Numerical Optimization in DROP

v4.59 2 December 2019

Convex Optimization - Introduction and Overview

Motivation, Background, and Setup

1. Mathematical Formulation and Framework: Convex Optimization is an important class of constrained optimization problems that subsumes linear and quadratic programming. Such problems are specified in the form

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

such that

$$x \in S$$

where the feasible set

$$S \subseteq \mathbb{R}^n$$

is a *convex* closed subset of $\mathbb{R}^n$ and f is a *convex function* (Hauser (2012)).

2. Local Minima as Global Minima: Convex optimization problems are important because all local minima are also guaranteed to be globally minimal, although this value may not be reached necessarily at a unique point.

Convex Sets and Convex Hull

1. Definition of a Convex Set: A convex set S satisfies the following property. For any two points x and y in S the point $(1-u)\cdot x + u\cdot y$ for

$$u \in [0,1]$$

lies in S as well. In other words, the line segment between any two points in S must also lie in S.
2. Definition of a Convex Hull: The smallest convex set containing another set A is called the *convex Hull* of $C(A)$.
3. Definition of a Convex Function: A function of a scalar field f is convex if it satisfies the following property:

$$f\big((1-u)\cdot x + u\cdot y\big) \leq (1-u)\cdot f(x) + u\cdot f(y)$$

for all x and y and any

$$u \in [0,1]$$

In other words, the function value between any two points x and y lies below the line of f connecting $f(x)$ and $f(y)$. An equivalent definition is that on the line segment between x and y the area over the graph of f forms a convex set.
4. Minima of a Convex Function: Convex functions are always somewhat "bowl shaped", have no local maxima (unless constant), and have no more than one local minimum value. If a local minimum exists, then it is also a global minimum. Hence gradient descent and Newton methods (with line search) are guaranteed to produce the global minimum when applied to such functions.

Properties of Convex Sets/Functions

1. Connection Property: Convex sets are connected.
2. Intersection Property: The intersection of any number of convex sets is also a convex set.
3. Axiomatic Convex Sets: The empty set, the whole space, any linear sub-spaces, and half-spaces are also convex.
4. Convex Set Dimensionality Reduction: If a convex set of S is of a lower dimension than its surrounding space $\mathbb{R}^n$ then S lies on a linear sub-space of $\mathbb{R}^n$.
5. Typical Convex Set Closure Properties: Some common convex functions include e^x, $-\log x$, x^k where k is an even number. Convex functions are closed under addition, the max operator, monotonic transformations of the x variable, and scaling by a non-negative constant.
6. Convex Set from Ellipsoidal Constraints: Constraints of the form

$$x^T A x \leq c$$

   produce a closed convex set (an ellipsoid) if A is positive semi-definite, and c is a non-negative number.
7. Transformation onto Semi-definite Programs: The set of positive semi-definite matrices is a closed convex set. Optimization problems with these constraints are known as *semi-definite programs* (SDPs). A surprising number of problems, including LPs and QPs, can be transformed into SDPs.

Convex Optimization Problems

1. General Form of Convex Optimization: An optimization problem in general

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

such that

$$g_i(x) \leq 0, i = 1, \cdots, m$$

and

$$h_j(x) = 0, j = 1, \cdots, p$$

is convex as long as f is convex, all of the g_i for

$$i = 1, \cdots, m$$

are convex, and all of the h_j for

$$j = 1, \cdots, p$$

are linear.

2. Elimination of the Equality Constraints: In convex optimization we will typically eliminate the equalities before optimizing, either by converting them into two inequalities

$$h_j(x) \leq 0$$

and

$$-h_j(x) \leq 0$$

or by performing a null-space transformation. Hence the h_j's can be dropped from typical treatments.

References

- Hauser (2012): Convex Optimization and Interior Point Methods

Newton's Method in Optimization

Method

1. Iterative Root Finding vs Optimization: In calculus Newton's method is an iterative method for finding the roots of a differentiable function f, i.e., solutions to the equation

$$f(x) = 0$$

In optimization the Newton's method is applied to the derivative f' of a twice-differentiable function f to find the roots of the derivative, i.e., solutions to

$$f'(x) = 0$$

also known as the stationary points of f.

2. The Stationary Point of f: In the $1D$ problem the Newton's method attempts to construct a sequence x_n from an initial guess x_0 that converges towards some value x^* satisfying

$$f(x^*) = 0$$

This x^* is a stationary point of f (Newton's Method in Optimization (Wiki)).

3. Taylor Expansion of f around x_n: One wishes to find Δx such that $f(x_n + \Delta x)$ is a maximum. Thus, one seeks to solve the equation that sets the derivative of the last expression with respect to Δx equal to zero:

$$0 = \frac{\partial}{\partial \Delta x}\left\{f(x_n) + f'(x_n)\Delta x + \frac{1}{2}f''(x_n)[\Delta x]^2\right\} = f'(x_n) + f''(x_n)\Delta x$$

4. Conditions for approximating the Stationary Point: For the value of

$$\Delta x = -\frac{f'(x_n)}{f''(x_n)}$$

which is a solution to the above equation, typically

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$

will be closer to the stationary point x^*. Provided that $f(x)$ is a twice-differentiable function and other technical conditions are satisfied, the sequence $x_1, \cdots, x_n$ will converge to the point x^* satisfying

$$f(x^*) = 0$$

5. Geometric Interpretation of the Newton's Method: The geometric interpretation of the Newton's method is that at each iteration one approximates $f(\vec{x})$ by a quadratic function $\vec{x}_n$ and then takes a step towards a maximum/minimum of that quadratic function. In higher dimensions, this stationary point may also be a saddle point. Of course, if $f(\vec{x})$ happens to *be* a quadratic function then the exact extremum is found in one step.

Higher Dimensions

1. Determination of the Iteration Increment: The above iteration scheme can be generalized to several dimensions by replacing the derivative with a gradient $\vec{\nabla} f(\vec{x})$

and the reciprocal of the second derivative with the inverse of the Hessian matrix $\mathbb{H}f(\vec{x})$. One thus obtains the iteration scheme

$$\vec{x}_{n+1} = \vec{x}_n + [\mathbb{H}f(\vec{x})]^{-1}\vec{\nabla}f(\vec{x}), n \geq 0$$

2. Adjustment to the Iteration Increment: Often Newton's method is modified to include a small step size

$$\gamma \in (0, 1)$$

instead of

$$\gamma = 1$$

to result in

$$\vec{x}_{n+1} = \vec{x}_n + \gamma[\mathbb{H}f(\vec{x})]^{-1}\vec{\nabla}f(\vec{x})$$

This is often done to ensure that the Wolfe conditions are satisfied at each step

$$\vec{x}_n \longrightarrow \vec{x}_{n+1}$$

of the iteration.

3. Newton's Method - Speed of Convergence: Where applicable Newton's method converges much faster towards the local maximum or minimum than gradient descent. In fact, every local minimum has a neighborhood $\mathbb{N}$ such that if one starts with

$$x_0 \in \mathbb{N}$$

Newton's method with step size

$$\gamma = 1$$

converges quadratically. The only requirements are that the Hessian be invertible, and it be a Lipschitz-continuous function of $\vec{x}$ in that neighborhood.

4. By-passing Explicit Hessian Computation: Finding the inverse of the Hessian in high dimensions can be an expensive exercise. In such cases, instead of directly inverting the Hessian it may be better to calculate the vector

$$\Delta\vec{x}_n = \vec{x}_{n+1} - \vec{x}_n$$

as a solution to the system of linear equations

$$[\mathbb{H}f(\vec{x}_n)]\vec{\nabla}f(\vec{x}_{n+1}) = -\vec{\nabla}f(\vec{x}_n)$$

This may be solved by various factorizations or approximately – and to great accuracy – using iterative methods.

5. Caveats of the Matrix Methods: Many of these methods are only applicable to certain types of equations – e.g., the Cholesky factorization and the conjugate gradient method will work only if $\mathbb{H}f(\vec{x}_n)$ is a positive definite matrix. While this may seem like a limitation, it is often a useful indicator of something gone wrong. For instance, if a maximization problem is being considered and $\mathbb{H}f(\vec{x}_n)$ is not positive definite, the iterations end up converging to a saddle point and not to a minimum.
6. Stable Variants of these Methods: On the other hand if constrained optimization is being considered – for example using Lagrange multipliers – the problem may become one of saddle point finding, in which case the Hessian will be symmetric indefinite and a solution for $\vec{x}_{n+1}$ needs to be done with approaches that will work for such situations, such as the LDL^T variant of the Cholesky factorization or the conjugate gradient method.

7. Techniques of Quasi-Newton Methods: There are exist various quasi-Newton methods where the approximation for the Hessian – or its direct inverse – is built up from the changes in the gradient.
8. Challenges with non-invertible Hessians: If a Hessian is close to a non-invertible matrix the inverted Hessian can be numerically unstable and the solution may diverge. In this case certain workarounds have been tried in the past, each of which have had varied success in differing setups.
9. Converting Hessian to Positive Definite: For example, one can modify the Hessian by adding a correction matrix B_n so as to make $\mathbb{H}f(\vec{x}_n) + B_n$ positive definite. On approach is to diagonalize $\mathbb{H}f(\vec{x}_n)$ and choose B_n so that $\mathbb{H}f(\vec{x}_n) + B_n$ has the same eigenvectors as $\mathbb{H}f(\vec{x}_n)$ but with each negative eigenvalue replaced by

$$\epsilon > 0$$

10. Approach used by Levenberg-Marquardt: An approach exploited by the Levenberg-Marquardt algorithm – which uses an approximate Hessian – is to add a scaled identity matrix $\mu\mathbb{I}$ to the Hessian, with the scale adjusted at every iteration as needed.
11. Comparison with Gradient Descent Approach: For large μ and small Hessian, the iterations behave like gradient descent with a step size $\frac{1}{\mu}$. This results in slower but more reliable convergence when the Hessian doesn't provide useful information.

Wolf Conditions

1. Motivation, Rationale, and Primary Purpose: In the unconstrained minimization problem setting, the **Wolf Conditions** are a set of inequalities for performing *inexact* line search, especially in quasi-Newton methods (Wolf (1969, 1971)).
2. Problem Setup - Function Extremization: In these methods the idea is to find

$$\min_{\vec{x}} f(\vec{x})$$

for some smooth

$$f: \mathbb{R}^n \longrightarrow \mathbb{R}$$

Each step often involves approximately solving the sub-problem

$$\min_{\alpha} f(\vec{x}_k + \alpha \vec{p}_k)$$

where $\vec{x}_k$ is the current best guess

$$\vec{p}_k \in \mathbb{R}^n$$

is a search direction, and

$$\alpha \in \mathbb{R}$$

is the step length.

3. Application of the Wolfe Conditions: The inexact line search provides an efficient way of computing an acceptable step length α that reduces the objective function sufficiently rather than minimizing the objective function over

$$\alpha \in \mathbb{R}^+$$

exactly. A line search algorithm can use the Wolfe conditions as a requirement for any guessed α before finding a new search direction $\vec{p}_k$.

Armijo Rule and Curvature Condition

1. The Inequalities of Wolfe Condition: A step length α_k is said to satisfy the *Wolfe conditions*, restricted to the direction $\vec{p}_k$, if the following inequalities hold:

$$f(\vec{x}_k + \alpha_k \vec{p}_k) \leq f(\vec{x}_k) + c_1 \alpha_k \vec{p}_k \cdot \vec{\nabla} f(\vec{x}_k)$$

$$\vec{p}_k \cdot \vec{\nabla} f(\vec{x}_k + \alpha_k \vec{p}_k) \geq c_2 \vec{p}_k \cdot \vec{\nabla} f(\vec{x}_k)$$

with

$$0 < c_1 < c_2 < 1$$

In examining the second condition once recalls that to ensure $\vec{p}_k$ is a descent direction one has

$$\vec{p}_k \cdot \vec{\nabla} f(\vec{x}_k) < 0$$

2. Choices for c_1 and c_2: c_1 is usually chosen quite small while c_2 is much larger. Nocedal and Wright (2006) provide example values of

$$c_1 = 10^{-4}$$

and

$$c_2 = 0.9$$

for Newton or quasi-Newton methods, and

$$c_2 = 0.1$$

for the non-linear conjugate gradient method. The first inequality is known as the **Armijo Rule** (Armijo (1966)) and ensures that the step length α_k decreases f sufficiently. The second inequality is called the **Curvature Condition**; it ensures that the slope has reduced sufficiently.

3. Strong Wolfe Conditions on Curvature: Denoting by ϕ a univariate function restricted to the direction $\vec{p}_k$ as

$$\phi(\alpha) = f(\vec{x}_k + \alpha_k \vec{p}_k)$$

there are situations where the Wolfe conditions can result in a value for the step length that is not close to a minimizer of ϕ. If one modifies the curvature condition to

$$\left|\vec{p}_k \cdot \vec{\nabla} f(\vec{x}_k + \alpha_k \vec{p}_k)\right| \leq c_2 \left|\vec{p}_k \cdot \vec{\nabla} f(\vec{x}_k)\right|$$

then the curvature condition along with the Armijo rule form the so-called **strong Wolfe conditions**, and force α_k to lie close to a critical point of ϕ.

