Numerical Methods in Economics

MIT Press, 1998

Notes for Chapter 4: Optimization

October 3, 2007

Optimization Problems

- Canonical problem:

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

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

One-D 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 $p(x)$
  - If $f''(a) > 0$, then p is convex, and $x_m = a - f'(a)/f''(a)$.
  - Hope: x_m is closer than a to the minimum.

- Newton's method:

  Algorithm 4.2 Newton's Method in $\mathbb{R}^1$

  Initialize. Choose initial guess x_0 and stopping parameters $\delta, \epsilon > 0$.

  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 defined in (4.1.2) converges quadratically to x^*; in particular,*

$$\lim_{n\to\infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|^2} = \frac{1}{2}\left|\frac{f'''(x^*)}{f''(x^*)}\right| \tag{4.1.3}$$

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

$$\max_{\theta} \quad \left(\frac{\theta}{2}\right)^{1/2} + 2\left(\frac{1-\theta}{3}\right)^{1/2} \tag{4.1.6}$$

  - Solution $\theta^* = 3/11 = .272727$
  - If $\theta_0 = 1/2$, 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:

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

  - Compute $f(x)$, $x \in X$, and locate max
  - Should always do some grid search first.
  - Grid search is sloooooooow

- Polytope Methods (a.k.a. Nelder-Mead, simplex, "amoeba")

Algorithm 4.3 Polytope Algorithm

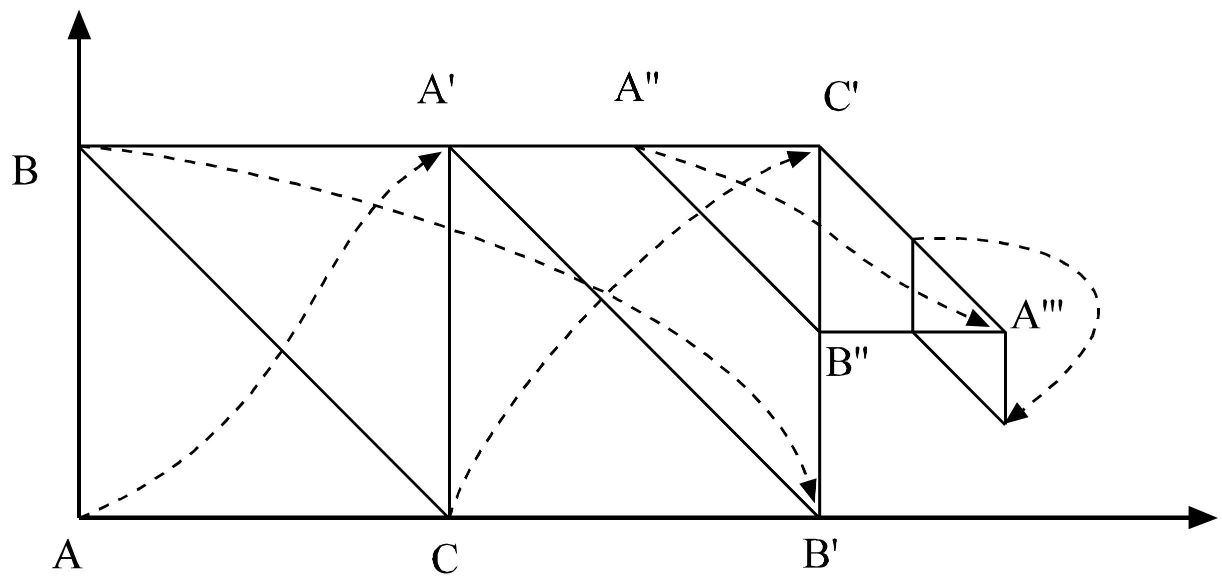
Initialize. Choose the stopping rule parameter ϵ. Choose an initial simplex $\{x^1, x^2, \cdots, x^{n+1}\}$.

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

Step 2. Look for least i s.t. $f(x^i) > f(y^i)$ 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 $x^i = \frac{1}{2}(x^i + x^{n+1})$, and go to step 1.



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, $H(x^k)$, and gradient, $\bigtriangledown f(x^k)$, and solve $H(x^k)s^k = -(\bigtriangledown f(x^k))^\top$ for the step s^k.

Step 2. $x^{k+1} = x^k + s^k$.

