covariance_matrix=cov_mat
)
print('Replication volatility =', replication_vol(opt_portfolio_70_12, weights_ref, c
print('Optimized portfolio vs benchmark:')
pd.DataFrame(np.vstack([opt_portfolio_70_12, weights_ref]).T, index=index, columns=['
```

```
Replication volatility = 0.01973251575044196
Optimized portfolio vs benchmark:
```

Out[17]:

	w70%	b
1	0.2404	0.2000
2	0.0788	0.1700
3	0.2202	0.1700
4	0.1978	0.1300
5	0.0667	0.1100
6	0.0893	0.1000
7	0.0000	0.0600
8	0.1068	0.0600

In [18]:

```
reduction_rates = 0.01 * np.arange(0, 100, 5)
rep_vols = []

for reduction_rate in reduction_rates:
    opt_portfolio = optimal_replicating_portfolio(
        reduction_rate=reduction_rate,
        carbon_intensity=carbon_intensity_12,
        reference_portfolio=weights_ref,
        covariance_matrix=cov_mat
    )
    rep_vols.append(replication_vol(opt_portfolio, weights_ref, cov_mat))
```

In [19]:

```
results_12 = pd.DataFrame(index=index)
results_12['b'] = weights_ref
results_12['20%'] = opt_portfolio_20_12
results_12['30%'] = opt_portfolio_30_12
results_12['50%'] = opt_portfolio_50_12
results_12['70%'] = opt_portfolio_70_12
results_12
```

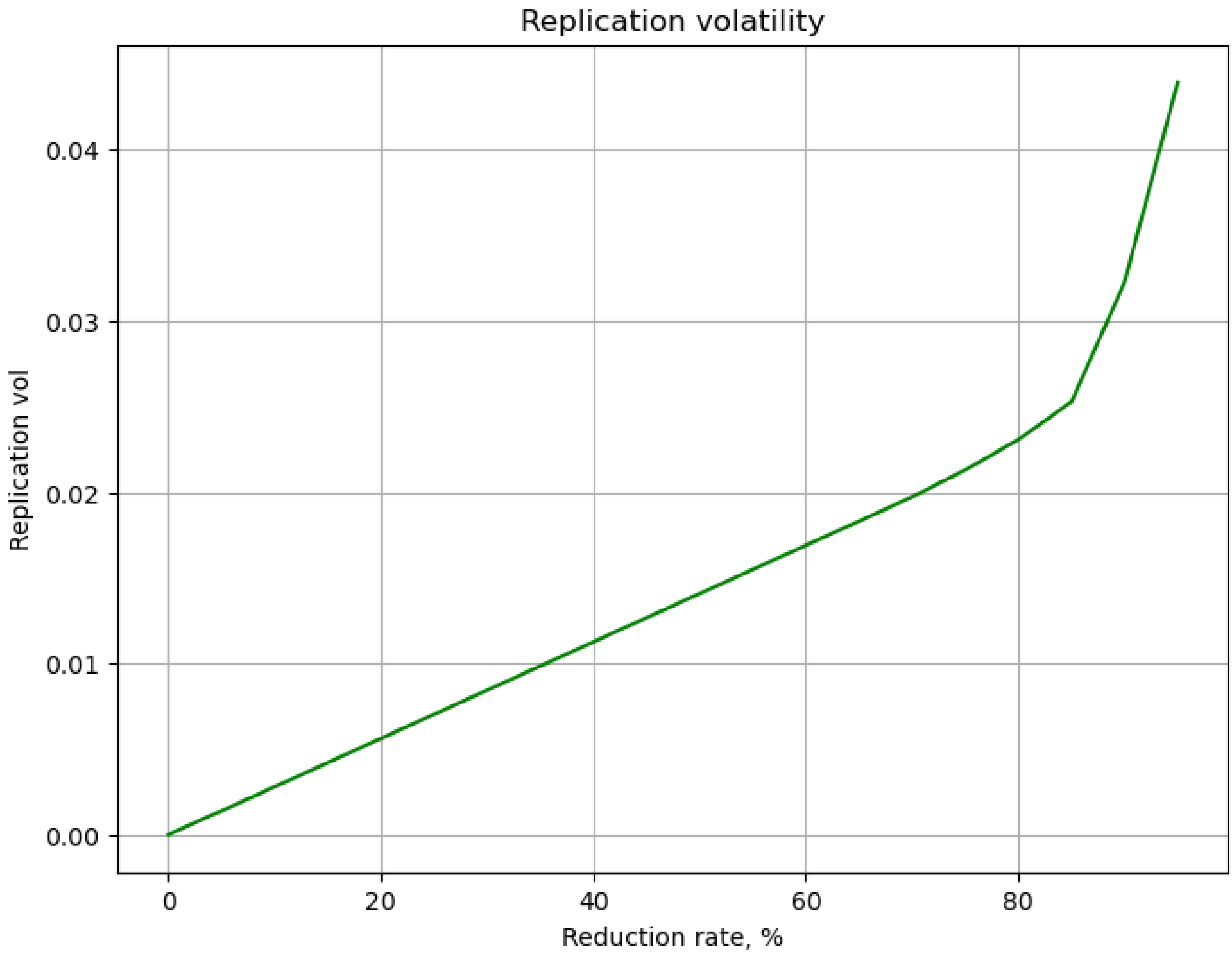
Out[19]:

	b	20%	30%	50%	70%
1	0.2000	0.2103	0.2155	0.2258	0.2404
2	0.1700	0.1449	0.1324	0.1073	0.0788
3	0.1700	0.1844	0.1916	0.2061	0.2202
4	0.1300	0.1484	0.1576	0.1760	0.1978
5	0.1100	0.0993	0.0940	0.0833	0.0667

6	0.1000	0.0969	0.0954	0.0923	0.0893
7	0.0600	0.0426	0.0339	0.0164	0.0000
8	0.0600	0.0731	0.0797	0.0928	0.1068

In [20]:

