

# Introduction to GAUSS (III): Numerical Optimization

Enrique Moral-Benito\*

OCTOBER 2008

## 1 Numerical Optimization

I suppose that until now, you have already faced a lot of optimization problems. However, in the next two years at CEMFI, and probably in the remaining of your careers, you will face much more... There are two problems that are very common in economics and they are formally equivalent: (i) maximization/minimization: likelihood, Generalized Method of Moments (GMM), Nonlinear Least Squares... (ii) zeros of a function: usually First Order Conditions (FOC).

In this third lesson, we will see that GAUSS is very useful for this problems.

### 1.1 Newton's method

It is often called Newton-Raphson method, however, the contribution of Raphson was to publish a simplified description of the method...

Sometimes we face a problem which cannot be solved by simple algebraic methods, that is, it does not have an explicit solution. We will see that the computer will give us a way of finding approximate solutions.

The method is designed to find the roots of a given function. To approximate a solution to an equation  $f(x) = 0$  to within a tolerance of  $\varepsilon$  beginning with an initial guess  $x_0$ , compute the sequence of approximations  $x_0, x_1, x_2, x_3, \dots$ , using the difference equation

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

stopping when  $|x_{n+1} - x_n| < \varepsilon$ . The graph in the blackboard will be very useful for understanding the method.

The multivariate case of the method can be summarized as follows: Let  $\mathbf{f} : \mathbb{R}^k \rightarrow \mathbb{R}^k$ . Suppose we want to solve:

$$\mathbf{f}(\mathbf{x}) = 0$$

---

\*E-mail: enrique.moral@gmail.com

where  $\mathbf{f}(\mathbf{x})$  and  $0$  are  $k \times 1$  vectors, and  $\mathbf{x} = (x_1, \dots, x_k)'$ .

Then, the Newton iteration is:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - [\mathbf{f}'(\mathbf{x}_n)]^{-1} \mathbf{f}(\mathbf{x}_n) \quad (2)$$

where  $\mathbf{x}_{n+1}$ ,  $\mathbf{x}_n$  and  $\mathbf{f}(\mathbf{x}_n)$  are  $k \times 1$  vectors and  $\mathbf{f}'(\mathbf{x}_n)$  is a  $k \times k$  Jacobian matrix.

Originally, Newton conceived the method with the objective of finding the roots of a given function. However, his brilliant idea is also very useful for solving maximization/minimization problems. This is so because you can easily switch from (i) maximization/minimization to (ii) zeros of a function, since finding the maximum of  $f(x)$  is equivalent to finding the zero of  $f'(x)$ <sup>1</sup>.

## 1.2 Optimum library in GAUSS

As with the graph library, you have to explicitly include the optimum library in your code with the command `library optimum;`

Optimum minimizes a function with respect to a set of parameters. The function to be minimized must be provided by you and defined in a procedure.

If you want to call optimum you have to write:

```
{arg,obj,grad,retcode}=optimum(&func,start);
```

where `start` is the vector of initial values for the parameters and `&func` is a pointer to the procedure that computes the function to be minimized. This procedure must have one input argument, a vector of parameter values, and one output argument, a scalar value of the function evaluated at the input vector of parameter values.

The output consists of the following:

**arg**: vector of parameter estimates

**obj**: scalar, value of function at minimum

**grad**: gradient evaluated at **arg**

**retcode**: scalar, return code. If normal convergence is achieved then **retcode**=0, otherwise a positive integer is returned indicating the reason for the abnormal termination: **1**. forced exit; **2**. maximum iterations exceeded; **3**. function calculation failed; **4**. gradient calculation failed; **5**. Hessian calculation failed...

---

<sup>1</sup>Note that we must be careful with the possibility of having a local maximum or a local minimum. If so, the gradient will be zero at different points. Therefore, finding a zero of  $f'(x)$  could not be the same as finding the global maximum/minimum of  $f(x)$ .

We will make use of the `optmum` library in order to maximize two different functions.

## Exercise 1: The Univariate Case

1. Start a new program (`new; cls;`), create a scalar  $N = 1000$  and load the `pgraph` and `optmum` libraries.
2. Generate a random sample of size  $N$  from a normal distribution with  $\mu = 2$  and  $\sigma = 1$ . (*Hint: `rndn` command*)
3. Write a procedure with the function:

$$L(\mu) = - \sum_{i=1}^N \frac{(x_i - \mu)^2}{2}$$

where  $\mu$  is the input of the procedure,  $L(\mu)$  is the output of the procedure and  $x_i$  ( $i = 1, \dots, N$ ) is  $i$ -th element of the vector that you have generated in step 2.

```
[f=(-1/2)*(sumc((x-mu).^2));]
```

4. Plot the function. Where is the maximum? (*Hint: create the support vector with the `seqa` command and evaluate the function in this support*)
5. By using the `optmum` library, maximize the function  $L(\mu)$  with respect to  $\mu$ . (*Hint: remember that `optmum` minimizes functions, but maximizing  $f(x)$  is equivalent to minimizing  $-f(x)$* )
6. What is the result?
7. Repeat the step 5 but using very different initial guesses (`start`). What happens?

