Optimization

Econ 5170

Computational Methods in Economics

2022-2023 Spring

Optimization Problems

- Canonical problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad g(x) & = 0 \\ h(x) & \leq 0 \end{aligned}$$

  - $f : \mathbb{R}^n \to \mathbb{R}$ is the *objective function*
  - $g : \mathbb{R}^n \to \mathbb{R}^m$ is the vector of m *equality constraints*
  - h : $\mathbb{R}^n \to \mathbb{R}^\ell$ is the vector of l *inequality constraints.*
- Examples:
  - Maximization of consumer utility subject to a budget constraint
  - Optimal incentive contracts
  - Portfolio optimization
  - Life-cycle consumption
- Assumptions
  - Always assume f, g, and h are continuous
  - Usually assume f, g, and h are C^1
  - Often assume f, g, and h are C^3

- Topics
  - Unconstrained optimization
    - Unconstrained optimization problems occur naturally - maximum likelihood, minimize moment criteria
    - They are also the foundation of constrained optimization methods
  - Constrained optimization
    - Optimal life-cycle problems with budget constraint
    - Maximize profit given production constraints
    - Optimal taxation given incentive compatibility constraints
    - Econometric estimation of structural models

One-dimensional Unconstrained Minimization: Newton's Method

$$\min_{x \in \mathbb{R}} \quad f(x),$$

- Assume $f(x)$ is C^2 functions $f(x)$
  - At a point a, the quadratic polynomial, $p(x)$

$$p(x) \equiv f(a) + f'(a)(x-a) + \frac{f''(a)}{2}(x-a)^2$$

    is the second-order approximation of $f(x)$ at a
  - Approximately minimize f by minimizing $p(x)$ and solve

$$p'(x) = f'(a) + f''(a)(x-a) = 0$$

  - If $f''(a) > 0$, then p is convex, and $x_m = a - f'(a)/f''(a)$.

Newton's Method

Algorithm: Newton's Method in $\mathbb{R}^1$
Initialize. Choose initial guess x_0 and stopping parameters δ, $\epsilon > 0$.
Step 1. $x_{k+1} = x_k - f'(x_k)/f''(x_k)$.
Step 2. If $|x_k - x_{k+1}| < \epsilon(1 + |x_k|)$ and $|f'(x_k)| < \delta$, STOP and report success; else go to step 1.

- Properties:
  - Newton's method finds critical points, that is, solutions to $f'(x) = 0$, not min or max.
  - If x_n converges to x^*, must check $f''(x^*)$ to check if min or max
  - Only find local extrema.
- Good news: convergence is locally quadratic.

Theorem 1 *Suppose that $f(x)$ is minimized at x^*, C^3 in a neighborhood of x^*, and that $f''(x^*) \neq 0$. Then there is some $\epsilon > 0$ such that if $|x_0 - x^*| < \epsilon$, then the x_n sequence converges quadratically to x^*; in particular,*

$$\lim_{n\to\infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|^2} = \frac{1}{2}\left|\frac{f'''(x^*)}{f''(x^*)}\right|$$

is the quadratic rate of convergence.

- Consumer problem example:
  - Consumer has \$1; price of x is \$2, price of y is \$3, utility function is $x^{1/2} + 2y^{1/2}$.
  - If θ is amount spent on x then we have

$$\max_{\theta} \left(\frac{\theta}{2}\right)^{1/2} + 2\left(\frac{1-\theta}{3}\right)^{1/2}$$

    Solution $\theta^* = 3/11 = 0.272727$
  - If θ_0=1/2, Newton iteration is

$$0.5, 0.2595917942, 0.2724249335, 0.2727271048, 0.2727272727$$

    and magnitude of the errors are

$$2.3(-1), 1.3(-2), 3.1(-4), 1.7(-7), 4.8(-14)$$

- Problems with Newton's method
  - May not converge if initial guess is too far away from solution.
  - $f''(x)$ may be difficult to calculate.