```
fig, ax = plt.subplots(figsize=(8, 6))
ax.plot(reduction_rates * 100, rep_vols, 'g')
ax.grid()
ax.set_title('Replication volatility')
ax.set_xlabel('Reduction rate, %')
ax.set_ylabel('Replication vol')
plt.show()
```



Question 3

In this question, we will reduce the WACI of the benchmark portfolio by rate $\mathcal{R}$ keeping the sectors weights equal for benchmark and optimized portfolio.

(a) Let s_i for $i = 1, 2$ denote the sector-mapping vector, i.e. $s_{i,j} = 1$ if j-th company is in i-th sector.

The sector-neutrality constraint reads:

$$s_i^T w = s_i^T b, \quad i = 1, 2.$$

(b) We add this constraints to the minimization problem considered in question 2:

$$\min \left\{ \frac{1}{2} w^T \Sigma w - w^T \Sigma b \right\}$$

$$
\text{s.t.} \begin{cases} \mathbf{1}^T w = 1, \\ s_1^T w = s_1^T b, \\ s_2^T w = s_2^T b, \\ \mathbf{0} \leq w \leq \mathbf{1}, \\ \mathcal{CI}(w) \leq (1 - \mathcal{R})\mathcal{CI}(b). \end{cases}
$$

(c) In the QP formulation one should only modify the matrix A

$$
\min \left\{ \frac{1}{2} w^T P w + w^T q \right\}
$$

$$
\text{s.t.} \begin{cases} Gx \leq h, \\ Ax = b, \\ w^- \leq w \leq w^+, \end{cases}
$$

where for our problem

$$
P = \Sigma, \quad q = -\Sigma b,
$$
$$
G = \mathcal{CI}^T, \quad h = (1 - \mathcal{R})\mathcal{CI}(b),
$$
$$
A = \begin{pmatrix} \mathbf{1}^T \\ s_1^T \\ s_2^T \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ s_1^T b \\ s_2^T b \end{pmatrix}
$$
$$
w^- = \mathbf{0}, \quad w^+ = \mathbf{1}.
$$

In [21]:

