Numerical methods for optimization - Project

BY

Crochemar Théo

January - February 2026

Contents

1 Choice of programming language

The programming language chosen was Python and all the computations were realized on Pycharm's 2024.3.4 version. In the .ipynb markdown cells indicate the corresponding codes to each questions.

2 Question 1 : Rewriting of the problem

We consider $n = 12$ assets with random returns $(R_i)_{i=1,\dots,n}$ and portfolio weights $x = (x_1, \dots, x_n) \in \mathbb{R}^n$. The portfolio return is

$$R = \sum_{i=1}^{n} x_i R_i.$$

We denote $\mu_i = \mathbb{E}[R_i]$ et $\Sigma_{ij} = \mathrm{Cov}(R_i, R_j)$, and we fix $\phi > 0$ (in the project $\phi = 5$). The objective is

$$J(x) = -\phi\,\mathbb{E}(R) + \mathrm{Var}(R),$$

under constraints

$$x_i \geq 0 \quad (i = 1, \dots, n), \qquad \sum_{i=1}^{n} x_i = 1.$$

The equality constraint allows one variable to be eliminated, for example x_n :

$$x_n = 1 - \sum_{i=1}^{n-1} x_i.$$

We define the reduced variable $x = (x_1, \dots, x_{n-1}) \in \mathbb{R}^{n-1}$ and replace the eliminated variable x_n by the expression above. The constrains then become

$$x_i \geq 0 \quad (i = 1, \dots, n-1), \qquad x_n \geq 0 \iff 1 - \sum_{i=1}^{n-1} x_i \geq 0.$$

We thus obtain a reduced problem of dimension one less $n - 1$:

$$\min_{x \in \mathbb{R}^{n-1}} \tilde{J}(x) \quad \text{sous} \quad x_i \geq 0 \ (i = 1, \dots, n-1), \ \ 1 - \sum_{i=1}^{n-1} x_i \geq 0,$$

where $\tilde{J}$ the objective function is obtained by substitution.

By substituting $x_n = 1 - \sum_{i=1}^{n-1} x_i$:

$$R = \sum_{i=1}^{n-1} x_i R_i + \Big(1 - \sum_{i=1}^{n-1} x_i\Big) R_n = R_n + \sum_{i=1}^{n-1} x_i (R_i - R_n).$$

Taking expectations :

$$\mathbb{E}(R) = \mathbb{E}(R_n) + \sum_{i=1}^{n-1} x_i \big(\mathbb{E}(R_i) - \mathbb{E}(R_n)\big).$$

Thus

$$-\phi\,\mathbb{E}(R) = -\phi\,\mathbb{E}(R_n) - \phi\sum_{i=1}^{n-1} x_i\big(\mathbb{E}(R_i) - \mathbb{E}(R_n)\big).$$

Let us define

$$A := \sum_{i=1}^{n-1} x_i(R_i - R_n).$$

Then $R = R_n + A$ and

$$\mathrm{Var}(R) = \mathrm{Var}(R_n + A) = \mathrm{Var}(R_n) + \mathrm{Var}(A) + 2\,\mathrm{Cov}(R_n, A).$$

(i) Constant term

$$\mathrm{Var}(R_n) = \mathrm{Cov}(R_n, R_n).$$

(ii) Quadratic term $\mathrm{Var}(A)$. We have

$$\mathrm{Var}(A) = \sum_{i=1}^{n-1}\sum_{j=1}^{n-1} x_i x_j\,\mathrm{Cov}(R_i - R_n,\ R_j - R_n).$$

