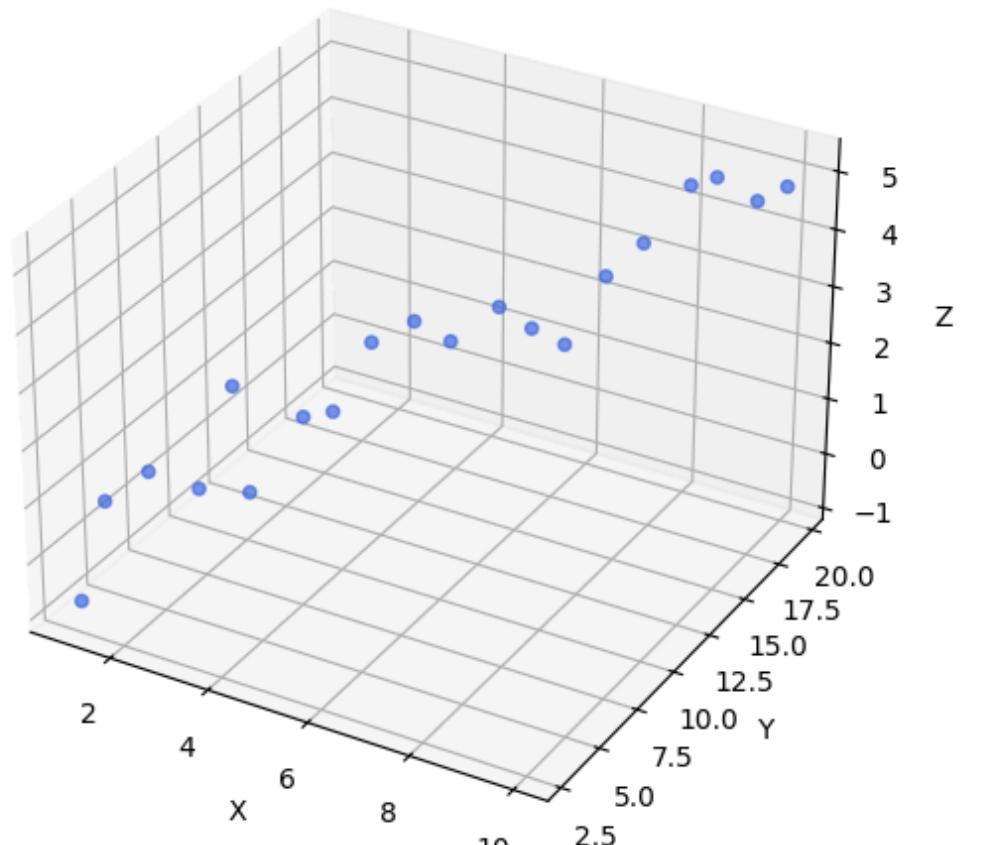


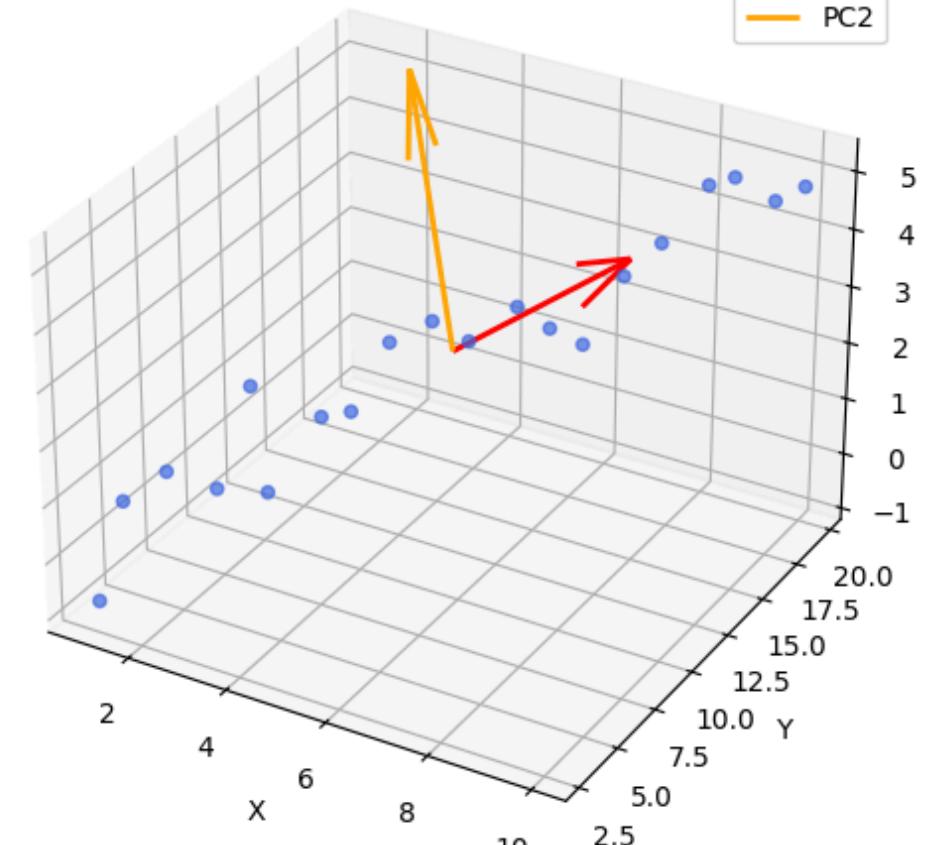
# REDUKCJA RYZYKA INWESTYCYJNEGO Z POMOCĄ ALGEBRY LINIOWEJ



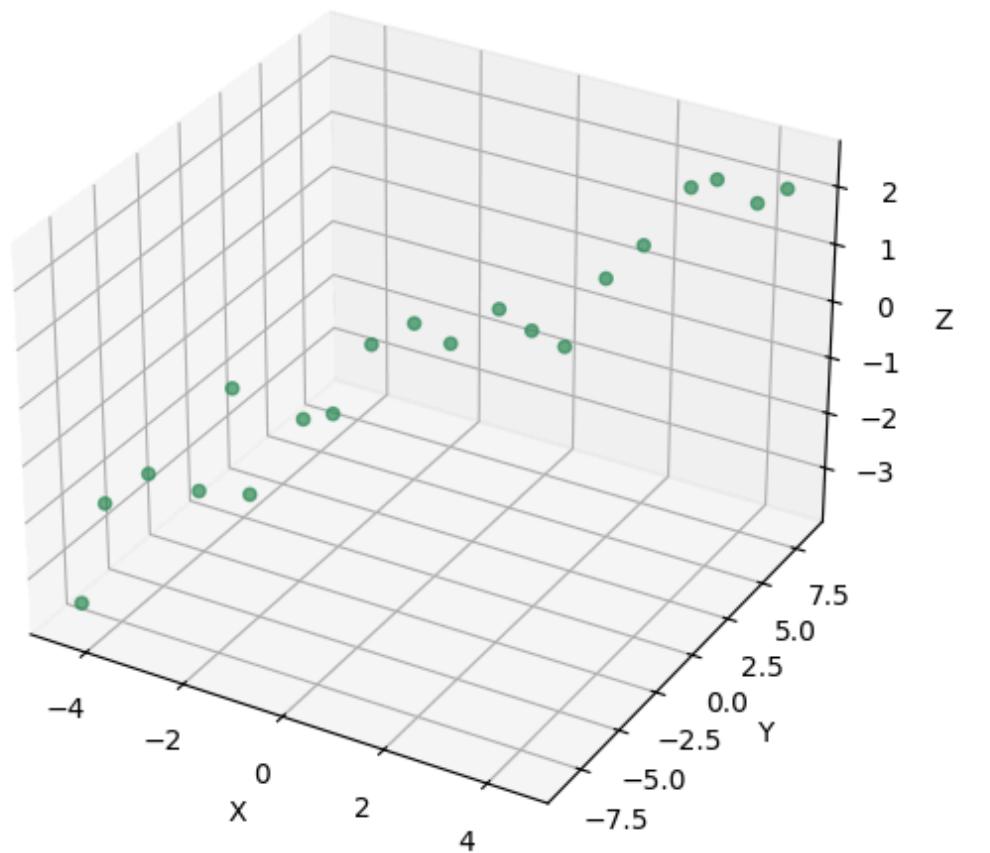
1. Oryginalne dane



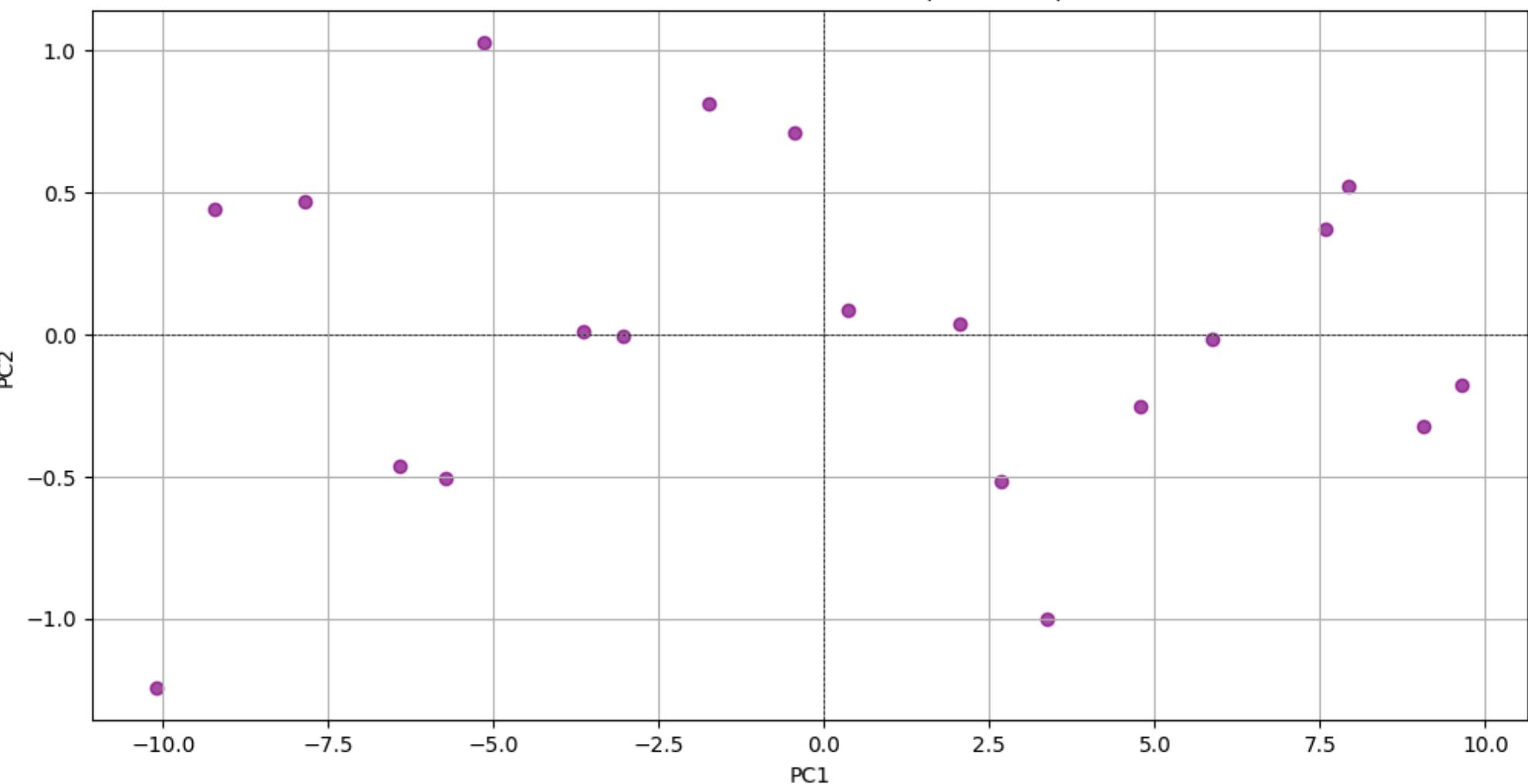
2. Oryginalne dane + PC1, PC2



3. Wycentrowane dane



4. Dane zredukowane do 2D (PC1 i PC2)



**DANE  $\sim \mathbf{X}$**

MACIERZ KOWARIANCJI:

$$C = \frac{1}{n-1} \times \mathbf{X}^T \mathbf{X}$$

Szukamy prostej, względem której dane sa najbardziej rozciagniete.

**Dlaczego wektory własne macierzy kowariancji sa potrzebne?**

$$Cv = \lambda v$$

Mnożąc wektor przez macierz kowariancji, nie zmieniamy jego kierunku.

**Idealny kandydat na oś!**

Im wieksza  $\lambda$ , tym wieksza wariancja.

