A Fast and Stable Heston Model Calibration on the GPU

Michael Aichinger[1], Andreas Binder[2],
Johannes Fürst[2], and Christian Kletzmayr[2]

[1] Johann Radon Institute for Computational and Applied Mathematics (RICAM), Austrian Academy of Sciences, Altenberger Strasse 69, A-4040 Linz, Austria
michael.aichinger@ricam.oeaw.ac.at
[2] MathConsult GmbH, Altenberger Strasse 69, A-4040 Linz, Austria

Abstract. For the analysis of many exotic financial derivatives, the Heston model, a stochastic volatility model, is widely used. Its specific parameters have to be identified from sets of options market data with different strike prices and maturities, leading to a minimization problem for the least square error between the model prices and the market prices. It is intrinsic to the Heston model that this error functional typically exhibits a large number of local minima, therefore techniques from global optimization have to be applied or combined with local optimization techniques to deliver a trustworthy optimum. To achieve results in reasonable time, we approach as follows: (1) For the evaluation of the objective function, we use a Fourier cosine method, optimized for parallelization, and (2) the local/global optimization scheme is carried out on parallel architectures. Results are reported for a multi GPU server and a multicore SGI Altix 4700.

Keywords: Heston Model, Calibration, Cosine Method, GPU, Multicore CPU.

1 Model and Methods

1.1 The Heston Model

The Heston stochastic volatility model [1] relaxes the constant volatility assumption in the classical Black Scholes model by incorporating an instantaneous short term variance process (CIR)

$$\begin{aligned} dS_t &= r(t)S_t dt + \sqrt{v_t}S_t dW_t^1 \\ dv_t &= \kappa(\theta - v_t)dt + \lambda\sqrt{v_t}dW_t^2 \qquad v_0 \geq 0 \end{aligned} \tag{1}$$

where r denotes the domestic yield curve, v_t denotes the stock price variance, dW_t^i are standard Brownian motions with correlation ρ, κ is the the mean reversion parameter, Θ is the long term level and λ is the volatility of variance parameter. The variance process is always positive if $2\kappa\theta > \sigma^2$ (Feller condition). The characteristic function of the Heston model is analytically available. (see for example [2]).

M.R. Guarracino et al. (Eds.): Euro-Par 2010 Workshops, LNCS 6586, pp. 431–438, 2011.

1.2 The Calibration Problem

Before complex instruments can be priced, the model parameters p_i have to be calibrated to market prices of liquid instruments, e.g. by minimizing the least squares error of model prices compared to market prices [2].

$$\sum_j ||V_j^{Mod}(\{p_i\}) - V_j^{Mar}||^2 \to \min \tag{2}$$

where the V_j are single calibration instruments, i.e. European call/put options with with different strike prices and different expiries. One of the difficulties in solving the resulting inverse problem is that the market information is insufficient to completely identify a pricing model, which means that several sets of model parameters may reproduce the market prices, leading to ill-conditioned problems and model uncertainty. To overcome the ill-conditioned nature of the problem, regularization methods can be used to guarantee stable parameters [2]. The corresponding optimization problem, formulated as a minimization problem of a least squares functional, is not necessarily convex as a function of the model parameters and is therefore hard to solve. It may happen that several, even a large amount of local minima exist. Another problem that may arise is that the objective function may exhibit an extremely "flat" behaviour such that even if only a unique minimum exists, a parameter set is accepted as optimal although it is far away from the true optimum. Two groups of algorithms can be applied to solve these optimization problems. The first group are locally convergent algorithms which will find a minimum but not necessarily the global one (e.g. Levenberg-Marquardt). The second group of algorithms are globally convergent, which should theoretically (CPU time going to infinity) be able to find the global minimum (simulated annealing, particle swarm methods, evolutionary algorithms, ...) [4]. The disadvantage of the second group is the enormous amount of computation time in comparison to the algorithms of the first group to obtain results. Our key idea to overcome this drawback is to take the best parts of both worlds to improve the quality of the results for the first group and to speed up the computation for algorithms of the second group. We start with quasi-random low-discrepancy sequences to get a good coverage of the parameter space and evaluate the residual function (2) for each of these N_I points. After sorting, a gradient based algorithm (Levenberg-Marquardt) is started from the N_B best points of the initial set. The Levenberg-Marquardt algorithm is an iterative technique that locates the minimum of a function that is expressed as the sum of squares of nonlinear functions. It has become a standard technique for nonlinear least-squares problems and can be thought of as a combination of steepest descent and the Gauss-Newton method [9].

