Department of Mathematics at University of California, San Diego

Computational Finance

Optimization Techniques [**L**ecture 2]

Michael Holst

January 17, 2018

Contents

List of Algorithms

1 Optimization Techniques

One of the first applications of the derivative in first-year undergraduate mathematics is to find maxima and minima of real-value functions of a single real variable (or *univariate* functions), with no constraints placed on that variable. This was likely the first exposure at the undergraduate level to what is known as *unconstrained optimization*. In Section 1.1, we examine a particularly important optimization problem arising in finance, and we will use this problem as our motivation for understanding optimization problems and developing solution techniques. In Section 1.2, we begin by reviewing the basic approach we took in Calculus involving the search for *critical points* of unconstrained optimization problems in one variable, and show how it extends naturally to several variables. We then discuss how one incorporates both equality and inequality constraints placed on the independent variable, leading to what is referred to as *constrained optimization*. Given some function $f\colon \mathbb{R}^n \to \mathbb{R}$, and a set $S \subset \mathbb{R}^n$, we will be interested in the problem:

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f(x) \\ &\text{s.t. } x \in S, \end{aligned}$$

where here and throughout we abbreviate "such that" as "s.t.". The function f is called the *objective function*, and S is called the *feasible region*. The feasible region S is typically represented by equations and/or inequalities known as *constraints*. We will discuss in some detail the general form of this problem, its associated optimality conditions, and techniques for its solution, along with some special cases of this problem known as *Linear Programming (LP)*, *Quadratic Programming (QP)*, and some related problems. We will also look at a couple of more advanced topics, such as non-smooth optimization and optimization under uncertainty. Finally, in Section 1.3, we will focus entirely on QP Problems along with methods for their solution, and finally come back to some specific QP problems arising in finance.

Two good references for all of the optimization material appearing in these notes are [2, 1].

1.1 Examples from Finance Involving Optimization

Finance is rich with optimization problems: portfolio construction, models of markets, mean-variance optimization for portfolios, construction of index funds, and value-at-risk models, to name only a few. Let's look at a specific example to motivate our development of some tools for solving optimization problems in finance.

1.1.1 Portfolio Optimization

The theory of optimal selection of portfolios is due to Harry Markowitz in the 1950's. His work formalized the "diversification principle" in portfolio selection, and earned him the 1990 Nobel Prize in Economics. The basic idea here is to consider an investor with a certain amount of money to invest in n different securities. For each security S_i, $i = 1, \ldots, n$, we assume we have estimates of expected return μ_i and variance σ_i^2. Also, for any two securities i and j, we assume we have their correlation ρ_{ij}. If x_i represents the total portion of funds allocated for security S_i, one can compute the expected return and variance:

$$\begin{aligned} E[x] &= \sum_{i=1}^{n} x_i \mu_i = \mu^T x, \\ V[x] &= \sum_{i,j=1}^{n} \rho_{ij} \sigma_i \sigma_j x_i x_j = x^T Q x, \end{aligned}$$

where we have defined:

$$Q_{ij} = \rho_{ij}\sigma_i\sigma_j, \quad \rho_{ii} = 1, \quad \mu = [\mu_1, \mu_2, \ldots, \mu_n]^T.$$

If we assume that we have a single unit of money (or have normalized or units so that we start with a single unit of money), then

$$\sum_{i=1}^{n} x_i = 1,$$

is a reasonable "feasibility" constraint (we spend all our money). There may be other feasibility constraints also, but we will just use this constraint here. Note that if we define the constant n-vector of ones as $e = [1, 1, \ldots, 1]^T$, then we can write this feasibility constraint in matrix form as:

$$e^T x = 1.$$

One now defines an *efficient portfolio* as one which gives the maximal expected return over all portfolios with the same variance, or (equivalently as it turns out) one that gives the minimal variance among all portfolios that have an expected return above some given target. The *efficient frontier* is defined as the collection of efficient portfolios. The *Markowitz portfolio optimization problem*, or *mean-variance optimization (MVO)* can be written in several mathematically equivalent ways. Here is the version we will focus on: We look for a minimum variance portfolio of securities that gives the expected return target R. Mathematically, this reads:

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} x^T Q x \\ &e^T x = 1, \\ &\mu^T x \geq R, \\ &x \geq 0, \end{aligned}$$

where the last constraint is taken to mean: $x_i \geq 0, i = 1, \ldots, n$. The first constraint is the feasibility constraint (we spend all our money). The second constraint is that the expected return is at least R. The third constraint is to exclude "short sales", i.e., we can only invest a non-negative amount of money in a particular security. The objective function is the quadratic polynomial

$$f(x) = x^T Q x = \sum_{i,j=1}^{n} Q_{ij} x_i x_j.$$

This type of optimization problem is called a *constrained quadratic minimization problem*, or more commonly a *Quadratic Program*, or *QP* for short. We build some tools specifically for QP problems in Section 1.3 below.

1.2 Overview of Optimization Problems and Techniques

In this section, we first review the approach we took Calculus for finding minimizers of real-valued functions of a single real variable. We then show how this approach extends naturally to real-valued functions of several variables, giving us a general set of techniques for solving *unconstrained optimization* problems. We then discuss how one incorporates both equality and inequality constraints placed on the independent variables, leading to what is referred to as *constrained optimization*. We will discuss briefly some special cases of this problem known as *Linear Programming (LP)* and *Quadratic Programming (QP)*. We will also look at a couple of more advanced topics, such as non-smooth optimization and optimization under uncertainty.

1.2.1 Unconstrained Univariate and Multivariate Optimization

Unconstrained optimization, where we have only an objective function to consider and no constraints on the independent variable, is much easier to conceptualize than constrained optimization, as well as to develop techniques for solving problems. This is the case for both real-valued functions of a single variable, as well as for real-valued functions of several variables. The idea we remember from calculus is that if one is given a smooth (infinitely differentiable) function $f\colon \mathbb{R} \to \mathbb{R}$, then its (local) maximum and minima are the points along the curve where the slope of the curve is "flat". Calculus gives us an actual mathematical object for the slope, namely $f'(x)$, and so we are led to the equation for the *critical points* of f:

$$f'(x) = 0. \tag{1.1}$$

In other words, if $x \in \mathbb{R}$ solves a minimization or a maximization problem:

$$\min_{x\in\mathbb{R}} f(x), \qquad \text{or} \qquad \max_{x\in\mathbb{R}} f(x), \tag{1.2}$$

then x must satisfy (1.1). It is worth noting once and for all that if we are given a maximization problem involving a function g, we can always turn it into an equivalent minimization problem by defining $f(x) = -g(x)$. Therefore, here (and throughout the study of optimization), we tend to focus only on the case of minimization problems. When we refer to optimization problems below, either unconstrained or constrained, we will be focusing primarily on minimization problems.

Here is an example: Let $f(x) = x^2 - 4x$. The equation for the critical point is $f'(x) = 2x - 4 = 0$, and by solving for x, we find the (unique) critical point, $x = 2$. If we draw the curve of $f(x)$, then we see in fact that $f(x)$ does appear to have a unique global minimum value of -4 at the point $x = 2$. Can we know ahead of time whether or not the critical point is gives a maximum or a minimum of $f(x)$? The answer is yes, if we are willing to compute one more derivative. If $f''(x) > 0$, then we know that the *curvature* of f at x is positive, and in that case f is called concave up, or *convex*; this tells us that x is a (local) minimizer of f. If $f''(x) < 0$, then f is concave down (or just *concave*), and so we know x is a (local) maximizer of f. If $f''(x) = 0$, then we potentially have multiple local minimizers or maximizers near x, or possible a *saddle point*, in which case $f(x)$ is neither a local maximum or minimum. In our example above, we had $f(x) = x^2 - 4x$ and $f'(x) = 2x - 4$, and so $f''(x) = 2$. Since this is strictly positive, we know $f(x)$ is convex at $x = 2$, and so f has a local minimizer at $x = 2$. In fact, since $f''(x)$ is constant, $f(x)$ is convex at every x, and since x was the unique critical point, this tells us that x is in fact the unique *global* minimizer of f.