# Rozkład SVD i macierz kowariancji

Rozkład SVD macierzy danych:

$$X = USV^T$$

$$X^T = VSU^T$$

Macierz kowariancji:

$$C = \frac{X^T X}{n - 1} = V \frac{S^2}{n - 1} V^T$$

Wartości własne:

$$\lambda_i = \frac{\sigma_i^2}{n - 1}$$

## Własności macierzy kowariancji

- $C$  jest macierzą symetryczną
- $C$  ma rozkład własny  $C = V\Lambda V^T$

## Redukcja wymiaru

Jak chcemy zredukować wymiar:  $X \cdot V_1$

```
1 import yfinance as yf
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5
6 tickers = [
7     'AAPL', 'MSFT', 'NVDA', 'GOOGL', 'META',
8     'JPM', 'V', 'MA',
9     'PG', 'KO', 'PEP',
10    'XOM', 'CVX',
11    'JNJ', 'UNH'
12]
13
14
15 data = yf.download(tickers, start="2023-06-01", end="2025-06-01", auto_adjust=True)[['Close']]
16 returns = data.pct_change().dropna()
17
18 # PCA przez SVD na zwrotach
19 X = returns.values
20 X_centered = X - np.mean(X, axis=0)
21
22 U, S, Vt = np.linalg.svd(X_centered, full_matrices=False)
23 principal_components = Vt
24 X_pca = X_centered @ principal_components.T
25
26 explained_variance = (S ** 2) / (X.shape[0] - 1)
27 explained_variance_ratio = explained_variance / np.sum(explained_variance)
28
29 print("Udział wariancji (na zwrotach):")
30 for i, ratio in enumerate(explained_variance_ratio[:10]):
31     print(f"PC{i+1}: {ratio:.2%}")
32
33
34 plt.figure(figsize=(10, 6))
35 plt.scatter(X_pca[:, 0], X_pca[:, 1], alpha=0.5, c='tomato', edgecolor='k')
36 plt.xlabel('Główna składowa 1')
37 plt.ylabel('Główna składowa 2')
38 plt.title('PCA (na dziennych stopach zwrotu)')
39 plt.grid(True)
40 plt.tight_layout()
41 plt.show()
42
```

```
num_components = 4
selected_components = principal_components[:num_components]

weights = selected_components.T @ np.diag(1 / explained_variance[:num_components])
weights = weights.sum(axis=1)

# nie robimy shortow
weights = np.maximum(weights, 0)

weights = weights / weights.sum()

print("Wagi portfela optymalizowanego PCA:")
for ticker, weight in zip(tickers, weights):
    print(f"{ticker}: {weight:.2%}")

equal_weights = np.ones(len(tickers)) / len(tickers)

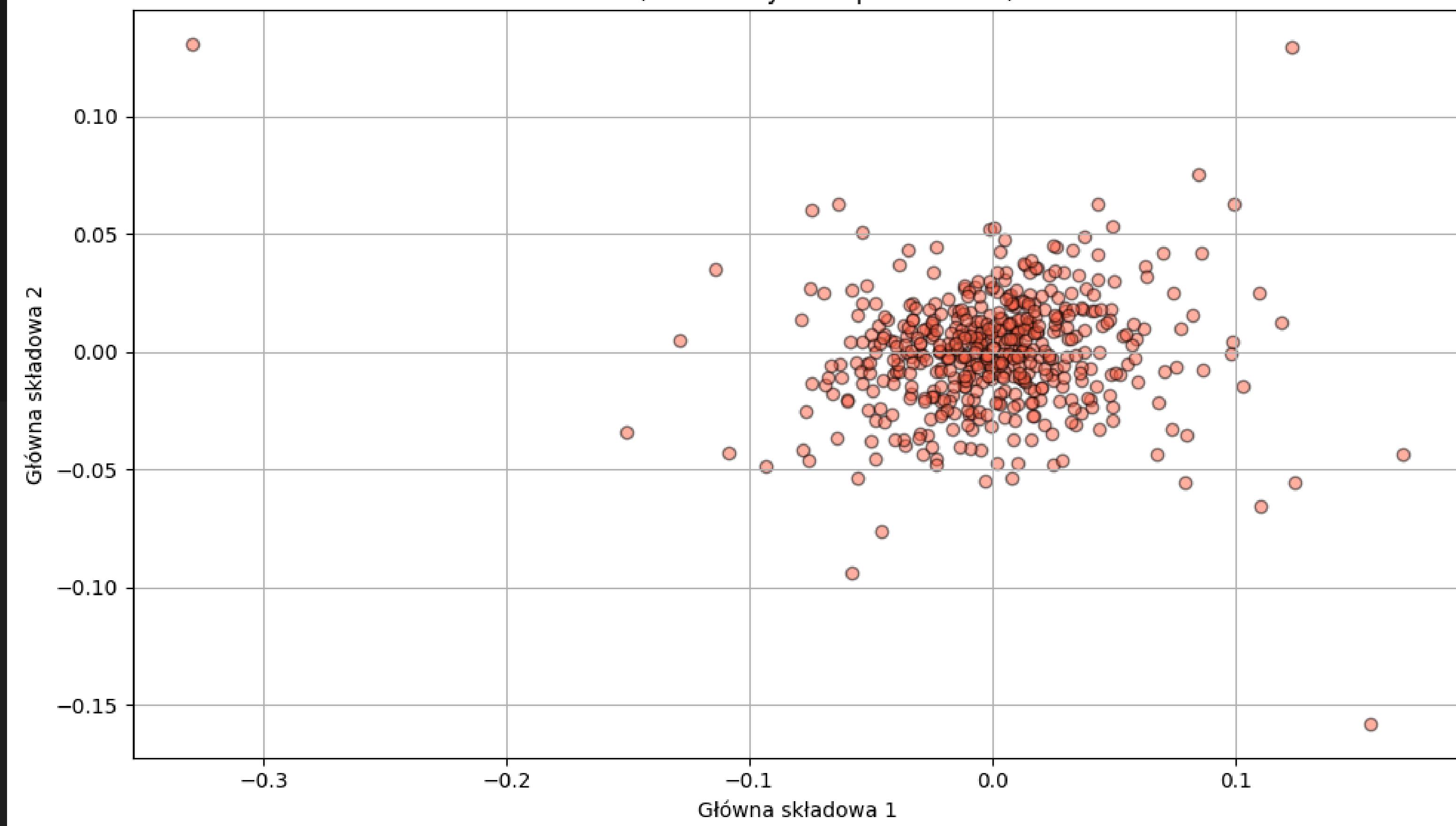
portfolio_pca = (returns * weights).sum(axis=1)
portfolio_equal = (returns * equal_weights).sum(axis=1)

plt.figure(figsize=(10, 5))
plt.plot(np.cumprod(1 + portfolio_pca), label='Portfel PCA')
plt.plot(np.cumprod(1 + portfolio_equal), label='Portfel równych wag')
plt.title("Porównanie strategii")
plt.xlabel("Dni")
plt.ylabel("Wartość portfela (start = 1)")
plt.legend()
plt.grid()
plt.show()
```