Step 3. If $\| x^k - x^{k+1} \| < \epsilon(1+ \| x^k \|)$, go to step 4; else go to step 1.

Step 4. If $\| \bigtriangledown f(x^{k+1}) \| < \delta(1 + |f(x^{k+1})|)$, STOP and report success; else STOP and report convergence to nonoptimal point.

- Stopping rule: Choose ε and δ to be bigger than square root of machine epsilon.

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 $\| x^0 - x^* \| < \epsilon$, then the sequence defined in (4.3.1) 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 $H(x^k)s^k = -(\bigtriangledown f(x^k))^\top$ for s^k
  - May converge to local solution, not global solution
  - We now consider methods which solve 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 $\lambda_k = \arg\min_\lambda f(x^k + \lambda s^k)$.
Step 3.	$x^{k+1} = x^k + \lambda_k s^k$.
Step 4.	If $\parallel x^k - x^{k+1} \parallel < \epsilon(1+ \parallel x^k \parallel)$, go to step 5; else go to step 1.
Step 5.	If $\parallel \bigtriangledown f(x^{k+1}) \parallel < \delta(1 + f(x^{k+1}))$, STOP and report success; else STOP and report convergence to nonoptimal point.

- Possible direction set methods
  - Coordinate Directions
    - Let search directions be coordinate, x_1, x_2, etc.
    - Search direction $s_{2n+k} = x_k$
  - Steepest Descent: $s_k = \bigtriangledown f(x^k)$
  - Newton's Method with Line Search: $H_k s^k = -(\bigtriangledown f(x^k))^\top$
- These will always converge to a local optimum.

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 $H_k s^k = -(\bigtriangledown f(x^k))^\top$ for the search direction s^k.
Step 2.	Solve $\lambda_k = \arg\min_\lambda f(x^k + \lambda s^k)$
Step 3.	$x^{k+1} = x^k + \lambda_k s^k$.
Step 4.	Compute H_{k+1} using H_k, $\bigtriangledown f(x^{k+1})$, x^{k+1}, $\bigtriangledown f(x^k)$, etc.
Step 5.	If $\parallel x^k - x^{k+1} \parallel < \epsilon(1+ \parallel x^k \parallel)$, go to step 6;. else go to step 1
Step 6.	If $\parallel \bigtriangledown f(x^{k+1}) \parallel < \delta\lvert 1 + f(x^{k+1})\rvert$, STOP and report success; else STOP and report convergence to nonoptimal point.

- Example: BFGS:

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

  - Preserves positive definiteness
  - Uses only gradients that are already needed
  - Warning: denominators may get too small; should keep them away from zero since small z_k does not necessarily stop iteration.

- Note: The Hessian iterates H_k may not converge to true Hessian at solution, even if x_k converges to solution.

Monopoly Example

- We look at a simple monopoly pricing example:
  - Utility function: if M is spending on other goods,

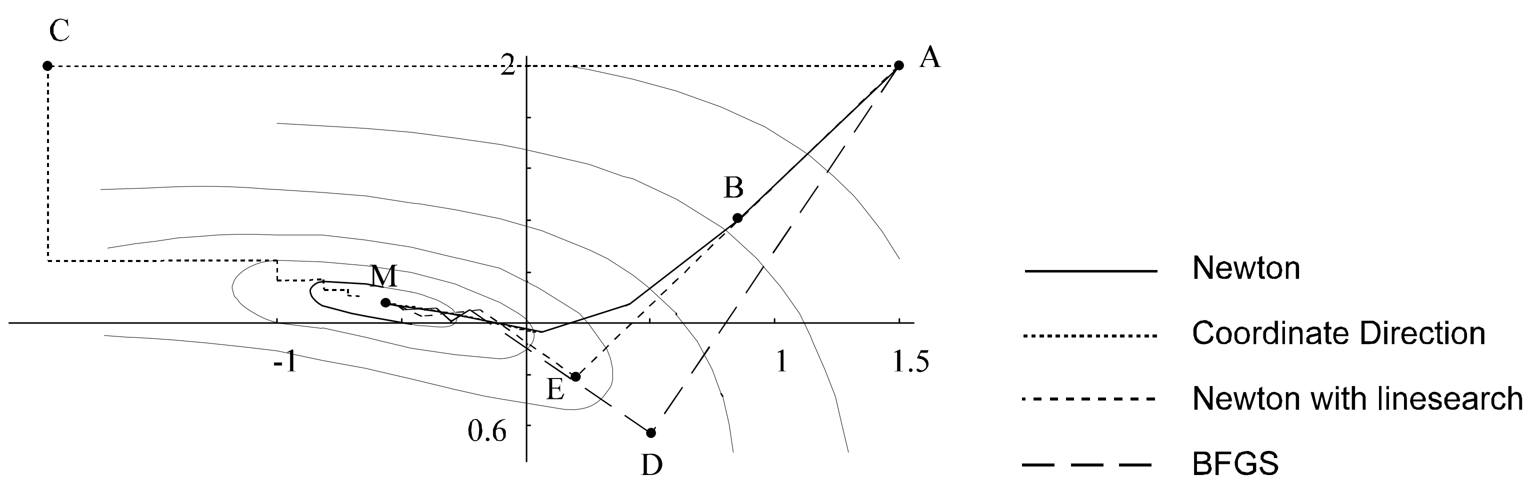
$$U(Y,Z) = (Y^\alpha + Z^\alpha)^{\eta/\alpha} + M = u(Y,Z) + M,$$

