Jump-Diffusion Calibration using Differential Evolution

Ardia, David and Ospina, Juan and Giraldo, Giraldo

aeris CAPITAL AG Switzerland, School of Statistics, National University of Colombia, Medellin, Colombia

16 October 2010

Jump-Diffusion Calibration using Differential Evolution[☆]

David Ardia[a,*], Juan David Ospina Arango[b], Norman Diego Giraldo Gómez[b]

[a] *aeris CAPITAL AG, Switzerland*
[b] *School of Statistics, National University of Colombia, Medellin, Colombia*

Abstract

The estimation of a jump-diffusion model via Differential Evolution is presented. Finding the maximum likelihood estimator for such processes is a tedious task due to the multimodality of the likelihood function. The performance of the Differential Evolution algorithm is compared to standard optimization techniques.

Keywords: Jump-diffusion, maximum likelihood, optimization, Differential Evolution

1. Jump Diffusion models

Jump-diffusion models are continuous-time stochastic processes introduced in quantitative finance by Merton (1973), extending the work on option pricing by Black and Scholes (1973). These models are used to reproduce stylized facts observed in asset price dynamics, such as mean-reversion and jumps. Various specifications have been proposed in the literature; see Cont and Tankov (2004) for an extensive review. In what follows, we consider the standard jump-diffusion model with time invariant coefficients, constant volatility and Gaussian distributed jumps. This model can be expressed as

$$dX(t) = X(t)\ (\mu dt + \sigma dW(t) + J(t)dP(t))\ , \tag{1}$$

where $X(t)$ is the process that describes the price of a financial asset, with $\mathbb{P}\left(X(0) > 0\right) = 1$, $\mu \in \mathbb{R}$ is the process drift coefficient and $\sigma^2 > 0$ is the instantaneous process variance, $W(t)$ is a standard Wiener process, $P(t)$ is Poisson process with constant intensity $\lambda > 0$ and $J\left(t\right)$ is the process generating the jump size, that together with $P(t)$ forms a compound Poisson process. The solution of (1) is

$$X(t) = X(0)\ \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W(t) + \sum_{k=1}^{P(t)} Q_k\right), \tag{2}$$

where Q_k is implicitly defined such that $J\left(T_k\right) = \exp(Q_k) - 1$, T_k being the time at which the kth jump of the Poisson process occurs; see Ospina Arango (2009). If $P(t) = 0$, the sum is zero by convention. We assume that Q_k is an independent and identically normally distributed sequence with mean μ_Q and variance σ_Q^2.

[☆]The views expressed in this paper are the sole responsibility of the authors and do not necessarily reflect those of aeris CAPITAL AG or any of its affiliates. Any remaining errors or shortcomings are the authors' responsibility. The authors are grateful to Kris Boudt, Michel Dubois, Lennart F. Hoogerheide and Enrico Schumann for helpful comments.

[*]Corresponding author; tel.: +41 (55)5 111 222; fax. +41 (55)5 111 223.
Email addresses: da@aeris-capital.com (David Ardia), jdospina@gmail.com (Juan David Ospina Arango), ndgirald@unal.edu.co (Norman Diego Giraldo Gómez)

The log-return of $X(t)$ over a Δt-period is defined as $\Delta Y(t) \doteq \Delta \log X(t) = \log X(t+\Delta t) - \log X(t)$ and, from (2), its dynamic is given by

$$\Delta Y(t) = \left(\mu - \frac{\sigma^2}{2}\right)\Delta t + \sigma \Delta W(t) + \sum_{k=1}^{\Delta P(t)} Q_k \,. \tag{3}$$

