1 Numerical Optimization

1.1 Introduction

As economist we are interested in solving optimization problems. These are problems of the form:

$$\begin{gathered}\text{min/max objective function} \\ \text{by choosing } x \in \mathbb{R} \\ \text{subject to equality/inequality constraints.}\end{gathered}$$

Often these problems cannot by solved analytically. Thus we must employ some numerical methods to solve them numerically. But choosing which numerical method to use to solve a problem can be difficult. Hence we need some criteria by which to determine how well suited a method is to the problem at hand.

Optimization algorithms are iterative. They begin with an initial guess of the optimal values and generate a sequence of improved estimates. We can judge how well suited an algorithm is for solving a particular problem by assessing it's performance along 3 different lines:

- robustness : How versatile is the algorithm? Can it handle different variations of the problem? How sensitive is it to the initial guess?
- efficiency : How fast is the algorithm? How much computer memory does it require?
- accuracy : How accurate is the solution? How sensitive is it to numerical error?

1.2 One Measure of Efficiency-Rates of Convergence

One measure of efficiency of an algorithm is its rate of convergence. There are two classes of of convergence rates: quotient rates (Q-rates) and root rates (R-rates).

1.2.1 Q-rates

Quotient-Linear Let $\{x_k\}$ be a sequence in $\mathbb{R}$ that converges to x^* then the convergence is **Q-linear** if there exists $r \in (0,1)$ such that $\frac{|x_{k+1}-x^*|}{|x_k-x^*|} \leq r$ for all k sufficiently large.

Example 1 *Consider the sequence* $\{1+(0.5)^k\}$ *which converges to* 1. *Show that it converges Q-linearly.*

$$\frac{\left|1+(0.5)^{k+1}-1\right|}{|1+(0.5)^k-1|} = 0.5.$$

Quotient-Superlinear Let $\{x_k\}$ be a sequence in $\mathbb{R}$ that converges to x^* then the convergence is **Q-superlinear** if

$$\lim_{k\to\infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|} = 0.$$

Example 2 *Consider the sequence* $\{1 + k^{-k}\}$ *which converges to* 1. *Show that it converges Q-superlinearly.*

$$\frac{\left|1 + (k+1)^{-(k+1)} - 1\right|}{|1 + k^{-k} - 1|} = \frac{\left|(k+1)^{-k}(k+1)^{-1}\right|}{|k^{-k}|} = 0.$$

Quotient-Quadratic Let $\{x_k\}$ be a sequence in $\mathbb{R}$ that converges to x^* then the convergence is **Q-quadratic** if there exists $M > 0$ (not necessarily < 1) such that $\frac{|x_{k+1}-x^*|}{|x_k-x^*|^2} \leq M$ for all k sufficiently large.

Example 3 *Consider the sequence* $\{1 + (0.5)^{2^k}\}$ *which converges to* 1. *Show that it converges Q-quadratically.*

$$\frac{\left|1 + (0.5)^{2^{k+1}} - 1\right|}{\left|1 + (0.5)^{2^k} - 1\right|^2} = \frac{\left|(0.5)^{2^{k+1}}\right|}{\left|(0.5)^{2^{k+1}}\right|} = 1.$$

In general we say that a sequence converges with order p ($p > 1$) if there exists a positive constant M such that $\frac{|x_{k+1}-x^*|}{|x_k-x^*|^p} \leq M$ for all k sufficiently large.

1.2.2 R-rates

Root-Linear Let $\{x_k\}$ be a sequence in $\mathbb{R}$ that converges to x^* then the convergence is **R-linear** if there exists a sequence $\{v_k\}$ with $v_k \geq 0$ for all k, $\{v_k\}$ converges Q-linearly to 0, and $|x_k - x^*| \leq v_k$ for all k.

Similarly convergence is *R-superlinear* if $\{v_k\}$ converges Q-superlinearly to 0 and convergence is *R-quadratic* if $\{v_k\}$ converges Q-quadratically to 0.

Example 4 *Consider the sequence* $x_k = \begin{cases} 1 + (0.5)^k & k \text{ even} \\ 1 & k \text{ odd} \end{cases}$. *Show that it converges R-linearly to* 1.

