The Heston–Hull–White Model Part III: Design and Implementation

Holger Kammeyer
University of Goettingen

Joerg Kienitz
Dt. Postbank AG, e-mail: joerg.kienitz@postbank.de

1 Introduction

This is the third article in a series of three on financial modeling using the Heston–Hull–White model. The aim of this series is to show the full-life cycle of model development and implementation.

The Heston–Hull–White model is a hybrid equity model exhibiting both stochastic volatility and stochastic rates. Our encompassing goal is to develop a software that features calibration of the model to market data as well as option pricing by Fourier and by Monte Carlo methods. In the first article of our series (Kammeyer and Kienitz, 2012a), we presented the mathematical background as well as justification as to why considering non-constant rates should be beneficial. The second article (Kammeyer and Kienitz, 2012b), was concerned with the algorithms and numerics involved. We provided code snippets for crucial parts of the implementation and concluded with some numerical examples and possible extensions. In this final article, we will consider the software design in the large and assemble the parts we collected before.

The outline is as follows. Mainly for reference, we will give a quick reminder of the most important results in Parts I and II on the Hull–White–Model and on Carr–Madan pricing in section 2. Sections 3, 4, and 5 then present the software design of Carr–Madan fast Fourier transform pricing, Monte Carlo pricing and calibration of the model parameters to market data. The implementation was done in C++. To present object oriented programming, the Unified Modeling Language has become a standard. Each of the sections 3, 4, and 5 splits up into static and dynamic design and these are illustrated by UML class and interaction diagrams respectively. Section 6 concludes. The complete source code is available from the authors (see our contacts on page references). We wish to thank Hendrick Kammeyer for helpful insights on UML.

2 Carr–Madan pricing and the Heston–Hull–White model

The main purpose of our implementation is to compute the price of a European call or put option. The payoff of such an option is given by $(S_T - K)_+$ where S denotes the value of its underlying, T denotes the expiration time and K is its strike price. The hybrid Heston–Hull–White model assumes that S evolves according to certain dynamics. These are given by the following system of stochastic differential equations subject to the filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t), Q)$.

$$dS_t = r_t S_t dt + \sqrt{v_t} S_t dW_t^1 \tag{1}$$

$$dv_t = \kappa(\overline{v} - v_t)dt + \omega\sqrt{v_t}dW_t^2 \tag{2}$$

$$dr_t = \lambda(\theta(t) - r_t)dt + \eta dW_t^3 \tag{3}$$

We agree that the Wiener process W^3 be independent of W^1 and W^2. Let $D(t_0, t) = \exp(-\int_{t_0}^{t} r_t dt)$ be the stochastic discount factor with t_0 denoting present time. Then note that in the representation (1), (2), and (3), the discounted underlying $D(t_0, t)S_t$ is a martingale under Q to begin with. A concise description of the parameters was given in Kammeyer and Kienitz (2012a), section 2. They can be split up in three groups.

The call price we wish to compute is given by

$$C = C_{\mathcal{F}_{t_0}}(K, T) = \mathbb{E}[D(t_0, T)(S_T - K)_+ | \mathcal{F}_{t_0}].$$

Call Pricing Carr and Madan (see Carr and Madan, 1999 and Kammeyer and Kienitz, 2012b, section 2.1) suggest to compute this price as

$$C = \frac{\exp(-\alpha k)}{\pi} \int_0^\infty \exp(-iuk)\psi_T(u)du. \tag{4}$$

Here $k = \log K$ is the logarithm of the strike, $\alpha \in \mathbb{R}_+$ is the Carr–Madan damping parameter and $i = \sqrt{-1}$. Let $P(t_0, t) = \mathbb{E}[D(t_0, t) | \mathcal{F}_{t_0}]$ be the zero coupon bond value. Then the T-forward measure Q^T is defined by the Radon-Nikodym derivative $\frac{dQ^T}{dQ} = \frac{D(0,T)}{P(0,T)}$. This gives rise to the T–forward characteristic function $\phi_T(u) = \mathbb{E}^{Q^T}[\exp(iuX_T) | \mathcal{F}_{t_0}]$ of $X = \log S$. Finally, ψ_T can be computed from ϕ_T as follows.

$$\psi_T(u) = \frac{\phi_T(u - (\alpha + 1)i)}{\alpha^2 + \alpha - u^2 + i(2\alpha + 1)u} \tag{5}$$