In the multivariate case where $F\colon \mathbb{R}^n \to \mathbb{R}^m$, the only case of interest for optimization is the case $m = 1$, i.e., for functions of the form $f\colon \mathbb{R}^n \to \mathbb{R}$. The reason is that the "target" for f must be an ordered set such as $\mathbb{R}$, so that we can talk about whether f is larger or smaller at various values of x. This allows us to pose an *unconstrained optimization problem* of the form:

$$\min_{x\in\mathbb{R}^n} f(x). \tag{1.3}$$

Just as in the case of a single real variable, we have the notion of *critical points*. However, we now must consider *all partial derivatives*, and look for points $x \in \mathbb{R}$ such that all partial derivatives vanish. This is one of the uses of the gradient vector we encountered earlier; it is the convenient column vector collection of all of the first partial derivatives of $f(x)$, and we have our critical point equation:

$$\nabla f(x) = 0.$$

This is referred to as the *first-order necessary condition for optimality*; the idea is that if $x \in \mathbb{R}^n$ is a point at which f is (locally) minimized, then the tangent plane to the surface traced out by

f must be flat, analogous to the case in one dimension. The expression for this flatness is that all of the partial derivatives of f, evaluated at that one point x, must be zero.

Here is an example: Let $f(x) = x_1^2 + 2x_2^2 + 3$, so that $f\colon \mathbb{R}^2 \to \mathbb{R}$. The first-order necessary condition for optimality reads:

$$\nabla f(x) = \begin{bmatrix} 2x_1 \\ 4x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The unique solution to this equation is $x_1 = x_2 = 0$, so the unique critical point is $x = [0,0]^T$. If we draw $f(x)$ as a two-dimensional surface over the $x_1 - x_2$ plane, we see that it takes the shape of a bowel that touches the plane $f(x) = 3$ at the point $x = [0,0]^T$. The picture implies that this is a local minimizer, and in fact a unique global minimizer. Again the same question arises: can we know this is the case before looking at a picture? The answer is again yes, but we must generalize what we did earlier in terms of looking at the second derivative in the case of a scalar function. It turns out that the appropriate generalization if the condition $f''(x) > 0$ is to have the *eigenvalues of the Hessian Matrix* of $f(x)$ be positive. We discussed the Hessian matrix of $f(x)$ earlier; it is is simply the square matrix of all second partial derivatives of $f(x)$ arranged as a square $n \times n$ matrix. The *second-order necessary and sufficient condition for optimality* of $f(x)$ at a point x is then the following together:

1) The first-order necessary condition holds at x;
2) The eigenvalues of the Hessian matrix $f''(x)$ at x are positive.

We recall that an equivalent condition to having positive eigenvalues for a symmetric matrix $A \in \mathbb{R}^{n\times n}$ is the condition that A is *symmetric positive definite (SPD)*:

$$x^T A x > 0, \quad \forall x \in \mathbb{R}^n,\ x \neq 0.$$

Therefore, we can view the second order condition alternatively as:

1) $\nabla f(x) = 0$ at x.
2) $f''(x)$ is positive definite at x.

Here is again our example: Let $f(x) = x_1^2 + 2x_2^2 + 3$, so that $f\colon \mathbb{R}^2 \to \mathbb{R}$. The first-order condition for optimality reads:

$$\nabla f(x) = \begin{bmatrix} 2x_1 \\ 4x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

and we saw that $x = [0,0]^T$ was the unique critical point. If we now compute $f''(x)$ we find:

$$f''(x) = \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}.$$

Since a diagonal matrix has its eigenvalues displayed on its diagonal, we can skip the characteristic equation and quadratic formula and just read off the two eigenvalues: $\lambda_1 = 2$, $\lambda_2 = 4$. Since these are both positive, we know that $x = [0,0]^T$ is a local minimizer of $f(x)$. Moreover, since $f''(x)$ is a constant matrix in this case, it is positive definite at every x. Since $x = [0,0]^T$ was the unique critical point, it must be the unique global minimizer of $f(x)$.

Computing Minimizers Directly: Method of Steepest Descent. From the discussion above, we now know how to identify a minimizer when we have a candidate in our hands (we just check the first- and second-order conditions). However, what about finding a minimizer to begin with? As we saw in the examples above, the first-order condition gives us a set of equations that we can

in principle solve, and we can then check the second-order condition once we have a candidate minimizer.

However, solving the equations for the first-order condition can be formidable. We could instead look for minimizers directly by focusing on the objective function, by trying to move "down hill" from the current position on the surface generated by f at each step of an algorithm. We could then use the necessary condition $\nabla f(x^k) = 0$ to *check* whether x^k is a critical point, and if so then check the eigenvalues of $f''(x^k)$, but we would not actually have to *solve* the necessary condition equations for x^k in this approach.

The down-hill direction from the current position on the surface that has the steepest slope, known as the *direction of steepest descent*, can be shown to be minus the gradient of the objective function, or:

$$p^k = -\nabla f(x^k). \tag{1.4}$$

The idea is to step a certain distance α in the direction of steepest descent p^k, and then call this our new approximation:

$$x^{k+1} = x^k + \alpha p^k.$$

How far do we go in the direction p^k? We can either take a unit step $\alpha = 1$ in the direction p^k, or we can let f tell us how far to step, by solving a one-dimensional minimization problem:

$$\min_{\alpha \in \mathbb{R}} f(x^k + \alpha p^k).$$

This problem can be solved by findings its critical points using any root finding algorithm for real-valued functions of a single real variable (such as the bisection method, or a one-dimensional Newton iteration).

If we were to build an iteration that successively tries to move in the direction of steepest descent, we would end up with Algorithm 1. (This algorithm could be improved in various ways, such as a more sophisticated and robust stopping criterion.)

Algorithm 1 Method of Steepest Descent

- Pick an initial guess: $x^0 \in \mathbb{R}^n$.
- Select a tolerance: $0 < TOL \ll 1$.
- Initialize the iteration: $k = 0$.
- While $(\|\nabla f(x^k)\| \geq TOL)$ do:

  1) $p^k = -\nabla f(x^k)$ (Steepest Descent Direction p^k)
  2) $x^{k+1} = x^k + \alpha_k p^k$ ($\alpha_k = 1$, or find α_k by minimizing $f(x^k + \alpha_k p^k)$)
  3) $k = k + 1$ (Update k)

- End While

Computing Minimizers Indirectly: Solving for Critical Points. If we want to go directly for the critical points rather than letting f gently guide us down to them, then we would need to solve the equations representing the first-order necessary condition:

$$F(x) = 0, \qquad F\colon \mathbb{R}^n \to \mathbb{R}^n, \tag{1.5}$$

where $F(x) = \nabla f(x)$. How do we actually solve such a (generally nonlinear) equation?

Algorithm 2 Global Inexact Newton Iteration

- Pick an initial guess: $x^0 \in \mathbb{R}^n$
- Select a tolerance: $0 < TOL \ll 1$
- Select a robustness parameter: $0 < \mu < 1$
- Initialize the iteration: $k = 0$
- While ($\|F(x^k)\| \geq TOL$) do:

  1) Solve for the Newton direction p^k:
  $F'(x^k)p^k = -F(x^k) + r^k$

  2) Find the damping parameter α_k:
  $\alpha_k = 1$
  While $\|F(x^k + \alpha_k p^k)\| > (1 - \alpha_k \mu)\|F(x^k)\|$ do:
  $\alpha_k = \alpha_k / 2$
  End While

  3) Update the solution:
  $x^{k+1} = x^k + \alpha_k p^k$

  4) Increment k:
  $k = k + 1$

- End While

As it turns out, the most effective techniques are based on the method we derived in the first lecture: Newton's method. We will repeat the algorithm here, but will add a few more bells and whistles. The version of the Newton iteration given here as Algorithm 2, which we call *Global Inexact Newton Iteration*, is modified in several ways from the vanilla Newton Algorithm from the first lecture in order to improve its performance. This version of Newton's method is quite robust, and is used as the basic nonlinear equation solver in many application areas, including optimization. (More sophisticated stopping criteria can be employed that are more robust to varying input functions than the simple condition that is shown above in the outer-most WHILE loop.)

