CHAPTER 5. OPTIMIZATION

This chapter is based on Chapter 4 of the book by Miranda and Fackler.

We are given a real-valued function *f* defined on $X \subseteq R^n$ and asked to find an $x^* \in X$ such that $f(x^*) \geq f(x)$ for all $x \in X$. We denote this problem

$$\max_{x \in X} f(x)$$

and call *f* the objective function, *X* the feasible set and *x**, if it exists, a maximum. In this chapter we will assume that the feasible set plays no role in the optimisation problem, or that $X = R^n$. Miranda and Fackler extends the analysis to the case when $X \subset R^n$.

A point x^* is a *local maximum* of *f* if there is a neighbourhood *N* of *x** such that $f(x^*) \geq f(x)$ for all $x \in N$. The point x^* is a *strict local maximum* if, additionally, $f(x^*) > f(x)$ for all $x \neq x^*$ in *N*.

If x^* is a local maximum of *f* and *f* is twice continuously differentiable, then *f'(x*)*=0 and *f''(x*)* is negative semi-definite. Conversely, if *f'(x*)*=0 and *f''(x)* is negative semi-definite in a neighbourhood *N* of *x**, then *x** is a local maximum. If additionally *f''(x)* is negative definite, then *x** is a strict local maximum. If *f* is concave on R^n and *x** is a local maximum of *f*, then *x** is a *global maximum* of *f* on R^n.

To solve a minimization problem, one simply maximizes the negative of the objective function. The result of previous paragraph hold for minimization, provided one change concavity of *f* to convexity and negative (semi) definiteness of *f''* to positive (semi) definiteness.

1. The golden search method

We saw in chapter 3 the bisection method, which is the simplest and most robust method for computing the root of a continuous real-valued function on a bounded interval of the real line. The golden search method shares these qualities for uni-dimensional optimisation problems.

We want to find a local maximum of a continuous univariate function *f(x)* on the interval $[a, b]$. We pick any two numbers in the interval, x_1 and x_2, with $x_1 < x_2$. We evaluate the function at these two points and replace the original interval with $[a, x_2]$ if $f(x_{1)} > f(x_2)$, or with $[x_1, b]$ if $f(x_1) \leq f(x_2)$. A local maximum must be contained in the new interval because the endpoints of the new interval have smaller function values than a point on the interval's interior (or the local maximum is at one of the original endpoints). We can repeat this procedure producing a sequence of progressively smaller intervals that are guaranteed to contain a local maximum, until the length of the interval is shorter than some desired tolerance level.

How to pick the interior evaluation points? First, we want the length of the new interval to be independent of whether the upper or lower bound is replaced, which means $x_2 - a = b - x_1$. Second, on successive iteration, one should be able to reuse an interior point from the previous iteration so that only one new function evaluation is performed per iteration. That means that if we have retained $[a, x_2]$ at the first iteration, the two points, which will be picked in this interval in the second iteration will be y_1 and $y_2 = x_1$. If we have retained $[x_1, b]$ at the first iteration the two points, which will be picked in the second iteration will be $y_1 = x_2$ and y_2. We can prove that these conditions are satisfied by selecting

$x_i = a + \alpha_i(b - a)$

where $\alpha_1 = \frac{3-\sqrt{5}}{2} \simeq 0.382$ and $\alpha_2 = \frac{\sqrt{5}-1}{2} \simeq 0.618$.

α_2 is the golden ratio conjugate $\Phi = \frac{1}{\varphi} = \varphi - 1$, where $\varphi = \frac{\sqrt{5}+1}{2}$ is the *golden ratio*. The golden ratio is a mysterious number well known by painters and architects. Two quantities *a* and *b* are said to be in the golden ratio if

$\frac{a+b}{a} = \frac{a}{b}$.

You can find much more on Wikipedia.