Special attention is turned to the stability of the optimal parameters and of the prices for the exotic instruments obtained under these parameters with respect to time. Crucial for this kind of algorithm is the fast evaluation of the residual value as well as the corresponding gradients with respect to the model parameters.

1.3 Fast Evaluation of European Style Options and Their Derivatives

If the model under consideration does not provide an analytic solution for the European option price efficient numerical methods are required to price European options in order to calibrate the parameters of the model to given market data.

Starting point is the risk neutral valuation formula

$$v(x,t_0) = e^{-r\Delta t}\mathbb{E}Q[v(y,T)|x] = e^{-r\Delta t}\int_{\mathbb{R}} v(y,T)f(y|x)dy \tag{3}$$

where v denotes the option value, Δt is the difference between the maturity T and the valuation date t_0, $f(y|x)$ is the probability density of y given x and r is the risk neutral interest rate. Some of the state of the art numerical integration techniques have in common that they rely on a transformation to the Fourier domain [7]. The main reason for this is that the probability density function appears in the integration in the original pricing domain but is not known analytically for many important pricing processes. Instead, the characteristic functions of these processes, the Fourier transforms of the respective density functions, can often be expressed analytically. The density and its characteristic function form a Fourier pair and the idea of the Fourier-Cosine method [6] is to reconstruct the whole Fourier integral in - not only the integrand - from its Fourier-cosine expansion. In the case of European options one ends up with

$$L[v(x,t_0)] = e^{-r\Delta t}\sum_{k=0}^{\prime N}\mathrm{Re}\left\{L\left[[\phi\left(\frac{k\pi}{b-a},x\right)\right]\exp\left(-i\frac{ka\pi}{b-a}\right)\right\}V_k \tag{4}$$

V_k depends on the payoff of an European option at maturity T. $L[\cdot]$ denotes an operator being id, if the option value or $L = \frac{\partial}{\partial p_i}$ if the gradient of the option value with respect to the i^{th} model parameter should be calculated.

2 Implementation Details

We have implemented the calibration routine on two parallel hardware platforms, a CPU multicore server SGI Altix 4700 with 256 cores in clusters of four with one terabyte memory and a GPU server with two C1060 (240 streaming processor cores) and one GTX 260 (192 CUDA cores) graphic cards from Nvidia, two Intel E5520 CPUs and 24 gigabyte memory. For parallelization Open-MP has been used on the SGI machine and a combination of Open-MP and the Nvidia Cuda framework has been used on the GPU Server.

The key point in programming effective GPU algorithms is the optimal memory management. We used constant memory (which is read-only but cached) for look-up tables and ensured that all memory transactions on global memory are coalesced. Special emphasize has been put on the minimization of memory transfers between host and global memory since this is bottleneck in terms of

speed. To achieve the highest performance, it is necessary to get the best combination of the number of registers, the number of threads and the amount of shared memory. All GPU computations has been performed in double precision to ensure the accuracy necessary for calibration purposes. However, at this point in time double precision is much slower than single precision (typically ba a factor of eight). Switching to new hardware with FERMI technology will probably speed up the whole algorithm.

On each of our testing platforms the function evaluations for the N_I starting points are distributed between the processing units - GPUs or Cores. Whereas the parallelization is straight forward in the CPU case, details of the implementation on the GPU can be found in figure (1). The residuals (2) are sorted on