The terminology *Global* in the name of Algorithm 2 comes from the introduction of the damping parameter into Step 2), which implements *backtracking line-search* to allow the method to improve the solution as much as possible at each iteration. (This idea is stolen from the Steepest Descent algorithm above.) It helps control the behavior of Newton's method when the initial guess is not within the basin of attraction for the method to begin to contract quadratically. The search for a good damping parameter is the inner-most WHILE loop in Algorithm 2; one can show that Newton's method always produces a *direction of decrease* for $\|F(x)\|$, meaning that it is always possible to find a sufficiently small α_k so that the condition of the WHILE loop fails, and then the loop terminates with that finite positive α_k to use as the damping parameter. The robustness parameter μ features into this search for α_k; it is there to make sure the damping allows for *sufficient decrease* in $\|F(x)\|$ so that the overall Newton iteration does not stall.

The terminology *Inexact* in the name of Algorithm 2 comes from a modification to Step 1) that allows one to solve the linear system for p^k only approximately rather than exactly; this accounts for the appearance of r^k in Step 1). The accuracy of the solution p^k, measured by the magnitude of the residual vector r^k, becomes critical only later in the iteration, where one

begins to need $r^k = 0$. By allowing for the approximate solution for p^k instead of the exact solution for early iterations makes it possible to produce a more efficient method overall.

Note that with $F(x) = \nabla f(x)$, and then also $F'(x) = f''(x)$, the only fundamental difference between Algorithm 2 and Algorithm 1, other than the extra bells and whistles that we added to Algorithm 2, is the choice of the direction vector p^k at each step:

1) Steepest Descent: $p^k = -\nabla f(x^k)$.
2) Newton Method: $p^k = -[f''(x^k)]^{-1}\nabla f(x^k)$.

In other words, Steepest Descent is exactly the Newton method, except that Steepest Descent approximates the Hessian matrix $f''(x)$ by the identity matrix. The Newton method uses more information about the function, and for that reason generally outperforms Steepest Descent.

Newton's method is in the best case quadratically convergent, and with the addition of the damping parameter in Step 2), can generally be made superlinearly convergent even in the case of a poor initial approximation. As noted, Newton's method outperforms Steepest Descent (which generally converges linearly), and it is still generally considered the "go-to" method for nearly any type of nonlinear equation involving functions of the form $F\colon \mathbb{R}^n \to \mathbb{R}^n$. It also forms the basis of the most effective algorithms that have been developed for unconstrained and constrained optimization, based on the approach of solving the equations for the first-order necessary condition.

1.2.2 Equality- and Inequality-Constrained Optimization

Our portfolio optimization example was both nonlinear (the objective function was quadratic) and came with constraints. It is one particular example from a large class of problems of the following form. Given a function $f\colon \mathbb{R}^n \to \mathbb{R}$, and a set $S \subset \mathbb{R}^n$, we are interested in the problem:

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f(x) \\ &\text{s.t. } x \in S. \end{aligned}$$

Note that a mathematically equivalent formulation is:

$$\text{Find } x^* \in S \text{ s.t. } f(x^*) \leq f(x), \quad \forall x \in S.$$

This (either formulation) is called a *constrained optimization* problem; f is called the *objective function*, and S is called the *feasible region*. If S is empty, the problem is called *infeasible*. If there is a sequence of elements $\{x^k\}_{k=1}^{\infty}$ from S such that $f(x^k) \to -\infty$ as $k \to \infty$ (so f is not bounded from below), we say the problem is *unbounded*. If the problem is neither infeasible nor unbounded, then it is possible that we might be able to find a solution to the problem; the solution x^* is then called the *global minimizer* of f over S. If x^* is such that $f(x^*) < f(x), \forall x \in S, x \neq x^*$, then we say that x^* is a *strict global minimizer*. This implies that x^* is a *unique solution* to the problem.

Unfortunately, for general optimization problems, most of the time we will have to be content with finding *local minimizers*:

$$\text{Find } x^* \in S \text{ s.t. } f(x^*) \leq f(x), \quad \forall x \in S \cap B_{x^*}(\epsilon),$$

where here $B_{x^*}(\epsilon)$ is the open "ball" of radius ϵ centered at x^*:

$$B_{x^*}(\epsilon) = \{\ x \in \mathbb{R}^n \mid \|x - x^*\| < \epsilon\ \}.$$

Usually one describes the feasible set S using equations and/or inequalities; for example,

$$S = \{\ x \in \mathbb{R}^n \mid g_i(x) = 0,\ i \in \mathcal{E}, \quad g_i(x) \geq 0,\ i \in \mathcal{I}\ \}$$

where $\mathcal{E}$ and $\mathcal{I}$ are index sets for *equality and inequality constraints*, respectively. If we define S in this way, our problem now reads:

$$\min_{x \in \mathbb{R}^n} f(x), \tag{1.6}$$

$$g_i(x) = 0, \quad i \in \mathcal{E}; \tag{1.7}$$

$$g_i(x) \geq 0, \quad i \in \mathcal{I}. \tag{1.8}$$

This is the most general form of constrained nonlinear optimization we will typically come across in finance and in other applications; we will refer to it as the *Nonlinear Programming (NLP) Problem*.

Method of Lagrange Multipliers for General Constraints. In the case of unconstrained optimization, we used the first-order necessary condition to turn the optimization problem into the solution of a nonlinear equation for a critical point, followed by use of the second-order condition to check that the critical point was a local minimizer. The computationally challenging part of the process is the use of Newton's method for solving the nonlinear equation for the critical point, yet armed with Newton's method the problem is within reach.

Can we incorporate the equality and inequality constraints into a similar algorithm, so that we can find a (local) minimizer of f that is in the feasible region defined by the constraints? Is there an analogue of the first- and second-order conditions for optimality in this general case of an NLP with constraints? The answer to both questions is yes; the key is to start with a new function that we call a *Lagrangian* that is built from a combination of f and the constraints g_i:

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i g_i(x). \tag{1.9}$$

The λ_i are additional variables that we are introducing into the problem, called *Lagrange Multipliers*. One can view the Lagrangian $\mathcal{L}(x, \lambda)$ as *penalizing* $f(x)$ for violating any constraint. The key mathematical result we need at this point is something that connects the (constrained) stationary point of f with the (unconstrained) stationary points of $\mathcal{L}$. To state the theorems we will rely on, we first need a definition.

Definition 1.1. *Let $x \in \mathbb{R}^n$ satisfy: $g_i(x) = 0$, $\forall i \in \mathcal{E}$, and $g_i(x) \geq 0$, $\forall i \in \mathcal{I}$. Let $\mathcal{J} \subset \mathcal{I}$ be the set of indices for which $g_i(x) = 0$. The set $\mathcal{E} \cup \mathcal{J}$ is called the* active constraints. *If the constraint gradient vectors $\nabla g_i(x)$ for each $i \in \mathcal{E} \cup \mathcal{J}$ are linearly independent, then $x \in \mathbb{R}^n$ is called a* regular point *of the constraints.*

Assuming that $x \in \mathbb{R}^n$ is a *regular point* of the constraints ensures that the constraints are not *degenerate* in some way when evaluated at x; we note that this condition will be automatically satisfied, for example, in the case of linear constraints. We can now state the analogue of the first- and second-order conditions for optimality which incorporate constraints; these are known as the *KKT (Karash-Kuhn-Tucker) Theorems*.