## Udział wariancji (na zwrotach):

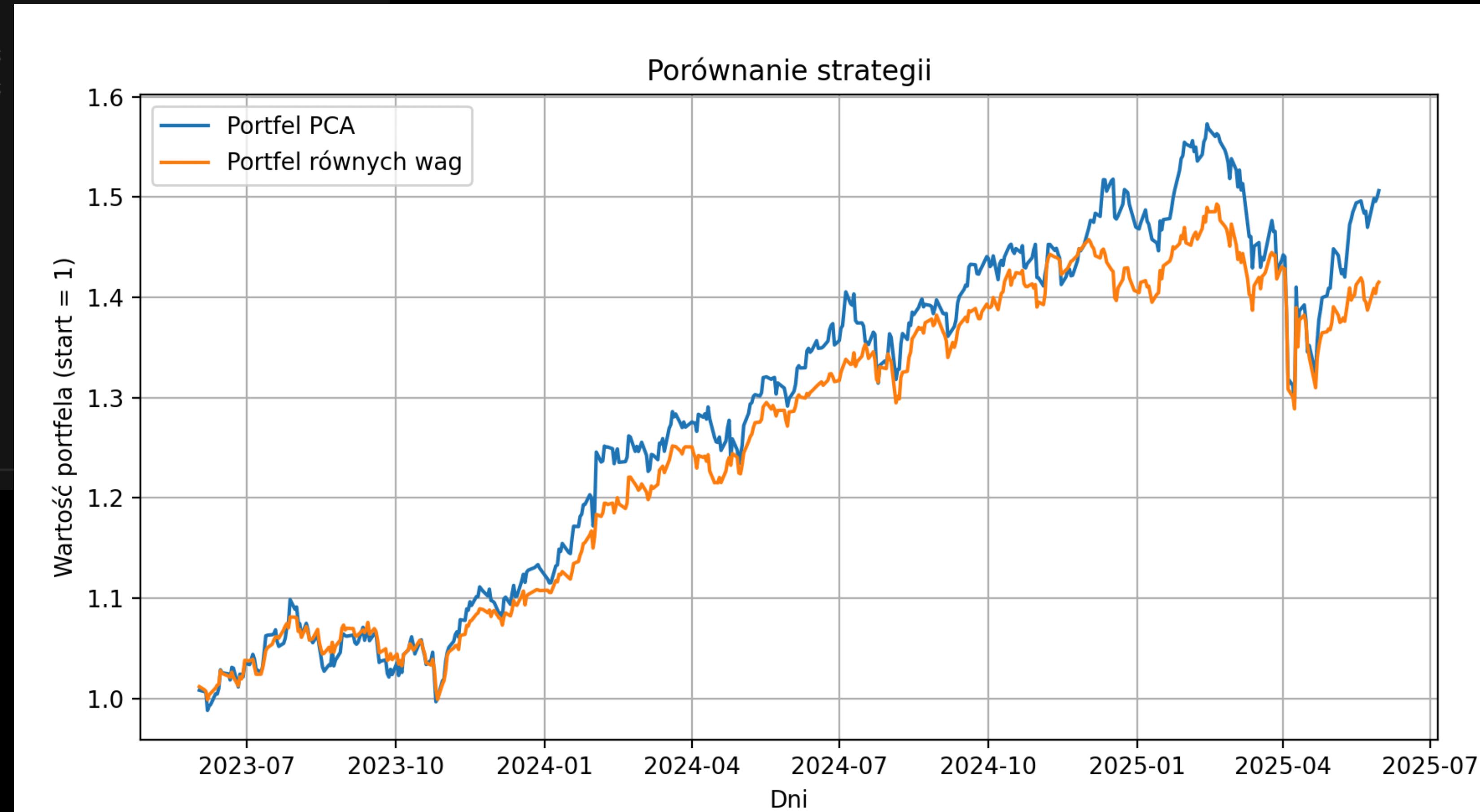
PC1: 39.39%  
PC2: 14.74%  
PC3: 11.05%  
PC4: 8.22%  
PC5: 5.69%  
PC6: 4.87%  
PC7: 3.74%  
PC8: 3.32%  
PC9: 2.08%  
PC10: 2.00%

PCA (na dziennych stopach zwrotu)



## Wagi portfela optymalizowanego PCA:

AAPL: 11.82%  
MSFT: 0.00%  
NVDA: 15.54%  
GOOGL: 5.53%  
META: 0.00%  
JPM: 7.80%  
V: 5.43%  
MA: 23.97%  
PG: 7.85%  
KO: 0.00%  
PEP: 8.91%  
XOM: 8.07%  
CVX: 0.00%  
JNJ: 5.08%  
UNH: 0.00%



# PROGRAMOWANIE LINIOWE

## Programowanie liniowe

Rozwiązywanie układów nierówności liniowych to chleb powszedni dla algebraików. Naturalnie więc, badanie przedziałów i optymalizacja obliczeń, działania firmy czy portfela inwestycyjnego - algebra liniowa może nam w tym pomóc. Chciałbym pokazać, jak wykorzystać programowanie liniowe do zoptymalizowania portfela inwestycyjnego.

## Przykładowy problem

	Matematyka Dyskretna (MD)	Analiza
Czas [h]	1	2
Wytrzymałość [pkt]	3	2

## Sformułowanie matematyczne

Funkcja celu:

$$z(x) = 5x_1 + 4x_2 \rightarrow \max$$

Ograniczenia:

$$\begin{cases} 3x_1 + 2x_2 \leq 12 \\ x_1 + 2x_2 \leq 8 \\ x_1, x_2 \geq 0 \end{cases}$$

## Zadanko

Student ma do wyboru dwa rodzaje sesji nauki: z Matematyki Dyskretnej i z Analizy. Każda sesja charakteryzuje się:

- Sesja MD: kosztuje 1 godzinę i 3 punkty wytrzymałości
- Sesja Analizy: kosztuje 2 godziny i 2 punkty wytrzymałości

Każda sesja nauki daje punkty przygotowania:

- Sesja MD: 5 pkt przygotowania
- Sesja Analizy: 4 pkt przygotowania

Student ma do dyspozycji 8 godzin nauki dziennie i maksymalnie 12 punktów wytrzymałości. Jak powinien wyglądać zoptymalizowany dzień nauki?

$$\text{Max } z = 5x_1 + 4x_2$$

$C_B$  to uogólnione zmienne w równaniu

$s_1, s_2 \sim$  zmiennego bazowego

$$\begin{cases} x_1 + 2x_2 \leq 8 \\ 3x_1 + 2x_2 \leq 12 \\ x_1, x_2 \geq 0 \end{cases}$$

(Działającym zmiennym swobodnym zmiennych)  $\begin{cases} x_1 + 2x_2 + s_1 = 8 \\ 3x_1 + 2x_2 + s_2 = 12 \end{cases}$

Baza	$C_B$	$x_1$	$x_2$	$s_1$	$s_2$	$b$ (wyniki)
	5	1	0	0	0	6 (wyniki)
$s_1$	0	0	$\frac{1}{3}$	1	$-\frac{1}{3}$	4
$x_2$	5	1	$\frac{2}{3}$	0	$\frac{1}{3}$	4
$Z_j$	5	$\frac{10}{3}$	0	$\frac{5}{3}$	20	
$C_j - Z_j$	0	$\frac{2}{3}$	0	$-\frac{5}{3}$		

symbol zmiennego w funkcji, jeśli jednostkę zmienność zostanie włączona do równiania

$Z_j$  otrzymujemy mnożąc wartości

I

Baza	$C_B$	$x_1$	$x_2$	$s_1$	$s_2$	$b$ (wyniki)
	5	1	0	0	0	6 (wyniki)
$s_1$	0	0	$\frac{1}{3}$	1	$-\frac{1}{3}$	4
$x_2$	5	1	$\frac{2}{3}$	0	$\frac{1}{3}$	4
$Z_j$	5	$\frac{10}{3}$	0	$\frac{5}{3}$	20	
$C_j - Z_j$	0	$\frac{2}{3}$	0	$-\frac{5}{3}$		

$$\text{Many podstawic} \quad \begin{vmatrix} C_B \\ 0 \end{vmatrix} \cdot \begin{vmatrix} x_1 \\ 1 \end{vmatrix} \quad Z_j = 0 \cdot 1 + 0 \cdot 3$$

$$\text{dopuszczalne rozwiązań} \quad x_1 = 0, x_2 = 0 \quad s_1 = 8 \quad s_2 = 12$$

Jak wygląda iteracja?