Multidimensional Unconstrained Optimization: Comparison Methods

- Grid Search
  - Pick a finite set of points, X; for example, a Cartesian grid:

$$V = \{v_i \mid i = 1, \ldots, n\}$$
$$X = \{x \in \mathbb{R}^n \mid \forall i, x_i \in V\}$$

  - Compute $f(x)$, $x \in \mathbf{X}$, and locate max
  - Grid search is often the first method to use.
    - Only involves function evaluations
    - It is embarrassingly parallelizable
    - It should get you a good initial guess
  - A good initial guess is not critical for grid search, but is for all good algorithms
  - Grid search is slooooooow, so you should always switch to something better
- General lesson: start with a reliable but slow method to find good initial guess for a faster method

- Polytope Methods (a.k.a. Nelder-Mead, simplex, “amoeba”) — Matlab code: fminsearch

Algorithm 4.3 Polytope Algorithm

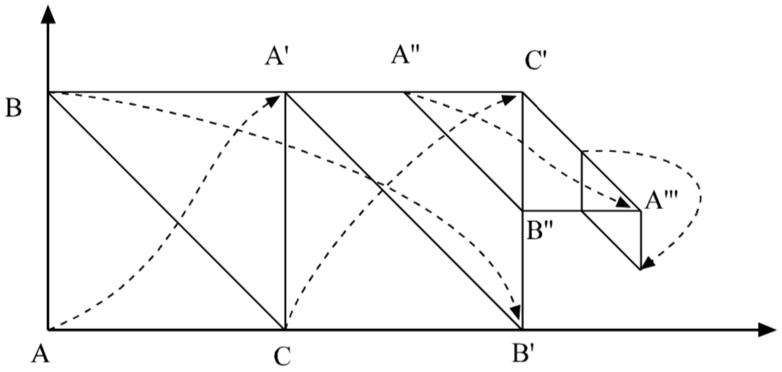
Initialize. Choose the stopping rule parameter ϵ. Choose an initial simplex $\left\{x^1, x^2, \cdots, x^{n+1}\right\}$

Step 1. Reorder vertices so $f\left(x^i\right) \geq f\left(x^{i+1}\right), i = 1, \cdots, n$.

Step 2. Look for least i, s.t. $f\left(x^i\right) > f\left(y^i\right)$ where y^i is reflection of x^i. If such an i exists, set $x^i = y^i$, and go to step 1. Otherwise, go to step 3 .

Step 3. Stopping rule: If the width of the current simplex is less than ϵ, STOP. Otherwise, go to step 4.

Step 4. Shrink simplex: For $i = 1, 2, \cdots, n$ set $x^i = \frac{1}{2}\left(x^i + x^{n+1}\right)$, and go to step 1.



The intuition is to replace the worst point with a point reflected through the centroid of the remaining n points.

Multidimensional Optimization: Newton's Method

- Idea: Given x^k, compute local quadratic approximation, $p(x)$, of $f(x)$ around x^k, and let x^{k+1} be max of $p(x)$

Algorithm 4.4 Newton's Method in $\mathbb{R}^n$

Initialize. Choose x^0 and stopping parameters δ and $\epsilon > 0$.
Step 1. Compute Hessian, $H\left(x^k\right)$, and gradient, $\nabla f\left(x^k\right)$, and solve $H\left(x^k\right) s^k = -\left(\nabla f\left(x^k\right)\right)^\top$ for the step s^k.
Step 2. $x^{k+1} = x^k + s^k$.
Step 3. If $\left\|x^k - x^{k+1}\right\| < \epsilon\left(1 + \left\|x^k\right\|\right)$ go to step 4, else go to step 1.
Step 4. If $\left\|\nabla f\left(x^{k+1}\right)\right\| < \delta\left(1 + \left|f\left(x^{k+1}\right)\right|\right)$, STOP and report success; else STOP and report convergence to nonoptimal point.