  - Output Y and Z implies prices of u_Y and u_Z.
  - Monopoly problem is

$$\max_{Y,Z} \Pi(Y,Z) \equiv Y u_Y(Y,Z) + Z u_Z(Y,Z) - C_Y(Y) - C_Z(Z), \tag{1}$$

  - Restate in terms of $y \equiv \ln Y$ and $z \equiv \ln Z$, $\pi(y,z) \equiv \Pi\left(e^y, e^z\right)$

$$\max_{y,z} \pi(y,z), \tag{2}$$



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
  - utility function $\sum_{t=1}^{T} \beta^t u(c_t)$.
- Define S_t to be end-of-period savings:

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

  - The constraint $S_T = 0 = S_0$
  - Substitute $c_t = S_{t-1}(1+r) + w_t - S_t$
- Problem now has $T - 1$ choices:

$$\begin{array}{ll} \max_{S_t} \sum_{t=1}^{T} \beta^t u(S_{t-1}(1+r) + w_t - S_t) \\ s.t. \quad S_T = S_0 = 0 \end{array} \tag{3}$$

  - Appears intractable for large T.
  - However, there are two ways to exploit the special structure of this problem and to efficiently solve this problem.

- Newton's method
  - Looks impractical if T large.
  - Hessian is tridiagonal (a sparse matrix), so Newton step is easy to compute.
  - Sparse Hessians are common in dynamic problems
  - *You* must recognize this and implement Newton or quasi-Newton method with sparse Hessians

Domain Problems

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

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

- Newton's method will take the guess S^k and compute a new guess S^{k+1}.
- Problem: S^{k+1} could imply negative consumption, $S_{t-1}(1+r)+w_t-S_t$, 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.

Nonlinear Least Squares

- Objective function has form, $f^i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., m$.:

$$\min_x \frac{1}{2} \sum_{i=1}^{m} f^i(x)^2 \equiv S(x),$$

- Idea: use simple approximation of Hessian
- In econometric applications
  - $f^i(x)$ are $g(\beta, y^i)$,
    - $x = \beta$ is parameter vector
    - y^i are the data.
    - $g(\beta, y^i)$ is residual for observation i
  - $S(\beta)$ is the sum of squared residuals at β.
- Let $f(x)$ denote the column vector $(f^i(x))_{i=1}^m$.
  - Let $J(x)$ be the Jacobian of $f(x) \equiv (f^1(x), \ldots, f^m(x))^\top$.
  - Let $f^i_\ell \equiv \frac{\partial f^i}{\partial x_\ell}$ and $f^i_{j\ell} \equiv \frac{\partial^2 f^i}{\partial x_j \partial x_\ell}$.
  - The gradient of $S(x)$ is $J(x)^\top f$: $S_\ell(x) = \sum_{i=1}^m f^i_\ell(x) f^i(x)$.
  - The Hessian of $S(x)$ is $J(x)^\top J(x) + G(x)$, where

$$G_{j\ell}(x) = \sum_{i=1}^{m} f^i_{j\ell}(x) f^i(x).$$

- Special structure of the gradient and Hessian.
  - $f_j^i(x)$ terms are needed to compute gradient of $S(x)$.
  - If $f(x) = 0$, then Hessian is just $J(x)^\top J(x)$: easy to compute.
  - A problem where $f(x)$ is small at the solution is called a *small residual problem*; otherwise, it is a *large residual problem*.
- Gauss-Newton algorithm
  - Do Newton except use $J(x)^\top J(x)$ for Hessian approx.

