Direct Search Algorithm for Numerical Solution of GARCH(1,1) Problem

I.I. MAGLEVANNY¹, V.A. SMOLAR² and T.I. KARYAKINA¹

¹Volgograd State Social Pedagogical University, 27 Lenin Avenue, Volgograd 400131, Russia
²Volgograd State Technical University, 28 Lenin Avenue, Volgograd 400131, Russia

*Corresponding author

Keywords: Simulation of volatility, Econometric calculations with provision for machine arithmetic, GARCH(1,1) model.

Abstract. Problems of estimation and forecasting of volatility are of considerable interest in various applications of economy and finance. In this paper, we propose an open program code (by C ++ language), which implements a robust heuristic algorithm for the numerical evaluation of the parameters of the linear model GARCH (1,1), with provision for its machine-dependent aspects. The program can be easily implemented by any programming language, does not require third-party mathematical procedures, so as selection of the initial values of the parameters for the model volatility. A numerical example is presented.

Introduction

Time-variation in the conditional variance of financial time-series is important when pricing derivatives, calculating measures of risk, and hedging against portfolio risk. Therefore, there has been an enormous interest amongst researchers and practitioners to model the conditional variance. As a result, a large number of such models have been developed [1].

Different algorithms for building the econometric models and their optimization have a long history of providing and commercial applications and are quite sufficiently investigated from formally theoretical viewpoint [1]. But analysis of current state of modern free software in a region of econometric calculations may characterize it as being unsatisfactory. In different packages different program realizations are presented with different numerical optimization methods and parameters defined by default, that leads to different estimations of model parameters for the same input data. This fact was mentioned in papers [2,3], where numerous commercial program packages were tested and some reasons for result deviations were outlined. Proprietary, the absence of open code, indistinctness of input language, absence of information on assumptions used and machine-oriented criteria, impede seriously the possibilities of investigators in practical econometrics.

Peculiarities of Optimization Problems of Financial Mathematics

The optimization problems of financial mathematics are hard from the numerical viewpoint: 1) they are the constraind minimization problems (with constrains of nonequalities type), while the majority of proven and practically well-established minimization algorithms are unconstrained. That circumstance often is resulted in emergency stopping of the program execution due to the fact that the descent step generated by quasi-newton methods [4] leads to exceeding the limits of admitted region in parameter space; 2) in a number of issues not the likelihood function but conditional likelihood is maximized that needs information on history of the process. That is why at simulation the long realizations of financial time series is used, that formally is justified by “infinite memory” of recurrent relations while theoretical modeling of process history. But the finite character of machine arithmetic [4] implicitly leads to “noisiness” of objective function stipulated by peculiarities of given computer and prescribed accuracy of calculations. As a result from the computational viewpoint the objective
function becomes to be stochastic and non-differentiable. That circumstance makes precarious the use of iterative algorithms based on finite difference approximation of derivatives.

That is why the elaboration of adequate numerical methods in econometrics necessitates handling the direct search methods, that do not use explicitly the derivatives of optimized function [5]. We propose an open program code (by C++ language), which implements a robust heuristic algorithm for the numerical evaluation of parameters of nonlinear optimizing models, based on the concept of direct search with application of general scheme of alternating-variable descent method (the cyclic method of relaxation). By that for one-dimensional optimization we have modified the well known golden section method [6], but our algorithm does not require the unimodality of minimized function and after finite number of steps always finds the local minimum on a segment. The stopping criteria are machine- and problem-oriented which contemplate the peculiarities of machine arithmetic. The program realization was carried out by C++ that gives quick and effective executable code. This program may be easily realized by any programming language, it does not require third party mathematical procedures and choice of initial values for estimation of model (it is sufficient to establish the upper and lower limits values for estimated parameters) the minimal number of objective function calculations.

**GARCH(1, 1) Model**

As an example of application of our program the econometric model of GARCH(1,1) model is presented. The GARCH(1,1) model uses the maximal likelihood method and is realized in different application packages [1]. But so far the machine-oriented peculiarities of numerical realization of solution of problems of this type were not considered in literature practically.