In case of our Heston–Hull–White model, we have seen that ϕ_T is built from the characteristic function $\phi_R(u, t_0, T) = \mathbb{E}[\exp(iuR_T | \mathcal{F}_{t_0}]$ of the integrated

Table 1: Parameters

Option parameters	Heston parameters	Hull White parameters
maturity: T	long term variance: $\bar{v}$	speed of mean reversion: λ
strike: K	instantaneous variance: v_0	volatility of short rate: η
spot: S_{t_0}	speed of mean reversion: κ	discount curve / yield curve: $\theta(t)$
	correlation of W^1 and W^2: ρ	
	second order volatility: ω	

Hull–White short rate process $R_T = \int_{t_0}^{T} r_s ds$ and from the characteristic function $\phi_H(u, t_0, T) = \mathbb{E}[\exp(iuX_t) | \mathcal{F}_{t_0}]$ of the pure Heston stochastic volatility model. The latter is obtained by simply setting λ and η to zero. Indeed,

$$P(t,T)\phi_T(u) = \phi_R(u + \mathrm{i}, t_0, T)\phi_H(u, t_0, T). \tag{6}$$

All Hull–White and Heston model parameters from the above list as well as the spot price are hidden in these two characteristic functions (given in Kammeyer and Kienitz, 2012a, section 4). By discrete approximation of the Fourier transform in (4), we obtain for the call price C,

$$C \approx \frac{\exp(-\alpha k)}{\pi} \sum_{j=0}^{N-1} \exp\left(-\frac{2\pi}{N} \mathrm{i} j u\right) (-1)^j \psi_T(j\eta) \frac{\eta}{3} (3 + (-1)^{j+1} - \delta_{j0}). \tag{7}$$

Here N and η are length and fineness of a grid of evaluation points, δ_{j_0} is the Kronecker delta and u is a discrete substitute for k which now runs from 0 to $N-1$. In view of this expression, the calculation essentially amounts to a single fast Fourier transform (FFT) which yields option prices for a whole range of strikes indexed by u.

3 FFT pricing design

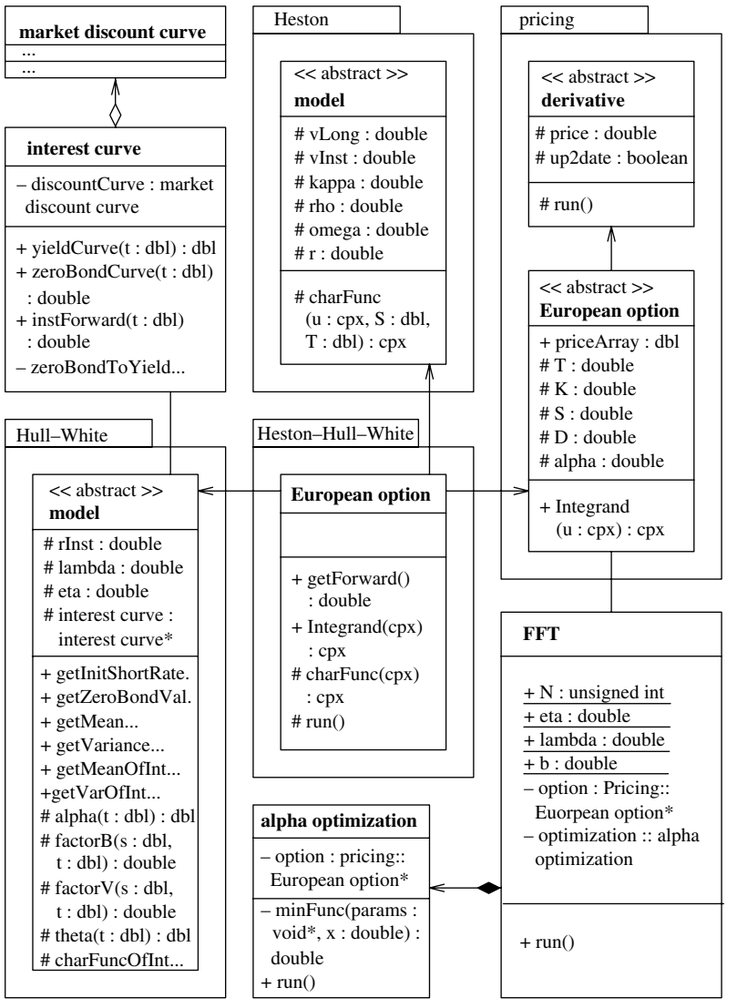
We wish to design a software that incorporates various components for fast pricing of European call and put options. We implement the Carr–Madan pricing framework consisting of FFT algorithm and an optimization routine for the damping parameter α. Abstract classes for derivatives should guarantee extensibility to various instruments other than vanilla options. Both the Heston and the Hull–White model should be reflected individually in the class structure. Finally, the Hull–White model is based on the market term structure which our concept must consider.