$$s^k = -(J(x^k)^\top J(x^k))^{-1}(\nabla f(x^k))^\top \tag{4.5.1}$$

    and avoid computing second derivatives of f.
  - Natural to use for small residual problems.
  - Works very well when it works.

- Problems.
  - $J(x)^\top J(x)$ is likely to be poorly conditioned, since it is the "square" of a matrix.
  - $J(x)$ may be poorly conditioned itself, particularly in statistical contexts.
  - Gauss-Newton step may not be a descent direction.
- Solution: Levenberg-Marquardt algorithm.
  - Use $J(x)^\top J(x) + \lambda I$ for some scalar λ (I is identity matrix):

$$s^k = -(J(x^k)^\top J(x^k) + \lambda I)^{-1}(\nabla f(x^k))^\top$$

  - The λI term reduces conditioning problems by "adding a little piece of the identity matrix"
  - s^k will be descent direction for large λ since s^k gets closer to steepest descent direction λ.

Linear Programming

- Canonical linear programming problem is

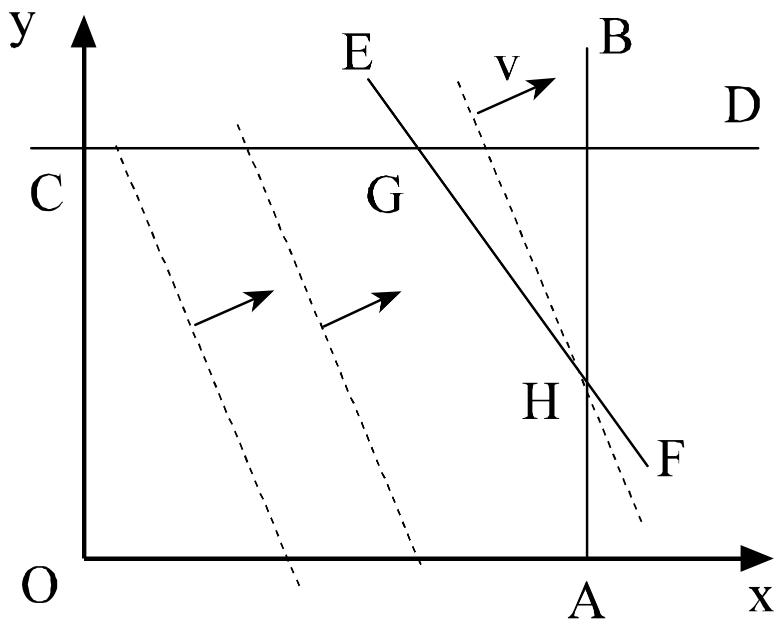
$$\begin{aligned} &\min_x a^\top x \\ &s.t. Cx = b, \\ &x \geq 0. \end{aligned} \tag{4}$$

  - $Dx \leq f$: use *slack variables*, s, and constraints $Dx + s = f, s \geq 0$.
  - $Dx \geq f$: use $Dx - s = f, s \geq 0$, s is vector of *surplus variables*.
  - $x \geq d$: define $y = x - d$ and min over y
  - x_i free: define $x_i = y_i - z_i$, add constraints $y_i, z_i \geq 0$, and min over (y_i, z_i).