```
def sector_neutral_optimal_replicating_portfolio(
    reduction_rate: float,
    carbon_intensity: np.ndarray,
    reference_portfolio: np.ndarray,
    covariance_matrix: np.ndarray,
    sector_mappings: np.ndarray
):
    A = np.vstack([
        np.ones_like(reference_portfolio),
        sector_mappings
    ])
    b = np.concatenate([[1], sector_mappings @ reference_portfolio])[:, None]
    np.ones_like(reference_portfolio)
    optimal_portfolio = solve_qp(
        P=covariance_matrix,
        q=-covariance_matrix @ reference_portfolio,
        G=carbon_intensity,
        h=np.array([(1 - reduction_rate) * carbon_intensity @ reference_portfolio]),
        A=A,
        b=b,
        lb=np.zeros_like(reference_portfolio),
        ub=np.ones_like(reference_portfolio),
        solver="clarabel"
    )
    return optimal_portfolio
```

(d) $\mathcal{R} = 20\%$. We will also consider carbon intensities of scopes 1 + 2 + 3.

In [22]:

```
sector_mappings = np.vstack([sector_1, sector_2])
```

In [23]:

```
reduction_rate = 0.2

opt_portfolio_20_123 = sector_neutral_optimal_replicating_portfolio(
    reduction_rate=reduction_rate,
    carbon_intensity=carbon_intensity_123,
    reference_portfolio=weights_ref,
    covariance_matrix=cov_mat,
    sector_mappings=sector_mappings
)
```

```
print('Replication volatility =', replication_vol(opt_portfolio_20_123, weights_ref,
print('Optimized portfolio vs benchmark:')
pd.DataFrame(np.vstack([opt_portfolio_20_123, weights_ref]).T, index=index, columns=[
```

```
Replication volatility = 0.00767451889883405
Optimized portfolio vs benchmark:
```

Out[23]:

	w20%	b
1	0.1946	0.2000
2	0.1517	0.1700
3	0.1794	0.1700
4	0.1349	0.1300
5	0.1002	0.1100
6	0.0911	0.1000
7	0.0374	0.0600
8	0.1107	0.0600

Sanity check

In [24]:

```
print('Weights sum =', np.sum(opt_portfolio_20_123))
print('Portfolio CI =', carbon_intensity_123 @ opt_portfolio_20_123)
print('0.8 * CI(b) =', (1 - reduction_rate) * waci_ref_123)
print('Weight of sector 1 (optimized)=', sector_1 @ opt_portfolio_20_123)
print('Weight of sector 1 (reference)=', sector_1 @ weights_ref)
```

```
Weights sum = 0.9999999999999825
Portfolio CI = 29.830476780307695
0.8 * CI(b) = 29.83049011437536
Weight of sector 1 (optimized)= 0.5999999999999983
Weight of sector 1 (reference)= 0.6
```

(d) Same question for $\mathcal{R} = 30\%, 50\%, 70\%$.

In [25]:

```
reduction_rate = 0.3

opt_portfolio_30_123 = sector_neutral_optimal_replicating_portfolio(
    reduction_rate=reduction_rate,
    carbon_intensity=carbon_intensity_123,
    reference_portfolio=weights_ref,
    covariance_matrix=cov_mat,
    sector_mappings=sector_mappings
)
print('Replication volatility =', replication_vol(opt_portfolio_30_123, weights_ref,
print('Optimized portfolio vs benchmark:')
pd.DataFrame(np.vstack([opt_portfolio_30_123, weights_ref]).T, index=index, columns=[
```

```
Replication volatility = 0.01151177006035631
Optimized portfolio vs benchmark:
```

Out[25]:

	w30%	b
1	0.1920	0.2000
2	0.1425	0.1700
3	0.1840	0.1700
4	0.1374	0.1300
5	0.0953	0.1100
6	0.0866	0.1000
7	0.0262	0.0600
8	0.1360	0.0600

In [26]:

```python
reduction_rate = 0.5

opt_portfolio_50_123 = sector_neutral_optimal_replicating_portfolio(
    reduction_rate=0.5,
    carbon_intensity=carbon_intensity_123,
    reference_portfolio=weights_ref,
    covariance_matrix=cov_mat,
    sector_mappings=sector_mappings
)
print('Replication volatility =', replication_vol(opt_portfolio_50_123, weights_ref,
print('Optimized portfolio vs benchmark:')
pd.DataFrame(np.vstack([opt_portfolio_50_123, weights_ref]).T, index=index, columns=[
```

```
Replication volatility = 0.01918627701732338
Optimized portfolio vs benchmark:
```

Out[26]:

	w50%	b
1	0.1866	0.2000
2	0.1242	0.1700
3	0.1934	0.1700
4	0.1423	0.1300
5	0.0855	0.1100
6	0.0777	0.1000
7	0.0036	0.0600
8	0.1867	0.0600

In [27]:

```python
reduction_rate = 0.7

opt_portfolio_70_123 = sector_neutral_optimal_replicating_portfolio(
    reduction_rate=0.7,
    carbon_intensity=carbon_intensity_123,
    reference_portfolio=weights_ref,
    covariance_matrix=cov_mat,
    sector_mappings=sector_mappings
)
print('Replication volatility =', replication_vol(opt_portfolio_70_123, weights_ref,
print('Optimized portfolio vs benchmark:')
pd.DataFrame(np.vstack([opt_portfolio_70_123, weights_ref]).T, index=index, columns=[
```

```
Replication volatility = 0.02825432656049339
Optimized portfolio vs benchmark:
```

Out[27]:

	w70%	b
1	0.2051	0.2000
2	0.0881	0.1700
3	0.1776	0.1700
4	0.1558	0.1300
5	0.0034	0.1100
6	0.0614	0.1000
7	0.0000	0.0600
8	0.3085	0.0600

In [28]:

```python
reduction_rates = 0.01 * np.arange(0, 90, 5)
rep_vols = []
rep_vols_esct_neutr = []
```