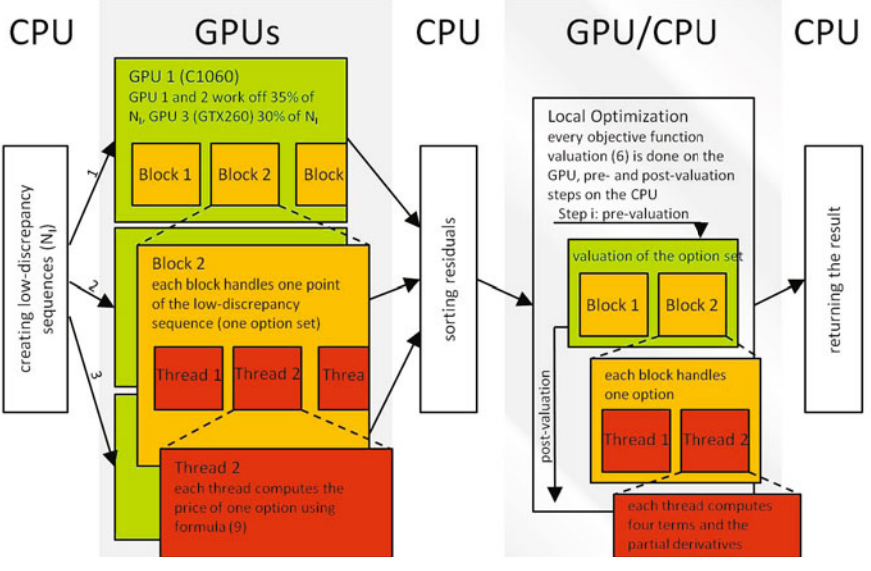

Fig. 1. Implementation details of the calibration algorithm - a combination of quasi-random low-discrepancy sequences and a gradient based Levenberg-Marquardt algorithm

the CPU, and Levenberg-Marquardt algorithms are started from the N_B points with the lowest residuals. Within the Levenberg-Marquardt algorithm, it depends whether the evaluation of the objective function (2) and the derivatives with respect to the model parameters are again performed in a parallel manner. In the case of the Multicore CPU machine each core is used to perform a Levenberg-Marquardt optimization and the residual and the derivatives are computed sequentially. Also in the GPU case, each GPU performs such an optimization but the threads and multiprocessors on each GPU are used to parallelize the summation in (4).

The main difference in the implementation between the GPU and the CPU is the number of summands used in (4). To get the best performance on the GPU,

each thread on each Multiprocessor should be equally busy. Therefore it is advantageous to keep the number of summands N fixed. On the other hand, on a CPU or a MultiCore CPU, a processing unit can start with the next parameter set after finishing a valuation task therefore it is advantageous to have a stopping criterion. We use the absolute value of the characteristic function, i.e. the envelope function as stopping criterion and abort the summation if $|\phi(u)| < \epsilon$.

3 Results

Market Data
All results presented in this section are calculated for a set of options on the FTSE-100 index for May 1^{st} 2008. The forward rates (GBP) on that date ranged between $3.4\%\, p.a.$ and $4.52\%\, p.a.$ (continuous compounding).

Evaluation of Points in the Parameter Space
We start with reporting the performance of the first part of our combined algorithm - the evaluation of the residual for a large number of points in the five dimensional parameter space. As mentioned before, the speed of the calculation of one option set (different strikes and maturities) is crucial. Therefore we report the performance using the GPU compared with a single core of the CPU of our GPU Server and using several CPUs on our multicore CPU server in Tab.1. Starting from N_I points we evaluate the objective function for each of these parameter sets. Increasing N_I allows to improve the best residual, as $N_{I_1} \subset N_{I_2}$ if $|N_{I_2}| > |N_{I_1}|$ when using quasi-random low discrepancy sequences (here Sobol points). Unfortunately it is possible that the optimal parameter set violates the Feller condition.

Table 1. This table shows the average computation time (in milliseconds) for one option set (256 options) on the GPU (C1060),on a single core of the GPU server, on three GPUs (2xC1060+1GTX260) and for 8 and 32 cores on the Altix for a fixed number of summands denoted by $N = 512$ as well as for an abort criterion denoted by $\epsilon = 10^{-8}$