- Basic method is the *simplex method*. Figure 4.4 shows example:

$$\begin{aligned} \min_{x,y} \;& -2x - y \\ s.t. \;& x + y \leq 4, \quad x, y \geq 0, \\ & x \leq 3, \quad\; y \leq 2. \end{aligned}$$

  - 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.
  - Figure 4.4 has two directions: from A: to B and to O, with B better.
  - Follow that direction to next vertex on boundary, and go back to step 1.
  - Continue until no direction reduces the objective: point H.
  - Stops in finite time since there are only a finite set of vertices.


y
E
B
v
D
C
G
H
F
O
A
x

- General History
  - Goes back to Dantzig (1951).
  - Fast on average.
  - Worst case time is exponential in number of variables and constraints
  - Software implementations vary in numerical stability
- Interior point methods
  - Developed in 1980's
  - Better on large problems

Constrained Nonlinear Optimization

- General problem:

$$\begin{aligned} &\min_x f(x) \\ &s.t.\ g(x) = 0 \\ &\quad\ \ \ h(x) \leq 0 \end{aligned} \tag{4.7.1}$$

  - $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$: 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

- Linear Independence Constraint Qualification (LICQ): The binding constraints at the solution are linearly independent

- Kuhn-Tucker 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*,

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^\top g(x) + \mu^\top h(x) \tag{4.7.2}$$

  If LICQ holds then the multipliers are unique; otherwise, they are called "unbounded".

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

$$\begin{aligned} f_x + \lambda^\top g_x + \mu^\top h_x &= 0 \\ \mu_i h^i(x) &= 0\,, \quad i = 1, \cdots, \ell \\ g(x) &= 0 \\ h(x) &\leq 0 \\ \mu &\geq 0 \end{aligned} \tag{4.7.3}$$

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 $\mathcal{P}$ problem, corresponding to a combination of binding and nonbinding inequality constraints

$$\begin{aligned} g(x) &= 0 \\ h^i(x) &= 0\,, \quad i \in \mathcal{P}, \\ \mu^i &= 0\,, \quad i \in \mathcal{J} - \mathcal{P}\ , \\ f_x + \lambda^\top g_x + \mu^\top h_x &= 0. \end{aligned} \tag{4.7.4}$$

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

Penalty Function Approach

- Most 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

$$\begin{aligned} \min_x \;& f(x) \\ s.t. \;& g(x) = a, \\ & h(x) \leq b. \end{aligned} \tag{4.7.5}$$

construct the penalty function problem

$$\min_x \; f(x) + \frac{1}{2} P \left(\sum_i \left(g^i(x) - a_i \right)^2 + \sum_j \left(\max\left[0, h^j(x) - b_j\right] \right)^2 \right) \tag{4.7.6}$$

where $P > 0$ is the penalty parameter.

  - Denote the penalized objective in (4.7.6) $F(x; P, a, b)$.
  - Include a and b as parameters of $F(x; P, a, b)$.
  - If P is "infinite," then (4.7.5) and (4.7.6) are identical.
  - Hopefully, for large P, their solutions will be close.

- 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
$$x^{k+1} \in \arg\min_x \; F(x; P_{k+1}, a, b) \tag{4.7.7}$$
    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 (which is hopefully not too ill-conditioned)
- Shadow prices in (4.7.5) and (4.7.7):
  - Shadow price of a_i in (4.7.6) is $F_{a_i} = P(g^i(x) - a_i)$.
  - Shadow price of b_j in (4.7.6) is F_{b_j}; $P(h^j(x) - b_j)$ if binding, 0 otherwise.
- Theorem: Penalty method works with convergence of x and shadow prices as P_k diverges (under mild conditions)

- 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

$$\begin{array}{l} \max_{y,z} \sqrt{yz} \\ s.t. \quad y + 2z \leq 5. \end{array} \tag{4.7.8}$$

    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)

Sequential Quadratic Method

- Special methods are available when we have a quadratic objective and linear constraints

$$\begin{aligned} \min_x\,&(x-a)^\top A\,(x-a) \\ s.t.\;\; & b\,(x-s)=0 \\ & c\,(x-q)\le 0 \end{aligned}$$

- Sequential Quadratic Method
  - Solution is stationary point of Lagrangian

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

  - Suppose that the current guesses are (x^k,λ^k,μ^k).
  - Let step size s^{k+1} solve approximating quadratic problem

$$\begin{aligned} \min_s\,&\mathcal{L}_x(x^k,\lambda^k,\mu^k)(x^k-s)+(x^k-s)^\top\mathcal{L}_{xx}(x^k,\lambda^k,\mu^k)(x^k-s) \\ s.t.\;\; & g(x^k)+g_x(x^k)(x^k-s)=0 \\ & h(x^k)+h_x(x^k)(x^k-s)\le 0 \end{aligned}$$

  - The next iterate is $x^{k+1}=x^k+\phi s^{k+1}$ for some ϕ
    - Could use linesearch to choose ϕ, or must take $\phi=1$.
    - λ^k and μ^k are also updated but we do not describe the detail here.
  - Proceed through a sequence of quadratic problems.
  - S.Q. method inherits many properties of Newton's method
    - rapid local convergence
    - can use quasi-Newton to approximate Hessian.