Wybieramy zmienną z największą wartością  $C_j - Z_j$  jej odpowiadającą. W 1 iteracji  $x_1$  musi zastąpić  $s_1$  lub  $s_2$ . Która usunąć? Dzielimy b przez kolumnę  $x_1$  w kolejności dla  $s_1$ : 8

dla  $s_2$ : 4, mniejsza zmiana wartości którą byta w kolumnie  $x_1$ , i rządzie  $s_2$  to tzw. pivot element

- rząd  $x_1$  uzywamy dla wszystkich el z  $s_1$  przez pivot

- na przecięciu kol  $x_1$  i indeksu  $s_1$  potrzebujemy零 -Robin

jeżeli obliczamy od starego rządu się nowy rząd  $x_1$ , tyle razem ile trzeba.

II

Baza	$C_B$	$x_1$	$x_2$	$s_1$	$s_2$	$b$ (wyniki)
	5	1	0	0	0	6 (wyniki)
$x_2$	0	1	$\frac{3}{4}$	$-\frac{1}{4}$	3	
$x_1$	5	1	0	$-\frac{1}{2}$	2	
$Z_j$	5	$\frac{5}{4}$	$\frac{1}{4}$	1	23	
$C_j - Z_j$	0	0	$-\frac{3}{4}$	-1		

$$G - z_j \begin{array}{c|ccccc} & 0 & \frac{3}{2} & 0 & \frac{5}{3} & 20 \\ \hline & \frac{3}{2} & 0 & 0 & -\frac{5}{3} & \end{array}$$

II  $x_2 \leftrightarrow s_1$

Baza CB	$x_1$	$x_2$	$s_1$	$s_2$	$b$ (wyniki)
$x_2$	5	0	0	0	
$x_1$	5	1	$\frac{3}{5}$	$-\frac{1}{5}$	3
$z_j$	5	5	$\frac{-3}{5}$	1	23
$G - z_j$	0	0	$\frac{3}{5}$	-1	

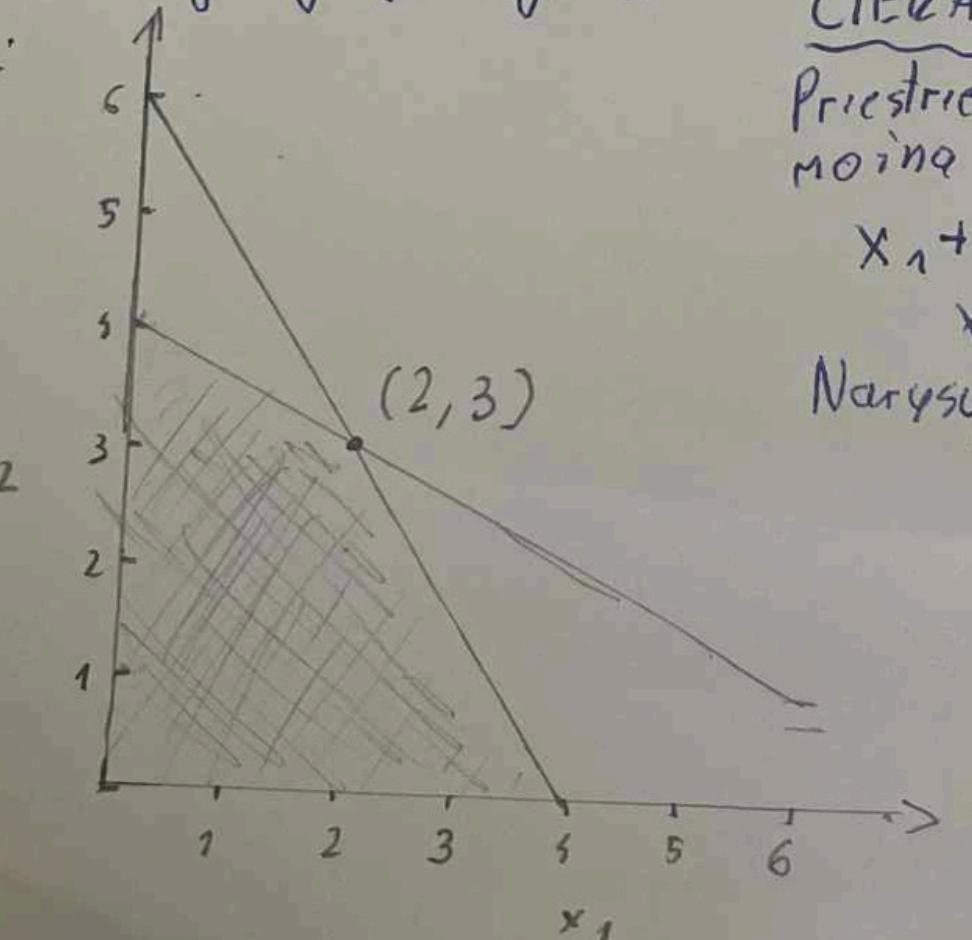
Optimum zostaje osiągnięte, jeśli wszystkie wartości w  $G - z_j$  są nieujemne

Rozwiązanie:

$$x_1 = 2$$

$$x_2 = 3$$

$$Z = 28$$



Wybieramy zmienne z największą wartością  $c_j - z_j$  jej odpowiadającą. W 1 iteracji  $x_1$  musi zastąpić  $s_1$  lub  $s_2$ . Które wybrać?

Dzielimy b przez kolumnę  $x_1$  wektorowo

dla  $s_1 : 8$

dla  $s_2 : 5$ , mniejsza zmiana

- wartość która była w kolumnie  $x_1$  i rządzie  $s_2$  to tzw. pivot element

- rząd  $x_1$  użyskujemy dzieląc wszystkie el z  $s_1$  przez pivot

- na przecięciu kol  $x_1$  i rędu  $s_1$  potrzebujemy zera. Robimy to sobie odgrywając od starego rządu  $s_1$  nowy rząd  $x_1$ , tyle raz, ile trzeba.

### CIEKAWE

Priestępce dopuszczalnych rozwiązań można reprezentować geometrycznie:

$$x_1 + 2x_2 = 8$$

$$x_2 = \frac{8 - x_1}{2}$$

$$3x_1 + 2x_2 = 12$$

$$x_2 = \frac{12 - 3x_1}{2}$$

Narysujmy to:

Metoda simplex również ma swoją interpretację geometryczną.