Now, by bilinearity of the covariance

$$\begin{aligned}\mathrm{Cov}(R_i - R_n,\ R_j - R_n) &= \mathrm{Cov}(R_i, R_j) - \mathrm{Cov}(R_i, R_n) - \mathrm{Cov}(R_n, R_j) + \mathrm{Cov}(R_n, R_n)\\ &= \mathrm{Cov}(R_i, R_j) - \mathrm{Cov}(R_i, R_n) - \mathrm{Cov}(R_j, R_n) + \mathrm{Cov}(R_n, R_n),\end{aligned}$$

since
$\mathrm{Cov}(R_n, R_j) = \mathrm{Cov}(R_j, R_n)$.
Thus

$$\mathrm{Var}(A) = \sum_{i=1}^{n-1}\sum_{j=1}^{n-1} x_i x_j\left[\mathrm{Cov}(R_i, R_j) - \mathrm{Cov}(R_i, R_n) - \mathrm{Cov}(R_j, R_n) + \mathrm{Cov}(R_n, R_n)\right].$$

(iii) Linear term $2\,\mathrm{Cov}(R_n, A)$. By linearity of the covariance with respect to its second argument,

$$\mathrm{Cov}(R_n, A) = \sum_{i=1}^{n-1} x_i\,\mathrm{Cov}(R_n, R_i - R_n).$$

Moreover,

$$\mathrm{Cov}(R_n, R_i - R_n) = \mathrm{Cov}(R_n, R_i) - \mathrm{Cov}(R_n, R_n) = \mathrm{Cov}(R_i, R_n) - \mathrm{Cov}(R_n, R_n).$$

Therefore

$$2\,\mathrm{Cov}(R_n, A) = \sum_{i=1}^{n-1} x_i\Big[2\,\mathrm{Cov}(R_i, R_n) - 2\,\mathrm{Cov}(R_n, R_n)\Big].$$

By combining the results of the previous sections, we obtain

$$\begin{aligned}\tilde{J}(x) &= -\phi\,\mathbb{E}(R) + \mathrm{Var}(R)\\ &= -\phi\,\mathbb{E}(R_n) + \mathrm{Cov}(R_n, R_n)\\ &\quad+ \sum_{i=1}^{n-1} x_i \Big[-\phi\big(\mathbb{E}(R_i) - \mathbb{E}(R_n)\big) - 2\,\mathrm{Cov}(R_n, R_n) + 2\,\mathrm{Cov}(R_i, R_n)\Big]\\ &\quad+ \sum_{i=1}^{n-1}\sum_{j=1}^{n-1} x_i x_j \Big[\mathrm{Cov}(R_i, R_j) - \mathrm{Cov}(R_i, R_n) - \mathrm{Cov}(R_j, R_n) + \mathrm{Cov}(R_n, R_n)\Big].\end{aligned}$$

This is what we were looking for, so the reduced problem can be rewritten as :

$$\min_{x\in\mathbb{R}^{n-1}} \tilde{J}(x) \quad \text{subject to} \quad x_i \geq 0 \ (i = 1, \dots, n-1), \ \ 1 - \sum_{i=1}^{n-1} x_i \geq 0.$$

3 Question 2: Steepest descent with Wolfe step lengths

In this question, we minimize the reduced solution.

$$J^e(x) = x^\top Q x + q^\top x + r, \qquad x \in \mathbb{R}^{11},$$

obtained in question 1 by eliminating the equality constraint $\sum_{i=1}^{12} x_i = 1$ through $x_{12} = 1 - \sum_{i=1}^{11} x_i$. The gradient is

$$\nabla J^e(x) = 2Qx + q.$$

Starting from an initial point $x_0 \in \mathbb{R}^{11}$, we iterate

$$p_k = -\nabla J^e(x_k), \qquad x_{k+1} = x_k + \alpha_k p_k,$$

where p_k is the steepest descent direction.

Let $\varphi(\alpha) = J^e(x_k + \alpha p_k)$. At each iteration, the step length $\alpha_k > 0$ is computed by a line-search procedure enforcing the strong Wolfe conditions:

$$\varphi(\alpha_k) \leq \varphi(0) + c_1 \alpha_k \varphi'(0), \qquad \big|\varphi'(\alpha_k)\big| \leq -c_2\,\varphi'(0),$$

with parameters $c_1 = 10^{-4}$ and $c_2 = 0.9$. The line search uses a bracketing phase and a `zoom` procedure ,based on bisection, to ensure both sufficient decrease and curvature.