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

$$\begin{aligned} &\min_x f(x) \\ &s.t.\ g(x) = 0 \\ &\quad\ \ h(x) \leq 0 \end{aligned} \tag{4.7.1}$$

- 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.
- Example: Consumer demand problem

$$\begin{aligned} &\max_{y,z} u(y, z) \\ &s.t.\ p\ y + q\ z \leq I. \end{aligned}$$

  – Penalty method

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

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

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

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

- Solutions
  - 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

$$\max_{\widetilde{y},\widetilde{z}} \; u(e^{\widetilde{y}}, e^{\widetilde{z}}) - \frac{1}{2}P(\max[0,\ p\ e^{\widetilde{y}} + q\ e^{\widetilde{z}} - I])^2$$

    * 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)
  - 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 (4.7.5), even if only a few bind at solution.
- Solution: refine K-T with a *good sequence* of subproblems.
  - Let $\mathcal{J}$ be the set $\{1, 2, \cdots, \ell\}$
  - for $\mathcal{P} \subset \mathcal{J}$, define the $\mathcal{P}$ problem

$$\begin{aligned} &\min_x f(x) \\ &s.t.\ \ g(x) = 0, \qquad\qquad (\mathcal{P}) \\ &\qquad h^i(x) \leq 0, \quad i \in \mathcal{P}. \end{aligned} \tag{4.7.10}$$

  - Choose an initial set of constraints, $\mathcal{P}$, and start to solve (4.7.10-$\mathcal{P}$).
  - Periodically drop constraints in $\mathcal{P}$ which fail to bind
  - Periodically add constraints which are violated.
  - Increase penalty parameters
- The simplex method for linear programing is really an active set method.

Efficient Outcomes with Adverse Selection

- Rothschild-Stiglitz-Wilson (RSW) model of insurance markets with adverse selection; we formulate it as an endowment problem
- All agents receive either e_1 or e_2, $e_1 > e_2$
  - type H: probability π^H of receiving e_1
  - type L : probability π^L of receiving e_1, $\pi^H > \pi^L$.
  - θ^H $(\theta^L = 1 - \theta^H)$ is fraction of type $H(L)$ agents.
  - Risks are independent across agents;
  - Infinite number of each type; invoke LLN.
- Social planner
  - offers insurance contracts; redistributes income across states and people
  - sees only individual's realized income, not his type
  - must break even.

- $y = (y_1, y_2)$ is net state-contingent total income
  - pays $e_1 - y_1$ to insurer and consumes y_1 if income is e_1
  - receives $y_2 - e_2$ and consumes y_2 otherwise.
- Type t expected utility with net income, y^t.

$$U^t(y^t) = \pi^t u^t(y_1^t) + (1 - \pi^t) u^t(y_2^t), \;\; t = H, L,$$

- Planner's profits are

$$\begin{aligned} \Pi(y^H, y^L) = &\, \theta^H(\pi^H(e_1 - y_1^H) + (1 - \pi^H)(e_2 - y_2^H)) \\ &+ \theta^L(\pi^L(e_1 - y_1^L) + (1 - \pi^L)(e_2 - y_2^L)). \end{aligned} \tag{4.8.3}$$

- Social planner offers menu $\left(y^H, y^L\right)$ and lets agents choose
- $y^H, y^L \in \mathbb{R}^2$ *constrained efficient* if it solves

$$\begin{aligned} \max \;\; & \lambda U^H(y^H) + (1 - \lambda) U^L(y^L) \\ s.t. \;\; & U^H(y^H) \geq U^H(y^L), \\ & U^L(y^L) \geq U^L(y^H), \\ & \Pi(y^H, y^L) \geq 0, \end{aligned} \tag{4.8.4}$$

where $0 \leq \lambda \leq 1$ is the welfare weight of type H agents.