Theorem 1.2 (First-Order Necessary Condition for NLP Optimality)**.** *Let x^* be a local minimizer of (1.6), feasible (eqns (1.7)–(1.8) hold), and regular. Then, there exists λ_i, $i \in \mathcal{E} \cup \mathcal{I}$ such that:*

1) $\nabla f(x^*) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i \nabla g_i(x^*) = 0$,

2) $\lambda_i g_i(x^*) = 0$, $i \in \mathcal{I}$,

3) $\lambda_i \geq 0$, $i \in \mathcal{I}$.

Theorem 1.3 (Second-Order Sufficient Condition for NLP Optimality)**.** *Assume f, g_i, $i \in \mathcal{E} \cup \mathcal{I}$ are twice continuously differentiable. Let x^* be feasible (eqns (1.7)–(1.8) hold) and regular. Let $A(x^*)$ denote the Jacobian matrix of the active constraints at x^*, and let $N(x^*)$ have as its columns a basis for the null-space of $A(x^*)$. If there exists λ_i, $i \in \mathcal{E} \cup \mathcal{I}$ such that both:*

1) The conclusions of Theorem 1.2 hold;

2) $N^T(x^*)[f''(x^*) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i g_i''(x^*)]N(x^*)$ is positive definite;

then x^* is a local minimizer of (1.6)–(1.8).

The proofs of these two key theorems are beyond the scope of this course. (However, the proofs require only basic ideas from Calculus and Linear Algebra that we have reviewed earlier in the notes.) The usefulness for us here is that, similar to the unconstrained case, the first-order necessary condition (Theorem 1.2) can be used as the basis for building an algorithm around solving equations for critical points. In particular, what the theorem says is that we need to find a pair $x \in \mathbb{R}^n$, $\lambda \in \mathbb{R}^{m+p}$, where m is the number of equality constraints and p is the number of inequality constraints, such that:

$$\nabla f(x^*) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i \nabla g_i(x^*) = 0, \tag{1.10}$$

$$g_i(x^*) = 0, \quad i \in \mathcal{E}, \tag{1.11}$$

$$\lambda_i g_i(x^*) = 0, \quad \lambda_i \geq 0, \quad i \in \mathcal{I}. \tag{1.12}$$

Equation (1.10) contains n equations, which arise from the Lagrangian (1.9) by differentiation with respect to x. Equation (1.11) contains m equations corresponding to the equality constraints (the indices i in $\mathcal{E}$), which arise from the Lagrangian (1.9) by differentiation with respect to λ_i for $i \in \mathcal{E}$. Equation (1.12) contains p equations corresponding to the inequality constraints (the indices i in $\mathcal{I}$); these arise from the Lagrangian (1.9) by differentiation with respect to λ_i for $i \in \mathcal{I}$, and then using the corresponding λ_i as what are known as *slack variables*. This is then a system of $n + m + p$ equations in the $n + m + p$ unknowns, and one could potentially try to solve this system using Newton's Method. However, one difficulty that does not arise in the unconstrained case is the need for maintaining $\lambda_i \geq 0$ for $i \in \mathcal{I}$, which shows up in Equation (1.12). The most effective techniques developed so far are to modify Newton's method so that the iterates remain strictly in the interior of the feasible region defined by the combination of equality and inequality constraints; these methods are called *Interior Point (IP) Methods*. We note that our primary interest here is in developing algorithms for solving QP problems arising in finance; therefore, we will postpone our discussion of IP Methods until Section 1.3 below, and we restrict our discussion in that section to IP methods specifically designed for QP problems.

Method of Lagrange Multipliers for Equality Constraints. Before leaving this section on the general NLP problem, it is worth discussing the important case where there are only equality constraints. In this case, even though the problem can still be quite general (both the objective function and the remaining equality constraints can be arbitrarily nonlinear), we will be able to avoid developing an IP method, and instead will be able to simply apply the Newton method without modification (i.e., we can use Algorithm 2 as it stands).

We now explain how the Lagrange-multiplier framework makes this possible in the case of pure equality constraints. In this version of the NLP problem, the number of inequality constraints $p = 0$, so that corresponding index set is empty: $\mathcal{I} = \{\}$. We will therefore label the remaining equality constraints with indices from the nonempty index set $\mathcal{E}$ as $g_i(x) = 0$, $i = 1, \ldots, m$. The first-order necessary condition from Theorem 1.2 can now be written more simply as Theorem 1.4 below, where we include in the statement of the theorem an equivalent characterization involving the null-space of the constraint Jacobian.

Theorem 1.4 (First-Order Necessary Condition for NLP with Pure Equality Constraints)**.** *Let x^* be a local minimizer of (1.6), feasible (eqn (1.7) holds), and regular (with respect to the equality constraints (1.7)). Then, there exists $\lambda \in \mathbb{R}^m$ such that:*

1) $\nabla f(x^*) - g'(x^*)^T\lambda = 0,$

or, equivalently

1) $N(x^*)^T \nabla f(x^*) = 0,$

where $N(x^*)$ denotes a matrix with columns formed from a basis for the null-space of $g'(x^*)$.

We also can state a simpler version of the second-order condition from Theorem 1.3:

Theorem 1.5 (Second-Order Sufficient Condition for NLP with Pure Equality Constraints)**.** *Assume f, g, are twice continuously differentiable. Let x^* be feasible (eqn (1.7) holds), and regular (with respect to the equality constraints (1.7)). If there exists $\lambda \in \mathbb{R}^m$ such that both:*

1) The conclusions of Theorem 1.4 hold;

2) $N^T(x^*)[f''(x^*) - \sum_{i=1}^m \lambda_i g_i''(x^*)]N(x^*)$ *is positive definite;*

where $N(x^*)$ denotes a matrix with columns formed from a basis for the null-space of $g'(x^*)$, then x^* is a local minimizer of (1.6)–(1.7).

The quantity $N(x)^T\nabla f(x) = 0$ is called the *reduced gradient* of f at x. Similarly, the quantity $N^T(x^*)[f''(x^*) - \sum_{i=1}^m \lambda_i g_i''(x^*)]N(x^*)$ is called the *reduced Hessian* of f at x.

In summary, the first-order necessary condition from Theorem 1.4 now becomes simply the coupled system of equations:

$$F(x,\lambda) = \begin{bmatrix} F_1(x,\lambda) \\ F_2(x,\lambda) \end{bmatrix} = \begin{bmatrix} \nabla f(x) - g'(x)^T\lambda \\ g(x) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{1.13}$$

The unknowns are the variables $x \in \mathbb{R}^n$ and the Lagrange multipliers $\lambda \in \mathbb{R}^m$, giving a total of $n+m$ unknowns. The first block in the equations (1.13) represent n equations, and the second block are the m constraint equations, giving a total of $n+m$ equations. Since we will need it below, let us also compute the $(n+m)\times(n+m)$ Jacobian matrix associated with $F(x,\lambda)$. Denoting as D_x and D_λ the partial derivatives with respect to all of the variables in the vectors x and λ, respectively, the Jacobian matrix $F'(x,\lambda)$ has the form:

$$F'(x,\lambda) = \begin{bmatrix} D_xF_1(x,\lambda) & D_\lambda F_1(x,\lambda) \\ D_xF_2(x,\lambda) & D_\lambda F_2(x,\lambda) \end{bmatrix} = \begin{bmatrix} f''(x) - \sum_{i=1}^m \lambda_i g_i''(x) & -g'(x)^T \\ g'(x) & 0 \end{bmatrix}. \tag{1.14}$$

This can be viewed as a 2×2 block matrix, where the upper-left block is an $n\times n$ matrix; the lower-right block is an $m\times m$ matrix of zeros; the lower-left block is the $m\times n$ Jacobian matrix of the constraints; and finally the upper-right block is the $n\times m$ transpose of the constraint Jacobian. Since there are no other constraints on the variables (x,λ) in the system (1.13) we are free to apply the Global Inexact Newton Iteration in Algorithm 2 directly to solve this system, using (1.14) as the associated Jacobian matrix. We have written this out completely as Algorithm 3. (Again, the stopping criterion to terminate the outer WHILE loop can be made to be more sophisticated and robust.)