It can be shown that the distribution of $\Delta Y(t)$ is an infinite mixture of Gaussian distributions which renders the estimation intractable; see Beckers (1981) and Honoré (1998). Ball and Torous (1983) present a simplified version by assuming that if the jumps' occurrence rate is small, then in a sufficiently short time period only one jump can occur. In this case, $\Delta P(t)$ can be approximated by a Bernoulli random variable for small $\lambda \Delta t$, and the density of $\Delta Y(t)$ is

$$f_{\Delta Y}(y) = (1 - \lambda \Delta t)\, f_{\Delta D}(y) + \lambda \Delta t \,(f_{\Delta D} * f_Q)\,(y)\,, \tag{4}$$

where $f_{\Delta D}$ is the density of the diffusion part and f_Q is the density assumed for the jumps. The convolution operator $*$ is defined as $(f_{\Delta D} * f_Q)(y) \doteq \int_{-\infty}^{\infty} f_{\Delta D}(u) f_Q(u-y) du$. The diffusion density $f_{\Delta D}$ is a normal density with mean $(\mu - \sigma^2/2)\Delta t$ and variance $\sigma^2 \Delta t$. If f_Q is also a normal density, then $(f_{\Delta D} * f_Q)$ is normal with mean $(\mu - \sigma^2/2)\Delta t + \mu_Q$ and variance $\sigma^2 \Delta t + \sigma_Q^2$. For a series of observed log-returns $\Delta y_1, \ldots, \Delta y_T$, the log-likelihood of the model parameters $\theta \doteq (\mu, \sigma, \lambda, \mu_Q, \sigma_Q)'$ is obtained in a straightforward manner from (4) as

$$\log \mathcal{L}\,(\theta \,|\, \Delta y_1, \ldots, \Delta y_T) = \sum_{t=1}^{T} \log f_{\Delta Y}\,(\Delta y_t \,|\, \theta)\,, \tag{5}$$

and the maximum likelihood estimator $\hat{\theta}$ is obtained by maximizing (5). Kiefer (1978) shows that there does not exist a unique optimum $\hat{\theta}$ in such a mixture setting. The estimation of the model is therefore a difficult task and a robust global optimizer such as Differential Evolution is required.

2. Differential Evolution

Differential Evolution (DE) is a search heuristic introduced by Storn and Price (1997) and belongs to the class of genetic algorithms. The algorithm uses biology-inspired operations of crossover, mutation, and selection on a population in order to minimize an objective function over the course of successive generations. Its remarkable performance as a global optimization algorithm on continuous problems has been extensively explored; see, e.g., Price et al. (2006).

Let NP denote the number of parameter vectors (members) $\boldsymbol{x} \in \mathbb{R}^d$ in the population. In order to create the initial generation, NP guesses for the optimal value of the parameter vector are made, either using random values between bounds or using values given by the user. Each generation involves creation of a new population from the current population members $\{\boldsymbol{x}_i \,|\, i = 1, \ldots, NP\}$, where i indexes the vectors that make up the population. This is accomplished using *differential mutation* of the population members. An initial mutant parameter vector $\boldsymbol{v}_i$ is created by choosing three members of the population, $\boldsymbol{x}_{r_0}$, $\boldsymbol{x}_{r_1}$ and $\boldsymbol{x}_{r_2}$, at random. Then $\boldsymbol{v}_i$ is generated as $\boldsymbol{v}_i \doteq \boldsymbol{x}_{r_0} + F \cdot (\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2})$, where F is a positive scale factor whose effective values are typically less than one. After the first mutation operation, mutation is continued until d mutations have been made, with a crossover probability $CR \in [0, 1]$. The crossover probability CR controls the fraction of the parameter values that are copied from the mutant. If an element of the trial parameter vector is found to violate the bounds after mutation and crossover, it is reset in such a way that the bounds are respected. Then, the objective function values associated with the children are determined. If a trial vector has equal or lower objective function value than the previous vector