We can write in a working directory, which I called Chapter 5 the script M-file golden.m

```
alpha1 = (3-sqrt(5))/2;
alpha2 = (sqrt(5)-1)/2;
```

```
x1 = a+alpha1*(b-a);f1 = f(x1);
x2 = a+alpha2*(b-a);f2 = f(x2);
d = alpha1*alpha2*(b-a);
while d>tol
  d = d*alpha2;
  if f2<f1 % x2 is new upper bound
    x2 = x1; x1 = x1-d;
    f2 = f1; f1 = f(x1);
  else     % x1 is new lower bound
    x1 = x2; x2 = x2+d;
    f1 = f2; f2 = f(x2);
  end
end
if f2>f1
    x1 = x2;
else
    x = x1; end
```

We will use this program to compute the maximum of the function $f(x) = xcos(x^2)$ on the interval [0, 3]. So, we write a function M-file f.m

```
function y=f(x);
y=x*cos(x^2);
```

Finally, we type in the command window

>> tol=1e-09;

>> a=0;b=3;

>> golden;x

We get

x =

0.8083

There exist extensions of this method to the optimisation of multivariate functions. We will not look at them in this course.

2. The Newton-Raphson method

We saw that if x^* is a local maximum of the objective function f and f is twice continuously differentiable, then $f'(x^*)=0$ and $f''(x^*)$ is negative semi-definite. Conversely, if $f'(x^*)=0$ and $f''(x)$ is negative semi-definite in a neighbourhood N of x^*, then x^* is a local maximum.

This means that a local maximum of *f(x)* is a root of the gradient *f'(x)* of this function. But a root of the gradient *f'(x)* will be a maximum of *f(x)* only if the Hessian *f''(x)* is negative semi-definite in a neighbourhood of this root. Notice that if *f* is a *n* variable function, then its gradient is a *n* dimension vector and its Hessian a *nxn* matrix.

The Newton-Raphson method is identical to applying Newton's method to compute the root of the gradient of the objective function. So, we deduce from equation (3) of chapter 3 the iteration rule

$$x^{(k+1)} = x^{(k)} - \left[f''\left(x^{(k)}\right)\right]^{-1} f'\left(x^{(k)}\right) \qquad (1)$$

The analyst has to supply a guess $x^{(0)}$ to begin the iteration.

This method is very efficient if *f* is concave everywhere or at least on a large subset of R^n. Otherwise, the method can converge to a root of the Jacobian where the Hessian is not a negative semi-definite matrix in the neighbourhood of this root. Then, we will have computed a minimum or a saddle-point of the objective function, not a maximum!!!!

For this reason the Newton-Raphson method is rarely used in practice, and then only if the objective function is globally concave.

Equation (1) expresses that in iteration *k+1* we move from $x^{(k)}$ to $x^{(k+1)}$ in the direction $x^{(k+1)} - x^{(k)} = \left[f''\left(x^{(k)}\right)\right]^{-1} f'\left(x^{(k)}\right)$, which is called the *Newton-Raphson step*. The idea of the other methods will be to move in a direction such that *f(x)* begins by increasing when one starts from $x^{(k)}$. We will stop moving when *f(x)* decreases by too much. Of course, we can look for the maximum of *f(x)* in this direction, for instance by using the golden search method. But actually, as we are going through a sequence of iterations, we will be satisfied if *f(x)* increases by a significant amount in the direction of the iteration, even if we have not reached its maximum in this direction.

3. The Quasi-Newton method: how to find a good search direction

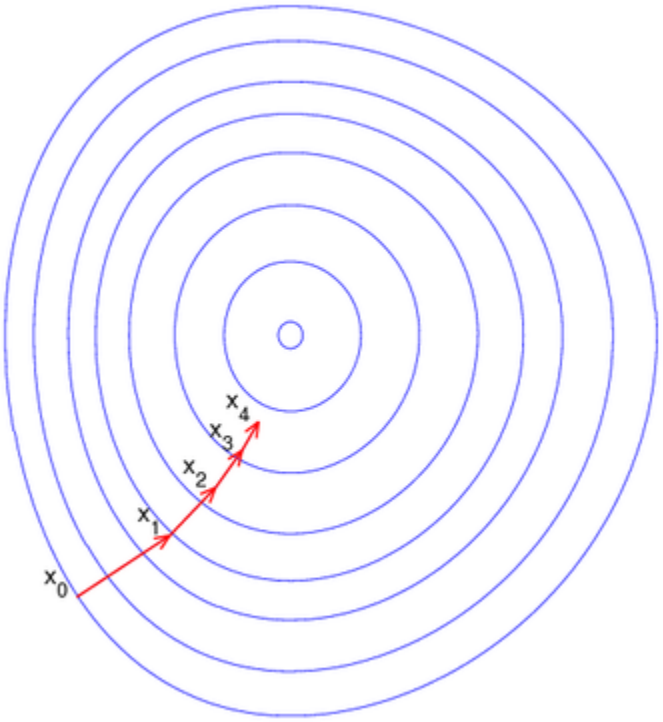
The idea of these methods is to substitute equation (1) by