3.1 Static design

As a matter of fact, the Carr–Madan FFT pricing method is suitable for any model whose characteristic function is analytically computable. For the sake of building customizable software, we have thus taken care to separate this pricing framework in our software structure from the implementation of any specific model. We have seen in (5) that such a model, in our case the Heston–Hull–White model, enters the picture in terms of its characteristic function only.

In the UML class diagram of Figure 1, the right column of the diagram constitutes the Carr–Madan pricing framework. It consists of the *pricing* package with an associated *FFT* class. A European call option within this package is represented by a class *European option* which inherits from a class *derivative*. These classes are labeled by the stereotype ≪*abstract*≫ because only

Figure 1: UML class diagram for the Carr–Madan FFT pricing method.



instances of specializing classes will be created. The instances of the associated *FFT* class will manage the Fourier pricing routine. The *FFT* class includes by composition a class *alpha optimization* to find the optimal Carr–Madan parameter α (again see Kammeyer and Kienitz, 2012b, and the original paper by Kahl and Lord, 2007).

Now the *Heston–Hull–White* package in the center of the diagram includes a class named *European option* as well. As opposed to its abstract right hand mother, this class represents a special European option modeled according to the concrete Heston–Hull–White dynamics. Since our hybrid model ramifies in both the *Hull–White* short rate *model* and the *Heston* stochastic volatility *model*, compare (6), it inherits from either abstract model class as indicated by the remaining two arrows starting in it. The class *interest curve* in turn is associated with *Hull–White::model* and contains by aggregation a class of type *market discount curve* (realized by the *numeric matrix* class of Duffy and Kienitz, 2009). Note that discount curves and yield curves as the given $\theta(t)$ of the Heston–Hull–White model correspond one-to-one. For reference, some UML vocabulary is given in Figures 2 and 3.

Figure 2: Some UML notation.

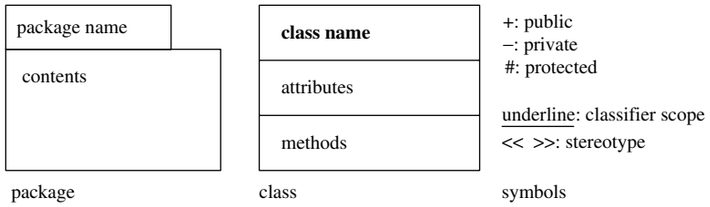
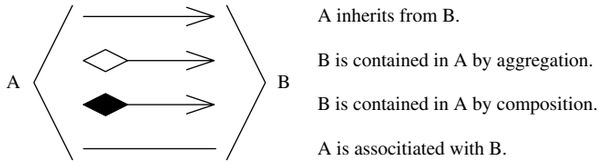


Figure 3: Arrows symbolizing various types of UML class relationships.



3.2 Dynamic design

The interaction structure corresponding to the use case of valuating a European option is given in the UML collaboration diagram 4. In the beginning of the pricing process an object of the *Heston–Hull–White::European option* class is constructed. In doing so, the parameters listed on page list of parameters are stored as attributes coming from the *pricing::European option*, *Heston::model* and *Hull–White::model* class respectively. The most complicated attribute is the market discount curve. It is stored in an object of the homonymous class which is aggregated by an interest curve object. Note that in our C++ implementation, the association of the latter with the *Hull–White::model* class is realized by a pointer. Now up to some correction terms, the price formula (7) is just a discrete Fourier transform of the function ψ_T given in (5). This function depends on the Carr–Madan damping parameter α for which the (in some well-defined sense) optimal choice is made by an instance of the alpha optimization class. We remark that for the optimization routine, the characteristic functions of the Heston–Hull–White model have to be evaluated already. To get a clear arrangement however, we left this part out in the diagram. Having found the optimal α, our *FFT* object sets up the fast Fourier transform. For each time the FFT algorithm evaluates ψ_T, the characteristic functions of the Heston model and the Hull–White model are called. For the latter function, time zero forwards as well as the term structure $\theta(t)$ must be known. These are computed by the interest curve object which collects market data from the aggregated market discount curve. Finally, the initial *Heston–Hull–White::European option* rescales the returned Fourier transform by $\frac{\exp(-\alpha k)}{\pi}$ as required by (7) and interpolates the price.