Rationale for the Wolfe Conditions

1. Convergence of the Gradient to Zero: The principal reason for imposing the Wolfe conditions in an optimization algorithm where

$$\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{p}_k$$

is to ensure the convergence of the gradient to zero. In particular if the cosine of the angle between $\vec{p}_k$ and the gradient

$$\cos\theta = \frac{\vec{\nabla} f(\vec{x}_k) \cdot \vec{p}_k}{\left\|\vec{\nabla} f(\vec{x}_k)\right\| \|\vec{p}_k\|}$$

is bounded away from zero, and Armijo rule and curvature conditions (modified) hold,

$$\vec{\nabla} f(\vec{x}_k) \longrightarrow 0$$

2. Positive Definiteness of Update Matrix: An additional motivation, in the case of a quasi-Newton method, is that if

$$\vec{p}_k = {B_k}^{-1} \vec{\nabla} f(\vec{x}_k)$$

where the matrix B_k is updated by the BFGS or the DFP formula, then if B_k is positive, the (modified) curvature condition implies that B_{k+1} is also positive definite.

References

- Armijo, L. (1966): Minimization of Functions having Lipschitz Continuous First Derivatives *Pacific Journal of Mathematics* **16 (1)** 1-3
- Nocedal, J., and S. Wright (2006): *Numerical Optimization 2nd Edition* **Springer**
- Wikipedia (2019): Newton's Method in Optimization
- Wikipedia (2019): Wolfe Conditions
- Wolfe, P. (1969): Convergence Conditions for Ascent Methods *SIAM Review* **11 (2)** 226-235
- Wolfe, P. (1971): Convergence Conditions for Ascent Methods II: Some Corrections *SIAM Review* **13 (2)** 185-188

Constrained Optimization

Constrained Optimization – Definition and Description

1. Constrained Optimization Definition – Variable Constraints: In mathematical optimization **constrained optimization** is the process of optimizing an objective function with respect to some variables in the presence of constraints on those variables.
2. Constrained Optimization Definition - Objective Function: The objective function is either a cost function or an energy function that is to be minimized, or a reward function or a utility function that has to be maximized.
3. Hard vs Soft Variable Constraints: Constraints can either be *hard constraints* that set conditions on variables required to be satisfied, or *soft constraints* that have some variable values that are penalized in the objective function if – and based on the extent that – the conditions on the variables are not satisfied (Constrained Optimization (Wiki)).

General Form

1. Constrained Optimization Problem – Mathematical Specification: A general constrained minimization problem may be written as follows:

$$\min_{x} f(x)$$

subject to

$$g_i(x) = c_i$$

for

$$i = 1, \cdots, n$$

equality constraints and

$$h_j(x) \geq d_j$$

for

$$j = 1, \cdots, m$$

inequality constraints where

$$g_i(x) = c_i$$

for

$$i = 1, \cdots, n$$

and

$$h_j(x) \geq d_j$$

for

$$j = 1, \cdots, m$$

are the constraints to be satisfied; these are called the hard constraints.

2. Constrained Optimizers - Soft Objective Function: In some problems, often called *constraint optimization problems*, the objective function is a sum of the cost functions, each of which penalized the extent if any to which a soft constraint – a constraint that is preferred but not required to be satisfied - is violated.

Solution Methods

1. Adaptation from Unconstrained Algorithms: Many unconstrained optimization algorithms can be adapted to the constrained case, often via the use of a penalty method. However, the search steps taken by the unconstrained method may be unacceptable for the constrained problem, leading to a lack of convergence. This is referred to as the Maratos effect (Sun and Yua (2010)).
2. Equality Constraints Lagrange Multiplier Technique: If the constrained problem has only equality constraints the method of Lagrange multipliers can be used to convert it to an unconstrained problem whose number of variables is the original number of variables plus the number of constraints.
3. Equality Constraints - Cross Variable Substitution: Alternatively, if the constraints are all equality constraints and linear, they can be solved for some of the variables in terms of the others, and the former can be substituted out of the objective function, leaving an unconstrained problem in a smaller number of variables.
4. Inequality Constraints - Conditions on Variables: With inequality constraints the problem can be characterized in terms of Geometric Optimality Conditions, Fritz-John Conditions, and Karush-Kuhn-Tucker Conditions in which simple problems may be directly solvable.
5. Linear Programming – Simplex/Interior Point: If the objective function and all of the hard constraints are linear, then the problem is a linear programming one. This can be solved by the Simplex method, which usually works in polynomial time in the

problem size but is not guaranteed to, or by interior point methods, which are guaranteed to work in polynomial time.

6. Quadratic Programming - Objective Function Constraints: If all the hard constraints are linear but the objective function is quadratic the problem is a quadratic programming problem.
7. Quadratic Programming Convex Objective Function: Quadratic Programming problems can be solved by the ellipsoid method in polynomial time if the objective function is convex; otherwise the problem is NP hard.

Constraint Optimization: Branch and Bound

1. Idea behind Branch and Bound: Constraint optimization can be solved by branch-and-bound algorithms. These are back-tracking algorithms that store the cost of the best solution found during execution and use it eventually to avoid part of the search.
2. Back Tracking vs. Solution Extension: More precisely whenever the algorithm encounters a partial solution that cannot be extended to form a solution with better cost than the best stored cost, the algorithm back tracks instead of trying to extend this solution.
3. Efficient of Back Tracking Algorithms: Assuming that the cost is to be minimized the efficiency of these algorithms depends on how the cost can be obtained from extending a partial solution is evaluated. Indeed, if the algorithms can back track from a partial solution, part of the search can be skipped. The lower the estimated cost the better the algorithm, as a lower estimated cost is more likely to be lower than the best cost of the solution found so far.
4. Algorithm Lower and Upper Bounds: On the other hand, this estimated cost cannot be lower than the effective cost obtained by extending the solution, as otherwise the algorithm could backtrack while a solution better than the best found so far exists. As a result, the algorithm requires an upper bound on the cost that can be obtained by extending a partial solution and this upper bound should be as small as possible.

5. Hansen's Branch-and-Bound Variation: A variation of the above method called Hansen's method uses interval methods (Leader (2004)). It inherently implements rectangular constraints.

Branch-and-Bound: First-Choice Bounding Conditions

1. Separately treating each Soft Constraint: One way of evaluating this upper bound for a partial solution is to consider each soft constraint separately. For each soft constraint the maximal possible value for any assignment to the unassigned variables is assumed. The sum of these variables is an upper bound because the soft constraints cannot assume a higher value.
2. Exact Nature of Upper Bound: It is exact because the maximal nature of soft constraints may derive from different evaluations: a soft constraint may be maximal for

$$x = a$$

while another soft constraint is maximal for

$$x = b$$

Branch-and-Bound Russian Doll Search

1. Branch-and-Bound Sub-problems: This method runs a branch-and-bound algorithm on n problems where n is the number of variables (Verfaillie, Lemaitre, and Schiex (1996)). Each such sub-problem is the sub-problem obtained by dropping the sequence $x_1, \cdots, x_i$ from the original problem along with the constraints containing them.

2. Sub-problem Cost Upper Bound: After the problem on variables $x_{i+1}, \cdots, x_n$ is solved its optimal cost can be used as an upper bound while solving the other problems.
3. Assigned/Unassigned Variables Sub-problem: In particular the cost of estimating a solution having $x_{i+1}, \cdots, x_n$ is added to the cost that derives from the evaluated variables. Virtually this corresponds to ignoring the evaluate variables and solving the problem on the unassigned ones, except that the latter problem has been already solved.
4. Sub-problem Total Cost Update: More precisely the cost of the constraints containing both the assigned and the unassigned variables is estimated as above, or using another arbitrary method; the cost of soft constraints using unassigned variables is instead estimated using the optimal solution of the corresponding problem, which is already known at this point.
5. Similarities with Dynamic Programming Approach: Like Dynamic Programming Russian Doll Search solves the sub-problems in order to solve the whole problem.
6. Differences with Dynamic Programming Approach: However, whereas Dynamic Programming directly combines the results obtained on the sub-problems to get the result of the whole program, the Russian Doll Search only uses them as bounds during the search.

Branch-and-Bound – Bucket Elimination

1. Elimination of a Specified Variable: The bucket elimination algorithm can be used for constraint optimization. A given variable can be removed from the problem by replacing all the soft constraints containing it with a new soft constraint.
2. Removed Variable Constraint-Cost Expression: The cost of the constraint expression is estimated assuming the maximal objective function value for each of the removed variable. Formally if x is the variable to be removed, $C_1, \cdots, C_n$ are the constraints containing it, and $y_1, \cdots, y_m$ are the variables containing x, the new soft constraint is defined by

$$C(y_1 = a_1, \cdots, y_n = a_n) = \max_a \sum_i C_i(x = a, y_1 = a_1, \cdots, y_n = a_n)$$

3. Bucket of all Variable Constraints: Bucket elimination works with an arbitrary ordering of the variables. Every variable is associated with a bucket of constraints; the bucket of a variable contains all the constraints having the variable has the highest in the order.
4. Order of the Bucket Elimination: Bucket elimination proceeds from the last variable to the first. For each variable, all constraints of the bucket are replaced as above to remove the variable. The resulting constraint is then placed in the appropriate bucket.

References

- Leader, J. J. (2004): *Numerical Analysis and Scientific Computation* **Addison Wesley**
- Sun, W., and Y. X. Yua (2010): *Optimization Theory and Methods: Non-linear Programming* **Springer**
- Verfaillie, G., M. Lemaitre, and T. Schiex (1996): Russian Doll Search for solving Constraint Optimization Problems *Proceedings of the 13th National Conference on Artificial Intelligence and 8th Innovative Applications of Artificial Intelligence Conference* **Portland** 181-187
- Wikipedia (2019): Constrained Optimization

Lagrange Multipliers

Motivation, Definition, and Problem Formulation

1. Purpose behind the Lagrange Methodology: In mathematical optimization the **method of Lagrange multipliers** (Lagrange (1811)) is a strategy for finding the local maxima and minima subject to equality constraints (Lagrange Multiplier (Wiki)).
2. Optimization Problem Mathematical Setup: Consider the optimization problem of maximizing $f(x, y)$ subject to

$$g(x, y) = 0$$

   A new variable λ is introduced (this is called the Lagrange multiplier) and the Lagrange function (or Lagrangian) defined by

$$\mathcal{L}(x, y, \lambda) = f(x, y) - \lambda g(x, y)$$

   is studied. Here the term λ may be either added or subtracted.
3. Lagrange Multipliers as Stationary Points: If $f(x_0, y_0)$ is the maximum of $f(x, y)$ for the original constrained problem, then there exists a λ_0 such that (x_0, y_0, λ_0) is a stationary point for the Lagrangian function (stationary points are those where the partial derivatives of $\mathcal{L}$ are zero).
4. Necessary Conditions for Constrained Optimality: However not all stationary points yield a solution to the original problem. Thus, the method of Lagrange multipliers yields necessary conditions for optimality in constrained problems (Hiriart-Urruty and Lemarechal (1993), Bertsekas (1999), Vapnyarskii (2001), Lemarechal (2001), Lasdon (2002)).

5. Sufficient Conditions for Constrained Optimality: Sufficient conditions for a maximum or a minimum also exist. Sufficient conditions for a constrained local maximum or a minimum can be stated in terms of a sequence of principal minors, i.e., determinants of the upper-left-justified sub-matrices, of the bordered Hessian matrix of the second derivatives of the Lagrangian expression (Chiang (1984)).

Introduction, Background, and Overview

1. Advantage of the Lagrange Multiplier Formulation: One of the most common problems in calculus is that of finding the extrema of a function, but it is often difficult to find a closed form for the function being extremized. Such difficulties often arise when one wishes to extremize the function subject to fixed outside constraints or conditions. The method of Lagrange multipliers is a powerful tool for solving this class of problems without the need to explicitly solve for the conditions and use them to eliminate the extra variables.
2. Intuition behind the Lagrange Multiplier Methodology: Consider the 2D problem introduced above: maximize $f(x,y)$ subject to

$$g(x,y) = 0$$

The method of Lagrange multipliers relies on the intuition that at the maximum $f(x,y)$ cannot be increasing in the direction of any neighboring point where

$$g(x,y) = 0$$

If it did, one could walk along

$$g(x,y) = 0$$

to get higher, meaning that the starting point wasn't actually the maximum.

3. Function Realization/Constraint Value Contours: The contours of f given by

$$f(x,y) = d$$

for various values of d as well as the contours of g given by

$$g(x,y) = 0$$

may be visualized as being parallel/tangential to each other.