## Exercise 2: The Multivariate Case

1. Start a new program (`new;`) in a new file, create a scalar  $N = 1000$  and load the `optmum` library.
2. Generate a random sample of size  $N$  from a normal distribution with  $\mu = 2$  and  $\sigma = 1$ . (*Hint: `rndn` command*)
3. Write a procedure with the function:

$$L(\mu, \sigma) = -\frac{N}{2} \ln(2\pi) - \frac{N}{2} \ln(\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^N (x_i - \mu)^2$$

where  $\mu$  and  $\sigma$  are the inputs of the procedure,  $L(\mu, \sigma)$  is the output of the procedure and  $x_i$  ( $i = 1, \dots, N$ ) is  $i$ -th element of the vector that you have generated in step 2. (*Remark: the input of the procedure is now a 2 by 1 vector*)

```
[f=-(N/2)*ln(2*pi)-(N/2)*ln(sigma^2)+(-1/(2*(sigma^2)))*(sumc((x-mu).^2));]
```

4. By using the `optmum` library, maximize the function  $L(\mu, \sigma)$  with respect to  $\mu$  and  $\sigma$ . (*Hint: create the initial guess as follows: `start={5,5};`*)
5. What is the result?
6. Repeat the step 4 but using very different initial guesses (`start`). For example try `start={78,95};`. What happens?

# Appendix

## A1. Newton's Method in the Multivariate Case. An example.

Perhaps, the following example results familiar for you, as it is the maximum likelihood estimator for the mean and the variance in the normal linear regression model.

Suppose we have the following function  $\mathbf{f} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ :

$$\mathbf{f}(\mu, \sigma^2) = \begin{pmatrix} \frac{1}{\sigma^2} \sum (x_i - \mu) \\ -\frac{n}{2\sigma^2} + \frac{\sum(x_i - \mu)^2}{2\sigma^4} \end{pmatrix}$$

where  $x_i$  is the  $i$ th element of a random sample of size  $n$  drawn from a  $N(\mu, \sigma^2)$  population.

We want to find the zeros of  $\mathbf{f}$ , that is to say:

$$\begin{pmatrix} \frac{1}{\sigma^2} \sum (x_i - \mu) \\ -\frac{n}{2\sigma^2} + \frac{\sum(x_i - \mu)^2}{2\sigma^4} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

The 2x2 Jacobian matrix of the function  $\mathbf{f}(\mu, \sigma^2)$  would be:

$$\begin{pmatrix} -\frac{n}{\sigma^2} & -\frac{\sum(x_i - \mu)}{\sigma^4} \\ -\frac{\sum(x_i - \mu)}{\sigma^4} & \frac{n}{2\sigma^4} - \frac{\sum(x_i - \mu)^2}{\sigma^6} \end{pmatrix}$$

Therefore, taking the initial point  $(\mu^{(0)}, \sigma^{2(0)})'$  the first Newton's iteration would be:

$$\begin{pmatrix} \mu^{(1)} \\ \sigma^{2(1)} \end{pmatrix} = \begin{pmatrix} \mu^{(0)} \\ \sigma^{2(0)} \end{pmatrix} - \begin{pmatrix} -\frac{n}{\sigma^{2(0)}} & -\frac{\sum(x_i - \mu^{(0)})}{\sigma^{4(0)}} \\ -\frac{\sum(x_i - \mu^{(0)})}{\sigma^{4(0)}} & \frac{n}{2\sigma^{4(0)}} - \frac{\sum(x_i - \mu^{(0)})^2}{\sigma^{6(0)}} \end{pmatrix}^{-1} \begin{pmatrix} \frac{1}{\sigma^{2(0)}} \sum (x_i - \mu^{(0)}) \\ -\frac{n}{2\sigma^{2(0)}} + \frac{\sum(x_i - \mu^{(0)})^2}{2\sigma^{4(0)}} \end{pmatrix}$$

## A2. Numerical Differentiation (Mostly from Wikipedia, the free encyclopedia)

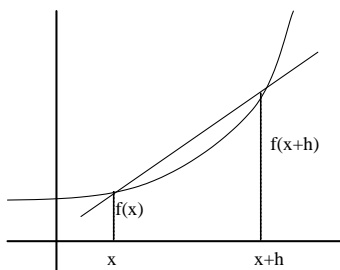
When you need to evaluate the derivatives of a given function in a given point, as we need for the implementation of the Newton's method, there are two options:

1. You can write the analytical derivative and then evaluate it at the point of interest.
2. You can use the numerical derivative at the point of interest.

The GAUSS commands `gradp` and `hessp` use the second option.

Numerical differentiation is a technique of numerical analysis to produce an estimate of the derivative of a mathematical function.

A simple two-point estimation is to compute the slope of a nearby secant line through the points  $(x, f(x))$  and  $(x + h, f(x + h))$ .  $h$  represents a small change in  $x$  and it can be either positive or negative.



The slope of this line is:

$$\frac{f(x+h) - f(x)}{h}$$

The slope of this secant line differs from the slope of the tangent line by an amount proportional to  $h$ . As  $h$  approaches zero, the slope of the secant line approaches the slope of the tangent line. Therefore, the true derivative of  $f$  at  $x$  is:

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

An important consideration in practice is how small of an  $h$  to choose. If chosen too small, the subtraction will yield a large rounding error (difference between the calculated approximation of a number and its exact mathematical value). If too large, the calculation of the slope of the secant line will be more accurate, but the estimate of the slope of the tangent by using the secant could be worse.