4 Monte Carlo pricing design

The closed form expression (4) is due to the simple payoff of a plain vanilla option. For the valuation of exotic derivatives with path dependent payoffs, no closed form pricing formula will be available in general. For these sophisticated instruments, the implementation of a Monte Carlo framework for the Heston–Hull–White model proves to be convenient. We presented the corresponding technical background along with code snippets of the quadratic exponential discretization scheme (due to Andersen,

Figure 4: UML collaboration diagram for the analytical pricing method. By UML default, an asterisk (*) indicates that the step is repeated several times.

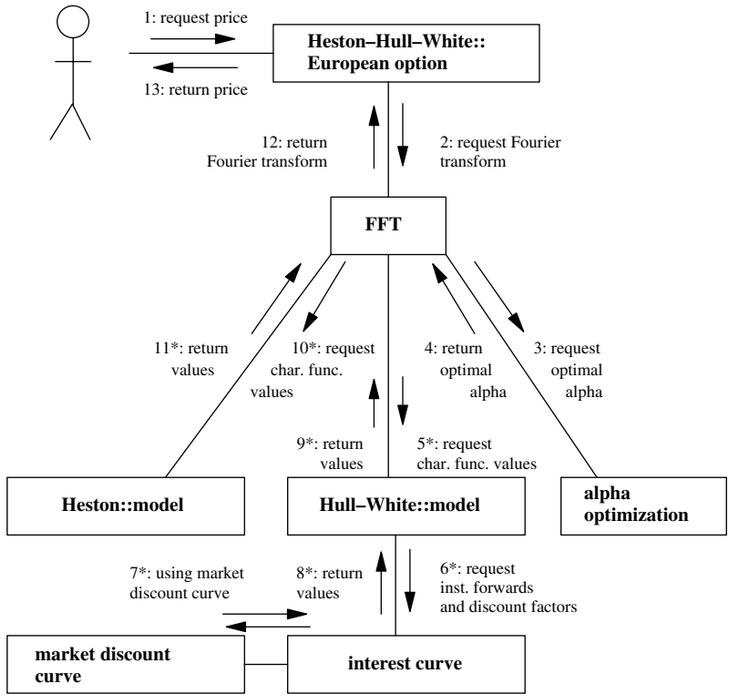
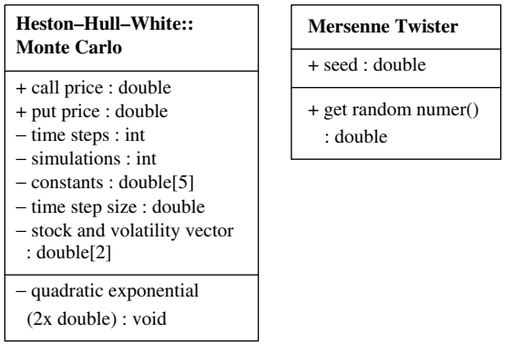


Figure 5: Additional classes for Monte Carlo pricing.

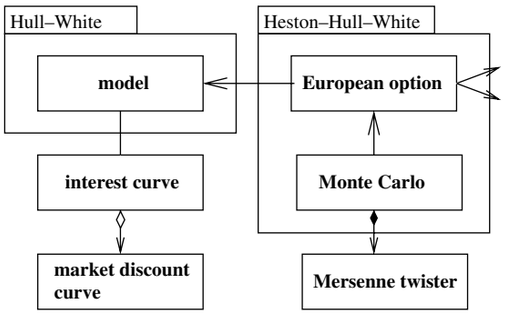


2007) in Kammeyer and Kienitz (2012b), section 3. The Mersenne Twister as implemented by Saito and Matsumoto (2008) will serve us as random number generator.