Let the observed realization of stochastic process is represented by time series \( r_t, t = 0, \ldots, T-1 \), where \( T \) is the number of observations. The standard specification of GARCH(1,1) model has a form

\[
\begin{align*}
    r_t &= \mu + \epsilon_t, \\
    \epsilon_t &= \psi |\epsilon_{t-1} = \sigma_t \xi_t, \\
    \sigma_t^2 &= \omega + \alpha (r_{t-1} - \mu)^2 + \beta \sigma_{t-1}^2
\end{align*}
\]  

Here \( \psi_{t-1} \) is the information set, \( \mu \) is the conditional mathematical expectation, \( \sigma_t^2 \) is the conditional variance (volatility), \( \epsilon_t, \xi_t \) are the normalized normally distributed independent random variables. The value \( \epsilon_t \) represents the random component of the time series. The model parameters should be found in a region \( \omega > 0, \alpha > 0, \beta > 0, \alpha + \beta < 1 \). The principle of maximal likelihood is the problem of minimization of objective function

\[
f(\mu, \omega, \alpha, \beta) = \frac{1}{2} \left[ T \ln(2T) + \sum_{t=1}^{T} \left( \ln(\sigma_t^2) + \frac{(r_t - \mu)^2}{2\sigma_t^2} \right) \right] \rightarrow \min.
\]  

**C++ Program Code**

The text of the program realizing the direct search of minimum for function of many variables is presented below.

```c
#include <math.h>

// The one-dimensional minimization by golden section method
static inline double amax1(double x, double y){if(x>y) return x;else return y;}

static double mach(void)//returns the machine epsilon
double eps=1.,eps1=2;while(eps1!=1){eps/=2.;eps1=1.+eps;} return(2.*eps);}

// The function returns the minimal value minimization on the segment [a,b] at point xmin
double GoldSectDim1(double a, double b, //The boundaries of segment
```
double (*func)(double);  // The pointer to minimized function
double steptol;  // The stopping criterion on the step value
double typx;  // The character value of argument at minimum point
double &xmin;  // The minimum point, output value

if(steptol<=0.) steptol=sqrt(mach());
typx=(typx!=0.)?fabs(typx):1;
for(i=1;i<=3;i++) if(f[i]<fmin) {xmin=x[i];fmin=f[i];}
if(xmin<=x[1]) x[3]=x[2];
else x[0]=x[1];
x[1]=x[2]-x[0];
f[1]=func(x[1]);

while(fabs(x[2]-x[1])/amax1(fabs(x[2]),typx)>steptol){
    xmin=x[0];
fmin=f[0];
    for(i=1;i<=3;i++) if(f[i]<fmin) {xmin=x[i];fmin=f[i];}
    if(xmin<=x[1]) x[3]=x[2];
    else x[0]=x[1];
x[1]=x[2]-x[0];
f[1]=func(x[1]);
    if(x[1]>x[2]) {double a=x[1]; x[1]=x[2];
      x[2]=a;
      a=f[1]; f[1]=f[2]; f[2]=a; }
  }
  return func(xmin); // The program of alternating-variable descent method
}

The program of alternating-variable descent method
// Global variables
int ivar;
int NGlobal;
double *xGlobal;
double (*funcGlobal)(int Ndim, double x[]);

static double OneDimension(double y){
xGlobal[ivar]=y;
return funcGlobal(NGlobal,xGlobal);
}

int umstop(int n,double xc[], double xp[], double typx[], double steptol,
double fc,double fp, double typf, double ftol, double it, double itlim){
    if(it>=itlim) return 3;
    if(fabs(fp-fc)/amax1(typf,fp)<ftol) return 1;
    double stepmax=0.;
    for(int i=0;i<n;i++) stepmax=
      amax1(stepmax,fabs(xp[i]-xc[i])/amax1(fabs(xp[i]),typx[i]));
    if(stepmax<=steptol) return 2;
    return 0;
}

double DirectSearch(int Ndim, //The dimension of parameters space
        double a[], //The lower bounds in parameter space
        double b[], //The upper bounds in parameter space
        double (*func)(int Ndim, double x[]), //The pointer to objective function
        double steptol, //The stopping criterion on step value
        double ftol, //The stopping criterion on objective function value
        double itlim, //The maximum iteration number
        double typx[], //The typical values of argument
        double typf, // The typical values of objective function
        double &xmin[], //The minimum point in parameter space
        int &termcode, //The stopping code, showing the reason to stop.
        double &it//The number of iterations accomplished
        )
{::NGlobal=Ndim;::funcGlobal=func;if(steptol<=0.) steptol=sqrt(mach());
if(ftol<=0.) ftol=steptol*steptol;if((typf!=0.)typf=fabs(typf);else typf=1.;
for(int i=0;i<Ndim;i++) typx[i]=(typx[i]!=0.)?fabs(typx[i]):1;
double *xp=new double[Ndim];
for(i=0;i<Ndim;i++) xp[i]=0.5*(a[i]+b[i]);
fc=funcGlobal(Ndim,xc);termcode=0;