## Problem optymalizacji portfela z MAD jako programowanie liniowe

### Zmienne decyzyjne

- $w_i$  - waga  $i$ -tego aktywa w portfelu,  $i = 1, \dots, n$
- $z_t$  - wartość bezwzględna odchylenia portfela od oczekiwanej zwrotu w okresie  $t$ ,  $t = 1, \dots, T$

### Parametry

- $r_{t,i}$  - historyczny zwrot  $i$ -tego aktywa w okresie  $t$
- $\mu_i = \frac{1}{T} \sum_{t=1}^T r_{t,i}$  - średni zwrot  $i$ -tego aktywa
- $\mu_p$  - docelowy oczekiwany zwrot portfela
- $T$  - liczba okresów obserwacji
- $n$  - liczba aktywów

### Funkcja celu

Minimalizacja średniego odchylenia absolutnego (MAD):

$$\text{minimize } \frac{1}{T} \sum_{t=1}^T z_t$$

### Ograniczenia

$$(1) \text{ Odchylenia bezwzględne: } z_t \geq \sum_{i=1}^n w_i r_{t,i} - \mu_p, \quad t = 1, \dots, T$$
$$z_t \geq - \left( \sum_{i=1}^n w_i r_{t,i} - \mu_p \right), \quad t = 1, \dots, T$$

$$(2) \text{ Suma wag: } \sum_{i=1}^n w_i = 1$$

$$(3) \text{ Oczekiwany zwrot: } \sum_{i=1}^n w_i \mu_i = \mu_p$$

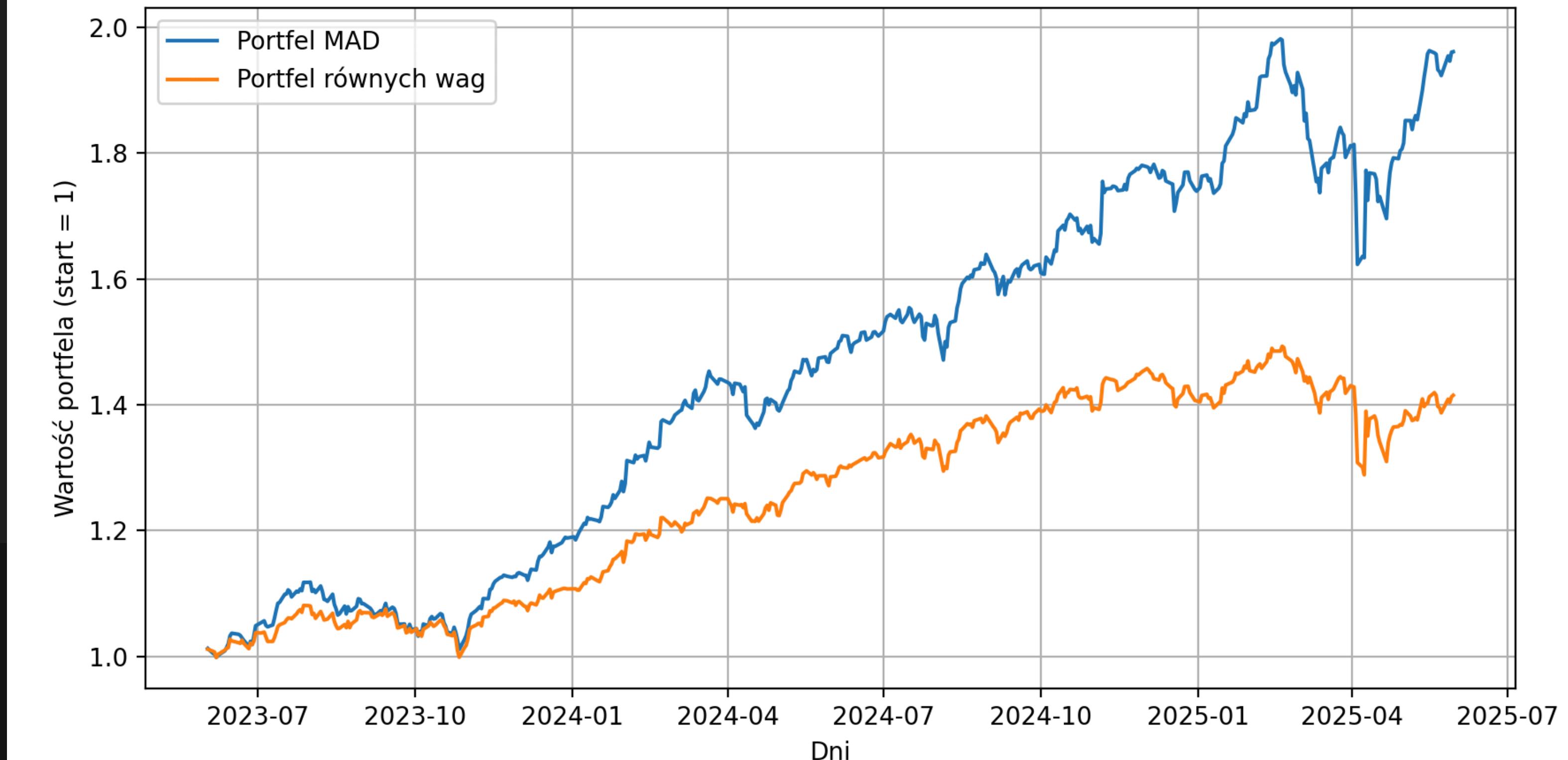
$$(4) \text{ Brak krótkiej sprzedaży: } w_i \geq 0, \quad i = 1, \dots, n$$
$$z_t \geq 0, \quad t = 1, \dots, T$$

```
1 import yfinance as yf
2 import numpy as np
3 import pulp
4
5 tickers = [
6     'AAPL', 'MSFT', 'NVDA', 'GOOGL', 'META',
7     'JPM', 'V', 'MA',
8     'PG', 'KO', 'PEP',
9     'XOM', 'CVX',
10    'JNJ', 'UNH'
11 ]
12 start_date = '2023-01-01'
13 end_date = '2025-01-01'
14 data = yf.download(tickers, start=start_date, end=end_date, auto_adjust=False)
15 r = data['Adj Close'].pct_change().dropna().to_numpy() # Macierz zwrotów: r_{i,t} (T x n)
16 n = len(tickers)
17 T = r.shape[0]
18 mu = np.mean(r, axis=0) # Oczekiwane zwroty aktywów ( $\mu_i$ ), średnia z  $r_{i,t}$  dla każdego i
19
20
21 print("Oczekiwane dzienne zwroty aktywów ( $\mu_i$ ):")
22 for i, ticker in enumerate(tickers):
23     print(f"{ticker}: {mu[i]:.6f}")
24 min_mu = np.min(mu)
25 max_mu = np.max(mu)
26 print(f"Zakres dla  $\mu_p$ : [{min_mu:.6f}, {max_mu:.6f}]")
27
28 # oczekiwane  $\mu_p$  musi być w granicach [min( $\mu_i$ ), max( $\mu_i$ )
29 # ja sobie wyborę takie:
30 mu_p = 0.0014
31
32 print(f"Wybrano  $\mu_p$ : {mu_p:.6f}")
33
```