Finally, note that (1.13) can be viewed as arising as the critical point equation, sometimes called the *condition of stationarity*, of the Lagrangian

$$\mathcal{L}(x,\lambda) = f(x) - \sum_{i=1}^m \lambda_i g_i(x). \tag{1.15}$$

Algorithm 3 Method of Lagrange Multipliers for Equality-Constrained NLP

- Pick an initial guess: $(x^0, \lambda^0) \in \mathbb{R}^n \times \mathbb{R}^m$.
- Select a tolerance: $0 < TOL \ll 1$
- Select a robustness parameter: $0 < \mu < 1$
- Initialize the iteration: $k = 0$
- While $(\|F(x^k, \lambda^k)\| \geq TOL)$ do:

  1) Solve for the Newton direction $[\delta x^k, \delta \lambda^k]^T$:

  $$F'(x^k, \lambda^k) \begin{bmatrix} \delta x^k \\ \delta \lambda^k \end{bmatrix} = -F(x^k, \lambda^k)$$

  2) Find the damping parameter α_k:

  $\alpha_k = 1$
  While $\|F(x^k + \alpha_k \delta x^k, \lambda^k + \alpha_k \delta \lambda^k)\| > (1 - \alpha_k \mu)\|F(x^k, \lambda^k)\|$ do:
  $\alpha_k = \alpha_k / 2$
  End While

  3) Update the solution:

  $$\begin{bmatrix} x^{k+1} \\ \lambda^{k+1} \end{bmatrix} = \begin{bmatrix} x^k \\ \lambda^k \end{bmatrix} + \alpha_k \begin{bmatrix} \delta x^k \\ \delta \lambda^k \end{bmatrix}$$

  4) Increment k:

  $k = k + 1$

- End While

The first block equation in (1.13) arises from differentiating (1.15) with respect to x, and the second block equation in (1.13) arises from differentiating (1.15) with respect to λ. Therefore, in the case case of pure equality constraints (with no inequality constraints), the method of Lagrange multipliers may be viewed as a technique for turning a constrained minimization problem into an unconstrained one, where we simply look for critical points of the Lagrangian (1.15) without regard to constraints; the constraints are now baked into the formulation. We can simply apply Algorithm 2 to the system (1.13) for its complete solution, which for completeness we have presented here as Algorithm 3.

Before moving on to the next topic, it is useful to look at a specific example of a pure equality-constrained NLP:

$$\begin{aligned} &\min_{x \in \mathbb{R}^2} f(x) \\ &\text{s.t. } g(x) = 0, \end{aligned}$$

where

$$f(x) = x_1^2 + x_2^2 - 3x_1x_2 + x_1 + x_2, \qquad g(x) = x_1 + 2x_2 - 2. \tag{1.16}$$

For this simple example, we can compute the various objects of interest:

$$\nabla f(x) = \begin{bmatrix} 2x_1 - 3x_2 + 1 \\ 2x_2 - 3x_1 + 1 \end{bmatrix}, \quad \nabla f''(x) = \begin{bmatrix} 2 & -3 \\ -3 & 2 \end{bmatrix}, \quad g'(x) = \begin{bmatrix} 1 & 2 \end{bmatrix}, \quad g_i''(x) = 0.$$

If one were to use Algorithm 3, one would find that the point $x^* = [7/22, 15/22]^T$ and the Lagrange multiplier $\lambda = 5/22$, satisfy the first-order condition of Theorem 1.4:

$$g'(x)^T\lambda^* = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \lambda^* = \begin{bmatrix} \frac{5}{22} \\ \frac{5}{11} \end{bmatrix} = \nabla f(x^*).$$

We also note that the equivalent characterization of the first-order condition involving the null-space of the constraints is also satisfied; to see this, we need to identify a basis for the null-space. Since taking $Z = [2, -1]^T$ gives $g'(x)Z = 0$, for any $x \in \mathbb{R}^2$, we can use this matrix as Z. The equivalent first-order condition involving the *reduced gradient* is then:

$$Z^T\nabla f(x^*) = \begin{bmatrix} 2 & -1 \end{bmatrix} \begin{bmatrix} \frac{5}{22} \\ \frac{5}{11} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The eigenvalues of the Hessian are easily seen to be 5 and -1 (from our discussion earlier in the notes), and so the Hessian is indefinite. However, the *reduced Hessian*, as needed for the second-order condition of Theorem 1.5 is in fact positive definite:

$$Z(x^*)^T f''(x^*)Z(x^*) = \begin{bmatrix} 2 & -1 \end{bmatrix} \begin{bmatrix} 2 & -3 \\ -3 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \end{bmatrix} = 22 > 0.$$

Therefore, the point $x^* = [7/22, 15/22]^T$ and the Lagrange multiplier $\lambda = 5/22$ satisfies the first-order necessary condition for optimality in Theorem 1.4 and the second-order sufficient condition for optimality in Theorem 1.5.

1.2.3 Linear Programming (LP)

A special case of the NLP Problem is *Linear Programming*, or LP for short. This is when both the objective function f and all of the constraint functions g_i are linear, in which case the problem simplifies to:

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} c^T x, \\ &Ax = b, \\ &x \geq 0, \end{aligned}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $m \leq n$. We have n independent variables and m equality constraints. Since the inequality constraints now involve all of the variables, the number of inequality constraints is $p = n$. We have the same terminology as for the NLP: Feasible vs. infeasible, and bounded vs. unbounded. The standard solution techniques for LP are:

1) The Simplex Method.
2) Interior-Point Methods.

While the Interior-Point Methods are similar to the techniques we discuss below for QP Problems, Simplex Methods are completely different; we will not discuss LP problems further in these notes. Note that if any one of $f(x)$ or $g_i(x)$ (for even one i) is nonlinear, then this is a case of the fully general NLP problem, and the options open to us are then:

1) Gradient Search-Type Methods (Steepest Descent, Newton's Method, etc).
2) Interior-Point Methods (Based on Newton's Method).

3) Sequential Quadratic Programming (SQP, again based on Newton's Method).

However, if the constraints remain linear, and the objective function $f(x)$ is a purely quadratic nonlinearity, then we are facing what is known as the Quadratic Programming (QP) Problem. In this case, we have access to somewhat better techniques (based on simplifications that occur to the three types of methods above); we will discuss QP problems in the next section.

Some problems closely related to LP are the following. The *Conic Programming (CP)* Problem has the form:

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} c^T x, \\ &Ax = b, \\ &x \in C. \end{aligned}$$

When $C = \mathbb{R}^n_+$ (the set of n-vectors with non-negative entries), then this is just the LP Problem again. However, more generally one finds applications which require C to be a subset of $\mathbb{R}^n$ known as a *cone*. One can often modify LP methods to solve CP problems efficiently.

The *Integer Programming (IP)* Problem has the form:

$$\begin{aligned} &\min_{x \in \mathbb{N}^n} c^T x, \\ &Ax = b, \\ &x \geq 0, \quad x \in \mathbb{N}^n. \end{aligned}$$

Here, the free variable x is restricted to have non-negative integers $\mathbb{N}$ as its components; we denote this set as $\mathbb{N}^n$. Methods for solving IP problems are very different from those discussed here, although Interior-Point Methods for standard LP problems can play a role.