4. Invariance of $f(x,y)$ along the Constraint: Suppose one walks along the contour with

$$g(x,y) = 0$$

One is interested in funding points along $f(x,y)$ that do not change along the walk since these points may be maxima. There are two ways this can happen. First one could be walking along the contour line of $f(x,y)$ since by definition $f(x,y)$ does not change along the contour line. This would mean that the contour lines of $f(x,y)$ and $g(x,y)$ are parallel here. The second possibility is that one has reached a "level" part of $f(x,y)$ meaning that $f(x,y)$ cannot change in any direction.

5. Constraint/Objective Function Gradient Scaling: To check for the first possibility, one notices that the gradient of a function is perpendicular to its contour lines, and the contour lines of $f(x,y)$ and $g(x,y)$ are parallel if and only if the gradients of $f(x,y)$ and $g(x,y)$ are parallel. Thus, one wants points (x,y) where

$$g(x,y) = 0$$

and

$$\vec{\nabla}_{x,y} f(x,y) = -\lambda \vec{\nabla}_{x,y} g(x,y)$$

for some λ where

$$\vec{\nabla}_{x,y} f(x,y) = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

$$\vec{\nabla}_{x,y} g(x,y) = \left[\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}\right]$$

are the respective gradients. The constant λ is required because although the two gradient vectors are parallel, the magnitudes of the gradient vectors are generally not equal. This constant is called the Lagrange multiplier (the negative sign is a convention).

6. The "Level" Plateau of f: This method also addresses the second possibility; if f is level then its gradient is zero, and setting

$$\lambda = 0$$

is a solution regardless of g.

7. Formulation of the Lagrangian Function: To incorporate these conditions into one equation one introduces an auxiliary function

$$\mathcal{L}(x,y,\lambda) = f(x,y) - \lambda g(x,y)$$

and solves for

$$\vec{\nabla}_{x,y,\lambda} \mathcal{L}(x,y,\lambda) = 0$$

This is the method of Lagrange multipliers. Note that

$$\vec{\nabla}_{\lambda}\mathcal{L}(x,y,\lambda)=0$$

implies

$$g(x,y)=0$$

To summarize

$$\vec{\nabla}_{x,y,\lambda}\mathcal{L}(x,y,\lambda)=0$$

results in

$$\vec{\nabla}_{x,y}f(x,y)=-\lambda\vec{\nabla}_{x,y}g(x,y)$$

and

$$g(x,y)=0$$

The constrained extrema of f are the *critical points* but they are not necessarily the *local extrema* of $\mathcal{L}$.

8. Alternate Formulations for the Lagrangian: One may re-formulate the Lagrangian as a Hamiltonian, in which case the solutions are the local minima for the Hamiltonian. This is done in optimal control theory in the form of Pontryagin's minimum principle.
9. Non-Extremum Solutions to Lagrangian: The fact that the solutions to the Lagrangian are not necessarily the extrema also poses difficulties for numerical optimization. This can be addressed by computing the *magnitude* of the gradient, as the zeroes of the magnitude are necessarily the local minima, as illustrated later in the part on numerical optimization.

Handling Multiple Constraints

1. Notation Used for Multiple Constraints: Throughout this section the independent variables are denoted $x_1, \cdots, x_N$ and, as a group denoted as

$$p = (x_1, \cdots, x_N)$$

   Also, the function being analyzed will be denoted $f(p)$ and the constraints will be represented by the equations

$$g_1(p) = 0$$
$$\vdots$$
$$g_M(p) = 0$$

2. Basic Idea behind the Constraints: The basic idea remains essentially the same: if we consider only satisfy the constraints (i.e., are *in* the constraints) then a point $\big(p, f(p)\big)$ is a stationary point (i.e., a point in a *flat* region) of f if and only if the constraints at that point do not allow movement in a direction where f changes value. Once the stationary points are located further tests are needed to see if it is a minimum, a maximum, or just a stationary point that is neither.
3. Multi-dimensional Invariance of f: Consider the level set of f at $\big(p, f(p)\big)$. Let the set of vectors $\{v_L\}$ contain the directions in which one can move and still remain in the same level set, the directions where the value of f does not change (i.e., the change equals zero). For every vector v in $\{v_L\}$ the following relation must hold:

$$\frac{\partial f}{\partial x_1} v_1 + \cdots + \frac{\partial f}{\partial x_N} v_N = 0$$

   where the notation v_k above refers to the k^{th} component of the vector v.

4. Invocation of the Gradient of f: The equation above can be re-written in a more compact geometric form that helps the intuition:

$$\left[\frac{\partial f}{\partial x_1} \quad \cdots \quad \frac{\partial f}{\partial x_N}\right] \cdot \begin{bmatrix} v_1 \\ \vdots \\ v_N \end{bmatrix} = 0$$

i.e.

$$\vec{\nabla} f^T \cdot \vec{v} = 0$$

This makes it clear that if one is at p then *all* directions from this point that do *not* change the value of f *must be perpendicular to* $\vec{\nabla} f(p)$ (the gradient of f at p).

5. Usage of the Constraint Gradient: Each constraint limits the directions that one can move from a particular point and still satisfy the constraint. One uses the same set of procedures above to look for a set of vectors $\{v_C\}$ that contain the directions in which can move and still satisfy the constraint. As above, for every vector v in $\{v_C\}$ the following relation must hold:

$$\frac{\partial g}{\partial x_1} v_1 + \cdots + \frac{\partial g}{\partial x_N} v_N = 0$$

i.e.

$$\vec{\nabla} g^T \cdot \vec{v} = 0$$

From this it can be seen that at point p all directions from this point that will satisfy the constraint must be proportional to $\vec{\nabla} g(p)$.

6. Lagrange Multiplier Method Formal Definition: A point on f is a constrained stationary point if and only if the direction that changes f violates at least one of the constraints, i.e., it has no "component" in the legal space perpendicular to $\vec{\nabla} g(p)$.

Mathematically this means that the gradient of f at this constrained stationary point is perpendicular to the space spanned by the set of vectors $\{v_C\}$ which in turn is perpendicular to the gradients of the constraints g.

7. Single Constraint f/g Gradients: For a single constraint the above statement says that at stationary points the direction that changes f is the same as that violates the constraint. To determine if two vectors are in the same direction, note that if two vectors start from the same point then open vector can always reach the other by changing its length and/or flipping or the opposite way along the same direction line. This requires that

$$\vec{\nabla} f(p) = \lambda \vec{\nabla} g(p)$$

implying that

$$\vec{\nabla} f(p) - \lambda \vec{\nabla} g(p) = 0$$

8. Single Constraint Case - Compact Formulation: On adding another simultaneous equation to guarantee that this test is performed only at the point that satisfies the constraint two simultaneous equations result, which when solved identify all the constrained stationary points.

$$g(p) = 0$$

implies that p satisfies the constraint, thus

$$\vec{\nabla} f(p) - \lambda \vec{\nabla} g(p) = 0$$

is a stationary point.

9. Single Constraint Case Expanded Formulation: Fully expanded, there are simultaneous equations that need to be solved for $N + 1$ variables, which are $x_1, \cdots, x_N$ and λ:

$$g(x_1, \cdots, x_N) = 0$$

$$\frac{\partial f(x_1, \cdots, x_N)}{\partial x_1} - \lambda \frac{\partial g(x_1, \cdots, x_N)}{\partial x_1} = 0$$
$$\vdots$$
$$\frac{\partial f(x_1, \cdots, x_N)}{\partial x_N} - \lambda \frac{\partial g(x_1, \cdots, x_N)}{\partial x_N} = 0$$

10. Multiple Constraints and their Gradients: For more than one constraint a similar reasoning applies. Each gradient function g has a space of allowable directions at p – the space of vectors perpendicular to $\vec{\nabla} g(p)$. The set of directions allowed by all constraints is thus the space of directions perpendicular to all of the constraint gradients. Denoting the space of allowable moves by A and the span of gradients by S, using the discussion above

$$A = S^{\perp}$$

the space of vectors perpendicular to every element in S.

11. Multiple Constraints Case Gradient Formulation: If p is an optimum any element not perpendicular to $\vec{\nabla} f(p)$ is not an allowable direction. One can show that this implies

$$\vec{\nabla} f(p) \in A^{\perp} = S$$

Thus, there are scalars $\lambda_1, \cdots, \lambda_M$ such that

$$\vec{\nabla} f(p) = \sum_{k=1}^{M} \lambda_k \vec{\nabla} g_k(p)$$

implies

$$\vec{\nabla} f(p) - \sum_{k=1}^{M} \lambda_k \vec{\nabla} g_k(p) = 0$$

As before the simultaneous equations are added to guarantee that this test is performed only at a point that satisfies every constraint, and the resulting equations, when satisfied, identify all the constrained stationary points.

$$g_1(p) = \cdots = g_M(p) = 0$$

implies that p satisfies all constraints; further

$$\vec{\nabla} f(p) - \sum_{k=1}^{M} \lambda_k \vec{\nabla} g_k(p) = 0$$

indicates that p is a stationary point.

12. Multiple Gradient Case Lagrangian Formulation: The method is complete now from the standpoint of finding the stationary points, but these equations can be condensed into a more succinct and elegant form. The equations above look like partial derivatives of a larger scalar function $\mathcal{L}$ that takes all the $x_1, \cdots, x_N$ and all the $\lambda_1, \cdots, \lambda_M$ as inputs. Setting every equation to zero is exactly what one would have to do to solve for the *unconstrained* stationary points of the larger function. Finally, the larger function $\mathcal{L}$ with partial derivatives that are exactly the ones that we require can be constructed very simply as

$$\mathcal{L}(x_1, \cdots, x_N, \lambda_1, \cdots, \lambda_M) = f(x_1, \cdots, x_N) - \sum_{k=1}^{M} \lambda_k \vec{\nabla} g_k(x_1, \cdots, x_N)$$

Solving the above equation for its *unconstrained* stationary points generates exactly the same stationary points as solving for the *constrained* stationary points of f under the constraints $g_1, \cdots, g_M$

13. The Method of Lagrange Multipliers: In Lagrange's honor the above function is called a *Lagrangian*, the scalar multipliers $\lambda_1, \cdots, \lambda_M$ are called *Lagrange Multipliers*, and the method itself is called *The Method of Lagrange Multipliers*.
14. The Karush-Kuhn-Tucker Generalization: The method of Lagrange multipliers is generalized by the Karush-Kuhn-Tucker conditions, which also takes into account inequality constraints of the form

$$h(\vec{x}) \leq c$$

Modern Formulation via Differentiable Manifolds

1. Local Extrema of $\mathbb{R}^d \to \mathbb{R}^1$: Funding the local maxima of a function

$$f: \mathbb{U} \to \mathbb{R}$$

where $\mathbb{U}$ is an open subset of $\mathbb{R}^d$ is done by finding all points $x \in \mathbb{U}$ such that

$$\mathbb{D}_x f = 0$$

and then checking whether all of the eigenvalues of the Hessian $\mathbb{H}_x f$ are negative. Setting

$$\mathbb{D}_x f = 0$$

is a non-linear problem and, in general, arbitrarily difficult. After funding the critical points checking for the eigenvalues is a linear problem and thus easy.

2. Extrema under the Level-Set Constraint: When

$$g: \mathbb{R}^d \to \mathbb{R}^1$$

is a smooth function such that

$$\mathbb{D}_x g \neq 0$$

for all x in the level set

$$g(x) = c$$

then $g^{-1}(c)$ becomes an $n-1$ dimensional smooth manifold $\mathbb{M}$ by the level set theorem. Funding local maxima is by definition a local problem, so it can be done on the local charts of $\mathbb{M}$ after funding a diffeomorphism

$$\varphi: \mathbb{V} \to \mathbb{R}^{n-1}$$

from an open subset of

$$\mathbb{V} \subset \mathbb{M}$$

onto an open subset

$$\mathbb{U} \subset \mathbb{R}^{n-1}$$

and thus, one can apply the algorithm in the previous part to the function

$$f' = f \circ \varphi^{-1} \colon \mathbb{U} \to \mathbb{R}$$

3. Approach of Lagrange Multiplier Method: While the above idea sounds good it is difficult to compute φ^{-1} in practice. *The entire method of Lagrange multipliers reduces to skipping that step and finding the zeroes of* $\mathbb{D}_x f'$ *directly.* It follows from the construction in the level set theorem that $\mathbb{D}_x \varphi^{-1}$ is the inclusion map

$$\ker \mathbb{D}_{\varphi^{-1}(x)} g \subseteq \mathbb{R}^n$$

Therefore

$$0 = \mathbb{D}_x f' = \mathbb{D}_x (f \circ \varphi^{-1}) = \mathbb{D}_{\varphi^{-1}(x)} f \circ \mathbb{D}_x \varphi^{-1}$$

if and only if

$$\ker \mathbb{D}_y g \subseteq \ker \mathbb{D}_y f$$

from writing y for $\varphi^{-1}(x)$

4. Existence of the Linear Map: By the first isomorphism theorem this is true if and only if there exists a linear map

$$\mathcal{L} \colon \mathbb{R} \to \mathbb{R}$$

such that

$$\mathcal{L} \circ \mathbb{D}_y g = \mathbb{D}_y f$$

As a linear map one must have that

$$\mathcal{L}(x) = \lambda x$$

for a fixed

$$\lambda \in \mathbb{R}$$

Therefore, funding the critical point of f' is equivalent to solving the system of equations

$$\lambda \mathbb{D}_y g = \mathbb{D}_y f$$

$$g(y) = c$$

in the variables

$$y \in \mathbb{R}^{n-1}$$

and

$$\lambda \in \mathbb{R}$$

This is in general a non-linear system of n equations and n unknowns.