$$x^{(k+1)} = x^{(k)} - s^{(k)} B^{(k)} f'\left(x^{(k)}\right) \qquad (2)$$

$d^{(k)} = B^{(k)} f'\left(x^{(k)}\right)$ is the product of a *nxn* matrix $B^{(k)}$, which is used instead of the inverse Hessian, by the gradient of the objective function $f'\left(x^{(k)}\right)$, which is a *n* vector. Thus, $d^{(k)}$ is a *n* vector, which defines a direction, called *quasi-Newton step*. $s^{(k)}$ is a scalar indicating how far we progress along the quasi-Newton step. $s^{(k)}$ can be computed by a golden search method or by other univariate maximisation methods, which will be briefly presented in the next section.

Quasi-Newton methods differ in how matrix $B^{(k)}$ is computed. There are three main groups of methods.

1. The *method of the steepest ascent*. We set $B^{(k)} = -I$, where I is the identity matrix. So, the quasi-Newton step is identical to the gradient of the objective function at the current iterate $d^{(k)} = f'\left(x^{(k)}\right)$ .The choice of gradient as a step direction is intuitively appealing because the gradient always points in the direction which, to a first order, promises the greatest increase in f . I have taken the following illustration from Wikipedia.



Here f is assumed to be defined on the plane, and that its graph has an invert bowl shape. The blue curves are the contour lines, that is, the regions on which the value of f is constant. A red arrow originating at a point shows the direction of the negative gradient at that point. Note that the (negative) gradient at a point is orthogonal to the contour line going through that point. We see that gradient *ascent* leads us to the top of the invert bowl, that is, to the point where the value of the function f is maximal.

Or to tell these things in a simpler way: when you climb a hill you never follow the gradient line, but when you go down (and if you have good knees and ankles) you follow it.

The steepest ascent method is simple to implement, but it is numerically less efficient in practice than competing quasi-Newton methods that incorporate information regarding the curvature of the objective function.

2. If we know $x^{(k)}$ and $B^{(k)}$ we can compute $x^{(k+1)}$ with equation (1). We need a second iteration formula, to computes $B^{(k+1)}$ when we know $x^{(k)}$ and $x^{(k+1)}$. We have the Taylor's approximation

   $f'\left(x^{(k+1)}\right) \approx f'\left(x^{(k)}\right) + f''\left(x^{(k)}\right)\left(x^{(k+1)} - x^{(k)}\right)$

   We deduce from this equation

   $x^{(k+1)} - x^{(k)} \approx \left[f''\left(x^{(k)}\right)\right]^{-1}\left[f'\left(x^{(k+1)}\right) - f'\left(x^{(k)}\right)\right]$

   This equation suggests that the iteration formula on $B^{(k+1)}$ should satisfy the so-called *quasi-Newton condition*

   $d^{(k)} = B^{(k+1)}\left[f'\left(x^{(k+1)}\right) - f'\left(x^{(k)}\right)\right]$

   This condition imposes *n* conditions on the *nxn* $B^{(k+1)}$ matrix. We also require this matrix to be symmetric and negative definite, as must be true of the inverse Hessian at a local maximum. The negative definiteness of $B^{(k+1)}$ assures that the objective function value can be increased in the direction of the quasi-Newton step.