Multidimensional Optimization: Newton's Method

- Stopping rule: Don't be too fussy!
  - Good values for ε and δ are close to the square root of machine epsilon.
  - First use sloppy ε and δ, such as 10^{-3}.
  - Then reduce ε and δ until failure.
  - You can try to push them below square root of machine epsilon but you will probably not get too far.

Theorem 2 *Suppose that f(x) is C^3, minimized at x^*, and that $H(x^*)$ is nonsingular. Then there is some $\epsilon > 0$ such that if $\left\|x^0 - x^*\right\| < \epsilon$, then the sequence converges quadratically to x^*.*

- Problems with Newton's method:
  - May not converge
  - Computational demands may be excessive
    - need at least $\mathcal{O}(n^2)$ time to compute $H(x^k)$, perhaps more if one does not have efficient code for $H(x)$
    - need $\mathcal{O}(n^2)$ space for $H(x^k)$
    - need $\mathcal{O}(n^3)$ time to solve $H(x^k)s^k = -(\nabla f(x^k))^\top$ for s^k
  - May converge to local solution, not global solution
  - We now consider methods which address these problems.

Direction Set Methods

- Problem: may not converge, or go to wrong kind of extremum
- Solution: if we always move uphill, we will eventually get to a local maximum

Algorithm 4.5 Generic Direction Method

Initialize. Choose initial x^0 and stopping parameters δ and $\epsilon > 0$.
Step 1. Compute a search direction s^k.
Step 2. Solve $\lambda_k = \arg\min_\lambda f\left(x^k + \lambda s^k\right)$
Step 3. $x^{k+1} = x^k + \lambda_k s^k$.
Step 4. If $\left\|x^k - x^{k+1}\right\| < \epsilon\left(1 + \left\|x^k\right\|\right)$, go to step 5, else go to step 1.
Step 5. If $\left\|\nabla f\left(x^{k+1}\right)\right\| < \delta\left(1 + f\left(x^{k+1}\right)\right)$, STOP and report success; else STOP and report convergence to nonoptimal point.

Direction Set Methods

- Possible direction set methods
  - Steepest Descent: $s_k = \nabla f(x^k)$
  - Newton's Method with Line Search (Quasi-Newton Method): $H_k s^k = -(\nabla f(x^k))^\top$ — Matlab code: fminunc

Quasi-Newton Methods

- Problem: Hessians are expensive to compute
- Solution: Don't need true Hessians (see Carter, 1993), so approximate them

Generic Quasi-Newton Method

Initialize. Choose initial x^0, Hessian $H^0(I)$ and stopping parameters δ and $\epsilon > 0$

Step 1. Solve $H_k s^k = -\left(\nabla f\left(x^k\right)\right)^\top$ for the search direction s^k.

Step 2. Solve $\lambda_k = \arg\min_\lambda f\left(x^k + \lambda s^k\right)$

Step 3. $x^{k+1} = x^k + \lambda_k s^k$

Step 4. Compute H_{k+1} using $H_k, \nabla f\left(x^{k+1}\right), x^{k+1}, \nabla f\left(x^k\right)$, etc.

Step 5. If $\left\|x^k - x^{k+1}\right\| < \epsilon\left(1 + \left\|x^k\right\|\right)$, go to step 6; else go to step 1.

Step 6. If $\left\|\nabla f\left(x^{k+1}\right)\right\| < \delta\left|1 + f\left(x^{k+1}\right)\right|$, STOP and report success; else STOP and report convergence to nonoptimal point.

- Example: BFGS:

$$
\begin{aligned}
z_k &= x^{k+1} - x^k \\
y_k &= \left(\nabla f\left(x^{k+1}\right)\right)^\top - \left(\nabla f\left(x^k\right)\right)^\top \\
H_{k+1} &= H_k - \frac{H_k z_k z_k^\top H_k}{z_k^\top H_k z_k} + \frac{y_k y_k^\top}{y_k^\top z_k}
\end{aligned}
$$

  - Preserves positive definiteness
  - Uses only gradients that are already needed
- Note: The Hessian iterates H_k may not converge to true Hessian at solution, even if x_k converges to solution. Never use approximate Hessians to compute standard errors!