The iterations are stopped when

$$\|\nabla J^e(x_k)\| \leq 10^{-5}.$$

We used the starting point

$$x_0 = \mathbf{0} \in \mathbb{R}^{11},$$

which corresponds to a full portfolio allocating 100% to asset 12, since $x_{12} = 1 - \mathbf{e}^\top x$.

The method converged in

$$k = 55150 \text{ iterations.}$$

The computed minimizer is

$$x^* = \begin{pmatrix} 0.2685102 \\ -0.6015868 \\ -0.18651315 \\ 0.95014298 \\ 0.05797521 \\ -0.01420054 \\ -0.39572757 \\ 0.18019928 \\ 0.24183311 \\ 0.41863451 \\ 0.06426102 \end{pmatrix}$$

and the objective value at this point is

$$J^e(x^*) \approx 3.0412849729891605.$$

4 Question 3: Newton direction with Wolfe step lengths

We used the same starting point as in question 2:

$$x_0 = \mathbf{0} \in \mathbb{R}^{11}.$$

As expected for a quadratic objective, the method converged in

$$k = 1 \text{ iteration.}$$

The computed minimizer is

$$x^\star \approx \begin{pmatrix} 0.26851035 \\ -0.60158785 \\ -0.18651334 \\ 0.95014403 \\ 0.05797526 \\ -0.01420058 \\ -0.39572781 \\ 0.18019933 \\ 0.24183317 \\ 0.41863455 \\ 0.06426108 \end{pmatrix}, \qquad J^e(x^\star) \approx 3.041284972984613$$

4.1 Discussion vs Question 2 :

The Newton method returns the same minimizer, up to a 10^{-6} decimal, as the steepest descent method of Question 2, since both solve the same unconstrained reduced problem. However, steepest descent required a very large number of iterations, 55150 iterations, because it uses only first-order information and can exhibit zig-zag behavior on ill-conditioned quadratic objectives. In contrast, Newton's method uses second-order curvature information through the Hessian and, for a quadratic objective with constant Hessian, reaches the exact minimizer in one iteration.

5 Question 4: Linear Conjugate Gradient

5.1 Why linear CG applies here

From the previous questions, the reduced objective is the quadratic function

$$J^e(x) = x^\top Q x + q^\top x + r, \qquad x \in \mathbb{R}^{11},$$

with

$$\nabla J^e(x) = 2Qx + q.$$

At the minimizer $x^\star$ we have $\nabla J^e(x^\star) = 0$, hence $x^\star$ solves the linear system

$$(2Q)x^\star = -q.$$

Therefore, minimizing J^e is equivalent to solving $Hx = b$ with

$$H = 2Q, \qquad b = -q.$$

Since H is symmetric and in this instance positive definite, it is natural to use the *linear conjugate gradient* (CG) method to solve $Hx = b$.

5.2 Algorithm of a linear CG

Given an initial x_0, define

$$r_0 = b - Hx_0, \qquad p_0 = r_0.$$

For $k = 0, 1, 2, \dots$, iterate:

$$\alpha_k = \frac{r_k^\top r_k}{p_k^\top H p_k}, \qquad x_{k+1} = x_k + \alpha_k p_k,$$

$$r_{k+1} = r_k - \alpha_k H p_k, \qquad \beta_k = \frac{r_{k+1}^\top r_{k+1}}{r_k^\top r_k}, \qquad p_{k+1} = r_{k+1} + \beta_k p_k.$$

We stop when $\|r_k\| = \|b - Hx_k\| \le \texttt{tol}$.

5.3 Expected number of iterations

The linear CG method converges in at most n iterations. Here $n = 11$, so one could expect convergence in at most

$$11 \text{ iterations}$$

However, the loss of conjugacy and the conditioning effects may lead to more than n iterations in practice.