Let $v_k = (0.5)^k$ *then from above* $\{v_k\}$ *converges Q-linearly to* 0 *and*

$$\left|1 + (0.5)^k - 1\right| \leq (0.5)^k,$$

and

$$|1 - 1| \leq (0.5)^k,$$

for all k. *So* v_k *dominates* x_k.

Notice that root convergence is concerned only with the overall rate of decrease of the error while quotient convergence requires the error to decrease at each iteration of the algorithm. Thus Q-convergence is a stronger form of convergence than R-convergence and R-convergence implies Q-convergence. Q-convergence is more common and often just shortened to convergence.

A popular notation used to express the rate of convergence of a sequence or an algorithm (since the algorithm implies an underlying sequence of successive approximations to the solution) to 0 is called 'oh' notation. We say that a sequence is $o(n)$ if it converges Q-linearly to 0, $o(n^2)$ if it converges Q-quadratically, and, in general, $o(n^p)$ if its Q-rate of convergence is of order p. For R-rates of convergence we use 'big-Oh' notation. For example if a sequence converges R-linearly to 0 we say it is $O(n)$, etc.

1.3 Root-Finding Methods

Often we interested in finding x^* such that $f(x^*) = 0$. Methods used to solve problems of this form are call root-finding or zero-finding methods.

1.3.1 Bisection Method

The simplest root-finding method is the bisection method.

Idea: Suppose that $f(\cdot)$ is a continuous function defined over an interval $[a, b]$ and $f(a)$ and $f(b)$ have opposite signs. Then by the intermediate value theorem, there exists $c \in [a, b]$ such that $f(c) = 0$. Thus we can define smaller and smaller intervals such that each interval contains c by looking at subintervals of the current interval and choosing the one in which $f(\cdot)$ changes signs.

Algorithm:

Initial:

- Set $i = 0$.
- Set the tolerances for $f(\cdot)$ and x: ε, δ.
- Set maximum number of iterations to avoid endless iterating: Maxiter.
- Set $c_0 = \frac{a+b}{2} = \frac{a_0+b_0}{2}$.

For each iteration i:

- If $|f(c_i)| < \varepsilon$ and/or $|c_i - c_{i-1}| < \delta$
  STOP–done.
- If $i >$Maxiter
  STOP–maximum number of iterations reached.

- If $f(c_i) \in [f(a_i), 0)$
  Set $a_{i+1} = c_i$, $b_{i+1} = b_i$, and $c_{i+1} = \frac{a_{i+1}+b_{i+1}}{2}$.
- If $f(c_i) \in (0, f(b_i)]$
  Set $b_{i+1} = c_i$, $a_{i+1} = a_i$, and $c_{i+1} = \frac{a_{i+1}+b_{i+1}}{2}$.

Limitations: Convergence is very slow, see exercise 1. Only finds roots in which the function takes opposite signs on either side.

Exercises:

1. Suppose that $f(x)$ is continuous and strictly monotonic for $x \in [a, b] \subset \mathbb{R}$. And that $f(a)$ and $f(b)$ have opposite signs. Prove that the bisection method will converge to a zero of the function. What is the rate of convergence?

Solution: We know that a root exists by the intermediate value theorem. Call this root $r \in [a, b]$. Define $\{c_n\}_{n=0}^{\infty}$ to be the sequence of midpoints generated by the bisection algorithm. Then we need to show that

$$\lim_{n\to\infty} c_n = r,$$

which is equivalent to showing that

$$\lim_{n\to\infty} |r - c_n| = 0.$$

Since $r \in [a_n, b_n]$ and $c_n \in [a_n, b_n]$ the following must hold for all n,

$$|r - c_n| \leq \frac{b_n - a_n}{2}.$$

Using induction it is easy to show that

$$|r - c_n| \leq \frac{b_0 - a_0}{2^{n+1}},$$

holds for all n. Thus

$$0 \leq \lim_{n\to\infty} |r - c_n| \leq \lim_{n\to\infty} \frac{b_0 - a_0}{2^{n+1}} = 0,$$

The rate of convergence is r-linear. Since $|r - c_n|$ is dominated by $v_k = \frac{b_0-a_0}{2^{n+1}}$ which converges q-linearly to 0. Notice that convergence of $\{c_n\}$ is not q-linear.

2. Implement the bisection method using matlab. Test your program with an appropriate function.

1.3.2 Newton's Method

Probably the most popular method for finding a root is Newton's Method.