Example: A Dynamic Optimization Problem

- Life-cycle savings problem.
  - an individual lives for T periods
  - earns wages w_t in period $t, t = 1, \cdots, T$
  - consumes c_t in period t
  - earns interest on savings per period at rate r
  - define S_t to be end-of-period savings:

$$S_{t+1} = (1 + r)S_t + w_{t+1} - c_{t+1}$$

  - Set initial wealth: $S_0 = 0$
  - Utility function $\sum_{t=1}^{T} \beta^t u(c_t) + W(S_T)$
  - Substitute $c_t = S_{t-1}(1 + r) + w_t - S_t$
- Problem now has T choices:

$$\max_{S_t} \sum_{t=1}^{T} \beta^t u\left(S_{t-1}(1 + r) + w_t - S_t\right) + W\left(S_T\right)$$

- Newton's method looks impractical if T large. BUT
  - Hessian is tridiagonal (a sparse matrix)
    - The choice of S_t interacts only with the choices for S_{t-1} and S_{t+1}
    - Newton step is easy to compute.
    - The normal Hessian has size T^2
    - The tridiagonal matrix has size $3T$
  - Sparse Hessians are common in dynamic problems because time t variables interact only with time $t-1$ and time $t+1$ variables.
  - You must recognize this and implement Newton or quasi-Newton method with sparse Hessians.

Domain Problems

- Suppose $S_0 = 0$ and you want to solve

$$\max_{S_t} \sum_{t=1}^{T} \beta^t \log\left(S_{t-1}(1+r) + w_t - S_t\right) + W\left(S_T\right)$$

- Newton's method takes the guess S^k and computes a new guess S^{k+1}.
- Problem: S^{k+1} could imply consumption, $c_t = S_{t-1}(1+r) + w_t - S_t$, will be negative at some t causing computer to crash.
- A possible solution: Alter objective function
  - E.G.; replace u (c) = log c with, for some small $\varepsilon > 0$

$$\widetilde{u}(c) = \begin{cases} u(c), & c > \varepsilon \\ u(\varepsilon) + u'(\varepsilon)(c-\varepsilon) + u''(\varepsilon)(c-\varepsilon)^2/2, & c \leq \varepsilon \end{cases}$$

  - Maintains curvature
  - Equals real $u(c)$ on most of domain, which hopefully includes solution
  - Not as easy to apply to multivariate functions
- General solution: add constraints to keep this from happening.

Constrained Linear Optimization – Linear Programming

- Canonical linear programming problem is

$$\min_x a^\top x$$
$$\text{s.t. } Cx = b$$
$$x \geq 0$$

- $Dx \leq f$: use *slack variables*, s, and constraints $Dx + s = f, s \geq 0$.
- $Dx \geq f$: use $Dx - s = f, s \geq 0$.
- $x \geq d$: define $y = x - d$ and min over y

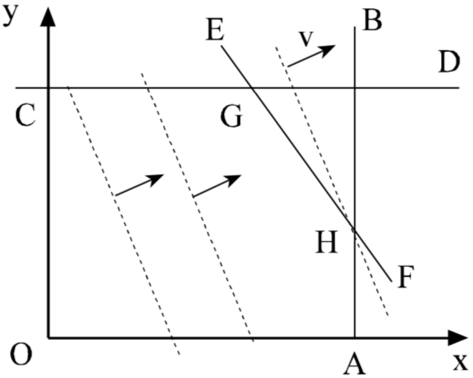
Linear Programming

- Basic method is the *simplex method*.
  - Initialization: Find some point on boundary of constraints, such as A.
  - Step 1: Note which constraints are active at A and points nearby. Find feasible directions and choose steepest descent direction.
  - Step 2: Follow that direction to next vertex on boundary, and go back to step 1.
  - Continue until no direction reduces the objective.
  - Stops in finite time since there are only a finite set of vertices.