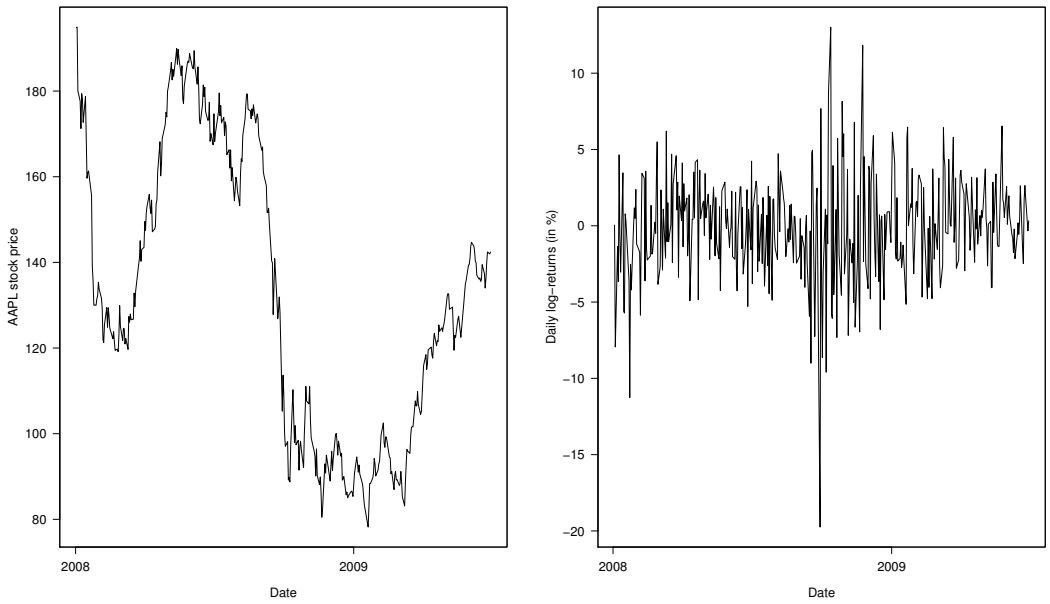

Figure 1: Prices of Apple Inc. stock (left) and log-returns (right) for a period ranging from January 1, 2008, to June 30, 2009, for a total of 377 observations.

it replaces the previous vector in the population; otherwise the previous vector remains. For more details, see Price et al. (2006) and Storn and Price (1997).

Several implementations of DE are currently available. A web-based list of DE programs for general purpose optimization is maintained by Rainer Storn at `http://www.icsi.berkeley.edu/~storn/code.html`. In what follows, we will rely on the package **DEoptim** (Ardia and Mullen, 2009; Mullen et al., 2009; Ardia et al., 2010) which implements DE in the R language and environment for statistical computing (R Development Core Team, 2009). R enables rapid prototyping of objective functions, access to a wide array of tools for statistical modeling, and ability to generate customized plots of results with ease. **DEoptim** is available at `http://cran.r-project.org/web/packages/DEoptim/`.

3. Illustration

We illustrate the performance of DE by fitting model (3) to financial data. We consider daily log-returns of the stock Apple Inc. (AAPL) for a period ranging from January 1, 2008, to June 30, 2009, for a total of 377 observations. The data set is downloaded from `http://www.finance.yahoo.com`. The stock prices and the corresponding log-returns (in percent) are displayed in Figures 1.

In order to find the maximum likelihood estimator we minimize the negative value of the log-likelihood function (NLL) given in (5). The function `DEoptim` of the package **DEoptim** is run with the default parameters (i.e., 200 populations of size $NP = 50$ are generated with $F = 0.8$ and $CR = 0.5$); see the Appendix for the R code. For comparison purposes, the objective function is also minimized using standard optimization functions available in R. More specifically, we use the function `optim` with method `L-BFGS-B`, and the functions `nlminb` and `constrOptim`. The method `L-BFGS-B` is a box-constrained quasi-Newton method which uses function values and numerical gradients to build up a picture of the surface to be optimized. `nlminb` offers unconstrained and constrained optimization using PORT routines. `constrOptim` minimizes a function subject to linear inequality constraints using an adaptive barrier algorithm. For all methods we use the default

values of the control parameters. Lower boundaries are set to $\theta_{\text{LB}} \doteq (1, -10, 0.0001, -10, 0.0001)'$ and upper boundaries to $\theta_{\text{UB}} \doteq (100, 10, 10, 10, 10)'$.