5.4 Numerical results

We solved $Hx = b$ with

$$H = 2Q, \quad b = -q, \quad x_0 = \mathbf{0} \in \mathbb{R}^{11},$$

and stopping criterion

$$\|b - Hx_k\| \leq 10^{-10}.$$

The method converged in

$$k = 16 \text{ iterations},$$

and produced

$$x^\star \approx \begin{pmatrix} 0.26851035 \\ -0.60158785 \\ -0.18651334 \\ 0.95014403 \\ 0.05797526 \\ -0.01420058 \\ -0.39572781 \\ 0.18019933 \\ 0.24183317 \\ 0.41863455 \\ 0.06426108 \end{pmatrix} \qquad J^e(x^\star) \approx 3.0412849729832487$$

The values match questions 2 and 3 up to a 10^{-10} decimal.

5.5 Discussion

All three methods (steepest descent, Newton, and linear CG) target the same unconstrained reduced problem, so they converge to the same minimizer $x^\star$ (up to numerical precision).

- **Question 2 (steepest descent + Wolfe):** required a very large number of iterations because steepest descent uses only first-order information and exhibits slow convergence on ill-conditioned quadratic objectives. The code could have been optimized in order to get less iterations but the number needed would still be more than a thousand.
- **Question 3 (Newton + Wolfe):** converged in one iteration since J^e is quadratic and the exact Hessian is constant, so Newton jumps directly to the solution of $2Qx + q = 0$.
- **Question 4 (linear CG):** converged in a small number of iterations. In theory one expects at most 11 iterations in exact arithmetic (dimension of the system), but in practice we observed 16 iterations due to finite-precision effects and conditioning of the system.

6 Question 5: Solution with SciPy

To validate our implementations, we also solved the reduced unconstrained problem

$$\min_{x \in \mathbb{R}^{11}} J^e(x)$$

using a standard optimization library in Python. We used `scipy.optimize.minimize` with the **BFGS** method which is a quasi-Newton. Since the objective is smooth and quadratic, BFGS is expected to converge rapidly. We supplied the analytic gradient $\nabla J^e(x) = 2Qx + q$ to improve accuracy and speed.

6.1 Library results

Starting from the same initial point as in other questions,

$$x_0 = \mathbf{0} \in \mathbb{R}^{11},$$

SciPy reported successful convergence after

$$\texttt{k} = 16 \text{ iterations}$$

with 21 function evaluations and 21 gradient evaluations. The obtained objective value and gradient norm at the returned point are:

$$J^e(x^\star_{\text{SciPy}}) = 3.0412849729841582, \qquad \|\nabla J^e(x^\star_{\text{SciPy}})\| \approx 5.90 \times 10^{-11}.$$

6.2 Comparison with our Newton+Wolfe method from Question 3

We compare the library solution to our Newton method with strong Wolfe line search, which converged in one iteration:

$$\text{iterations} = 1, \qquad J^e(x^\star_{\text{Newton}}) = 3.041284972984613, \qquad \|\nabla J^e(x^\star_{\text{Newton}})\| \approx 2.23 \times 10^{-12}.$$

The two minimizers are numerically indistinguishable up to a 10^{-13} decimal:

$$\|x^\star_{\text{SciPy}} - x^\star_{\text{Newton}}\| \approx 6.39 \times 10^{-13}, \qquad \left|J^e(x^\star_{\text{SciPy}}) - J^e(x^\star_{\text{Newton}})\right| \approx 4.55 \times 10^{-13}.$$

6.3 Discussion

The SciPy solution matches our Newton solution up to a 10^{-13} decimal, which confirms that our previous algorithms converge to the correct minimizer of the reduced quadratic problem. Comparing the number of iterations, BFGS required 16, while Newton reached the optimum in 1, as expected for a quadratic objective with an exact Hessian who is constant.

7 Question 6: The constrained problem

We now minimize the reduced objective $J^e(x)$ under the inequality constraints

$$x_1 \geq 0, \ldots, x_{11} \geq 0, \qquad 1 - \sum_{i=1}^{11} x_i \geq 0,$$

which is equivalent to working with the full weight vector

$$w = (w_1, \ldots, w_{12}) \in \mathbb{R}^{12}, \qquad w_i \geq 0, \ \sum_{i=1}^{12} w_i = 1,$$

by setting $w_i = x_i$ for $i = 1, \dots, 11$ and $w_{12} = 1 - \sum_{i=1}^{11} x_i$. In the full variables, the objective is

$$J(w) = w^\top \Sigma w - \phi\, \mu^\top w \quad (\phi = 5),$$

and the feasible set is the probability simplex.