5. Multiple Constraints Surjective Linear Map: In the case of general constraints one works with

$$g: \mathbb{R}^n \to \mathbb{R}^m$$

and replaces the condition

$$\mathbb{D}_x g \neq 0$$

for all

$$x \in g^{-1}(c)$$

with the requirement that $\mathbb{D}_x g$ be surjective at all such points. In this case $\mathcal{L}$ will be a linear map $\mathbb{R}^m \to \mathbb{R}$ i.e., a row vector with m entries.

Interpretation of the Lagrange Multipliers

1. As a Rate of Change Quantity: Often the Lagrange multipliers have an interpretation as some quantity of interest. For example, if the Lagrangian expression is

$$\begin{aligned}\mathcal{L}(x_1, \cdots, x_N, \lambda_1, \cdots, \lambda_M) \\ = f(x_1, \cdots, x_N) + \lambda_1[c_1 - g_1(x_1, \cdots, x_N)] + \cdots \\ + \lambda_M[c_M - g(x_1, \cdots, x_N)]\end{aligned}$$

then

$$\frac{\partial \mathcal{L}}{\partial c_k} = \lambda_k$$

So λ_k is the rate of change of the quantity being optimized as a function of the constraint variable.

2. Examples of Lagrange Multiplier Roles: As examples, in Lagrangian mechanics the equations of motion are derived by finding stationary points of action, i.e., the time integral of the difference between the potential and the kinetic energies. Thus, the force on a particle due to a scalar potential

$$\vec{F} = -\vec{\nabla}V$$

can be interpreted as a Lagrange multiplier determining the change in action – transfer of potential to kinetic energy – following a variation in the particle's constrained trajectory. In control theory this formulated instead as co-state equations.

3. Marginal Effect of the Constraint: Moreover, by the control theorem the optimal value of a Lagrange multiplier has an interpretation as the marginal effect of the corresponding constraint constant upon on the optimal attainable value of the original objective function. Denoting the optimum with an asterisk it can be shown that

$$\frac{\partial f\left(x_1{}^*(c_1, \cdots, c_M), \cdots, x_N{}^*(c_1, \cdots, c_M)\right)}{\partial c_k} = \lambda_k{}^*$$

4. Lagrange Multiplier Usage in Economics: For example, in economics the optimal profit to a player is calculated subject to a constrained space of actions, where the Lagrange multiplier is the change in the optimal value of the objective function (profit) due to the relaxation of a given constraint – e.g., through a change in the income. In such a context λ^* is the marginal cost of the constraint, and is referred to as the shadow price.

Lagrange Application: Maximal Information Entropy

1. Information Entropy Maximization – Problem Setup: Suppose on wishes to find the discrete probability distribution on the points $\{p_1, \cdots, p_n\}$ with maximal information entropy. This is the same as saying that one wishes to find the least biased probability on the points $\{p_1, \cdots, p_n\}$. In other words, one wishes to maximize the Shannon entropy equation

$$f(p_1, \cdots, p_n) = -\sum_{j=1}^{n} p_j \log p_j$$

2. Cumulative Discrete Probability Normalization Constraint: For this to be a probability distribution the sum of the probabilities p_j at each point x_i must equal 1, so the constraint becomes

$$g(p_1, \cdots, p_n) = \sum_{j=1}^{n} p_j \log p_j$$

3. Application of the Lagrange Multipliers: Using Lagrange multipliers to find the point of maximum entropy $\vec{p}^*$ across all discrete probability distributions $\vec{p}$ on $\{x_1, \cdots, x_n\}$ requires that

$$\frac{\partial}{\partial \vec{p}} [f + \lambda(g - 1)]\Big|_{\vec{p}=\vec{p}^*} = 0$$

which gives a system of n equations with indices

$$k = 1, \cdots, n$$

such that

$$\frac{\partial}{\partial p_k}\left[-\sum_{j=1}^{n} p_j \log p_j + \lambda\left(\sum_{j=1}^{n} p_j - 1\right)\right]\Bigg|_{\vec{p}=\vec{p}^*} = 0$$

4. Solution to the Discrete Probability: Carrying out the differentiation on these n equations one gets

$$-[1 + \log p_k{}^*] + \lambda = 0$$

This shows that all $p_k{}^*$ are the same because they depend only on λ. By using the constraint

$$\sum_{j=1}^{n} p_j = 1$$

one finds that

$$p_k{}^* = \frac{1}{n}$$

Hence the uniform distribution is the distribution with the greatest entropy, using distributions on n points.

Lagrange Application: Numerical Optimization Techniques

1. Local Minima vs. Saddle Points: The critical points of the Lagrangian occur at saddle points rather than the local minima or maxima (Heath (2005)). Unfortunately, many numerical optimization techniques such as hill climbing, gradient descent, some of the quasi-Newton methods, among others, are designed to find local minima (or maxima) and not the saddle points.
2. Conversion to an Extremization Problem: For this reason one must modify the formulation to ensure that this is a minimization problem – for example by extremizing the square of the gradient of the Lagrangian – or use an optimization technique that finds stationary points and not necessarily extrema, e.g., such as Newton's method without an extremum seeking line search.

3. "Saddle" Critical Points - An Example: As a simple example consider the problem of finding the value of x that minimizes the function

$$f(x) = x^2$$

constrained such that

$$x^2 = 1$$

Clearly this problem is pathological because there are only two values that satisfy the constraint, but it is useful for illustration purposes because the corresponding unconstrained function can be visualized in three dimensions.

4. Application of the Lagrange Multipliers: Using Lagrange multipliers this problem can be converted into an unconstrained optimization problem

$$\mathcal{L}(x, \lambda) = x^2 + \lambda(x^2 - 1)$$

The two critical points occur at the saddle points where

$$x = +1$$

and

$$x = -1$$

5. Transformation to Local Extremum Problem: In order to solve this problem with a numerical optimization technique one must first transform this problem such that the critical points occur at the local minima. This is done by computing the magnitude of the gradient of the unconstrained optimization problem.

6. Objective Variate/Constraint Multiplier Jacobian: First one computes the partial derivative of the unconstrained problem with respect to each variable;

$$\frac{\partial \mathcal{L}}{\partial x} = 2x + 2\lambda x$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = x^2 - 1$$

If the target function is not easily differentiable the differential with respect to each variable can be approximated using the divided differences technique, i.e.

$$\frac{\partial \mathcal{L}(x,\lambda)}{\partial x} \approx \frac{\mathcal{L}(x+\epsilon,\lambda) - \mathcal{L}(x-\epsilon,\lambda)}{2\epsilon}$$

$$\frac{\partial \mathcal{L}(x,\lambda)}{\partial \lambda} \approx \frac{\mathcal{L}(x,\lambda+\epsilon) - \mathcal{L}(x,\lambda-\epsilon)}{2\epsilon}$$

where ϵ is a small value.

7. Computing the Magnitude of the Gradient: Next one computes the magnitude of the gradient, which is the square root of the sum of squares of the partial derivatives:

$$h(x,\lambda) = \sqrt{(2x + 2\lambda x)^2 + (x^2 - 1)^2}$$

$$\approx \sqrt{\left[\frac{\mathcal{L}(x+\epsilon,\lambda) - \mathcal{L}(x-\epsilon,\lambda)}{2\epsilon}\right]^2 + \left[\frac{\mathcal{L}(x,\lambda+\epsilon) - \mathcal{L}(x,\lambda-\epsilon)}{2\epsilon}\right]^2}$$

Since the magnitude is always non-negative optimizing over the squared magnitude is equivalent to optimizing over the magnitude. Thus, the square root may be omitted from these equations with no expected differences in the results of the optimization.

8. Critical Points as Local Extrema: The critical points of h occur at

$$x = +1$$

and

$$x = -1$$

just as in $\mathcal{L}$. However, the critical points in h occur at local minima, so the numerical optimization techniques can be used to find them.

Lagrange Multipliers – Common Practice Applications

1. Lagrange Multipliers Applied in Economics: Constrained optimization plays a central role in economics. For example, the choice problem for a customer is represented as one of maximizing a utility function subject to a budget constraint. The Lagrange multiplier has an interpretation as the shadow price associated with that constraint – in this instance the marginal utility of the income. Other examples include profit maximization for a firm, along with various macro-economic applications.
2. Lagrange Multipliers in Control Theory: In optimal control theory the Lagrange multipliers are represented as co-state variables and are re-formulated as the minimization of the Hamiltonian, e.g., they are the basis behind the Pontryagin's minimum principle.
3. Lagrange Multipliers in Non-linear Programming: The Lagrange multiplier method has several generalizations. In non-linear programming there are several multiplier rules, e.g., the Caratheodery-John Multiplier Rule and the Convex Multiplier Rule for inequality constraints (Pourciau (1980)).

References

- Bertsekas, D. P. (1999): *Nonlinear Programming 2nd Edition* **Athena Scientific** Cambridge MA
- Chiang, A. C. (1984): *Fundamental Methods of Mathematical Economics 3rd Edition* **McGraw-Hill**
- Heath, M. T. (2005): *Scientific Computing – An Introductory Survey* **McGraw-Hill**
- Hiriart-Urruty, J. B., and C. Lemarechal (1993): XII Abstract Duality for Practitioners, in: *Convex Analysis and Minimization Algorithms, Volume II: Advanced Theory and Bundle Methods* **Springer-Verlag** Berlin
- Lagrange, J. L. (1811): *Mecanique Analytique*
- Lasdon, L. S. (2002): *Optimization Theory for Large Systems* **Dover Publications** Mineola NY
- Lemarechal, C. (1993): Lagrangian Relaxation, in: *Computational Combinatorial Optimization: Papers from the Spring School held in Schloss Dagstuhl (Editors: M. Junger and D. Naddef)* **Springer-Verlag** Berlin
- Pourciau, B. H. (1980): Modern Multiplier Rules *American Mathematical Monthly* **87 (6)** 433-452
- Vapnyarskii, I. B. (2001): Lagrange Multiplier, in: *Encyclopedia of Mathematics (editor: M. Hazewinkel)* **Springer**
- Wikipedia (2019): Lagrange Multiplier

Spline Optimizer

Constrained Optimization using Lagrangian

1. Base Set up: Use the Lagrangian objective function to optimize a multi-variate function $L(x, y)$ to incorporate the constraint

$$g(x, y) = c$$

as

$$\Lambda(x, y, z) = L(x, y) + \lambda[g(x, y) - c]$$

Here λ is called the Lagrange multiplier, and use one Lagrange multiplier per constraint.

2. Optimize (Maximize/Minimize) the Lagrangian: Optimize (i.e., maximize/minimize) the Lagrangian with respect to x, y, and λ – thus

$$\frac{\partial \Lambda(x, y, z)}{\partial x} = 0$$

$$\frac{\partial \Lambda(x, y, z)}{\partial y} = 0$$

and

$$\frac{\partial \Lambda(x, y, z)}{\partial \lambda} = 0$$

Notice that

$$\frac{\partial \Lambda(x,y,z)}{\partial \lambda} = 0$$

automatically implies the validity of the constraint

$$g(x,y) = c$$

thereby accommodating it in a natural way.

- Further

$$\frac{\partial^2 \Lambda(x,y,z)}{\partial \lambda^2} = 0$$

always, since $\Lambda(x,y,z)$ is linear in λ. Further, since the constraint is true by the optimizer construction, there should be no explicit dependence on λ.

3. Comparison with unconstrained optimization:
   - Unconstrained Optimization results in

$$\frac{\partial L(x,y)}{\partial x} = 0$$

and

$$\frac{\partial L(x,y)}{\partial y} = 0$$

Constrained Optimization results in

$$\frac{\partial \Lambda(x,y,z)}{\partial x} = \frac{\partial L(x,y)}{\partial x} + \lambda \frac{\partial g(x,y)}{\partial x}$$

and

$$\frac{\partial \Lambda(x,y,z)}{\partial y} = \frac{\partial L(x,y)}{\partial y} + \lambda \frac{\partial g(x,y)}{\partial y}$$

$$\lambda = 0$$

automatically reduces the constrained case to the unconstrained.