Idea: Suppose that $f(\cdot)$ is a continuously differentiable function and there is at least one root x^* such that $f(x^*) = 0$ if we have a "good enough" guess of x^*, x_n then by Taylor's expansion,

$$\begin{aligned} 0 &= f(x^*) \\ &= f(x_n) + f'(x_n)(x^* - x_n) + o((x^* - x_n)^2) \\ &\approx f(x_n) + f'(x_n)(x^* - x_n). \end{aligned}$$

Thus

$$x^* \approx x_n - [f'(x_n)]^{-1} f(x_n),$$

which suggests iterating on

$$x_{n+1} = x_n - [f'(x_n)]^{-1} f(x_n).$$

Algorithm:

Initial:

- Set $i = 0$.
- Set the tolerances for $f(\cdot)$ and x: ε, δ.
- Set maximum number of iterations to avoid endless iterating: Maxiter.
- Pick initial guess c_0.

For each iteration i:

- If $|f(c_i)| < \varepsilon$ and/or $|c_i - c_{i-1}| < \delta$
  STOP–done.
- If $i >$Maxiter
  STOP–maximum number of iterations reached.
- $c_{i+1} = c_i - [f'(c_i)]^{-1} f(c_i)$.

Limitations: If $f'(c_n) = 0$ then there is a division by 0 problem. Numerically this can be a problem even if $f'(c_n)$ is close to 0. Newton's method may not always converge and/or it may not converge to the desired root when more than one root exist. The success of the method depends crucially on the initial guess.

Theorem 5 (Local convergence) *Consider $f : U \to V$ where $U, V \subset \mathbb{R}$ are complete normed spaces. Suppose x^* is a root of $f(\cdot)$ and assume that there exists a neighborhood around x^*, $N(x^*)$ such that $[f'(x)]^{-1}$ exists and is continuous on $N(x^*)$ and $c = \sup_{x \in N(x^*)} ||[f'(x)]^{-1}|| < \infty$. Assume $f'(x)$ is Lipschitz continuous on $N(x^*)$, or*

$$||f'(x) - f'(v)|| \leq L||x - v|| \quad \forall x, v \in N(x^*),$$

where $L > 0$ is a constant. Then there exists a $\delta > 0$ such that if $||x_0 - x^*|| < \delta$, then the sequence $\{x_n\}$ generated by the Newton algorithm converges to x^*.

Proof. Define

$$T(x) = x - [f'(x)]^{-1} f(x), \quad x \in N(x^*).$$

Notice that $T(x^*) = x^*$. For $x \in N(x^*)$, we have

$$\begin{aligned} T(x^*) - T(x) &= x - x^* - [f'(x)]^{-1} f(x) \\ &= [f'(x)]^{-1}\{f(x^*) - f(x) - f'(x)(x^* - x)\} \\ &= [f'(x)]^{-1} \int_0^1 [f'(x + t(x^* - x)) - f'(x)]dt(x^* - x), \end{aligned}$$

where the last equality can be seen by noting that if $u \equiv x + t(x^* - x)$ then $du = (x^* - x)dt$ and

$$\int_0^1 f'(x + t(x^* - x))dt = \int_x^{x^*} f'(u)\frac{du}{(x^* - x)} = \frac{f(x^*) - f(x)}{x^* - x}.$$

Taking the norm and using the Lipschitz condition gives

$$\begin{aligned} ||T(x^*) - T(x)|| &\leq ||[f'(x)]^{-1}|| \int_0^1 ||f'(x + t(x^* - x)) - f'(x)||dt||x^* - x|| \\ &\leq ||[f'(x)]^{-1}|| \int_0^1 Lt||x^* - x||dt||x^* - x||. \end{aligned}$$

Hence,

$$||T(x^*) - T(x)|| \leq \frac{cL}{2}||x^* - x||^2.$$

Thus, if we choose $\delta < 2/(cL)$ then T is a contraction mapping and by the Banach fixed-point theorem, T has a unique fixed point x^* and $\{x_n\}$ converges to x^*. ■

Exercises:

1. Prove that the rate of convergence of Newton's Method is Q-quadratic.

2. Implement Newton's method using matlab. Test your program with an appropriate function.

2 Augmented Lagrangian

2.1 Introduction

Variety of algorithms used to solve optimization problems of the form

$$\begin{gathered}\min/\max\ f(x)\\ \text{by choosing } x \in \mathbb{R}^n\\ \text{subject to } c_i(x) = 0,\ i \in \mathcal{E},\\ c_j(x) \geq 0,\ j \in \mathcal{J}.\end{gathered}$$

Some depend on characteristics of the objective function and/or the constraints. For example:

- *Linear programming problems*–objective function and constraints are all linear in the choice variables.
- *Quadratic programming problems*–objective function is quadratic and constraints are linear.
- *Nonlinear programming*–some constraints are nonlinear.

One group of algorithms which can be applied to nonlinear programming problems are called: *penalty, barrier, and augmented Lagrangian methods.*

2.2 Penalty Methods

Idea: By combining the objective function and constraints into a penalty function we can attack the problem by solving a series of unconstrained problems.

Example: Suppose we want to

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

s.t.

$$c_i(x) = 0,\ i \in \mathcal{E}.$$

Define the Penalty function as:

$$f(x) + \frac{1}{2\mu} \sum_{\mathcal{E}} c_i^2(x),$$

where $\mu > 0$ is the penalty parameter. Then minimize the penalty function for a series of decreasing values of μ until we reach a sufficiently accurate solution to the original constrained problem.

2.3 Barrier Methods

Idea: Add terms to the objective function which are insignificant when x is safely in the interior of the feasible set but approach infinity as x approaches the boundary condition.

Example: Suppose we want to

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

s.t.

$$c_i(x) \geq 0, \ i \in \mathcal{E}.$$

The logarithmic barrier function has the form:

$$f(x) - \mu \sum_{\mathcal{E}} \ln c_i(x),$$

where $\mu > 0$ is the barrier parameter. Then minimize the log barrier function for a series of decreasing values of μ until we reach a sufficiently accurate solution to the original constrained problem.

2.4 Augmented Lagrangian Methods

Idea: Combine the properties of the Lagrangian function and the quadratic penalty function.

Example: Suppose we want to

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

s.t.

$$c_i(x) = 0, \ i \in \mathcal{E}.$$

Augmented Lagrangian is:

$$\mathcal{L}_A(x, \lambda; \mu) = f(x) - \sum_{\mathcal{E}} \lambda_i c_i(x) + \frac{1}{2\mu} \sum_{\mathcal{E}} c_i^2(x),$$

Then find x which minimizes $\mathcal{L}_A$ for fixed values of λ and μ. Use the new x to update λ, possibly decrease λ and repeat until a solution to the minimization problem of the desired accuracy is found.

If we have inequality constraints so that the problem is

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

s.t.

$$c_j(x) \geq 0, \ j \in \mathcal{I},$$

the augmented lagrangian becomes

$$\mathcal{L}_A(x,\lambda;\mu) = f(x) - \sum_{\mathcal{E}} \lambda_i \min\left\{c_i(x), \mu\lambda\right\} + \frac{1}{2\mu}\sum_{\mathcal{E}} \left(\min\left\{c_i(x), \mu\lambda\right\}\right)^2.$$

The algorithm is

1. Choose μ^0, λ^0, and $x^s \in \mathbb{R}^n$; set $k = 0$.
2. Using x^s as your initial guess find an unconstrained minimizer of
$$x^k = \arg\min_{x\in\mathbb{R}^n} \mathcal{L}_A(x, \lambda^k; \mu^k)$$
3. If $\left(x^k, \mu^k\right)$ satisfies the KKT conditions for optimality, STOP.
4. Update the penalty parameter, μ.
$\mu^{k+1} = \rho\mu^k$ with
$$\rho = \begin{cases} 1, & \text{if } \sum_{\mathcal{E}} c_i^2(x^{k-1}) - \sum_{\mathcal{E}} c_i^2(x^k) \geq C, \\ \hat{\rho}, & \text{otherwise.} \end{cases}$$
5. Update the multiplier estimates
$$\lambda_i^{k+1} = \max\left\{\lambda_i^k - c_i(x_k)/\mu_k, 0\right\}, \ \forall\ j \in \mathcal{I}.$$
6. Set $x^s = x^k, k = k+1$, and go to Step 2.