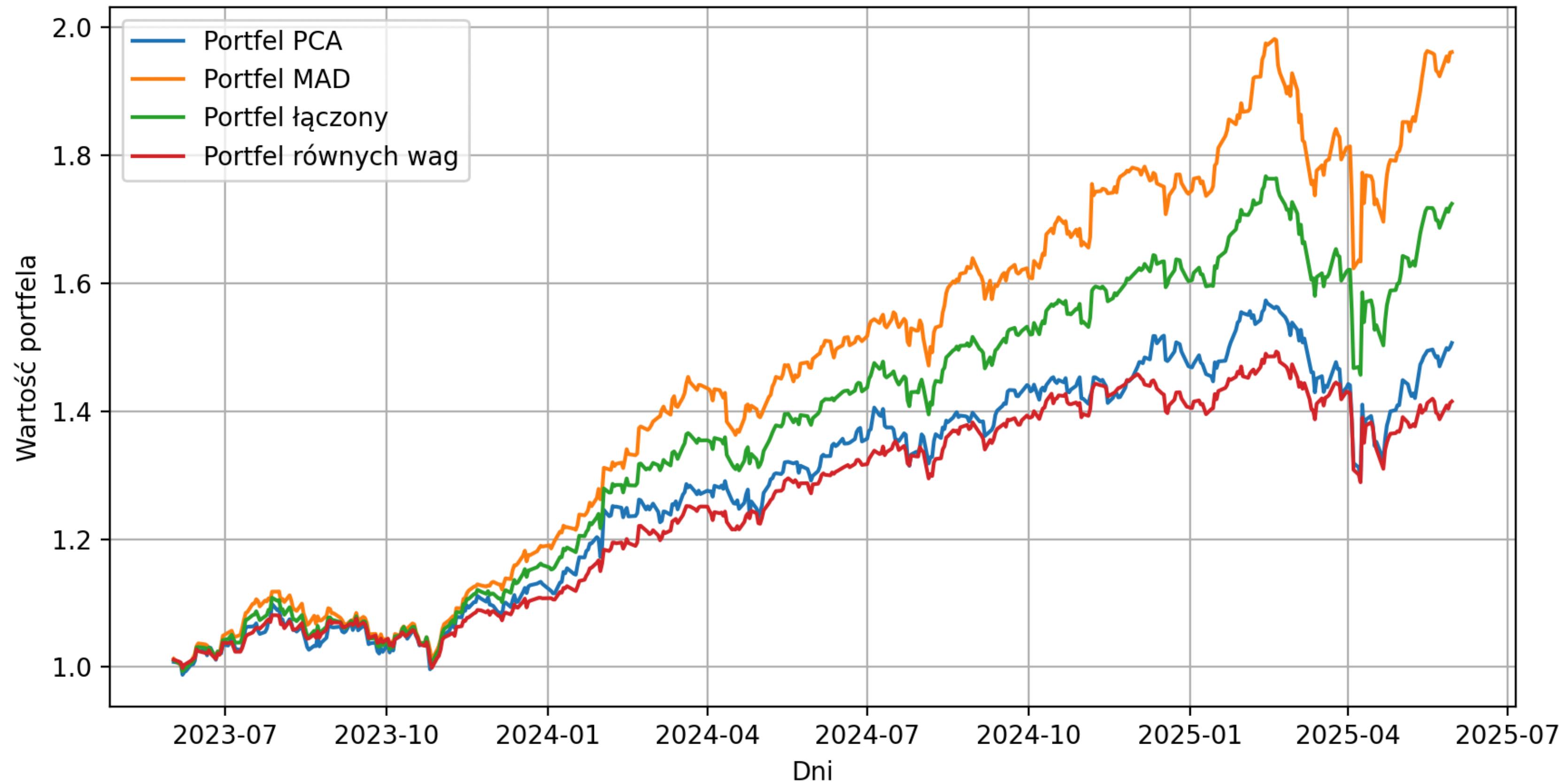
```
30 mu_p = 0.0014
31
32 print(f"Wybrano μ_p: {mu_p:.6f}")
33
34 # Definicja problemu programowania liniowego
35 problem = pulp.LpProblem("MAD_Portfolio_Optimization", pulp.LpMinimize)
36
37 # Zmienne decyzyjne
38 # w_i: wagi aktywów w portfelu (nieujemne, nie robimy shortow)
39 w = [pulp.LpVariable(f"w_{i}", lowBound=0, cat='Continuous') for i in range(n)]
40 # z_t: zmienne pomocnicze reprezentujące odchylenia bezwzględne dla każdego okresu t
41 z = [pulp.LpVariable(f"z_{t}", lowBound=0, cat='Continuous') for t in range(T)]
42
43 # Funkcja celu: minimalizacja MAD = (1/T) * sum(z_t), wzielismy właśnie MAD bo jest liniowe a nie kwadratowe tak
44 # jak standardowo w modelu Markowitza, bo nie wtedy musielibysmy rozważać bardziej zaawansowane programy
45 problem += (1/T) * pulp.lpSum(z), "Minimalizacja_MAD"
46
47
48 for t in range(T):
49     port_return = pulp.lpSum(w[i] * r[t, i] for i in range(n)) # Zwrot portfela w okresie t
50     problem += z[t] >= port_return - mu_p, f"z_t_geq_positive_dev_{t}"
51     problem += z[t] >= -(port_return - mu_p), f"z_t_geq_negative_dev_{t}"
52
53
54 problem += pulp.lpSum(w) == 1, "Suma_wag"
55
56
57 problem += pulp.lpSum(w[i] * mu[i] for i in range(n)) == mu_p, "Oczekiwany_zwrot"
58
59 status = problem.solve()
60
61 print("\nStatus:", pulp.LpStatus[status])
62 print(f"MAD portfela: {pulp.value(problem.objective):.6f}")
63 print("Wagi portfela:")
64 for i in range(n):
65     print(f"Aktywo {tickers[i]} (w_{i}): {pulp.value(w[i]):.4f}")
66 portfolio_return = sum(pulp.value(w[i]) * mu[i] for i in range(n))
67 print(f"Oczekiwany dzienny zwrot portfela (μ_p): {portfolio_return:.6f}")
68 portfolio_returns = np.sum(r * [pulp.value(w[i]) for i in range(n)], axis=1)
69 mad = np.mean(np.abs(portfolio_returns - mu_p))
70 print(f"Historyczne MAD portfela: {mad:.6f}")
71
```