- Advantage of the Lagrange Multiplier Incorporation into Optimization => This ends up converting the constrained formulation space over on to the unconstrained, thereby providing with as many equations as the number of optimization unknowns, and one equation each per every constraint.
- Drawback of the Lagrange Multiplier Optimization Incorporation => While it lets you passively move to the unknowns' space from the constrained formulation space, this may end up impacting the application of certain boundary conditions, such as financial boundary, natural boundary conditions, Not-A-Knot boundary condition, etc.

4. Constraints of inequality $g(x,y) < c$ or $g(x,y) > c$: Solutions to these constraints exist either inside the unconstrained set, in which case the unconstrained equations can be solved, or they don't, in which case convert the inequality to an equality and give it the Lagrange multiplier treatment.
5. Comparison with Convex Optimization: Convex optimization is predicated on the presence of at least one minimum/maximum in the zone of interest. One example is variance/covariance constrained optimization, where both the variance/covariance and the constraints are both quadratic. Eigenization (just another type of constrained variance/covariance optimization) is another.
6. Penalizing Optimizer as a Constrained Optimization Setup: Naively put

$$\Lambda = Good - Bad$$

where, from a calibration point of view, "Good" refers to closeness of fit, and "Bad" to the curvature/smoothness penalty. Thus, constrained optimization here corresponds to "maximize the Good, and minimize the Bad".

- De-coupling "Good/Bad" from closeness of fit and curvature/smoothness penalty, respectively, can lead to alternate optimization framework formulations in finance, along with its insights.

Least Squares Optimizer

1. Least Squares Optimization Formulation:

$$\mu_i(x_i) = y_i$$

$$\hat{\mu}_i(x_i) = \sum_{j=1}^{n} a_j B_{ij}(x_i)$$

$$S = \sum_{i=1}^{m} [\mu_i(x_i) - \hat{\mu}_i(x_i)]^2 = \sum_{i=1}^{m} \left[y_i - \sum_{j=1}^{n} a_j B_{ij}(x_i) \right]^2$$

$$= \sum_{i=1}^{m} \left\{ {y_i}^2 - 2 y_i \sum_{j=1}^{n} a_j B_{ij}(x_i) + \left[\sum_{j=1}^{n} a_j B_{ij}(x_i) \right]^2 \right\}$$

$$\frac{\partial S}{\partial a_j} = 2 \left[\sum_{i=1}^{m} a_j B_{ij}(x_i) - y_j \right] \left[\sum_{i=1}^{m} a_j B_{ij}(x_i) \right]$$

2. Least Squares Matrix Formulation:

$$Y = [y_1, \cdots, y_m]^T$$

$$B_i(\vec{x}) = \left[B_{i,1}(\vec{x}), \cdots, B_{i,m}(\vec{x})\right]^T$$

$$a = [a_1, \cdots, a_m]^T$$

Then

$$\left[\frac{\partial \vec{S}}{\partial A}\right] = 2AB(\vec{x})B^T(\vec{x})A - 2YB(\vec{x})$$

As expected,

$$y_i = \sum_{j=1}^{n} a_j B_{ij}(x_i)$$

is the optimized least squares tight fit.

Karush-Kuhn-Tucker Conditions

Introduction, Overview, Purpose, and Motivation

1. KKT Conditions - Necessity and Scope: In mathematical optimization the Karush-Kuhn-Tucker (KKT) Conditions – also known as Kuhn-Tucker Conditions – are the first order necessary conditions for a solution in non-linear programming to be optimal provided some regularity conditions are satisfied (Karush-Kuhn-Tucker Conditions (Wiki)).
2. KKT Conditions vs. Lagrange Multipliers: Allowing for inequality constraints the KKT approach to non-linear programming generalizes the method of Lagrange multipliers, which only allows for equality constraints.
3. Mathematical Foundation behind Optimization Algorithms: The system of equations corresponding to the KKT conditions is usually not solved directly except in a few special cases where a closed-form solution may be derived analytically. In general, many optimization algorithms can be interpreted as methods for numerically solving the KKT systems of equations (Boyd and van den Berghe (2009)).
4. KKT Conditions - Historical Attribution Revision: The KKT conditions were originally named after Harold W. Kuhn and Albert W. Tucker who first published the conditions in 1951 (Kuhn and Tucker (1951)). Later scholars discovered that then necessary conditions for this problem had been stated by William Karush in his master's thesis (Karush (1939), Kjeldsen (2000)).

Necessary Conditions for Optimization Problems

1. KKT Statement - Non-linear Optimization Problem: Consider the following non-linear optimization problem: Maximize $f(x)$ subject to

$$g_i(x) \leq 0$$

and

$$h_j(x) = 0$$

where x is the optimization variable, f is the *objective* or the *utility* function, $g_i(i = 1, \cdots, m)$ are the inequality constraint functions, and $h_j(j = 1, \cdots, l)$ are the equality constraint functions. The numbers of the inequality and the equality constraints are denoted m and l respectively.

2. KKT Statement - The Necessary Condition: Suppose that the objective function

$$f: \mathbb{R}^n \to \mathbb{R}$$

and the constraint functions

$$g_i: \mathbb{R}^n \to \mathbb{R}$$

and

$$h_j: \mathbb{R}^n \to \mathbb{R}$$

are continuously differentiable at the point x^*. If x^* is a local optimum that satisfies some regularity conditions below, there exist constants $\mu_i(i = 1, \cdots, m)$ and $\lambda_j(j = 1, \cdots, l)$ called KKT multipliers that have the following properties.

3. KKT Statement - Optimal Stationary Conditions: For maximizing $f(x)$

$$\vec{\nabla} f(x^*) = \sum_{i=1}^{m} \mu_i \vec{\nabla} g_i(x^*) + \sum_{j=1}^{l} \lambda_i \vec{\nabla} h_j(x^*)$$

For minimizing $f(x)$

$$-\vec{\nabla} f(x^*) = \sum_{i=1}^{m} \mu_i \vec{\nabla} g_i(x^*) + \sum_{j=1}^{l} \lambda_i \vec{\nabla} h_j(x^*)$$

4. KKT Statement - Primal Feasibility Conditions:

$$g_i(x^*) \leq 0$$

for all

$$i = 1, \cdots, m$$

and

$$h_j(x^*) = 0$$

for all

$$j = 1, \cdots, l$$

5. KKT Statement - Dual Feasibility Conditions:

$$\mu_i \geq 0$$

for all

$$i = 1, \cdots, m$$

6. KKT Statement – Complementary Slackness Conditions:

$$\mu_i g_i(x^*) = 0$$

for all

$$i = 1, \cdots, m$$

7. Reduction to the Lagrange Criterion: In the particular case

$$m = 0$$

where there are no inequality constraints the KKT conditions turn into the Lagrange conditions and the KKT multipliers become the Lagrange multipliers.

8. Non-differentiable Version of KKT: If some of the functions are non-differentiable sub-differentiable versions of the KKT conditions are available (Eustaquio, Karas, and Ribeiro (2008)).

Regularity Conditions or Constraint Qualifications

1. The KKT Necessary Regularity Conditions: In order for a minimum point x^* to satisfy the above KKT conditions the problem should satisfy some regularity conditions. The most common ones are listed below.
2. Linear Constraint Qualification: If g_i and h_j are affine functions the no other condition is needed to be satisfied.

3. Linear Independence Constraint Qualification (LICQ): The gradients of the active inequality constraints and the gradients of the equality constraints are linearly independent at x^*.
4. Mangasarian-Fromovitz Constraint Qualification (MFCQ): The gradients of the active inequality constraints and the gradients of the equality constraints are positive-linearly independent at x^*.
5. Constant Rank Constraint Qualification (CRCQ): For each subset of the gradients of the active inequality constraints and the gradients of the active equality constraints the rank at the vicinity of x^* is constant.
6. Constant Positive Linear Dependence Constraint Qualification (CPLD): For each subset of the gradients of the active inequality constraints and the gradients of the equality constraints, if it is positive-linearly dependent at x^*, then it is positive-linearly dependent at the vicinity of x^*.
7. Quasi-Normality Constraint Qualification (QNCQ): If the gradients of the active inequality constraints and the gradients of the active equality constraints are positive-linearly dependent at x^* with the associated multipliers λ_i for equalities and μ_j for inequalities then there is no sequence

$$x_k \to x^*$$

such that

$$\lambda_i \neq 0$$

implies

$$\lambda_i h_i(x_k) > 0$$

AND

$$\mu_j \neq 0$$

implies

$$\mu_j g_j(x_k) > 0$$

8. Slater Condition: For a convex problem there exists a point x such that

$$h_j(x) = 0$$

and

$$g_i(x) < 0$$

9. Positive Linear Dependency Condition Definition: The set of vectors $\{v_1, \cdots, v_n\}$ is positive linearly dependent if there exists

$$a_1 \geq 0, \cdots, a_n \geq 0$$

not all zero such that

$$a_1 v_1 + \cdots + a_n v_n = 0$$

10. Strength Order of Constraint Qualifications: It can be shown that

$$LICQ \Longrightarrow MFCQ \Longrightarrow CPLD \Longrightarrow QNCQ$$

and

$$LICQ \Longrightarrow CRCQ \Longrightarrow CPLD \Longrightarrow QNCQ$$

– the converses are not true – although MFCQ is not equivalent to CRCQ (Ruszczynski (2006)). In practice weaker constraint qualifications are preferred since they provide stronger optimality conditions.

Sufficient Conditions

1. The Second Order Sufficiency Conditions: In some cases, the necessary conditions are also sufficient for optimality. In general, the necessary conditions are not sufficient for optimality and more information is needed, such as the Second Order Sufficiency Conditions (SOSC). For smooth functions SOSC involve the second derivatives, which explains its name.
2. When Necessary Conditions are Sufficient: The necessary conditions are sufficient for optimality of the objective function f of a maximization problem is a concave function, the inequality constraints g_i are continuously differentiable convex functions, and the equality constraints h_j are affine functions.
3. The Type 1 Invex Functions: The broader class of functions in which the KKT conditions guarantee global optimality are the so-called Type 1 **invex functions** (Martin (1985), Hanson (1999)).
4. Second Order Sufficiency Conditions Formulation: For smooth non-linear optimization problems the second order sufficiency conditions are given as follows. Consider x^*, λ^*, and μ^* that find a local minimum using the Karush-Kuhn-Tucker conditions above. With μ^* such that the strict complementary condition is held at x^*, i.e.

$$\mu > 0$$

for all

$$s \neq 0$$

such that

$$\left[\frac{\partial g(x^*)}{\partial x}, \frac{\partial h(x^*)}{\partial x}\right] s = 0$$

the following equation must hold:

$$s^T \nabla_{xx}{}^2 \mathcal{L}(x^*, \lambda^*, \mu^*) s > 0$$

If the above condition is strictly met, the function is a strict constrained local minimum.

KKT Conditions Application - Economics

1. KKT Models as Theoretical Tools: Often in mathematical economics the KKT approach is used in theoretical models in order to obtain qualitative results.
2. Minimum Profit Constraint Revenue Maximization: Consider a firm that maximizes its sales revenue subject to a minimum profit constraint. Letting Q be the quantity of output produced – this is to be chosen - $R(Q)$ be the sales revenue with a positive first derivative and with a zero value at zero output, $C(Q)$ be the production cost with a positive first derivative and with a non-negative value at zero output, and G_{min} be the positive minimal acceptable level of profit, then the problem is a meaningful one if the revenue function levels off so it is eventually less steep than the cost function.
3. Application of the KKT Condition: The problem can be expressed in the following minimization form: Minimize $-R(Q)$ subject to

$$G_{min} \leq R(Q) - C(Q)$$

$$Q \geq 0$$

and the KKT conditions are

$$Q\left[\frac{\partial R}{\partial Q}(1+\mu) - \mu\frac{\partial C}{\partial Q}\right] = 0$$

$$R(Q) - C(Q) - G_{min} \geq 0$$

$$\mu \geq 0$$

$$\mu[R(Q) - C(Q) - G_{min}] = 0$$

4. Revenue Growth vs. Cost Growth: Since

$$Q = 0$$

would violate the minimum profit constraint

$$Q > 0$$

must hold, and hence the third condition implies that the first condition holds with equality. Solving that equality gives

$$\frac{\partial R}{\partial Q} = \frac{\mu}{1+\mu}\frac{\partial C}{\partial Q}$$

5. Impact of the Minimum Profit Constraint: Because it was given that $\frac{\partial R}{\partial Q}$ and $\frac{\partial C}{\partial Q}$ are strictly positive, the inequality along with the non-negativity condition on μ is

positive and so the revenue-maximizing firm operates at a level of output at which the marginal revenue $\frac{\partial R}{\partial Q}$ is less than the marginal cost $\frac{\partial C}{\partial Q}$ - a result that is of interest because it contrasts the behavior of a profit maximizing firm which operates at a level at which they are equal.