1.2.4 Quadratic Programming (QP)

When the constraints are linear and the objective function is quadratic (plus possibly lower-order terms that are linear and/or constant), then we have a standard Quadratic Programming Problem, or QP for short:

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} \frac{1}{2} x^T Q x + c^T x, \\ &Ax = b, \\ &x \geq 0, \end{aligned}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, $Q \in \mathbb{R}^{n \times n}$, with $m \leq n$. As in the case of LP, we have n independent variables and m equality constraints. Again, since the inequality constraints now involve all of the variables, the number of inequality constraints is $p = n$. We again have the same terminology as for the NLP and LP: Feasible vs. infeasible, and bounded vs. unbounded. We will discuss QP problems and algorithms for their solution in some detail in Section 1.3 below.

1.2.5 Non-smooth Optimization and Optimization with Uncertainty

A particularly difficult type of NLP problem arising in finance is know as the *Nonsmooth Optimization Problem*. These are problems for which the objective function and/or the constraints are not differentiable. Since the best techniques that have been developed for the NLP problem rely on using derivatives of both the objective function and the constraints as the foundation of

the algorithms, in the non-smooth case one has to develop alternative techniques based on the notion of *sub-gradients*.

In the case of the NLP, LP, and QP Problems discussed so far, we have only considered *deterministic optimization*, meaning that we assume that we know all data for the problem (the objective function and the constraints) deterministically. However, in applications either the objective function and/or some (or all) of the constraints may not be known deterministically. Two particularly important cases and corresponding techniques:

- Stochastic Programming: Techniques for handling cases where the data uncertainty is random, and can be explained by a probability distribution.
- Robust Optimization: These are techniques for making sure that the solution behavior of the optimization problem is well-behaved with respect to all possible configurations of the uncertain data.

These two general techniques can be applied to all of the types of optimization problems described above.

1.3 Methods and Software for Quadratic Programming

As noted earlier, finance is rich with optimization problems: data fitting, portfolio construction, models of markets, mean-variance optimization for portfolios, construction of index funds, value-at-risk models, to many others. One of the examples we discussed in some detail earlier was the theory of optimal selection of portfolios, due to Harry Markowitz in the 1950's. We noted that the *Markowitz portfolio optimization problem*, or *mean-variance optimization (MVO)* can be written in several mathematically equivalent ways. The version of the problem that gives rise to a QP was based on finding a minimum variance portfolio of securities that gives the expected return target R. We will now map this QP problem into what we have learned in previous sections about NLP.

1.3.1 The Quadratic Programming (QP) Problem

Solving a general constrained optimization problem (1.6)–(1.8) can be quite formidable, involving the solution of a large coupled system of nonlinear equations when using the method of Lagrange multipliers. However, when the constraints are linear and the objective function is quadratic (plus possibly lower-order terms that are linear and/or constant), then we have a simplification of the NLP to what is known as a *Quadratic Programming* problem, or QP for short:

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^T Q x + c^T x, \tag{1.17}$$

$$Ax = b, \tag{1.18}$$

$$x \geq 0, \tag{1.19}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, $Q \in \mathbb{R}^{n \times n}$, with $m \leq n$. As in the case of LP, we have n independent variables and m equality constraints. Again, since the inequality constraints now involve all of the variables, the number of inequality constraints is $p = n$. We again have the same terminology as for the NLP and LP: Feasible vs. infeasible, and bounded vs. unbounded. It is worth noting that similar to the case of LP, it is sometimes useful to formulate a mathematically

equivalent *dual problem*; in the case of QP, the dual problem has the form:

$$\max_{x,y\in\mathbb{R}^n} b^T y - \frac{1}{2}x^T Q x, \tag{1.20}$$

$$A^T y - Qx + s = c, \tag{1.21}$$

$$x \geq 0, \quad s \geq 0, \tag{1.22}$$

where $s \in \mathbb{R}^n$ is a set of *slack variables*.

Note that since

$$x^T Q x = \frac{1}{2}(Q + Q^T)x,$$

we can always assume Q is symmetric without loss of generality. When Q is also positive semi-definite (has only non-negative eigenvalues), then the objective function

$$f(x) = \frac{1}{2}x^T Q x + c^T x, \tag{1.23}$$

is a *convex function*, in that it satisfies the following inequality:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y), \quad \forall x, y \in \mathbb{R}^n, \quad \forall \alpha \in [0,1].$$

The point $z = \alpha x + (1-\alpha)y$ is called a *convex combination* of the points x and y. Any point z on the line connecting x and y can be realized by a particular choice of $\alpha \in [0,1]$. If $S \subset \mathbb{R}^n$ and it holds that:

$$z = \alpha x + (1-\alpha)y \in S, \quad \forall x, y \in \mathbb{R}^n, \quad \forall \alpha \in [0,1],$$

then we say that S is a *convex set*. In other words, S is convex if and only if the line connecting any two points from S is contained entirely in S. Note that any vector space such as $\mathbb{R}^n$ is automatically a convex set, since it is closed with respect to linear combinations.

As noted above, when Q is positive semi-definite, the QP objective function (1.23) is a convex function; in this case QP, is a called a *convex optimization problem*. This is quite useful, because all convex optimization problems have two very convenient features:

1) Any local minimizer is also a global minimizer.

2) There exist Algorithms to solve these problems in *polynomial time*.

An algorithm is said to be of *polynomial time* if its running time to produce a solution is bounded from above by an expression that is a polynomial in the size of the algorithm input. Problems for which a (deterministic) polynomial time algorithm exists are said to belong to *complexity class* $\mathcal{P}$. Another term for polynomial time (algorithms and/or problems) is *tractable*. Therefore, if Q is positive semi-definite, then the corresponding QP has a global solution(s), is in class $\mathcal{P}$, and can therefore be solved with efficient algorithms.

1.3.2 Optimality Conditions for QP

Our first order of business is to identify the QP analogue of the first- and second-order conditions for optimality conditions for the NLP, which we presented earlier as Theorems 1.2 and 1.3. The analogue of the first-order necessary condition for NLP is the following in the case of QP.

Theorem 1.6 (First-Order Necessary Condition for QP Optimality)**.** *Assume Q is positive semi-definite. Let x be a local solution of* (1.17)–(1.19)*, meaning it is a local minimizer of* (1.17)

that is also feasible (equations (1.18)–(1.19) hold). Then, there exists $y, s \in \mathbb{R}^n$ such that:

$$A^T y - Qx + s = c, \tag{1.24}$$
$$s \geq 0, \tag{1.25}$$
$$x_i s_i = 0, \quad i = 1, \ldots, n. \tag{1.26}$$

Moreover, x is a global solution of (1.17)–(1.19).

The analogue of the second-order sufficient condition for the NLP reduces to the following in the case of QP.

Theorem 1.7 (Second-Order Sufficient Condition for QP Optimality)**.** *Assume Q is positive semi-definite. Then the following five conditions for (x, y, s) are necessary and sufficient for x to be a global solution of (1.17)–(1.19).*

$$Ax = b, \tag{1.27}$$
$$x \geq 0, \tag{1.28}$$
$$A^T y - Qx + s = c, \tag{1.29}$$
$$s \geq 0, \tag{1.30}$$
$$x_i s_i = 0, \quad i = 1, \ldots, n. \tag{1.31}$$

Both QP and LP produce optimality conditions of the form:

1) Primal Feasibility: (1.27) and (1.28).
2) Dual Feasibility: (1.29) and (1.30).
3) Complementary Slackness: (1.31).

The challenge then is to develop an algorithm for satisfying all of the optimality conditions at the same time.

1.3.3 Interior Point (IP) Methods for QP

Interior Point (IP) Methods for QP (as well as other instances of NLP) are based on the following general approach:

1) Identify (x^0, y^0, s^0) so that the primal and dual feasibility conditions hold with *strict inequality*:

$$Ax^0 = b, \tag{1.32}$$
$$x^0 > 0, \tag{1.33}$$
$$A^T y^0 - Qx^0 + s^0 = c, \tag{1.34}$$
$$s^0 > 0. \tag{1.35}$$

   (The strict inequality gives rise to the terminology "interior point".)