   There are two popular methods of updating $B^{(k+1)}$ that satisfy these two constraints. The *Davidson-Fletcher-Powell (DFP) method* uses the updating scheme

   $$B^{(k+1)} = B^{(k)} + \frac{dd'}{d'u} - \frac{B^{(k)}uu'B^{(k)}}{u'B^{(k)}u}$$

   where $d = x^{(k+1)} - x^{(k)}$

   and $u = f'\left(x^{(k+1)}\right) - f'\left(x^{(k)}\right)$

   The *Broyden-Fletcher-Goldfarb-Shano (BFGS) method* uses the updating scheme

   $$B^{(k+1)} = B^{(k)} + \frac{1}{d'u}\left(wd' + dw' - \frac{w'u}{d'u}dd'\right)$$

   where $w = d - B^{(k)}u$

In both methods we can set $B^{(0)} = I$, as in the steepest ascent method. When we use one of these methods, sometimes it works, sometimes it does not work. So, you are advised trying both (and hope that at least one of them works).

There are a few important practical tricks with these three methods, which are explained in the book by Miranda and Fackler.

3. There are two important econometric problems where we can use other ways of computing matrix $B^{(k+1)}$.

   - The nonlinear univariate regression model is
     $y_i = f_i(\theta) + \varepsilon_i$, with θ a *k*-vector of parameters and *i=1,…,n* identifies the observations.
     Let us define $f(\theta)$ as the *n*-vector with generic element $f_i(\theta)$. The nonlinear least squares (NLLS) estimator of θ is the solution of the minimisation problem
     $$\min_{\theta} \frac{1}{2} f'(\theta) f(\theta)$$
     The gradient of the objective function is $\left[\frac{\partial f(\theta)}{\partial \theta}\right]' f(\theta)$
     The Hessian of the objective function is
     $$\left[\frac{\partial f(\theta)}{\partial \theta}\right]' \frac{\partial f(\theta)}{\partial \theta} + \sum_{i=1}^{n} f_i(\theta) \frac{\partial^2 f_i(\theta)}{\partial \theta \partial \theta'}$$
     If we ignore the second term in the Hessian, we are assumed of having a positive definite matrix with which to determine the search direction
     $$d = -\left\{\left[\frac{\partial f(\theta)}{\partial \theta}\right]' \frac{\partial f(\theta)}{\partial \theta}\right\}^{-1} \left[\frac{\partial f(\theta)}{\partial \theta}\right]' f(\theta)$$

   - In maximum likelihood problems we have to maximise the log-likelihood function, which is the sum of the logs of the likelihoods of each of the data points
     $$l(\theta; y) = \sum_{i=1}^{n} \ln f(y_i; \theta)$$

The *score function* at point *y*, $s(\theta; y)$, is defined as the column vector of derivatives of the log-likelihood function evaluated at this point. For instance, the score function at observation *i* is the *k* vector

$$s(\theta; y_i) = \frac{\partial l(\theta; y_i)}{\partial \theta}$$

A well known result in statistical theory is that the expectation of the outer product of the score function, $s(\theta; y)s(\theta; y)'$, is equal to the negative of the expectation of the second derivative of the likelihood function, which is known as the *information matrix*.