N_I	CPU-N	GPU-N	CPU-ϵ	8-CPU-N	32-CPU-N	8-CPU-ϵ	32-CPU-ϵ	3-GPU-N
2^7	217.24	4.05	15.59	9.46	3.52	1.40	1.42	4.19
2^8	217.02	3.29	16.29	9.55	2.75	1.41	0.68	2.34
2^{10}	216.97	2.85	16.92	9.26	2.39	1.14	0.50	1.30
2^{12}	217.03	2.71	17.10	9.27	2.38	1.05	0.47	1.02
2^{14}	217.00	2.68	16.96	9.20	2.34	1.03	0.47	0.97
2^{16}	216.96	2.67	16.94	9.23	2.33	0.93	0.45	0.95

The Gradient Algorithm and the Feller trap
Next we will focus on the gradient part of the calibration algorithm (Levenberg Marquardt). To check whether the analytical derivatives are advantageous to the numerical ones, calculated via finite difference formulas, we have used

our gradient based optimization routine. We have started from the following point in the five dimensional parameter space: $(2.5, 0.5, 0.5, -0.5, 2.5)$ and performed 50 iterations of the Levenberg-Marquardt algorithm (see Tab.2). Again a major drawback for the performance of the Levenberg-Marquardt is the Feller condition. Up to our experience, for some of the starting points the algorithm converges towards parameter sets not satisfying the Feller condition. When hitting the boundary of the area defined by the Feller condition we add/subtract a small number to move the parameter set to the allowed region - slowing down the performance of the whole algorithm. As Tab.3 shows this is especially true for the algorithm using analytic derivatives. Furthermore the values for the optimal parameters can be different for each of the methods used for calculating the derivatives (compare with Tab.2).

Table 2. This table shows the residual and the corresponding calibrated parameter set using the Levenberg Marquardt algorithm starting from $(2.5, 0.5, 0.5, 0.5, -0.5)$ and fixing the number of iterations to 50. The different lines correspond to the different methods for calculating the derivatives with respect to the Heston parameters: analytic means calculation using (4), bd means using a simple backward difference quotient (first order), fd means forward difference quotient (first order), cd means central difference quotient (second order) and cd2 means a higher order central difference quotient (fourth order).

Method	Residual	κ	θ	λ	ρ	v_0
analytic	0.00181063	0.092252	0.486551	0.695210	-0.384361	0.048140
fd	0.00181035	0.090878	0.493407	0.694971	-0.384394	0.048132
bd	0.00181089	0.093595	0.480052	0.695447	-0.384329	0.048147
cd	0.00181063	0.092256	0.486530	0.695211	-0.384361	0.048140
cd2	0.00181063	0.092249	0.486566	0.695210	-0.384361	0.048137

The Overall Calibration Algorithm

Finally we report some results for the whole calibration algorithm - the combination of the quasi-random low discrepancy sequences and the gradient based algorithm, the overall performance on our different computing systems and some comparison with other optimization algorithms. We have used a simulated annealing (SA) algorithm, a direct search simulated annealing (DSSA) algorithm and a differential evolution (DE) algorithm [4],[5] to get values for comparison. All of these algorithms give comparable results for the value of the objective function (2) but the parameter sets are different. The reason for this will be reported elsewhere [10]. In Tab.4 the results for these global optimization algorithms are reported. Furthermore we have added results obtained with our combination algorithm and emphasize the enormous advantage in performance obtained with our method. Table 5 shows the dependence of the residual and the optimal parameters from the starting points N_I and the number of points N_B chosen after sorting to start the gradient based algorithm from.

Table 3. This table shows the influence of the Feller condition on the results when starting from a certain point of the parameter space $(2.5, 0.5, 0.5, 0.5, -0.5)$. The number of iterations for the Levenberg-Marquardt algorithm has been fixed to 20 for these calculations.

Method	Feller y/n	Time[s]	κ	θ	λ	ρ	v_0	R
analytic	y	286	5.16374	0.0535189	0.743359	-0.656857	0.0532264	0.00317053
analytic	n	13	0.490489	0.128122	0.757395	-0.381153	0.0489238	0.00175283
fd	y	142	4.94054	0.0557186	0.741914	-0.634476	0.0474512	0.00300541
fd	n	24	0.599426	0.117699	0.803694	-0.376169	0.0490682	0.0017445
bd	y	40	8.32845	0.0523705	0.926955	-0.409117	0.0488767	0.00341173
bd	n	24	0.609957	0.116568	0.805291	-0.37605	0.049105	0.00174367
cd	y	65	4.45272	0.0525163	0.682833	-0.66807	0.0522633	0.00321284
cd	n	38	0.604866	0.11711	0.804514	-0.376109	0.0490869	0.00174406

Table 4. This table shows results obtained with SA, DSSA, DE and our hybrid method (HM). For the results obtained with the hybrid method we have used $N_I = 16384$, $N_B = 8$ and report the parameters of the best three points.