For example:

$$\min_{x,y} -2x - y$$
$$\text{s.t. } x + y \leq 4, \quad x, y \geq 0$$
$$x \leq 3, \quad y \leq 2$$

Linear Programming



- Initialization: Find A.
- Step 1: two directions: from A: to B and to O, with B better.
- Step 2: go to H

Constrained Nonlinear Optimization

General problem:

$$\min_x f(x)$$
$$\text{s.t. } g(x) = 0$$
$$h(x) \leq 0$$

- $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$: objective function with n choices
  - $g : X \subseteq \mathbb{R}^n \to \mathbb{R}^m$: m equality constraints
  - $h : X \subseteq \mathbb{R}^n \to \mathbb{R}^\ell$: ℓ inequality constraints
  - f, g, and h are C^2 on X

Constrained Nonlinear Optimization

- Karush-Kuhn-Tucker (KKT) theorem: if there is a local minimum at x^* then there are multipliers $\lambda^* \in \mathbb{R}^m$ and $\mu^* \in \mathbb{R}^\ell$ such that x^* is a *stationary*, or critical, point of $\mathcal{L}$, the *Lagrangian*,

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^\top g(x) + \mu^\top h(x)$$

- First-order conditions, $\mathcal{L}_x(x^*, \lambda^*, \mu^*) = 0$, imply that (λ^*, μ^*, x^*) solves

$$\begin{aligned} cf_x + \lambda^\top g_x + \mu^\top h_x &= 0 \\ \mu_i h^i(x) &= 0, \quad i = 1, \cdots, \ell \\ g(x) &= 0 \\ h(x) &\leq 0 \\ \mu &\geq 0 \end{aligned}$$

Constrained Nonlinear Optimization

- The KKT conditions are

$$
\begin{aligned}
\nabla_x L\left(x^*, \lambda^*\right) = 0 \text{ i.e. } \nabla f\left(x^*\right) &= \sum_{i \in E \cup I} \lambda_i^* \nabla g_i\left(x^*\right) \\
g_i\left(x^*\right) &= 0, \forall i \in E \\
g_i\left(x^*\right) &\geq 0, \forall i \in I \\
\lambda_i^* &\geq 0, \forall i \in I \\
\lambda_i^* g_i\left(x^*\right) &= 0, \forall i \in E \cup I
\end{aligned}
$$

- At a solution, x, all equality constraints must hold.

Constrained Nonlinear Optimization

- Some inequality constraints will be active, that is, equal zero. For each solution x, define the active set of constraints

$$A(x) = E \cup \{i \in I \mid g_i(x) = 0\}$$

- Given x^* and $A(x^*)$, we say that the linear independence constraint qualification (LICQ) holds if the set of active constraint gradients $\{\nabla g_i(x^*) \mid i \in A(x^*)\}$ is linearly independent.
- If LICQ holds then the multipliers are unique; otherwise, they are called "unbounded."

A Kuhn-Tucker Approach

- Idea: try all possible Kuhn-Tucker systems and pick best
  - Let $\mathcal{J}$ be the set $\{1, 2, \cdots, \ell\}$.
  - For a subset $\mathcal{P} \subset \mathcal{J}$, define the P problem, corresponding to a combination of binding and nonbinding inequality constraints

$$\begin{aligned} g(x) &= 0 \\ h^i(x) &= 0, \quad i \in \mathcal{P} \\ \mu^i &= 0, \quad i \in \mathcal{J} - \mathcal{P} \\ f_x + \lambda^\top g_x + \mu^\top h_x &= 0 \end{aligned}$$

  - Solve (or attempt to do so) each $\mathcal{P}$-problem
  - Choose the best solution among those $\mathcal{P}$-problem with solutions consistent with all constraints.
- We can do better in general.