We run the estimation 100 times for all optimization routines and use random starting values in the feasible parameter set when needed. Boxplots of the NLL values at optimum for the four optimizers are displayed in Figure 2. We notice that `L-BFGS-B`, `nlminb` and `constOptim` converge towards local optima for several runs; `nlminb` exhibits the best performance among them with 50% of the runs converging towards the *global* optimum at NLL = -754.66 (indicated by a horizontal dashed line). On the other hand, `DEoptim` is more stable over the runs and always converges towards to global optimum. Note that the global optimum is obtained by `DEoptim` after a longer run of 500 populations of size $NP = 100$. The global maximum likelihood estimator is $\hat{\theta} = (16.72, 0.0862, 0.4607, -0.0115, 0.0681)$, indicating around 17 jumps on average per year, with an average size of $\exp(-0.0115 + 0.0681^2/2) - 1 \approx -0.91\%$.

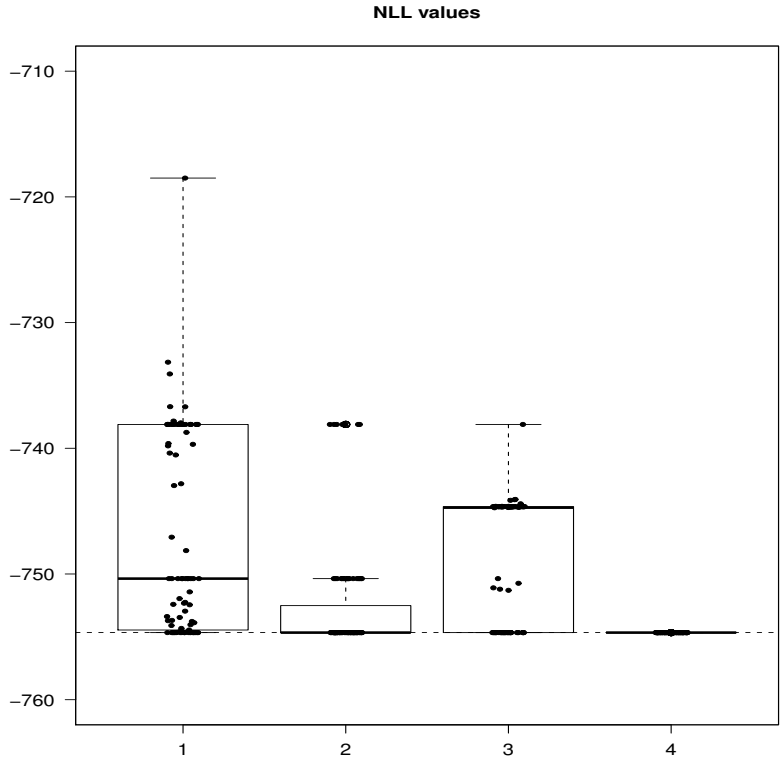

Figure 2: Boxplots of the 100 negative values of the log-likelihood function (NLL) at optimum $\hat{\theta}$ obtained by the various optimizers. (1) function `optim` with method `L-BFGS-B`, (2) function `nlminb`, (3) function `constrOptim`, (4) function `DEoptim`. Starting values are generated randomly in the feasible parameter set. Lower boundaries are set to $\theta_{\text{LB}} \doteq (1, -10, 0.0001, -10, 0.0001)'$ and upper boundaries to $\theta_{\text{UB}} \doteq (100, 10, 10, 10, 10)'$. Jittered black dots report the NLL values obtained for each run. Note that the graph is vertically scaled to the [-760,-710] interval (i.e., some outliers are not reported).

4. Conclusions

This note introduced Differential Evolution, a heuristic evolutionary method for global optimization that is effective on many problems of interest in science and technology. We illustrated the power of DE optimization through the use of the R package **DEoptim** by fitting a jump-diffusion model to financial data. Finally, we refer the reader to Ospina Arango (2009) for a more extensive study of jump-diffusion calibration using Differential Evolution.

References

Ardia, D., Boudt, K., Carl, P., Mullen, K., Peterson, B., 2010. Differential Evolution (**DEoptim**) for non-convex portfolio optimization.