Method	Time	κ	θ	λ	ρ	v_0	R
SA	$> 1h$	1.37489	0.0659624	0.42583	-0.521524	0.0442002	0.0026999
DSSA	$> 1h$	3.326651	0.056260	0.609410	-0.528481	0.045514	0.002731
DE	$> 1h$	2.19221	0.0606641	0.515656	-0.52504	0.0442017	0.002674
HM	$20s$	2.110270	0.060529	0.503548	-0.532090	0.045330	0.002684
HM	$20s$	2.851548	0.057947	0.574742	-0.512623	0.045789	0.002700
HM	$20s$	4.487206	0.054525	0.699049	-0.509853	0.045819	0.002808

Table 5. This table shows the residual for different combinations of initial points N_I and Levenberg-Marquardt starting points N_B.

$N_I \backslash N_B$	1	2	4	8	16	32	64	128
2^7	0.00441	0.00383	0.00359	0.00353	0.00316	0.00297	0.00293	0.00285
2^8	0.01350	0.01350	0.00343	0.00306	0.00300	0.00287	0.00287	0.00287
2^{10}	0.00394	0.00325	0.00311	0.00311	0.00311	0.00293	0.00288	0.00285
2^{12}	0.00311	0.00311	0.00311	0.00299	0.00290	0.00286	0.00275	0.00275
2^{14}	0.00286	0.00286	0.00286	0.00286	0.00285	0.00285	0.00283	0.00282
2^{16}	0.00286	0.00286	0.00286	0.00286	0.00286	0.00283	0.00283	0.00283

4 Conclusion

We have presented an algorithm for the calibration of the widely used Heston model which is theoretically capable of finding the global minimum. The combination of local and global algorithms together with parallelization on GPUs and CPUs led to a massive speed up compared to global algorithms. Including the

Feller condition has an enormous impact on the results and the performance - these problems will be addressed in further investigations.

Acknowledgement

Part of this work was supported by the Austrian Ministry of Economy, Family and Youth and by the Upper Austrian government within the framework of the "industrial competence centers" program. We thank Michael Schwaiger for useful discussions.

References

1. Heston, S.: A closed-form solution for options with stochastic volatility with applications to bond and currency options. Review of Financial Studies 6, 327–343 (1993)
2. Albrecher, H., Binder, A., Mayer, P.: Einführung in die Finanzmathematik. Birkhäuser, Basel (2009)
3. Albrecher, H., Mayer, P., Schoutens, W., Tistaert, J.: The Little Heston Trap. Wilmott Magazine, 83-92 (January 2007)
4. Brabazon: Biologically Inspired Algorithms for Financial Modelling. Springer, Heidelberg (2006)
5. Hedar, A.-R., Fukushima, M.: Hybrid simulated annealing and direct search method for nonlinear unconstrained global Optimization. Optimization Methods and Software 17, 891–912 (2002)
6. Fang, F., Osterlee, K.: A novel pricing method for European options based on Fourier-Cosine series expansion, MPRA Paper (2008)
7. Carr, P., Madan, D.B.: Option valuation using the fast Fourier transform. J. Comp. Finance 2, 61–73 (1999)
8. O'Sullivan, C.: Path dependent option pricing under Levy processes, EFA 2005 Moscow Meetings Paper, SSRN (February 2005), `http://ssrn.com/abstract=673424`
9. Madsen, K., Nielsen, H.B., Tingleff, O.: Methods for Non-Linear Least Sqares Problems, 2nd edn (2004)
10. Aichinger, M., Binder, A., Fürst, J., Kletzmayr, C.: Advanced Methods for the Calibration of Heston and Bates Models (to be submitted)