4.1 Static design

We introduce two new classes to feature Monte Carlo pricing in our software (Figure 5). The *Mersenne Twister* generates random numbers using a given seed. The *Monte Carlo* class transforms those to Brownian motion increments which according to the HHW dynamics (1)–(3) give rise to paths of the underlying stock price process S. Applying the payoff function to the paths, then averaging, yields the price. For comparison with the FFT prices we have

Figure 6: UML class diagram for the Monte Carlo pricing method.



implemented this for the payoff of the European option $(S_T - K)_+$. The private member variables of the *Monte Carlo* class include integers for the *number of time steps* of a path and the *number of paths* themselves. The former determines the fineness of a path, the latter determines the variance of the arithmetic mean of payoffs and in this sense the convergence to the correct value. The convergence rate of this and of any other Monte Carlo method is of order $(\sqrt{n})^{-1}$. Runtime increases linearly with either parameter. As explained in Kammeyer and Kienitz (2012b), section 3.1, the QE-discretization scheme requires certain constants as parameters which are stored in the 5-element array *constants*. The *time step size* is of course given by the time to maturity T of the derivative, divided by *number of time steps*. In Kammeyer and Kienitz (2012b), section 3.1, we have also described that the time steps of the underlying S are preceded by time steps of the volatility v. Both values are stored in the 2-element array *stock and volatility vector*. Calling the private member function *quadratic exponential* computes one time step due to the QE-scheme in a pure Heston model, thus requiring two random numbers as input, one for the Brownian motion driving the volatility, one for the Brownian motion of the underlying. As was also observed in Kammeyer and Kienitz (2012b), the independence of the Wiener process in (3) effects that only one random number per path has to be sampled for the short rate. There is thus no need for a 3-element vector. No further explanation is in order for the Mersenne twister random number generator class which is used as a complete black box. We refer to Saito and Matsumoto (2008) for any inquiry relating to its inner mechanism.

The class diagram in Figure 6 illustrates how the two new classes are connected to the existing tree.

4.2 Dynamic design

As was explained in the beginning of section 3.1 in Kammeyer and Kienitz (2012b), the generation of volatility and stock price paths can be separated completely from the generation of short rate paths. Therefore, the QE-scheme developed for the Heston model is applicable. It is called within two nested loops. The outer loop is counting the number of paths while the inner loop is counting the number of time steps. Every time the inner loop is complete, the integrated short rate has to be added as a drift to the (logarithmic) stock price path. We have seen in Kammeyer and Kienitz (2012a), section 3.2, that the integrated short rate is normally distributed with parameters mean and variance as given there in (20) and (21). The mean depends on the discount curve implied by the market, thus requiring a class collaboration as in steps 6–8 of Figure 4. The variance depends on the parameters λ, η and the expiration time T only. Drawing a single random number, the integrated short rate path can thus be built. Finally, the Monte Carlo price is computed as the arithmetic mean of all the payoff values corresponding to the simulated paths. As described, the procedure does not introduce any new collaboration structure and thus an interaction diagram for Monte Carlo pricing should be dispensable.

5 Calibration design

As in the case of Monte Carlo pricing, the independence assumption on the driving Wiener process in (3) of the stock price process (1) and of the volatility process (2) is crucial for calibration. It allows us to split up the calibration procedure of the Heston–Hull–White model to market data into two parts. First, the parameters mean reversion speed λ (`lambda`) and volatility of short rate η (`eta`) in the Hull–White model are calibrated using market prices of caplets and floorlets. Then, the remaining five Heston parameters long-term variance $\overline{v}$ (`vLong`), initial variance v_0 | 10 (`vInst`), speed of mean reversion κ (`kappa`), correlation ρ (`rho`), and volatility of variance ω (`omega`) will be calibrated using market European call option prices. In both cases an LBFGSB algorithm implemented in C by Ciyou Zhu[1] is used to minimize the function `minfunc` which is given by the sum of squared relative errors,

$$\texttt{minfunc} = \sum_{\text{market data}} \left(\frac{\text{market price} - \text{model price}}{\text{market price}} \right)^2 .$$

5.1 Static design

Our calibration algorithm has to compute many option prices. We use the faster FFT pricing method and not the Monte Carlo approach. In addition to the classes introduced for the FFT pricing procedure, we introduce three new classes *pricing::zero bond option*, *Hull–White::zero bond option* and *Hull–White::caplet floorlet*. These are given in Figure 7. Here again, we distinguish between a zero bond option as an abstract derivative in the *pricing* package and a zero bond option whose price is determined by the Hull–White dynamics in the *Hull–White* package. Also note that only zero bond parameters are attributes of the *pricing::zero bond option* class. Those parameters which stem from a zero bond option being a mere option will be inherited. The static class design for the Hull–White calibration is given in Figure 8. The key to sticking the *caplet floorlet* class to the existing class tree lies in the observation in Kammeyer and Kienitz (2012b) that caplets and floorlets can be expressed in terms of zero bond options. Thus, the *caplet floorlet class* inherits from the *zero bond option* class in the *Hull–White* package. The algorithm *lbfsgb minimize* follows the procedural programming paradigm. It

Figure 7: Additional classes to represent caplets and floorlets in terms of zero bond options.