Penalty Function Approach

- Many constrained optimization methods use a *penalty function* approach:
  - Replace constrained problem with related unconstrained problem.
  - Permit anything, but make it "painful" to violate constraints.
- Penalty function: for canonical problem

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = a, \\ & h(x) \le b. \end{aligned}$$

construct the *penalty function* problem $F(x; P, a, b)$

$$\min_x f(x) + \frac{1}{2} P \left(\sum_i \left(g^i(x) - a_i\right)^2 + \sum_j \left(\max\left[0, h^j(x) - b_j\right]\right)^2 \right)$$

where $P > 0$ is the penalty parameter.

  - If P is "infinite" then the two problems are identical.
  - Hopefully, for large P, their solutions will be close.

Penalty Function Approach

- Problem: for large P, the Hessian of F, F_{xx}, is ill-conditioned at x away from the solution.
- Solution: solve a sequence of problems.
  - Solve $\min_x F(x; P_1, a, b)$ with a small choice of P_1 to get x^1.
  - Then execute the iteration

$$x^{k+1} \in \arg\min_x F(x; P_{k+1}, a, b)$$

    where we use x^k as initial guess in iteration $k+1$, and $F_{xx}(x^k; P_{k+1}, a, b)$ as the initial Hessian guess

Penalty Function Approach

- Simple example
  - Consumer buys good y (price is 1) and good z (price is 2) with income 5.
  - Utility is $u(y, z) = \sqrt{yz}$.
  - Optimal consumption problem is

$$\begin{array}{l} \max_{y,z} \sqrt{yz} \\ \text{ s.t. } \; y + 2z \leq 5. \end{array}$$

    with solution $(y^*, z^*) = (5/2,\, 5/4)$, $\lambda^* = 8^{-1/2}$.
  - Penalty function is

$$u(y, z) - \frac{1}{2} P(\max[0, y + 2z - 5])^2$$

Iterates are in Table 4.7 (stagnation due to finite precision)

Table 4.7

Penalty function method applied to (4.7.8)

k	P_k	$(y,z)-(y^*,z^*)$	Constraint violation	λ error
0	10	(8.8(-3), .015)	1.0(−1)	−5.9(−3)
1	10^2	(8.8(−4), 1.5(−3))	1.0(−2)	−5.5(−4)
2	10^3	(5.5(−5), 1.7(−4))	1.0(−3)	2.1(−2)
3	10^4	(−2.5(−4), 1.7(−4))	1.0(−4)	1.7(−4)
4	10^5	(−2.8(−4), 1.7(−4))	1.0(−5)	2.3(−4)

Domain Problems

- Suppose $f : X \subseteq \mathbb{R}^n \to \mathbb{R}, g : X \subseteq \mathbb{R}^n \to \mathbb{R}^m, h : X \subseteq \mathbb{R}^n \to \mathbb{R}^\ell$, and we want to solve

$$\begin{aligned} \min_x \ & f(x) \\ \text{s.t.} \ & g(x) = 0 \\ & h(x) \leq 0 \end{aligned}$$

- The penalty function approach produces an unconstrained problem

$$\max_{x \in \mathbb{R}^n} F(x; P, a, b)$$

- Problem: $F(x; P, a, b)$ may not be defined for all x.

Domain Problems

- Example: Consumer demand problem

$$\begin{array}{l} \max_{y,z} u(y,z) \\ \text{ s.t. } \; py + qz \leq I \end{array}$$

  - Penalty method

$$\max_{y,z} u(y,z) - \frac{1}{2} P(\max[0, py + qz - I])^2$$

- Problem: $u(y,z)$ will not be defined for all y and z, such as

$$\begin{array}{l} u(y,z) = \log y + \log z \\ u(y,z) = y^{1/3} z^{1/4} \\ u(y,z) = \left(y^{1/6} + z^{1/6}\right)^{7/2} \end{array}$$

- Penalty method may crash when computer tries to evaluate $u(y,z)$!