## Porównanie strategii - MAD

MAD portfela: 0.005649  
Wagi portfela:  
Aktywo AAPL (w\_0): 0.0793  
Aktywo MSFT (w\_1): 0.0000  
Aktywo NVDA (w\_2): 0.0042  
Aktywo GOOGL (w\_3): 0.0241  
Aktywo META (w\_4): 0.1498  
Aktywo JPM (w\_5): 0.1078  
Aktywo V (w\_6): 0.0000  
Aktywo MA (w\_7): 0.1052  
Aktywo PG (w\_8): 0.0000  
Aktywo K0 (w\_9): 0.1109  
Aktywo PEP (w\_10): 0.0000  
Aktywo XOM (w\_11): 0.2422  
Aktywo CVX (w\_12): 0.0409  
Aktywo JNJ (w\_13): 0.0511  
Aktywo UNH (w\_14): 0.0844  
Oczekiwany dzienny zwrot portfela ( $\mu_p$ ): 0.001400  
Historyczne MAD portfela: 0.005649



## Porównanie wszystkich strategii



# Shrinkage Ledoita-Wolfa w analizie głównych składowych (PCA)

## Wprowadzenie

Wysokowymiarowe dane, w których liczba zmiennych  $p$  jest porównywalna lub wieksza niż liczba obserwacji  $n$ , mogą prowadzić do problemów z niestabilnością macierzy kowariancji, co utrudnia skuteczną analizę głównych składowych (PCA). Jedna z metod poprawiających jakość macierzy kowariancji jest tzw. **shrinkage Ledoita-Wolfa**.

## Shrinkage Ledoita-Wolfa

Metoda ta polega na obliczeniu nowej macierzy kowariancji na podstawie dwóch składników: macierzy kowariancji wyliczonej z danych oraz prostej, bardziej regularnej macierzy pomocniczej  $F$ . Obie te macierze są ze sobą mieszane w odpowiednich proporcjach:

$$\hat{\Sigma}_{LW} = (1 - \lambda)S + \lambda F,$$

gdzie:

- $S \in \mathbb{R}^{n \times n}$  to macierz kowariancji wyliczona z danych,
- $F$  to pomocnicza macierz kowariancji z uproszczona struktura,
- $\lambda \in [0, 1]$  to współczynnik decydujący o sile działania metody shrinkage.

W tym podejściu zakładamy, że macierz  $F$  zachowuje tylko wariancje zmiennych, ale nie zawiera żadnych zależności między nimi. Taka macierz wygląda następująco:

$$F = \text{diag}(S),$$

czyli:

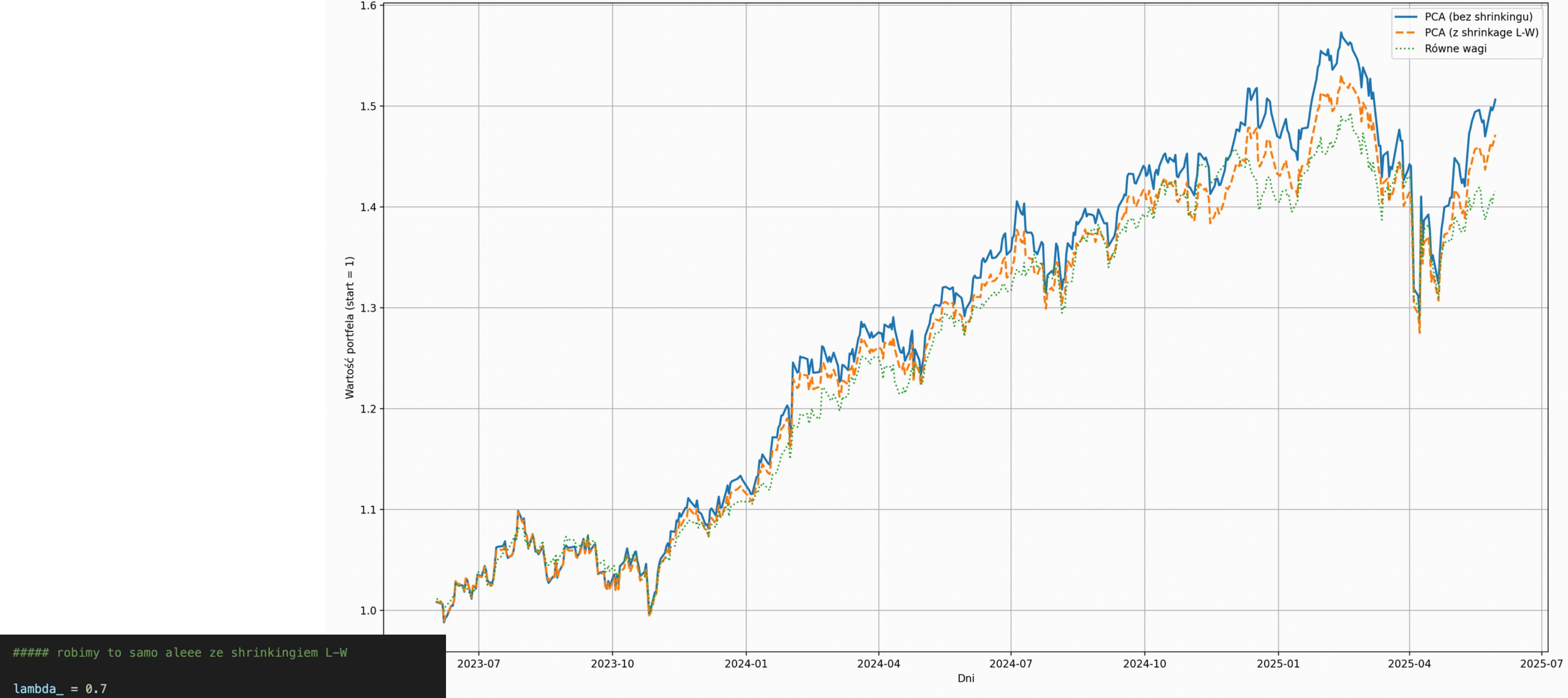
$$F_{ij} = \begin{cases} S_{ii}, & \text{gdy } i = j, \\ 0, & \text{gdy } i \neq j. \end{cases}$$

## Zastosowanie w PCA

Zastosowanie shrinkage Ledoita-Wolfa pozwala stworzyć bardziej uporzadkowaną macierz kowariancji, w której cześć informacji pochodzi z danych, a cześć jest uproszczona. Dzięki temu:

- wyniki PCA są bardziej stabilne,
- łatwiej jest wyciągać główne kierunki zmienności,
- zmniejsza się wpływ szumu i przypadkowych odchyлеń.

Porównanie portfeli: PCA vs PCA z shrinkage



```
##### robimy to samo aleee ze shrinkingiem L-W

lambda_ = 0.7
n = X_centered.shape[0]

S = (X_centered.T @ X_centered) / (n - 1)
F = np.diag(np.diag(S))
cov_shrink = (1 - lambda_) * S + lambda_ * F
# PCA weights na podstawie shrinkowanej macierzy
num_components = 4
selected_components = principal_components[:num_components]
```