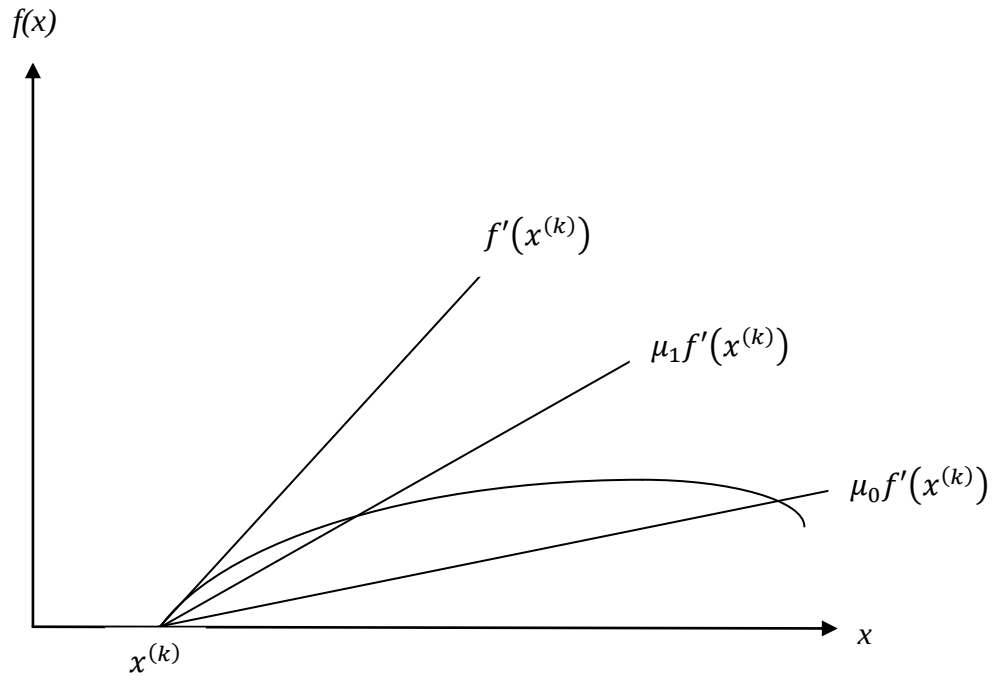
Either the information matrix or the sample average of the outer product of the score function provides a positive definite matrix that can be used to determine a search direction. In general the second matrix is much easier to compute. Then, the search direction is defined by

$$d = -\left[\sum_{i=1}^{n} s(\theta; y_i)\, s(\theta; y_i)'\right]^{-1} \sum_{i=1}^{n} s(\theta; y_i)$$

4. Line Search Methods

At iteration *k* we first have to compute a direction to move, starting at point $x^{(k)}$, then we have to determine where to stop in this direction to get point $x^{(k+1)}$. Section 3 explained how to compute this direction. We already told that point $x^{(k+1)}$ can be taken as the maximum of the objective function in this direction, and that this maximum can be computed by the golden search method. However, this computation can take time, and we do not need to compute the maximum of function *f*. The choice of a point $x^{(k+1)}$, which significantly increases the value of the objective function, will be quite satisfactory.

We will just give a basic idea of the line search methods by presenting the *Goldstein criterion* on the following graph.



We start at point $x^{(k)}$ and know the value of the derivative $f'(x^{(k)})$. By a system of trials and errors we compute a point $x^{(k+1)}$ such that

$\mu_0 f'(x^{(k)}) < \frac{f(x^{(k+1)})-f(x^{(k)})}{x^{(k+1)}-x^{(k)}} < \mu_1 f'(x^{(k+1)})$, with $0 < \mu_0 < \mu_1 < 1$

Thus, the point $(x^{(k+1)}, f(x^{(k+1)}))$ will be inside the angle between the two lines $\mu_0 f'(x^{(k)})$ and $\mu_1 f'(x^{(k)})$. In general this point will not be the maximum of function *f*. But we will have $f(x^{(k+1)}) > f(x^{(k)})$. Usually we take $\mu_0 = 0.25$ and $\mu_1 = 0.75$.

5. The end of the example of chapter 4

At the end of chapter 4 we drew a graph showing that the households' utility function is a concave function of the tariff rate *t* and the pollution tax rate *s*. the end of the program computes the optimal values of these two variables.

We get

Uc =

3.5574

Uopt =

3.5951

topt =

0.3459

sopt =

0.2028

In the table of chapter 4 we can see these results on the line T=1 and $\eta = 0.5$. I noted that the utility index in the model has the same dimension as a consumption index. When we move from zero to optimal tariff and tax this utility increases in percentage by

>> 100*(Uopt/Uc-1)

ans =

1.0608

An increase of households' consumption by 1 percent looks weak (the tariff rate increases from 0 to 34.59% and the taxation rate of manufacturing good increases from 0 to 20.28%). In general the evaluation by neoclassical computable general equilibrium models of the effects of big reforms of the tax or trade protection system lead to effects of this order of magnitude, an increase in consumption by no more than 1 or 2 percent.