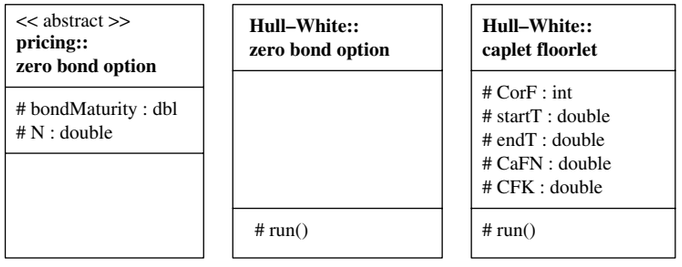


Figure 8: UML class diagram for the Hull–White calibration routine.

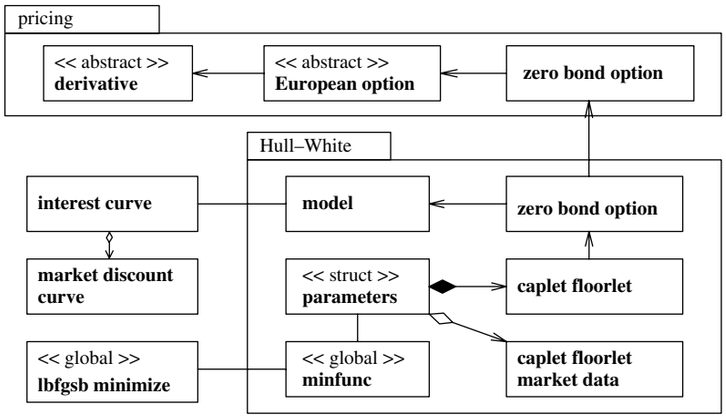


Figure 9: UML class diagram for the Heston calibration routine.

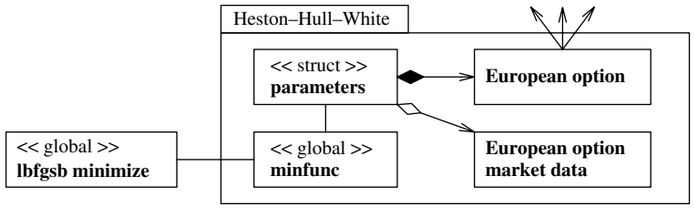
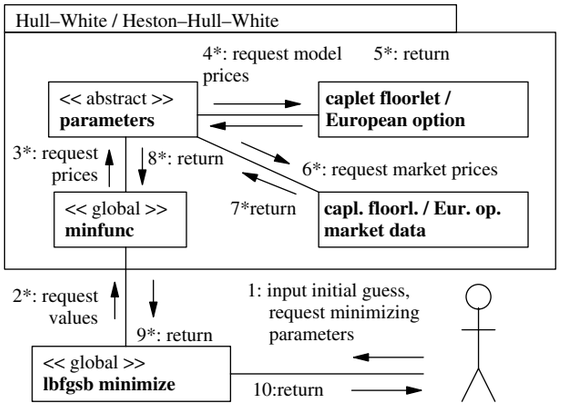


Figure 10: UML collaboration diagram for the calibration routine.



is given by a global function and requires a global function to be minimized. In UML this is reflected by the ≪*global*≫ stereotype. Since classes are not available in C, parameters for *minfunc* are stored in a structure as indicated by the ≪*struct*≫ stereotype. The association between *parameters*, *minfunc*, and *lbfgsb minimize* are again realized by pointers as is common in C and C++. The parameters associated with *minfunc* split up into *caplet floorlet market data* (realized by a *numeric matrix*) containing market data (starting time, ending time, nominal value, strike and price of a caplet or floorlet as quoted in the market) and an object of the *Hull–White::caplet floorlet* class. The latter has a method to compute the model price for a caplet or floorlet with the same properties as the one from the market.