```
for reduction_rate in reduction_rates:
    opt_portfolio = optimal_replicating_portfolio(
        reduction_rate=reduction_rate,
        carbon_intensity=carbon_intensity_123,
        reference_portfolio=weights_ref,
        covariance_matrix=cov_mat,
    )
    opt_portfolio_sect_neut = sector_neutral_optimal_replicating_portfolio(
        reduction_rate=reduction_rate,
        carbon_intensity=carbon_intensity_123,
        reference_portfolio=weights_ref,
        covariance_matrix=cov_mat,
        sector_mappings=sector_mappings
    )
    rep_vols.append(replication_vol(opt_portfolio, weights_ref, cov_mat))
    rep_vols_esct_neutr.append(replication_vol(opt_portfolio_sect_neut, weights_ref,
```

In [29]:

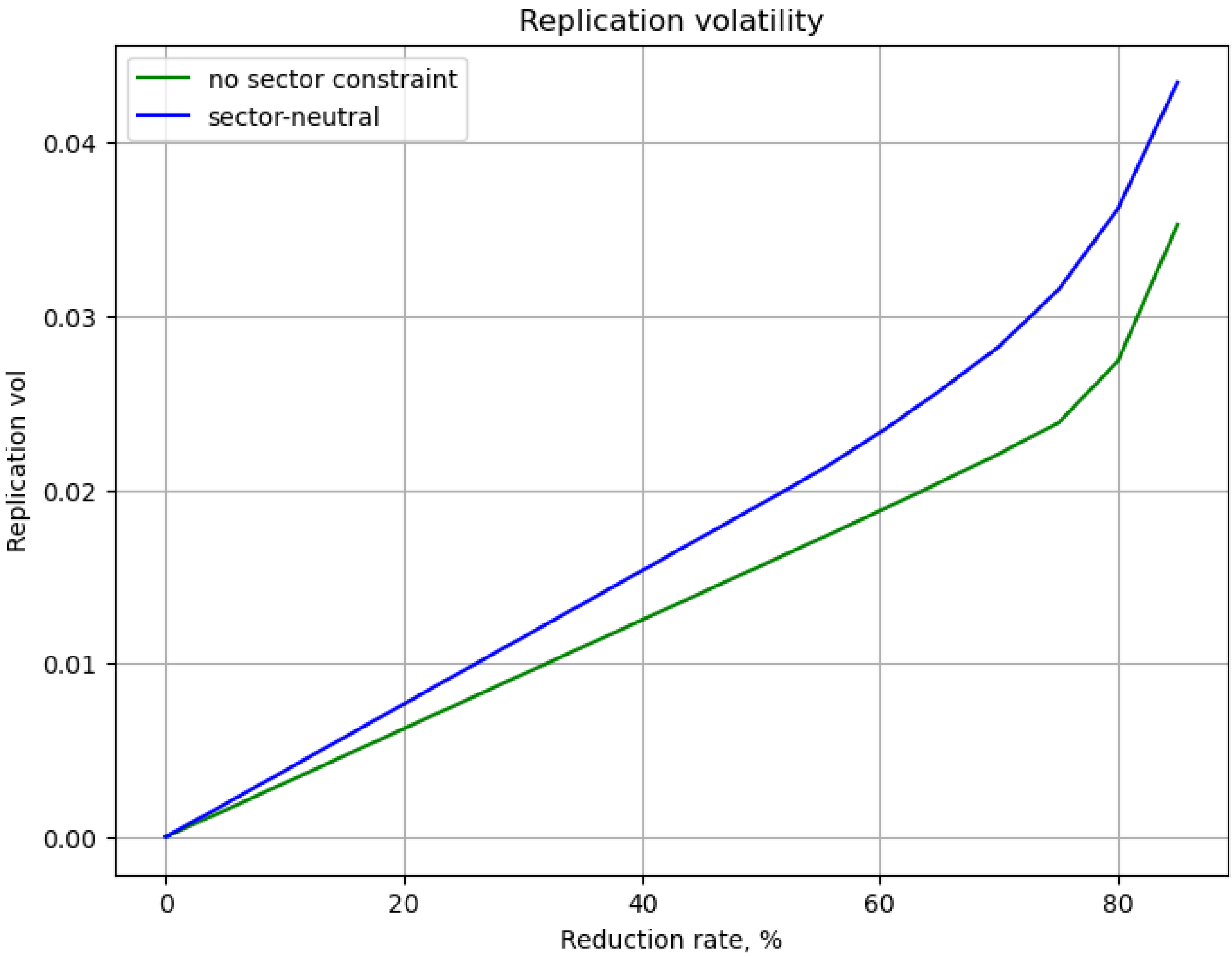
```
results_123 = pd.DataFrame(index=index)
results_123['b'] = weights_ref
results_123['20%'] = opt_portfolio_20_123
results_123['30%'] = opt_portfolio_30_123
results_123['50%'] = opt_portfolio_50_123
results_123['70%'] = opt_portfolio_70_123
results_123
```

Out[29]:

	b	20%	30%	50%	70%
1	0.2000	0.1946	0.1920	0.1866	0.2051
2	0.1700	0.1517	0.1425	0.1242	0.0881
3	0.1700	0.1794	0.1840	0.1934	0.1776
4	0.1300	0.1349	0.1374	0.1423	0.1558
5	0.1100	0.1002	0.0953	0.0855	0.0034
6	0.1000	0.0911	0.0866	0.0777	0.0614
7	0.0600	0.0374	0.0262	0.0036	0.0000
8	0.0600	0.1107	0.1360	0.1867	0.3085

In [30]:

```
fig, ax = plt.subplots(figsize=(8, 6))
ax.plot(reduction_rates * 100, rep_vols, 'g', label='no sector constraint')
ax.plot(reduction_rates * 100, rep_vols_esct_neutr, 'b', label='sector-neutral')
ax.grid()
ax.legend()
ax.set_title('Replication volatility')
ax.set_xlabel('Reduction rate, %')
ax.set_ylabel('Replication vol')
plt.show()
```



As expected, replication vol is higher when sector constraint is present.