Domain Problems

- Strategy 1: Transform variables
  - If functions are defined only for $x_i > 0$, then reformulate in terms of $z_i = \log x_i$
  - For example, let $\widetilde{y} = \log y, \widetilde{z} = \log z$ and solve

$$\max_{\widetilde{y},\widetilde{z}} u\left(e^{\widetilde{y}}, e^{\widetilde{z}}\right) - \frac{1}{2} P\left(\max\left[0, pe^{\widetilde{y}} + qe^{\widetilde{z}} - I\right]\right)^2$$

  - Problem: log transformation may not preserve shape; e.g., concave function of x may not be concave in $\log x$
- Strategy 2: Alter objective and constraint functions so that they are defined everywhere (see discussion above)

Domain Problems

- Strategy 3: Express the domain where functions are defined in terms of inequality constraints that are enforced by the algorithm at every step.
  - E.g., if utility function is $\log(x) + \log(y)$, then add constraints $x \geq \delta; y \geq \delta$ for some very small $\delta > 0$ (use, for example, $\delta \approx 10^{-6}$; don't use $\delta = 0$ since roundoff error may still allow negative x or y)
  - In general, you can avoid domain problems if you express the domain in terms of linear constraints.
  - If the domain is defined by nonlinear functions, then create new variables that can describe the domain in linear terms.

Active Set Approach

- Problems:
  - Kuhn-Tucker approach has too many combinations to check
    - some choices of $\mathcal{P}$ may have no solution
    - there may be multiple local solutions to others.
  - Penalty function methods are costly since all constraints are in the function, even if only a few bind at solution.

Active Set Approach

Solution: refine K-T with a good sequence of subproblems, ignoring constraints that you think won't be active at the solution.

- Let $\mathcal{J}$ be the set $\{1, 2, \cdots, \ell\}$
- for $\mathcal{P} \subset \mathcal{J}$, define the $\mathcal{P}$ problem

$$\begin{aligned} &\min_x f(x) \\ &\text{s.t. } g(x) = 0, \\ &h^i(x) \leq 0, \quad i \in \mathcal{P} \end{aligned}$$

- Choose an initial set of constraints, $\mathcal{P}$, and solve the optimization problem. If that solution satisfies all constraints, then you are done.

Active Set Approach

- Otherwise
  - Add constraints which are violated by most recent guess
  - Periodically drop constraints in $\mathcal{P}$ which fail to bind
  - Increase penalty parameters
  - Repeat
- The simplex method for linear programing is really an active set method.

Interior-Point methods

- Consider

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \; & c^\top x \\ \text{s.t.} \;\; & Ax = b \\ & x \geq 0 \end{aligned}$$

  where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and A is an $m \times n$ matrix.
- Karush-Kuhn-Tucker conditions for this optimization problem are as follows.

$$\begin{aligned} A^\top \lambda + s &= c \\ Ax &= b \\ x_i s_i &= 0, \quad i = 1, 2, \ldots, n \\ x &\geq 0 \\ s &\geq 0 \end{aligned}$$

- Interior-point methods solve a sequence of perturbed problems.
  - Consider the following perturbation of the KKT conditions.

$$
\begin{aligned}
A^\top \lambda + s &= c \\
Ax &= b \\
x_i s_i &= \mu, \quad i = 1, 2, \ldots, n \\
x &> 0 \\
s &> 0
\end{aligned}
$$

  - The complementarity condition $x_i s_i = 0$ is replaced by $x_i s_i = \mu$ for some positive scalar $\mu > 0$.

Interior-Point methods

- Assuming that a solution $(x^{(0)}, \lambda^{(0)}, s^{(0)})$ to this system is given for some initial value of $\mu^{(0)} > 0$, interior-point methods decrease the parameter μ and thereby generate a sequence of points $(x^{(k)}, \lambda^{(k)}, s^{(k)})$ that satisfy the non-negativity constraints on the variables strictly, $x^{(k)} > 0$ and $s^{(k)} > 0$.
- As μ is decreased to zero, a point satisfying the original first-order conditions is reached.
- The set of solutions to the perturbed system,

$$C = \{x(\mu), \lambda(\mu), s(\mu) \mid \mu > 0\}$$

  is called the central path.
- Matlab code, fmincon, contains both active set and interior point algorithms.