// Iterations
while(1){for(i=0;i<Ndim;i++){ivar=i;fp=GoldSectDim1(a[i],b[i],OneDimensional,0.,typx[i],xk);
xp[i]=xk;}
//Check the stopping condition
termmcode=umstop(Ndim,xc,xp,typx,steptol,fc,fp,typf,ftol,it,itlim);
if(termcode)break;it+=1;fc=fp;for(i=0;i<Ndim;i++)xc[i]=xp[i];}//end while
for(i=0;i<Ndim;i++)xmin[i]=xp[i];delete xp;delete xc;return fp;}

Below you’ll find the scaling and stopping parameters and others and recommendations on choice of appropriate values. The massive typx[n] is the input parameter, where the component x[i] is the incorrect results and, as a consequence, to incorrect statistical conclusions and recommendations. positive scalar showing the character value of x[i]. It is important to set the values typx[i], when it is expected then the values x[i] will differ significantly, in that case the program may work better then at typx[i]=1, which is set by default.

steptol is the positive scalar, defining the range, in which the the scaled distance between two successive approximations is considered sufficiently close to zero to stop the algorithm by condition

\[
\max \left\{ \frac{|x_i[i] - x_\ast[i]|}{\max(|x_i[i]|,typx[i])} \right\} \leq \text{steptol}
\] (3)

Here \(x_i\) and \(x_\ast\) are accordingly the initial an final poits at next iteration step. Thed value steptol 1 should be small. The proposed value by default is \(\text{steptol} = \sqrt{\text{machine epsilon}}\).

itlimit – the positive scalar, showing maximal number of iteration which may be accomplished.
termmcode – shows the reason of algorithm stop: termcode=1 – the algorithm is stopped by condition (6), termcode=2 – the algorithm is stopped by condition (3), termcode=3 – the limit of iteration was exceeded.

The text of the program realizing the estimation of paremeters of GARCH(1,1) model is presented below.

//The global parameters for calculation of objective function
int T;//The number of samples form time series
double *R;//The time series
double aminus1_2=0,sigminus1_2=0;
//The objective function
double func(int n,double zz[]){double
mu=zz[0],omega=zz[1],beta=zz[3],alpha=zz[2],aminus1_2,sigminus1_2;
if(n==5)aminus1_2=sigminus1_2=zz[4];else aminus1_2=sigminus1_2=::sigminus1_2;
double sum=T*log(2*M_PI);int t;
double sig2t=omega+alpha*aminus1_2+beta*sigminus1_2,at,addend;
for(t=0;t<T;t++){
if(t>0){at=R[t-1]-mu;sig2t=omega+alpha*at*at+beta*sig2t;}
at=R[t]-mu;sum+=log(sig2t)+at*at/sig2t;}
return 0.5*sum;}

void main(void)
{int n=4;// The number of estimated parameters
double lowerBounds[4]={0.001,0.001,0.001,0.001}; //The lower limits of parameter space
double upperBounds[4]={10.,10.,0.999,0.999}; // The upper limits of parameter space
double steptol=0,ftol=0,itlim=1000; //The tolerances
double typx[4]={0,0,0,0}; //The typical values of parameters
double typf=0; // The typical value of function
double xnew[4]; //The minimum value, output parameter
int termcode; //The stopping code which has stopped the algorithm
double it;//The iteration number
//Find the estimations of GARCH(1,1) model parameters
double fnew=
DirectSearch(n,lowerBounds,upperBounds,func,steptol,ftol,itlim,typx,typf,xnew,termcode,it);
//The output value fnew is the value of objective function at minimum point xnew
}

For testing the program the data and results presented in [7] based on optimization of logarithm of likelihood function with use of analytical expressions for gradient and gaussian were used. The following estimations of model parameters were received: $\mu =-0.000619$, $\omega =0.01076$, $\alpha =0.153134$, $\beta =0.80597$, what coincides with results of [7] up to 5 significant digits.

Summary

Many researches note that non-critical use of software of “black box” type leads frequently to incorrect results and, as a consequence, to incorrect statistical conclusions and recommendations. Thus the development of affordable open compact program code is actual for solution of particular econometrical problems.

References