- Example (Rothschild-Stiglitz, Wilson, Miyazaki, Spence):
  - $e_1 = 1$, $e_2 = 0$, $\pi^H = 0.8$, $\lambda = 1$
  - $u(c) = -e^{-4c}$
  - $P_k = 10^{1+k/2}$; $P_k = 10^k$ did not work as well

(Modification of Table 4.9)

Adverse selection example

π^L	θ^H	(y_1^H, y_2^H)	(y_1^L, y_2^L)	IV: $U^L(y^L) - U^L(y^H)$	Profit
0.70	0.10	$0.87, 0.51$	$0.70, 0.70$	$-1(-10)$	$-1(-10)$
0.50	0.10	$0.92, 0.35$	$0.50, 0.50$	$-5(-14)$	$-1(-14)$
0.70	0.75	$0.82, 0.79$	$0.77, 0.77$	$-1(-12)$	$-6(-13)$
0.50	0.75	$0.797, 0.789$	$0.794, 0.794$	$-2(-12)$	$-6(-13)$

- The results do reflect the predictions of adverse selection ("hidden information") theory.
  - If θ^H small, there is no cross-subsidy. Type H agents receive actuarially fair contracts but must face risk to keep type L agents from pretending to be H.
  - If θ^H large, cross-subsidies arise: the numerous type H agents take actuarially unfair contracts but receive safer allocations.
  - Type L agents always receive a risk-free consumption since no one wants to pretend to be L.

Computing Nash Equilibrium

- A game with n players.
  - Player i: strategy set $S_i = \{s_{i1}, s_{i2}, \cdots, s_{iJ_i}\}$.
  - $S = \Pi_{i=1}^{n} S_i$ is set strategy combinations.
  - $M_i(q_i, \sigma_{-i})$ is payoff to i from mixed strategy q_i if others play σ_{-i}.
- Consider the function

$$v(\sigma) = \sum_{i=1}^{n} \sum_{s_{ij} \in S_i} \left\{\max\left[M_i(s_{ij}, \sigma_{-i}) - M_i(\sigma), 0\right]\right\}^2 .$$

Theorem 3 *(McKelvey) The solutions to*

$$\begin{aligned} &\min_{\sigma} v(\sigma) \\ &\quad \sum \sigma^i(s_j) = 1 \\ &\quad \sigma^i(s_j) \geq 0 \end{aligned}$$

are the Nash equilibria of (M, S) and they are also the zeros of $v(\sigma)$, and conversely.

- Tradeoffs
  - Reduces Nash computation to a minimization problem
  - There may be local optima where $v(\sigma) > 0$ and are not equilibria.

- Example: simple coordination game:

		R	
L		1, 1	0, 0
		0, 0	1, 1

  – p^i_j is prob. that player i plays his jth strategy.

  – Payoff for each player is $p^1_1 p^2_1 + p^1_2 p^2_2$.

  – Lyapunov function for this game is

$$v(p^1_1, p^1_2, p^2_1, p^2_2) = \sum_{i,j=1}^{2} \max[0, p^j_i - (p^1_1 p^2_1 + p^1_2 p^2_2)]^2.$$

  – Three global min (and three equilibria) are

$$\begin{aligned}(p^1_1,\ p^1_2,\ p^2_1,\ p^2_2) = &(1,\ 0,\ 1,\ 0),\\ &(0.5,\ 0.5,\ 0.5,\ 0.5),\\ &(0,\ 1,\ 0,\ 1)\end{aligned}$$

  – BFGS did well except it got hung up on saddle point, but such hangups are easily fixed.

Table 4.10: Coordination game

Iterate	(p_1^1, p_1^2)	(p_1^1, p_1^2)	(p_1^1, p_1^2)	(p_1^1, p_1^2)
0	(0.1, 0.25)	(0.9, 0.2)	(0.8, 0.95)	(0.25, 0.25)
1	(0.175, 0.100)	(0.45, 0.60)	(0.959, 8.96)	(0.25, 0.25)
2	(0.110, 0.082)	(0.471, 0.561)	(0.994, 0.961)	(0.25, 0.25)
3	(0, 0)	(0.485, 0.509)	(1.00, 1.00)	(0.25, 0.25)
4	(0, 0)	(0.496, 0.502)	(1.00, 1.00)	(0.25, 0.25)
5	(0, 0)	(0.500, 0.500)	(1.00, 1.00)	(0.25, 0.25)