KKT Conditions Application – Value Function

1. Equality/Inequality Function Space Constraints: Reconsidering the optimization problem as a maximization problem with constant inequality constraints $v_1, \cdots, v_n$ Maximize $f(x)$ subject to

$$g_i(x) \leq a_i$$

$$h_j(x) = 0$$

2. Definition of the Value Function: The value function is defined as

$$V(a_1, \cdots, a_n) = \sup_x f(x)$$

subject to

$$g_i(x) \leq a_i$$

$$h_j(x) = 0$$

$$j \in \{1, \cdots, l\}$$

$$i \in \{1, \cdots, m\}$$

So, the domain of V is

$$a \in \mathbb{R}^m$$

for some

$$x \in X$$

$$g_i(x) \leq a_i$$

$$i \in \{1, \cdots, m\}$$

3. Interpretation of a_i and μ_i: Give this definition each coefficient μ_i is the rate at which the value of the function increases as a_i increases. Thus, if each a_i is interpreteed as a resource constraint the coefficients indicate how much increasing the resource increases the optimum value of f. This interpretation is especially important in economics and is used, for example, in utility maximization problems.

Generalizations

1. KKT Extensions – Fritz John Conditions: With an extra constant multiplier μ_0 which may be zero, in front of $\vec{\nabla} f(x^*)$ the KKT stationary conditions turn into

$$\mu_0 \vec{\nabla} f(x^*) + \sum_{i=1}^{m} \mu_i \vec{\nabla} g_i(x^*) + \sum_{j=1}^{l} \lambda_i \vec{\nabla} h_j(x^*)$$

which are called the Fritz John conditions.

2. KKT Class as FONC Support: The KKT conditions belong to the wider class of the First Order Necessary Conditions (FONC) which allow for non-smooth functions using sub-derivatives.

References

- Boyd, S., and L. van den Berghe (2004): *Convex Optimization* **Cambridge University Press**
- Eustaquio, R., E. Karas, and A. Ribeiro (2008): *Constraint Qualification under Non-linear Programming* Technical Report **Federal University of Parana**
- Hanson, M. A. (1999): Invexity and the Kuhn-Tucker Theorem *Journal of Mathematical Analysis and Applications* **236 (2)** 594-604
- Karush, W. (1939): *Minima of Functions of Several Variables with Inequalities as Side Constraints* Master's Thesis **University of Chicago** Chicago Illinois
- Kjeldsen, T. H. (2000): A Contextualized Analysis of the Kuhn-Tucker Theorem in Non-linear Programming: The Impact of World War II *Historia Mathematica* **27 (4)** 331-361
- Martin, D. H. (1985): The Essence of Invexity *Journal of Optimization Theory and Applications* **47 (1)** 65-76
- Ruszczynski, A. (2006): *Nonlinear Optimization* **Princeton University Press** Princeton NJ
- Wikipedia (2019): Karush Kuhn Tucker Conditions

Interior Point Method

Motivation, Background, and Literature Survey

1. Definition of Interior Point Methods: Interior Point Methods (also known as *barrier methods*) are a class of algorithms that solves linear and non-linear convex optimization problem (Interior Point Method (Wiki)).
3. John von Neumann's Early Approach: John von Neumann suggested an interior point method of linear programming that was neither a polynomial time method not an efficient method in practice (Dantzig and Thapa (2003)). In fact, it turned out to be slower in practice than the simplex method which is not a polynomial time method.
4. Karmarkar's Extension to the Simplex Method: Karmarkar (1984) developed a method for linear programming called the Karmarkar's algorithm which runs in provably polynomial time, and is also very efficient in practice. It enabled solutions to linear programming problems that were beyond the capabilities of the simplex method.
5. Traversal across the Feasible Region: Contrary to the Simplex method Karmarkar's algorithm reaches the best solution by traversing the interior of the feasible region. The method can be generalized to convex programming based on a self-concordant barrier function used to encode the convex set.
6. Transformation of the Convex Function: Any convex optimization problem can be transformed into minimizing (or maximizing) a linear function over a convex set by converting to an epigraph form (Boyd and van den Berghe (2004)). The idea of encoding the feasible set using a barrier and designing barrier methods was studied by Anthony V. Fiacco, Garth P. McCormick, and others in the early '60s. These ideas were mainly developed for general non-linear programming, but they were later abandoned due to the presence of more competitive methods for this class of problems (e.g., sequential quadratic programming).

7. Barriers to Encode Convex Set: Yuri Nesterov and Arkadi Nemirovskii came up with a special class of such barriers that can be used to encode any convex set. They provide guarantees that the number of iterations of the algorithm is bounded by a polynomial in the dimension and as well for the accuracy of the solution (Wright (2004)).
8. Interior Point Methods with Barriers: Karmarkar's breakthrough re-vitalized the study of interior point methods and barrier problems showing that it was possible to create an algorithm for linear programming characterized by polynomial time complexity, and, moreover, that was competitive with the simplex method. Already Khachiyan's ellipsoid method was a polynomial time algorithm; however, it was too slow to be of practical interest.
9. Interior Point Method Sub-classes: The class of primal dual path-following interior point methods is considered the most successful. Mehrotra's predictor-corrector algorithm provides the basis for most implementations of this class of methods (Mehrotra (1992), Potra and Wright (2000)).

Interior Point Methodology and Algorithm

1. Principle, Concept, and Approach Methodology: The main idea behind interior point methods is to iterate inside of the feasible set and progressively approach the boundary – if the minimum does lie on the boundary. This is done by transforming the original problem into a sequence of unconstrained optimization problems in which the objective function uses a *barrier function* that goes to ∞ at the boundaries of the feasible region. By reducing the strength of the barrier at each subsequent optimization the sequence of minima approach arbitrarily closely toward the minimum of the original problem.
2. Objective Function with Logarithmic Barriers: For the inequality constrained problem

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

such that

$$g_i(x) \leq 0, i = 1, \cdots, m$$

f is modified to obtain a *barrier function*

$$f_\alpha(x) = f(x) - \alpha \sum_{i=1}^{m} \log g_i(x)$$

3. Impact of the Barrier Strength Parameter: Since the logarithm goes to $-\infty$ as its argument approaches 0 the modified barrier function becomes arbitrarily large as x approaches the boundaries of the feasible region. The parameter α gives the barrier "strength". Its significance is that as α approaches 0 the optimum of $f_\alpha(x)$ approaches the optimum of $f(x)$. More precisely if f^* optimizes $f(x)$ and

$$x_\alpha{}^* \equiv \underset{x}{\arg min}\, f_\alpha(x)$$

then

$$\underset{\alpha \to 0}{Limit}\, f(x_\alpha{}^*) = f^*$$

4. Gradient of the Barrier Function: Newton's method is employed to find the minimum of $f_\alpha(x)$ The gradient is

$$\nabla f_\alpha(x) = \nabla f(x) - \alpha \sum_{i=1}^{m} \frac{\nabla g_i(x)}{g_i(x)}$$

5. Numerical Stability Close to the Boundary: One concern is that as $x_\alpha{}^*$ proceeds closer and closer to the boundary the numerical stability of the unconstrained optimization problem becomes worse and worse because the denominator approaches zero. Thus, it would be very challenging to apply the gradient descent to a small tolerance.
6. Use of KKT Type Multipliers: Therefore, another set of KKT multiplier like variables

$$\vec{\lambda} = (\lambda_1, \cdots, \lambda_m)$$

with

$$\lambda_i \equiv \frac{\alpha}{g_i(x)}$$

is introduced. So, in the $\left(x, \vec{\lambda}\right)$ space, one seeks the root of the equation

$$\nabla_x f_\alpha\left(x, \vec{\lambda}\right) = \nabla f(x) - \sum_{i=1}^{m} \lambda_i \nabla g_i(x) = 0$$

subject to the equality

$$\lambda_i g_i(x) = \alpha$$

for

$$i = 1, \cdots, m$$

7. Stability of the Modified Solution: This set of equations is far more numerically stable than

$$\nabla f_\alpha(x) = \nabla f(x) - \alpha \sum_{i=1}^{m} \frac{\nabla g_i(x)}{g_i(x)}$$

near the boundary. Note the similarity between these equations and the KKT equations. In fact, if α were 0 one gets the KKT equations except with the equalities stripped away.

8. Objective/Constraint Function - Partial Derivatives: To find a root $\left(x_\alpha{}^*, \vec{\lambda}_\alpha{}^*\right)$ where both of the functions are satisfied the Newton's method is applied. To do so one needs the partial derivatives of the above functions.

$$\nabla_{xx}{}^2 f_\alpha\left(x, \vec{\lambda}\right) = \nabla^2 f(x) - \sum_{i=1}^{m} \lambda_i \nabla^2 g_i(x)$$

$$\nabla_{\vec{\lambda}} \nabla_x f_\alpha(x) = -\sum_{i=1}^{m} \nabla g_i(x)$$

$$\nabla_x(\lambda_i g_i(x) - \alpha) = \lambda_i \nabla g_i(x)$$

$$\frac{\partial}{\partial \lambda_i}[\lambda_i g_i(x) - \alpha] = g_i(x)$$

for

$$i = 1, \cdots, m$$

9. Variate/Constraint Multipliers newton Increment: At the current iterate $\left(x_t, \vec{\lambda}_t\right)$ the Newton step $\left(\Delta x_t, \Delta\vec{\lambda}_t\right)$ is derived from the following system of equations.

$$\begin{bmatrix} \mathcal{H} & -\mathcal{G} \\ Diagonal(\vec{\lambda}_t) \cdot \mathcal{G}^T & Diagonal(\vec{g}) \end{bmatrix} \begin{bmatrix} \Delta x_t \\ \Delta \vec{\lambda}_t \end{bmatrix} = \begin{bmatrix} -\nabla f(x_t) + \mathcal{G} \cdot \vec{\lambda}_t \\ \alpha \mathbb{I} - Diagonal(\vec{g}) \cdot \vec{\lambda}_t \end{bmatrix}$$

where $\mathcal{H}$ denotes the Hessian

$$\nabla_{xx}{}^2 f_\alpha(x, \vec{\lambda}) = \nabla^2 f(x) - \sum_{i=1}^{m} \lambda_i \nabla^2 g_i(x)$$

evaluated at $(x_t, \vec{\lambda}_t)$, $\vec{g}$ denotes the $m \times 1$ vector of g_i's, $\mathcal{G}$ denotes the $n \times m$ matrix whose i^{th} column is $\nabla g_i(x_t)$, $Diagonal(\vec{v})$ produces a square matrix whose diagonal is the vector $\vec{v}$, and $\mathbb{I}$ denotes the $m \times 1$ vector of all 1's.

10. Variate Retention inside Feasible Region: Solving this system of equations provides a search direction that is then update via line search to avoid divergence. Note that to keep x drifting out of the feasible set one must enforce for all i the condition

$$g_i(x) \geq 0$$

or equivalently

$$\lambda_i \geq 0$$

during the line search.

11. Progressive Reduction of the Barrier Strength: Once the unconstrained search has converged α may be reduced (e.g., by multiplication by a small number) and the optimization can begin again. There is a trade-off in the convergence thresholds for each unconstrained search; too small and the algorithm must perform a lot of work even when α is high; but too large and the sequence of unconstrained optima does not converge quickly. A more balanced approach is to choose the threshold proportional to α.

12. Application to Non-Convex Functions: The formulation above did not explicitly require that the problem be convex. Interior point methods can certainly be used in general non-convex problems, but like any local optimization problem they are not guaranteed to a minimum. Furthermore, computing the Hessian matrix is quite often expensive.
13. Use in Linear/Quadratic Problems: They do, however, work quiet well in convex problems. For example, in quadratic programming the Hessian is constant, and in linear programming the Hessian is zero.
14. Initialization at a Feasible Point: The interior point method must be initialized at an interior point, or else the barrier function is undefined. To find the initial feasible point x the following optimization may be used.

$$\underset{x \in \mathbb{R}^n, s_1, \cdots, s_m}{max} \sum_{i=1}^{m} s_i$$

such that

$$g_i(x) - s_i \geq 0, i = 1, \cdots, ms_i \geq 0$$

If the problem is feasible then the optimal s_i will all be 0. It is easy to find an initial set of s_i for any given x simply by setting

$$s_i = \min(g_i(x), 0)$$

References

- Boyd, S., and L. van den Berghe (2004): *Convex Optimization* **Cambridge University Press**

- Dantzig, G. B., and M. N. Thapa (2003): *Linear Programming 2 – Theory and Extensions* **Springer-Verlag**
- Karmarkar, N. (1984): A New Polynomial-Time Algorithm for Linear Programming *Combinatorica* **4 (4)** 373-395
- Mehrotra, S. (1992): On the Implementation of the Primal-Dual Interior Point Method *SIAM Journal on Optimization* **2 (4)** 575-601
- Potra, F. A., and S. J. Wright (2000): Interior Point Methods *Journal of Computational and Applied Mathematics* **124 (1-2)** 281-302
- Wikipedia (2019): Interior Point Method
- Wright, M. H. (2004): The Interior-Point Revolution in Optimization; History, Recent Developments, and Lasting Consequences *Bulletin of the American Mathematical Society* **42 (1)** 39-56

Portfolio Selection with Cardinality and Bound Constraints

Synopsis

1. Portfolio Selection Problem – Formulation/Solution: Tadonki and Vial (2004) formulate and solve the portfolio selection problem with transaction costs and bound constraints on both the number of selected assets and the range of the corresponding investments.
2. Objective Function Risk/Return Tradeoff: The problem is to find a set of assets and the corresponding investment levels that suit the trade-off between risk and reward.
3. Combinatorial Mixed Integer Quadratic Problem: The underlying model results in a mixed integer quadratic problem. Tadonki and Vial (2004) show that the bound constraints on the investment yields a difficult combinatorial problem.
4. Feasibility Handling through Virtual Variable: To overcome this feasibility problem. Tadonki and Vial (2004) add a virtual variable that will remain selected if and only if the problem is infeasible.
5. Optimal Solution Greedy Heuristic Approximation: They propose a powerful greedy heuristic that provides a good approximation of the optimal solution in an affordable computation time.
6. Branch and Bound Global Solution: The algorithm for finding a global solution is a standard branch and bound processing.
7. Combining Heuristics with Cutting Planes: Tadonki and Vial (2004) use a cutting plane routine to solve the generic problem, and the heuristic to find potential upper bounds.
8. Test Runs on Benchmark Problems: Experimental results on a set of benchmark problems shows that the algorithm is quite efficient.