The same class relationship then applies to the calibration of the Heston parameters. Only the market data comes from European option prices instead of caplet and floorlet prices. This is shown in Figure 9. The three arrows starting in the European option class represent the inheritance of this class as given in Figure 1.

5.2 Dynamic design

The interaction structure (Figure 10) of both the Hull–White and the Heston calibration procedure is the same. Only the package name and the relevant derivative (*caplet/floorlet* or *European option*) need to be inserted as necessary. The course of action as described is straightforward. Note that when the model price for a European option is requested in step 4 of the Heston calibration, precisely the routine as given in Figure 4 is called.

6 Conclusion

We have come to the end of our presentation of the various stages of financial model development using the example of the Heston–Hull–White model. The reader should now be prepared to work with this model and to implement and customize it as needed. Again, it is encouraged to order the complete C++ source code free of charge for any noncommercial purpose at the contacts given below.

The Heston–Hull–White model is a powerful tool for pricing long maturity derivatives. We should however finish with a word of warning. The high performance of our pricing methods despite using a three factor model, heavily relies on our ad hoc assumption of an independent short rate. But as explained in Hunter and Picot (2005), realizing nonzero correlations of equity and interest rate is indispensable in order to accurately price *hybrid* derivatives. We have already mentioned in Kammeyer and Kienitz (2012a), that Grzelak and Oosterlee (2010) suggest a modification of the model which at least approximately covers the case of a full correlation matrix. It should be possible to extend our software correspondingly, making the hybrid world accessible with our approach.

Holger Kammeyer is studying toward his Ph.D. in mathematics. He researches geometric L2 invariants. He holds a diploma (with distinction) in mathematics from the University of Goettingen. Before he started his Ph.D. research, he worked as a teaching assistant, did an internship at Deutsche Postbank with the Quantiative Analysis group, and completed a one year graduate study at UC Berkeley.

Joerg Kienitz is the head of Quantitative Analysis at Deutsche Postbank AG. He is primarily involved in the development and implementation of models for pricing structured products, derivatives, and asset allocation. He authored a number of quantitative finance papers and his book *Monte Carlo Frameworks* was published with Wiley in 2009. A new Wiley book, *Financial Modelling: Theory, Implementation and Practice with Matlab Source Code* will be published in autumn 2012. He is member of the editorial board of the *International Review of Applied Financial Issues and Economics*. He holds a Ph.D. in stochastic analysis and probability theory.

ENDNOTE

1. Optimization Technology Center, Argonne National Laboratory.

REFERENCES

Andersen, L.B. 2007. *Efficient Simulation of the Heston Stochastic Volatility Model*. SSRN eLibrary.

Brigo, D. and Mercurio, F. 2001. *Interest Rate Models – Theory and Practice*. Springer Finance: Springer-Verlag, Berlin.

Carr, P. and Madan, D.B. 1999. Option valuation using the fast Fourier transform. *Journal of Computational Finance* 2(4), 61–73.
Duffy, D. and Kienitz, J. 2009. Monte Carlo Frameworks: Building Customisable High-Performance C++ Applications. Wiley: Chichester.
Grzelak, L.A. and Oosterlee, C.W. 2010. On the Heston Model with Stochastic Interest Rates. Delft Univ. of Technology Technical Report No. 09–05.
Hunter, C. and Picot, G. 2005. Hybrid Derivatives: Financial Engines of the Future. In Nicholson, L. ed. The Euromoney Derivatives & Risk Management Handbook 2005/2006, Euromoney Institutional Investor: London.
Kahl, C. and Lord, R. 2007. *Optimal Fourier Inversion in Semi-Analytical Option Pricing*. SSRN eLibrary.
Kammeyer, H. and Kienitz, J. 2012a. The Heston–Hull–White Model Part I: Finance and Analytics. *Wilmott magazine* January, 46–53.
Kammeyer, H. and Kienitz, J. 2012b. The Heston–Hull–White Model Part II: Numerics and Examples. *Wilmott magazine* March, 34–44.
Saito, M. and Matsumoto, M. 2008. SIMD-oriented Fast Mersenne Twister: A 128-bit Pseudorandom Number Generator. In Keller, A., Heinrich, S. and Niederreiter, eds. Monte Carlo and Quasi-Monte Carlo Methods 2006, Springer: Berlin, pp. 607–622.