7.1 Projected Gradient Descent on the simplex

We used a gradient projection method, and in practice we implemented it as a projected gradient descent with a projection onto the simplex. Starting from an initial feasible point w_0, the method performs a gradient step followed by a projection onto the simplex:

$$y_k = w_k - \alpha_k \nabla J(w_k), \qquad w_{k+1} = \Pi_\Delta(y_k),$$

where Π_Δ denotes the Euclidean projection onto $\Delta = \{w \in \mathbb{R}^{12} :\ w \geq 0,\ \sum_i w_i = 1\}$. The step length α_k is chosen by an Armijo backtracking procedure on the projected step.

As a stationarity measure for constrained optimization, we monitor the projected-gradient mapping

$$\|w_k - \Pi_\Delta(w_k - \nabla J(w_k))\|,$$

and stop when it is below a prescribed tolerance.

7.2 Numerical result

We initialized the algorithm with the uniform portfolio

$$w_0 = \left(\tfrac{1}{12}, \dots, \tfrac{1}{12}\right),$$

and obtained convergence after

$$k = 275 \text{ iterations.}$$

The computed optimal portfolio weights are

$$w^\star \approx \begin{pmatrix} 0.09546618 \\ 0 \\ 0 \\ 0.27193496 \\ 0.03241073 \\ 0 \\ 0 \\ 0.03515613 \\ 0.20256739 \\ 0.34348564 \\ 0.01897897 \\ 0 \end{pmatrix}, \qquad \sum_{i=1}^{12} w_i^\star \approx 1.$$

The corresponding objective value is

$$J(w^\star) \approx 7.918837100790903.$$

In reduced variables $x = (x_1, \ldots, x_{11}) = (w_1^\star, \ldots, w_{11}^\star)$, we have

$$x^\star \approx \begin{pmatrix} 0.09546618 \\ 0 \\ 0 \\ 0.27193496 \\ 0.03241073 \\ 0 \\ 0 \\ 0.03515613 \\ 0.20256739 \\ 0.34348564 \\ 0.01897897 \end{pmatrix}, \qquad 1 - \sum_{i=1}^{11} x_i^\star \approx 1.33 \times 10^{-15},$$

showing that the inequality constraint $1 - \sum_{i=1}^{11} x_i \geq 0$ is active, numerically $w_{12}^\star \approx 0$.

8 Conclusion

In this project, we studied a mean–variance portfolio optimization problem and solved it through several optimization paradigms. After rewriting the objective as a reduced quadratic function $J^e(x)$, we implemented and compared different methods. Steepest descent with Wolfe line search converged reliably but required a large number of iterations, illustrating its slow behavior on badly conditioned quadratic objectives. In contrast, Newton's method, with Wolfe step lengths, exploited curvature information and reached the minimizer in a single iteration, as expected for a quadratic objective with a constant Hessian. The linear conjugate gradient method provided an intermediate approach: by solving the optimality system $2Qx + q = 0$ iteratively, it converged in a small number of iterations and produced the same minimizer up to really small decision.

We also validated our implementations using a standard optimization library from SciPy. The library solution matched our Newton solution up to machine precision, which confirms the correctness of our derivations and numerical codes.

Finally, we addressed the practically relevant constrained problem, which amounts to optimizing over the simplex. Using a projection-based method, we obtained a sparse optimal portfolio, several weights equal to zero, and a higher objective value than in the unconstrained case, which is consistent with the loss of flexibility induced by constraints.

Overall, the experiments highlight the trade-off between robustness, computational cost, and convergence speed: projection-based first-order methods are simple and broadly applicable under constraints, while Newton and conjugate gradient methods are extremely efficient for unconstrained quadratic problems when curvature information , such as exact Hessian, can be exploited.