Introduction

1. Mean-Variance Portfolio Selection Formulation: The classical approach to the portfolio selection problem (Constantinides and Malliaris (1995)) is the Markowitz mean-variance formulation (Markowitz (1952, 1959)). The mean and the variance of a portfolio's return represent the benefit and the risk associated with the investment.
2. Standard Quadratic Convex Programming Problem: The approach boils down to a simple convex quadratic programming problem that is easily solved by standardized toolkits (CPLEX (2019), MOSEK (2019)). Various other aspects of the problem have also been studied by a number of authors (Mossin (1968), Samuelson (1969), Hakansson (1971), Zariphopoulou (1992), Li and Ng (2000), Zhou and Li (2000)).
3. Extensions to the Standard Framework: The problem can be considered with special properties; bounds on the total investment, fixed amount of investment, pre-determined set of assets, and lower/upper bounds on risk/rewards.
4. Accommodation of Discrete Portfolio Selection: In addition, in order to capture the realism of portfolio selection, a number of discrete constraints have been considered by different authors (Jobst, Horniman, Lucas, and Mitra (2001)), in particular, the *cardinality constraint*, which enables to take into account the expected number of assets in the portfolio.
5. Bounds on the Number of Assets: Jobst, Horniman, Lucas, and Mitra (2001) have studied the case where the number of assets to be selected is fixed. This chapter considers an upper bound instead.
6. Approaches for Handling Cardinality Constraints: For investors, the cardinality constraint is important for monitoring and control purposes. Chang, Meade, Beasley, and Sharaiha (1999) have proposed heuristics – genetic algorithm, tabu search, and simulated annealing – to solve for the cardinality constraint.
7. Lower Bounds on Invested Assets: Investors also use the so-called *buy-in thresholds*, which specifies the minimum investment level, and therefore eliminates small trades in the selection.

8. Upper Bounds on Invested Assets: This chapter also considers *buy-in limit*, which specifies the maximum investment level on an asset (Bienstock (1996), Lee and Mitchell (1997)). It is obvious that bounding the investments has an impact on the number of selected assets.
9. The Round-Lot Investment Constraint: Another interesting constraint is the round-lot constraint, which are discrete number is the assets taken as the basic units of the selection. The size of the portfolio is the integer linear combination of the round-lots. The feasibility problem coming from the round-lot constraints has been established to be NP-complete (Mansini and Speranza (1999)).
10. Incorporating Fixed Transaction Cost Constraint: This chapter extends the base mean-variance portfolio model by considering three special constraints. First, a transaction cost is assumed on each item. Prior treatments of these costs include Adcock and Meade (1994) and Young (1998).
11. Total Number of Assets Bound: Second, a bound on the total number of assets in the portfolio is imposed.
12. Lower and Upper Asset Bounds: Third, a lower and an upper limit is considered on the total number of items in the portfolio.
13. Mixed Integer Quadratic Programming Problem: The resulting mathematical problem is a quadratic mixed integer program.
14. Feasibility Handling via Virtual Barrier: The restriction on the investment levels yields a feasibility problem, which is another issue to consider. The standard approach to overcome this problem is to add a free virtual with a NULL reward and a high transaction cost.
15. Heuristics Approach for the Solution: Tadonki and Vial (2004) develop a heuristic that gives a good approximation of the problem in its generic form. The heuristic is used at different level trees of the branch-and-bound process to obtain upper bounds.
16. Analytic Center Cutting Plane Solver: The node problem is solved using an analytic center cutting planes solver named PROXACCPM (Gondzio, du Merle, Sarkissian, and Vial (1996)).

17. Organization of the Chapter: The chapter is organized as follows. The first section presents the formulation of the problem as it is considered in this treatment. This is followed by a complexity analysis of the problem.
18. Bender Decomposition and Greedy Heuristic: The Bender decomposition approach is presented in the next section, followed by the Tadonki and Vial (2004) greedy heuristic.
19. Cutting Plane Algorithm and Performance: The next section describes the cutting plane algorithm and its performance on the generic problem.
20. Chvatal-Gomory Cuts and Variants: This is followed by a presentation of the Chvatal-Gomory cuts and some variants.
21. Branching Rule and Node Selection: The penultimate section describes the branching rule and the node selection used in Tadonki and Vial (2004).
22. Discussion of Results and Conclusions: Computational Results are then presented, and the chapter is concluded in the final section.

Problem Formulation

1. Covariance Matrix and the Returns Vector: Let

$$V \in \mathbb{R}^{n \times n}$$

be the variance-covariance matrix among the n assets, and let

$$r \in \mathbb{R}^{n}$$

be the vector of the expected returns of the assets.

2. Asset Weights as the Decision Vector: The decision variable

$$x \in \mathbb{R}^{n}$$

respresents the fractional share of each asset in the portfolio. By definition, x belongs to the simplex, that is,

$$e^T x = 1$$

where

$$e = (1, \cdots, 1)$$

and

$$x \geq 0$$

3. Lower and Upper Bounds Vector: The lower and the upper bounds on the investment are given by the positive vectors

$$d, f \in \mathbb{R}^n$$

$$0 \leq d$$

$$f \leq 1$$

D and F denote the diagonal matrix with the main diagonal d and f, repectively.

4. Number of Assets Bound and the Transaction Cost: Finally, the fixed relative transaction cost is given by the positive vector

$$h \in \mathbb{R}^n$$

and

$$p \leq n$$

is a positive integer to bound the number of assets in the portfolio.

5. Objective Function and Constrains: To complete the formulation of the problem, the binary variables

$$y \in \{0,1\}^n$$

are introduced. The problem is:

$$\min_{x,y} \frac{1}{2} x^T V x - \mu r^T x + h^T y$$

$$e^T x = 1$$

$$Dy \leq x \leq Fy$$

$$e^T y \leq p$$

$$y \in \{0,1\}^n$$

6. Solving for the Optimal Allocation: The purpose of this chapter is to provide an efficient routine to solve the portfolio allocation problem based on the above formulation setup. The focus will be on a *Bender decomposition* approach together with a branch and bound technique.

Analysis of the Problem

1. Definition of the Optimization Parameters: For a given non-NULL vector

$$x \in \mathbb{R}^n$$

and two vectors

$$u, v \in \mathbb{R}^n$$

such that

$$u \leq v$$

$\varphi(x)$ - respectively $\psi(x, u, v)$ - is defined as the set of indices where the corresponding component value of x is non-NULL, i.e., x is between that of u and v:

$$\varphi(x) = \{i \in (1, \cdots, n)\ s.t.\ x_i \neq 0\}$$

$$\psi(x, u, v) = \{i \in (1, \cdots, n)\ s.t.\ u_i \leq x_i \leq v_i\}$$

2. Feasible Asset Subset - Lemma Statement: Given two vectors

$$u, v \in \mathbb{R}^n$$

and an integer

$$1 \leq p$$

there is a vector

$$x \in \mathbb{R}^n$$

such that

$$e^T x = 1$$

$$|\psi(x,u,v)| \leq p$$

$$\psi(x,u,v) = \varphi(x)$$

if and only if there is a subset

$$I \subset (1,\cdots,n)$$

with

$$|I| \leq p$$

such that

$$\sum_{i \in I} u_i \leq 1 \leq \sum_{i \in I} v_i$$

3. Feasible Asset subset Lemma Proof: Let

$$x \in \mathbb{R}^n$$

satisfying

$$e^T x = 1$$

$$|\psi(x,u,v)| \leq p$$

$$\psi(x,u,v) = \varphi(x)$$

It is easy to see that for

$$I = \varphi(x)$$

the relation

$$\sum_{i \in I} u_i \leq 1 \leq \sum_{i \in I} v_i$$

holds.

4. Proof of the Inverse Lemma: Let

$$I \subset (1, \cdots, n)$$

with

$$|I| \leq p$$

such that the relation

$$\sum_{i \in I} u_i \leq 1 \leq \sum_{i \in I} v_i$$

holds. The function τ defined by

$$\begin{matrix}\mathbb{R}^n \cap [u,v] \to \\ x \end{matrix} \begin{matrix}\mathbb{R}^n \\ \to \tau(x) = \sum_{i \in I} x_i\end{matrix}$$

is continuous and contains the value 1 inside its range, thus there is a vector

$$w \in [u,v]$$

such that

$$\tau(w) = 1$$

5. Recovery of x from w: The required vector x is then obtained as follows:

$$x_i = \begin{cases} w_i : i \in I \\ 0 : i \notin I \end{cases}$$

6. Condition for Portfolio Selection Feasibility: The above lemma provides a condition for the feasibility of the portfolio selection problem. However, the underlying combinatorial problem is NP-hard. This can be stated as follows.
7. Optimal Vector Determination Complexity - Inputs: An integer n, another integer p,

$$1 \leq p \leq n$$

and two vectors

$$u, v \in \mathbb{R}^n$$

$$0 < u$$

$$v < 1$$

such that

$$u \leq v$$

8. Optimal Vector Determination Complexity - Problem: The problem statement is: Is there a subset

$$I \subset (1, \cdots, n)$$

with

$$|I| \leq p$$

such that

$$\sum_{i \in I} u_i \leq 1 \leq \sum_{i \in I} v_i$$

9. NP-Complete Nature of the Subset Problem: To show that the problem is NP-hard, simply consider the case where

$$u = v$$

Here one obtains an instance which is equivalent to the *subset sum problem*, which is known to be NP-complete (Garey and Johnson (1979)).

10. Case where the Original Problem is Feasible: Consequently, one adds an artificial variable with any investment restriction, and which does not provide any profit. This ensures that the problem continues to be feasible.

11. Case where the Original Problem is Infeasible: However, of the considered case $u = v$ is infeasible, one seeks an exhaustive – and hence exponential – exploration process ending with the artificial point as the best solution found.
12. Problem Re-formulation with Augmented Variables: Technically, if V, μ, h, d and f are original parameters, the problem may be re-formulated using the following augmented variables: $\begin{pmatrix} V & 0 \\ 0 & \max(diagonal\, V) \end{pmatrix}$, $\begin{pmatrix} r \\ 0 \end{pmatrix}$, $\begin{pmatrix} h \\ \max h \end{pmatrix}$, $\begin{pmatrix} d \\ 0 \end{pmatrix}$, and $\begin{pmatrix} f \\ 1 \end{pmatrix}$. The rest of the chapter assumes that there is at least one variable with restriction on its investment.

Bender's Decomposition

1. Alternate Route to Computing Bounds: Bender's decomposition offers an alternative route to computing the bounds.
2. Convex Quadratic Program without Cardinality: Let $Q(y)$ be the value of the following convex quadratic program:

$$Q(y) = \underset{x,y}{min}\ \frac{1}{2}x^T Vx - \mu r^T x + h^T y$$

   The above quadratic program can be easily computed using standard approaches. Moreover, the dual variables obtained from the computation of $Q(y)$ can be used to provide a valid sub-gradient at y, as will be seen down.
3. Convex Quadratic Program with Cardinality: The above problem is difficult in two respects. First, the component $Q(y)$ is defined implicitly. It can only be approximated by the supporting hyper-planes, but this introduces a continuous variable

$$\varsigma \geq Q(y)$$

4. Complexity of Mixed-Integer Programming: Secondly, the outer approximation obtained by adding convexity cuts – the supporting hyper-planes – yields a mixed integer programming problem.
5. Relaxation of Binary to Linear Constraints: A possible course of action is to relax the binary constraints and solve the linear problem. If the solution of the relaxation is boolean in y, then the solution is globally optimal. However, we may not expect that it will be so in general.
6. Branch and Bound Heuristic Solutions: Consequently, exploration techniques like *branch* and *bound* should be considered. For this purpose, the next section provides a heuristic that results in a feasible approximation to the problem.