Ardia, D., Mullen, K., 2009. **DEoptim**: Differential Evolution Optimization in R. R package version 2.00-06. URL `http://CRAN.R-project.org/package=DEoptim`

Ball, C. A., Torous, W. N., 1983. A simplified jump process for common stock returns. Journal of Financial and Quantitative Analysis 18 (1), 53–65.

Beckers, S., 1981. A note on estimating the parameters of the jump-diffusion model of stock returns. Journal of Financial and Quantitative Analysis 16, 127–140.

Black, F., Scholes, M., 1973. The pricing of options and corporate liabilities. The Journal of Political Economy 81 (3), 637–654.

Cont, R., Tankov, P., 2004. Financial Modelling with Jumps. Chapman & Hall / CRC Press, ISBN 1584884134.

Honoré, P., 1998. Pitfalls in estimating jump-diffusion models.

Kiefer, N., 1978. Discrete parameter variation: Efficient estimation in diffusion process. Econometrica 46 (2), 427–434.

Merton, R. C., 1973. Theory of rational option pricing. The Bell Journal of Economics and Management Science 4 (1), 141–183.

Mullen, K. M., Ardia, D., Gil, D. L., Windover, D., Cline, J., 2009. **DEoptim**: An R package for global optimization by Differential Evolution.

Ospina Arango, J. D., 2009. Estimación de un modelo de difusión con saltos con distribución de error generalizada asimétrica usando algoritmos evolutivos. Master's thesis, Universidad Nacional de Colombia.

Price, K. V., Storn, R. M., Lampinen, J. A., 2006. Differential Evolution: A Practical Approach to Global Optimization. Springer-Verlag, Berlin, Germany, ISBN 3540209506.

R Development Core Team, 2009. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0. URL `http://www.R-project.org`

Storn, R., Price, K., 1997. Differential Evolution – A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11 (4), 341–359.

Estimation of a jump-diffusion process in R with the package DEoptim

```
## ==== LOAD PACKAGES ====
library("tseries")
library("DEoptim")

## ==== MIXTURE DENSITY AND LIKELIHOOD ====
## mixture density
fdy <- function(dy, dt, lambda, mu, sigma, muq, sigmaq) {
  mu1  <- (mu - sigma^2 / 2) * dt
  mu2  <- (mu - sigma^2 / 2) * dt + muq
  sig1 <- sigma * sqrt(dt)
  sig2 <- sqrt(sigma^2 * dt + sigmaq^2)
  pdf1 <- dnorm(dy, mean = mu1, sd = sig1)
  pdf2 <- dnorm(dy, mean = mu2, sd = sig2)
  pdf  <- (1 - lambda * dt) * pdf1 + (lambda * dt) * pdf2
  return(pdf)
}

## negloglikeliood function
negloglik <- function(theta, dy, dt) {
  L <- fdy(dy = dy, dt = dt, lambda = theta[1],
           mu = theta[2], sigma = theta[3],
           muq = theta[4], sigmaq = theta[5])
  nll <- -sum(log(L))
  if (is.nan(nll) | is.na(nll) | is.infinite(nll)) {
    nll <- 1e10
  }
 return(nll)
```

```
}

## ==== DATA ====
## load data set (need a web connection)
x  <- get.hist.quote(instrument = "AAPL",
      start = "2008-01-01", end = "2009-06-30",
      retclass = "zoo", quote = "AdjClose", compression = "d")

## log-returns
dy <- diff(log(as.vector(x)))
## assume 255 days in a year (trading days)
dt <- 1 / 255

## ==== DEOPTIM ESTIMATION ====
set.seed(1234)
outDE <- DEoptim(negloglik,
         lower = c(  1, -10, 1e-4, -10, 1e-4),
         upper = c(100,  10,   10,  10,   10),
         control = list(itermax = 500, NP = 100), dt = dt, dy = dy)
summary(outDE)
plot(outDE, type = 'l')
plot(outDE, plot.type = 'bestvalit', type = 'l')
```