2) Now generate (x^k, y^k, s^k) so that:

   - Solution (x^k, y^k, s^k) continues to satisfy conditions (1.32)–(1.35).
   - Solution (x^k, y^k, s^k) progressively approaches complementary slackness condition:

$$x_i s_i = 0, \quad i = 1, \ldots, n. \tag{1.36}$$

Interior Point methods (IP) based on the core idea above can be implemented using a modification of Newton's Method, and can be shown to converge in polynomial time. Since they are based on the second-order sufficient condition for optimality, which involve both primal and dual feasibility conditions, interior point methods are sometimes called *primal-dual interior point methods*. To discuss IP Methods for QP in more detail, it is useful to define two distinct sets represented by the non-strict and strict inequality constraints:

$$\mathcal{F} = \{\ (x, y, s) \mid Ax = b, \quad A^T y - Qx + s = c, \quad x \geq 0, \quad s \geq 0\ \}, \tag{1.37}$$

$$\mathcal{F}^0 = \{\ (x, y, s) \in \mathcal{F} \mid x > 0, \quad s > 0\ \}. \tag{1.38}$$

The set $\mathcal{F}$ represents the *feasible set*, whereas $\mathcal{F}^0$ is the *strictly feasible set*, also called the *relative interior* of $\mathcal{F}$. As noted above, IP Methods will generate iterates that begin inside $\mathcal{F}^0$, and remain there throughout the iteration.

In order to explain a bit about the structure of these algorithms, it is useful first to formulate the complementary condition (1.36) in matrix form. To this end, we define two square diagonal matrices $X, S \in \mathbb{R}^{n\times n}$ with the variables x and s laid out on their diagonals:

$$X_{ii} = x_i, \quad S_{ii} = s_i, \quad X_{ij} = S_{ij} = 0, \quad i \neq j. \tag{1.39}$$

The complementary condition (1.36) can now be written as:

$$XSe = 0,$$

where $e = [1, \ldots, 1]^T$ is the n-vector with all entries equal to one. A solution (x, y, s) satisfying the five conditions from Theorem 1.6 can be viewed now as a solution to the following system,

$$F(x, y, s) = \begin{bmatrix} F_1(x, y, s) \\ F_2(x, y, s) \\ F_3(x, y, s) \end{bmatrix} = \begin{bmatrix} A^T y - Qx + s - c \\ Ax - b \\ XSe \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \tag{1.40}$$

with the additional requirement that:

$$x \geq 0, \qquad s \geq 0. \tag{1.41}$$

The first block equation in (1.40) has n equations, the second block has m equations, and the third block has again n equations, giving $2n + m$ equations. We also have n unknowns represented by x, m unknowns represented by y (playing the role of Lagrange multipliers), and n unknowns represented by s (the slack variables), giving then $2n + m$ unknowns. Hence, we have then defined $F\colon \mathbb{R}^{2n+m} \to \mathbb{R}^{2n+m}$. If we did not have to worry about condition (1.41), then we could simply apply Newton's method to this system (1.40) of $2n + m$ equations in $2n + m$ unknowns. However, (1.41) represents $2n$ inequality constraints that we must also satisfy.

Recall Newton's method that we formulated as Algorithm 2. Applied to the problem (1.40) and for the moment ignoring the constraints (1.41), the first step of the algorithm involves solving the Jacobian system for the Newton direction:

$$F'(x^k, y^k, s^k) \begin{bmatrix} \delta x^k \\ \delta y^k \\ \delta s^k \end{bmatrix} = -F(x^k, y^k, s^k). \tag{1.42}$$

The solution to this linear system is the Newton direction $[\delta x^k, \delta y^k, \delta s^k]^T$. Denoting as D_x, D_y, and D_s the partial derivatives with respect to all of the variables in the vectors x, y, and s, respectively, the Jacobian matrix F' has the form:

$$F'(x^k, y^k, s^k) = \begin{bmatrix} D_x F_1 & D_y F_1 & D_s F_1 \\ D_x F_2 & D_y F_2 & D_s F_2 \\ D_x F_3 & D_y F_3 & D_s F_3 \end{bmatrix} = \begin{bmatrix} -Q & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix}, \tag{1.43}$$

where X^k and S^k are again diagonal matrices formed by taking x^k and s^k along their diagonals. Note that if we are starting with $(x^k, y^k, s^k) \in \mathcal{F}^0$, then

$$F(x^k, y^k, s^k) = \begin{bmatrix} F_1(x^k, y^k, s^k) \\ F_2(x^k, y^k, s^k) \\ F_3(x^k, y^k, s^k) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ X^k S^k e \end{bmatrix}. \tag{1.44}$$

Therefore, the Newton system takes the simplified form:

$$\begin{bmatrix} -Q & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \delta x^k \\ \delta y^k \\ \delta s^k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -X^k S^k e \end{bmatrix}. \tag{1.45}$$

At each step of the Newton iteration in an IP for solving a QP, one first updates the Jacobian matrix and right-hand side in (1.45) with the current x^k and s^k, and the solve the linear system (1.45) for (x^k, y^k, s^k). What remains is to decide how to do the update step in Newton's method:

$$\begin{bmatrix} x^{k+1} \\ y^{k+1} \\ s^{k+1} \end{bmatrix} = \begin{bmatrix} x^k \\ y^k \\ s^k \end{bmatrix} + \alpha_k \begin{bmatrix} \delta x^k \\ \delta y^k \\ \delta s^k \end{bmatrix}, \tag{1.46}$$

so that the new IP iterate $(x^{k+1}, y^{k+1}, s^{k+1})$ remains in $\mathcal{F}^0$. This is done by suitable choice of $\alpha_k \in (0, 1]$, called a *permissible step-length*. That such a choice of α_k is always possible follows from the assumption that $(x^k, y^k, s^k) \in \mathcal{F}^0$, and so that any sufficiently small change we make to each component (the smallness is controlled by the size the parameter α_k) will continue to ensure that each component of x and s remain strictly positive. The goal would be to find the *largest permissible step length* α_k, so that we move toward a solution as quickly as possible (which is what unconstrained Newton is trying to do), yet remain in $\mathcal{F}^0$.

This leads us finally to a complete algorithm for using an Interior Point Method to solve a QP problem, which we give as Algorithm 4. There is a little twist to the algorithm that we have not described yet, which is the appearance of the two parameters σ_k and μ_k. These two parameters (both real numbers) are designed to increase the robustness (resistance to failure) of the IP Method. The reason they are needed is that if one uses an IP Method without their inclusion, then for many problems the search for the step length α_k can "stall" the entire algorithm by requiring exceedingly small steps to remain feasible. The solution is to stay well within $\mathcal{F}^0$ rather than near its boundary. The parameter σ_k is referred to as a *centering* parameter, which attempts to steer the iteration toward what is known as the *central path* in $\mathcal{F}^0$. The setting of $\sigma_k = 1$ is called the *pure centering direction*; the setting of $\sigma_k = 0$ just removes σ_k and μ_k from the Newton system, giving us back the pure IP Method without the robustness condition added. One can show empirically that a choice of σ_k chosen near the center of the interval $[0, 1]$ generally produces the most effective algorithm.

The IP Method is the basis for the most effective methods for QP problems, and its generalizations to NLP similarly form the foundation of the solution of more general classes of problems.

1.3.4 Software Packages for using QP

There are a number of existing software packages, both open source and commercial, for solving QP (and also LP) problems. One of the best resources for the practical solution of QP and LP problems, as well as many other types of more general optimization problems, is the *NEOS* website:

Algorithm 4 Interior Point Method for QP

- Choose $(x^0, y^0, s^0) \in \mathcal{F}^0$.
- For $k = 0, 1, 2, \ldots$ do:

  1) Choose $\sigma_k \in [0, 1]$, set $\mu_k = (x^k)^T s^k / n$.