A Greedy Heuristic

1. Cardinality Constrained Quadratic Convex Problem: A key goal here is to provide a heuristic which results in a feasible solution to

$$\min_{x,y} \left(\frac{1}{2} x^T V x - \mu r^T x + h^T y \mid e^T x = 1, Dy \leq x \leq Fy, e^T y \leq p, y \in \{0,1\}^n \right)$$

2. Upper Constrained Quadratic Convex Problem: Let x be the solution to

$$\min_{x} \left(\frac{1}{2} x^T V x - \mu r^T x + p h^T x \mid e^T x = 1, 0 \leq x \leq f \right)$$

3. Lower Bound Adhering/Violating Allocations: Let J, K be the two sets of

$$I \subset (1, \cdots, n)$$

defined by

$$J = \{i \in I : x_i < d_i\}$$

$$K = \{i \in I : d_i \leq x_i \leq f_i\}$$

The following two exclusive cases are considered.

4. Target Optimal Allocation Cardinality Match: Considering the case

$$|K| \leq p$$

one defines $\tilde{x}$ by

$$\tilde{x}_i = \begin{cases} i \in K : x_i \\ i \in J : 0 \end{cases}$$

5. Target Optimal Allocation Cardinality Excess: Consider the case

$$|K| > p$$

Let

$$L \subset K$$

such that

$$|L| = p$$

and

$$\min_{i \in L} x_i \geq \max_{i \notin L} x_i$$

The subset L can be obtained by a s straight forward greedy algorithm which selects the index of the maximum value as a new candidate.

6. Candidate Subset Selection by Greedy Algorithm: One defines $\tilde{x}$ by

$$\tilde{x}_i = \begin{cases} i \in K : x_i \\ i \in J : 0 \end{cases}$$

Also define

$$\Delta = \sum_{i \in I} (x_i - \tilde{x}_i)$$

7. Allocation Gap Quadratic Convex Problem: If

$$\Delta > 0$$

the following problem is considered:

$$\min_{z} \left(\frac{1}{2} z^T V z - \mu r^T z + p h^T z \mid e^T z = \Delta, 0 \leq z \leq f - \tilde{x} \right)$$

8. Iterative Reduction of the Departure Metric: If z is a solution of the above optimization, then one considers $\tilde{x} + z$ as the approximation of

$$\min_{x,y} \left(\frac{1}{2} x^T V x - \mu r^T x + h^T y \mid e^T x = 1, Dy \leq x \leq Fy, e^T y \leq p, y \in \{0,1\}^n \right)$$

One then applies the process repeatedly until

$$\Delta = 0$$

or there is no more improvement. In the latter case, one finds the component

$$i \in K$$

(or L) such that

$$d_j \leq \tilde{x}_j + \Delta \leq f_j$$

which gives the minimum value for

$$\min\left(\frac{1}{2}[\tilde{x}_j + \Delta]^T V[\tilde{x}_j + \Delta] - \mu r^T[\tilde{x}_j + \Delta] + ph^T[\tilde{x}_j + \Delta]\right)$$

The solution is then obtained by $\tilde{x}_j$ to $\tilde{x}_j + \Delta$.

9. Allocation Fraction and Asset Index: At the end of the iterative process the solution x is retrieved from $\tilde{x}$. After computing the vector $\tilde{x}$ as described above, y is defined by

$$y_j = \begin{cases} 0 & \tilde{x}_j = 0 \\ 1 & \tilde{x}_j \neq 0 \end{cases}$$

Thus, it is clear that $(\tilde{x}, y)$ is a feasible approximation to

$$\min_{x,y}\left(\frac{1}{2}x^T V x - \mu r^T x + h^T y \mid e^T x = 1, Dy \leq x \leq Fy, e^T y \leq p, y \in \{0,1\}^n\right)$$

10. Applying the Heuristic to Fixed Bounds: The heuristic can also be applied when some components have a fixed value. In that case, one just needs to set the corresponding components of d and f to the same fixed value.
11. Performance of the Heuristic Greedy Algorithm: The table below displays the performance of the heuristic with a 2.5 GHz processor. As can be seen, the running time is globally acceptable. Moreover, it appears that the performance is better with a

larger value of p. The intuitive reason for this is that selecting more items – the non-NULL component of x – imparts a more significant reduction of the gap between current solution and what can be expected at further steps.

12. Heuristic Greedy Algorithm Performance Table: Source: Tadonki and Vial (2004)

n	p	Selected	Iterations	Time (seconds)
50	5	5	15	2.80
50	10	7	4	0.78
100	10	10	10	2.07
100	20	12	6	1.20
200	10	10	27	8.16
200	30	18	14	4.51
300	10	10	31	15.21
300	25	23	14	5.97
400	10	10	23	14.20
400	30	30	3	3.16
500	10	10	22	23.36
500	50	27	11	8.78

Cutting Planes Algorithm and PROXACCPM – Concept and Tool

1. NDO Oracle and Cutting Plane: The non-differentiable optimization (NDO) is based on the notions of *Oracle* and *Cutting Plane* (Kelley (1960)).
2. Hyperplane Supports for Polyhedral Domains: The main idea is to use hyperplanes that are linear supports of the objective function, to build a workable, polyhedral approximation of the function. This approximation is generally refined to approach an optimal solution.

3. Localization Set and the Query Point: The domain of the polyhedral approximation is called the *localization set*. The mechanism that supports the linear supports of the objective function at the so-called *query points* is fully problem dependent.
4. Generation of the Cutting Plane: In an NDO algorithm, the previous mechanism is assimilated into a *black box* and is usually named the *Oracle*. The Oracle output is one or more *cutting planes,* and the problem that is built around the generated cutting planes is called the *master problem*.
5. Steps involved in the Cutting Plane Algorithm: The cutting plane algorithm can be summarized as follows:
    a. Get an initial upper bound and a lower bound for the optimal solution.
    b. Find a point in the current localization set and the associated lower bound.
    c. Query the Oracle at the generated point. The Oracle generates one or more cuts for both the objective function and the sub-gradient. If the point is feasible, then the corresponding function value is used as a potential upper bound for the optimal value. If the point is infeasible, the cut is added as a domain constraint.
    d. Update the upper and the lower bounds and add the new cuts.
6. The Iteration Convergence/Termination Criterion: These steps are repeated until the gap between the upper bound and the lower bound falls below a prescribed optimality tolerance.

PROXACCPM Performance on the Generic Problem

1. PROXACCPM – Analytic Cutting Plane Implementation: PROXACCPM (Goffin, Haurie, and Vial (1992), Gondzio, du Merle, Sarkissian, and Vial (1996)) is a convex optimization solver based on the analytic center cutting plane algorithm. The notion of analytic center (Sonnevend (1986)) is the key of the query point selection procedure.

2. Convex Optimization under Cardinality Constraint: Recall that the problem solved is the following relaxation:

$$\min_{x,y} \left(\frac{1}{2} x^T V x - \mu r^T x + h^T y \mid e^T x = 1, Dy \le x \le Fy, e^T y \le p, 0 \le y \le 1 \right)$$

3. PROXACCPM Objective Function and Constraint Set: The PROXACCPM objective is defined by

$$\mathcal{F}(y) = \min_{x,y} \left(\frac{1}{2} x^T V x - \mu r^T x + h^T y \mid e^T x = 1, Dy \le x \le Fy \right)$$

4. Query Value and the Dual: The value of $\mathcal{F}$ is obtained by solving the corresponding quadratic programming problem, which also adds dual variables λ_l and λ_u for the inequalities

$$x \ge Dy$$

and

$$x \le Fy$$

respectively.

5. Sub-gradient at the Point y: One sub-gradient of $\mathcal{F}$ at the point y is given by

$$\mathcal{G}(y) = h + \lambda_l \otimes d - \lambda_u \otimes f$$

where $\otimes$ is a point-to-point product of two vectors.

6. Performance on Sub-problem Solving: Source: Tadonki and Vial (2004).

n	*p*	Iterations	Total (seconds)	Method (seconds)	Oracle (seconds)
50	5	56	8.54	0.63	7.81
50	10	58	8.76	0.67	8.03
100	10	88	15.55	1.28	14.09
100	20	90	15.65	1.30	14.20
200	10	93	27.04	1.71	25.12
200	30	88	25.04	1.56	23.25
300	10	86	39.24	1.93	37.10
300	25	89	40.26	2.11	37.99
400	10	96	71.42	2.93	68.07
400	30	90	65.56	2.50	62.74
500	10	89	98.52	3.61	94.52
500	50	144	165.98	8.75	156.45

References

- Adcock, C. J., and N. Meade (1994): A Simple Algorithm to incorporate Transaction Costs in Quadratic Optimization *European Journal of Operations Research* **79 (1)** 85-94
- Bienstock, D. (1996): Computational Study of a Family of Mixed-Integer Quadratic Programming Problems *Mathematical Programming* **74 (2)** 121-140
- Constantinides, G. M., and A. G. Malliaris (1995): Portfolio Theory *Handbook in Operations Research and Management Science* **9** 1-30
- Chang, T., J., N. Meade, J. E. Beasley, and Y. M. Sharaiha (2000): Heuristics for Cardinality Constrained Portfolio Optimization *Computers and Operations Research* **27 (13)** 1271-1302
- CPLEX (2019): https://en.wikipedia.org/wiki/CPLEX

- Garey, M. R., and D. S. Johnson (1979): *Computer and Intractability – A Guide to the Theory of NP-Completeness* **W H Freeman** New York
- Goffin, J. L. Haurie, A. Laurie, and J. P. Vial (1992): Decomposition and Non-differentiable Optimization with a Projective Algorithm *Management Science* **38 (2)** 284-302
- Gondzio, J., O. du Merle, R. Sarkissian, and J. P. Vial (1996): ACCPM – A library for Convex Optimization Based on Analytic Center Cutting Plane Method *European Journal of Operation Research* **94 (1)** 206-211
- Hakansson, N. H. (1971): Multi-period Mean-variance Analysis: Toward a General Theory of Portfolio Choice *Journal of Finance* **26 (4)** 857-884
- Jobst, N. J., M. D. Horniman, C. A. Lucas, and G. Mitra (2001): Computational Aspects of Alternative Portfolio Selection Models in the Presence of Discrete Asset Choice Constraints *Quantitative Finance* **1 (5)** 1-13
- Kelley, J. E. (1960): The Cutting Plane Method for Solving Convex Problems *Journal for the Society of the Industrial and Applied Mathematics* **8 (4)** 703-712
- Lee, E. K., and J. E. Mitchell (1997): Computational Experience of an Interior-Point SQP Algorithm in a Parallel Branch-and-Bound Framework, in: *High Performance Optimization, Applied Optimization* (*Frenk, H., K. Roos, T. Terlaky, S. Zhang editors*) **33 Springer** Boston, MA
- Li, D., and W. L. Ng (2000): Optimal Dynamic Portfolio Selection: Multi-period Mean-variance Formulation *Mathematical Finance* **10 (3)** 387-406
- Mansini, R., and M. G. Speranza (1999): Heuristic Algorithms for the Portfolio Selection Problem with Minimum Transaction Lots *European Journal of Operations Research* **114 (2)** 219-233
- Markowitz, H. (1952): Portfolio Selection *Journal of Finance* **7 (1)** 77-91
- Markowitz, H. (1959): *Portfolio Selection: Efficient Diversification of Investments* **John Wiley and Sons** New York
- MOSEK (2019): https://en.wikipedia.org/wiki/MOSEK
- Mossin, C. (1968): Optimal Multi-period Portfolio Policies *Journal of Business* **41 (2)** 215-229

- Samuelson, P. A. (1969): Life-time Portfolio Selection by Dynamic Stochastic Programming *Review of Economics and Statistics* **51 (3)** 239-246
- Sonnevend, G. (1986): An *Analytic Center* for Polyhedrons and New Classes of Global Algorithms for Linear (Smooth, Convex) Programming, in: *System Modeling and Optimization, Lecture Notes in Control and Information Sciences, Prekopa, A., B. Szelezsaan, and B. Strazicky editors* **84 Springer** Berlin
- Tadonki, C., and J. P. Vial (2004): Portfolio Selection with Cardinality and Bound Constraints
- Young, M. R. (1998): A Minimax Portfolio Selection Rule with Linear Programming Solution *Management Science* **44 (5)** 673-683
- Zariphopoulou, T. (1992): Investment-Consumption Models with Transaction fees and Markov-Chain Parameters *SIAM Journal of Control and Optimization* **30 (3)** 613-636
- Zhou, X. Y., and D. Li (2000): Continuous-Time Mean-variance Portfolio Selection: A Stochastic LQ Framework *Applied Mathematics and Optimization* **42 (1)** 19-33