  2) Solve the system for the Newton direction:

$$\begin{bmatrix} -Q & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \delta x^k \\ \delta y^k \\ \delta s^k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sigma_k \mu_k e - X^k S^k e \end{bmatrix} \tag{1.47}$$

  3) Find the largest permissible step length α_k for remaining in $\mathcal{F}^0$, meaning that:

$$x^k + \alpha_k \delta x^k > 0, \qquad s^k + \alpha_k \delta s^k > 0. \tag{1.48}$$

  4) Update current iterate so that it remains in $\mathcal{F}^0$:

$$\begin{bmatrix} x^{k+1} \\ y^{k+1} \\ s^{k+1} \end{bmatrix} = \begin{bmatrix} x^k \\ y^k \\ s^k \end{bmatrix} + \alpha_k \begin{bmatrix} \delta x^k \\ \delta y^k \\ \delta s^k \end{bmatrix} \tag{1.49}$$

- End For

http://www.neos-server.org/neos/

Particularly useful on the *NEOS* website is the *Optimization Guide*:

http://neos-guide.org/Optimization-Guide

which includes a thorough introduction to optimization, and the *Optimization Taxonomy*:

http://neos-guide.org/content/optimization-taxonomy

which shows the relationship between various optimization problems as an insightful *tree*.

Fortunately, MATLAB has a number of built-in functions for solving nonlinear equations and optimization problems, in many cases using the best available open source software from the numerical analysis community. The MATLAB Optimization Toolbox adds additional functionality for solving more complex optimization problems. We will explore some QP problems from finance, and their solution using MATLAB, in the homework.

1.3.5 Portfolio Optimization using QP

As we described briefly earlier, the theory of optimal selection of portfolios is due to Harry Markowitz in the 1950's. The basic idea is to consider an investor with a certain amount of money to invest in different securities. For each security S_i, $i = 1, \ldots, n$, we assume we have estimates of expected return μ_i and variance σ_i^2. Also, for any two securities i and j, we assume we have their correlation ρ_{ij}. If x_i represents the total portion of funds allocated for security i,

one can compute the expected return and variance:

$$E[x] = \sum_{i=1}^{n} x_i \mu_i = \mu^T x,$$
$$V[x] = \sum_{i,j=1}^{n} \rho_{ij} \sigma_i \sigma_j x_i x_j = x^T \Sigma_{ij} x,$$

where we have defined:

$$\Sigma_{ij} = \rho_{ij} \sigma_i \sigma_j, \quad \rho_{ii} = 1, \quad \mu = [\mu_1, \mu_2, \ldots, \mu_n]^T.$$

Clearly, Σ is symmetric positive semidefinite, since it is symmetric, and since variance is always non-negative. If we assume that there are no redundant assets (all our distinct), then it can be shown that this implies Σ is positive definite.

We considered a simplified set of portfolio constraints in our first look at portfolio optimization; here we want to allow that set to be a bit more general. We will define the set of *admissible portfolios* as those forming the following the nonempty polyhedral set:

$$\mathcal{X} = \{ \; x \in \mathbb{R}^n \mid Ax = b, \quad Cx \geq d \; \}, \tag{1.50}$$

where $A \in \mathbb{R}^{m \times n}$, $C \in \mathbb{R}^{p \times n}$, $b \in \mathbb{R}^m$, and $d \in \mathbb{R}^p$, where n is the number of variables (so $x \in \mathbb{R}^n$), m is the number of equality constraints, and p is the number of inequality constraints. In our first look at the MVO problem, we assumed that $\mathcal{X}$ had the simplified structure:

$$\hat{\mathcal{X}} = \{ \; x \in \mathbb{R}^n \mid e^T x = 1, \quad x \geq 0 \; \}, \tag{1.51}$$

which implied that $p = n$. Assuming we had a single unit of normalized money to work with, the first equation defining $\hat{\mathcal{X}}$ stated the constraint that we spent all our money. The inequality defining $\hat{\mathcal{X}}$ stated the constraint that we allowed no short sales. We will now focus on the more general feasible set $\mathcal{X}$ defined in (1.50) in this section; it of course includes the set $\hat{\mathcal{X}}$ defined in (1.51) as a special case.

We noted earlier that the *Markowitz portfolio optimization problem*, or *mean-variance optimization (MVO)* can be written in several mathematically equivalent ways. The version we mentioned earlier is the one will mostly focus on here: We look for a minimum variance portfolio of securities that gives the expected return target R. We wrote this down mathematically earlier (using slightly different notation due to the simplified constraints), and we repeat it again here using our more general constraints:

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^T \Sigma x, \tag{1.52}$$
$$\mu^T x \geq R, \tag{1.53}$$
$$Ax = b, \tag{1.54}$$
$$Cx \geq d, \tag{1.55}$$

where $A \in \mathbb{R}^{m \times n}$, $C \in \mathbb{R}^{p \times n}$, $b \in \mathbb{R}^m$, and $d \in \mathbb{R}^p$, where n is the number of variables, m is the number of equality constraints represented by A, and p is the number of inequality constraints represented by C. (Note that there is an additional inequality constraint, namely $\mu^T x \geq R$, which we keep track of separately.) The objective function is the quadratic polynomial

$$f(x) = \frac{1}{2} x^T \Sigma x = \frac{1}{2} \sum_{i,j=1}^{n} \Sigma_{ij} x_i x_j.$$

As we now know, this is a QP problem. Note that we have multiplied the objective function $f(x)$ by the constant $1/2$ for convenience, due to the way various expressions will simplify in the optimality conditions below.

Two other variations of the MVO problem that are sometimes considered as well are:

$$\begin{aligned}
&\max_{x \in \mathbb{R}^n} \mu^T x, \\
&x^T \Sigma x \leq \sigma^2, \\
&Ax = b, \\
&Cx \geq d,
\end{aligned}$$

and

$$\begin{aligned}
&\max_{x \in \mathbb{R}^n} \frac{\delta}{2} x^T \Sigma x, \\
&x^T \Sigma x \leq \sigma^2, \\
&Ax = b, \\
&Cx \geq d,
\end{aligned}$$

where in these two alternative formulations, σ^2 is a given *upper bound on the variance* of the allowed portfolio, and δ is a *risk-aversion parameter*.

Optimality Conditions for Portfolio Optimization. We can now just apply the KKT Theorems for QP to characterize the solution; these were Theorems 1.6 and 1.7. Since we have assumed that Σ is (symmetric) positive definite, we know that the QP problem we are facing has a global solution, and the first-order conditions are both necessary and sufficient for optimality. The first-order KKT conditions specifically for portfolio optimization are as follows (we simply write down the conditions appearing in Theorem 1.7 in the case of our particular QP problem).

We know that $x \in \mathbb{R}^n$ is an optimal solution of the QP problem (1.52) if and only if there exists $\lambda_R \in \mathbb{R}$, $\gamma_E \in \mathbb{R}^m$, and $\gamma_I \in \mathbb{R}^p$, such that the KKT conditions hold:

$$\Sigma x - \lambda_R \mu - A^T \gamma_E - C^T \gamma_i = 0, \tag{1.56}$$

$$Ax = b, \tag{1.57}$$

$$Cx \geq d, \tag{1.58}$$

$$\mu^T x \geq R, \tag{1.59}$$

$$\lambda_R \geq 0, \quad \lambda_R(\mu^T x - R) = 0, \tag{1.60}$$

$$\gamma_I \geq 0, \quad \gamma_I^T (Cx - d) = 0. \tag{1.61}$$

Solution of this simultaneous system of equations and inequalities is accomplished as in the case of the general QP problem, namely, one applies the IP Method given earlier as Algorithm 4.

References

[1] G. Cornuejols and R. Tutuncu. *Optimization Methods in Finance*. Cambridge University Press, New York, NY, 2011.

[2] P. Gill and M. Wright. *Numerical Optimization (Class Notes for Math 171B)*. UCSD (Available at Soft Reserves for Spring 2015), La Jolla, 1998.