The Chebyshev method for the implied volatility

Kathrin Glau^{1,2}, Paul Herold^{1}, Dilip B. Madan^{3}, Christian Pötz^{1,2}

^{1}Technical University of Munich, Germany,
^{2}Queen Mary University of London, UK,
^{3}Robert H. Smith School of Business, University of Maryland

November 22, 2018

Abstract

The implied volatility is a crucial element of any financial toolbox, since it is used for quoting and the hedging of options as well as for model calibration. In contrast to the Black-Scholes formula its inverse, the implied volatility, is not explicitly available and numerical approximation is required. We propose a bivariate interpolation of the implied volatility surface based on Chebyshev polynomials. This yields a closed-form approximation of the implied volatility, which is easy to implement and to maintain. We prove a subexponential error decay. This allows us to obtain an accuracy close to machine precision with polynomials of a low degree. We compare the performance of the method in terms of runtime and accuracy to the most common reference methods. In contrast to existing interpolation methods, the proposed method is able to compute the implied volatility for all relevant option data. In this context, numerical experiments confirm a considerable increase in efficiency, especially for large data sets.

Keywords Black-Scholes implied volatility, real-time evaluation, Chebyshev Polynomials, Polynomial Interpolation, Laplace implied volatility

1 Motivation

Ever since Black and Scholes (1973) and Merton (1973) introduced their option pricing model, the Black-Scholes formula has been omnipresent in the financial industry. The one parameter in the model that can not be observed using market data is the volatility of the underlying asset process. The Black-Scholes call price function is strictly monotone increasing in volatility. Manaster and Koehler (1982) show that under no-arbitrage assumptions there exists always a unique (positive) volatility such that the Black-Scholes model price equals the observed market price. This is called the *implied volatility*, one of the most important quantities in finance.

The implied volatility can be seen as a universal language in the daily business of trading, hedging, model calibration and more generally in risk management. Typically, trading desks quote option prices in implied volatilities instead of absolute prices. This allows traders to compare option prices on different underlyings such as equities, indices, currencies or commodities. For high frequency trading in particular, very accurate real-time evaluations of the implied volatility are required for large data sets. As stated in Baumeister (2013) and Salazar Celis (2017) in practice, often millions of option prices have to be inverted in real-time for instance by large data providers. Furthermore, the implied volatility is needed for the most common derivative hedging strategy, the so-called delta-hedging strategy. It is used to infer the sensitivity of the option price with respect to the underlying spot price, the option's delta. One takes an opposing position to the delta in the underlying asset as a hedge. Since the 1970s a large variety of asset price models that generalize and improve the Black-Scholes model have been introduced. Typically, these models are determined by a number of parameters that are fitted to observed option prices. In the context of this model calibration, the implied volatility enters the objective function. Instead of minimizing (for instance the quadratic) difference of model and market prices, the difference of the corresponding implied volatilities is used. This is a convenient normalization since options from deep in the money to far out of the money are transformed to the same scale. For calibration purposes, the implied volatility needs to be available rapidly—especially in view of routinely processed intraday recalibrations. Depending on the pricing routine employed, the accuracy needs to be medium or high. Moreover, a closed-form of the implied volatility function is advantageous since it allows the implementation of gradient-based optimization routines.

Unfortunately, the solution of this inverse problem is not available in an explicit form and thus a numerical approximation method is required. Since the implied volatility function is a crucial element of any financial toolbox, special care is called for. The method must allow the computation of implied volatilities for options in all of the different markets. Hence options with very low or high volatilities as well as options with moneyness varying from far out of the money to deep in the money have to be included. Therefore the method must cover a *large domain of input variables.* In order to satisfy the needs of the different applications the method should be *highly efficient* for a given requirement in terms of accuracy. Even for very large data sets the method must be able to deliver accurate *real-time evaluations* of the implied volatility. In view of the implied volatility as an ingredient of optimization routines, the approximation should be given in *closed-form* with accessible derivatives. Finally, the method should be *easy to implement and to maintain.* There exists a long list of papers dealing with this problem.

The first class of methods to determine the implied volatility are iterative root finders such as

- Newton-Raphson,
- *Matlab*s implied volatility function *blsimpv*,
- the iterative methods of Jäckel (2006) and Jäckel (2015).

The first approach dates back to Manaster and Koehler (1982) who provide a possible starting value which ensures that the Newton-Raphson algorithm will always converge and return the correct implied volatility. The *blsimpv* function is part of the financial toolbox in *Matlab* and uses an iterative scheme based on Brent-Dekker. The *blsimpv* function becomes very slow for larger data-sets and the number of iterations required and thus the speed of the Newton-Raphson algorithm depends highly on the observed market parameters. For many standard parameters it often converges fast but for more extreme parameters, the number of iterative steps increases significantly, see Section 7.

To overcome this problem, Jäckel (2006) exploits the limit behaviour of the normalised call price to provide a better initial guess, which reduces the iterative steps in a modified Newton method. In Jäckel (2015) this approach is further improved using rational approximation for the initial guess and Householder's method for the iteration. This reduces the number of iterative steps even further. One drawback of the method is that it comes with the burden of a relatively complex implementation and therefore a costly maintenance. Already the generation of the initial guess relies on the rational cubic interpolation of Delbourgo and Gregory (1985) and a transformation, which is highly sensitive in terms of the accuracy of the error function and the inverse of the normal distribution.

The second class of methods to compute the implied volatility are non-iterative approximations methods. These methods are popular since they provide

- fast computation of implied volatilities,
- easy implementation and maintenance,

- closed-form expressions,
- a simple interpretation of the formula.

First, analytical approximations for at the money and later near the money options have been developed. Typically, these methods depend on a series expansion of the call price at the money. Prominent examples are the approximation formulas of Brenner and Subrahmanyan (1988), Chance (1996), Corrado and Miller (1996), Chambers and Nawalkha (2001) and Lorig, Pagliarani, and Pascucci (2014). Typically, these methods suffer from a poor performance for out of the money options. More recently, Li (2008), Pistorius and Stolte (2012) and Salazar Celis (2017) have developed rational approximations of the implied volatility. Unfortunately, the domain for which the latter set up the interpolation is very restrictive and excludes option prices which occur in practice. In particular, options with relatively high or low volatilities cannot be handled. For example Figure 1.1 illustrates the moneyness and implied time-scaled volatilities of options on the DAX index traded on 6/20/2017 (Source Thomson Reuters Eikon). In this example, only 85% of all put options and 92% of the call options are covered. Although the domain was designed for equity options, even in this case the formula cannot be applied to all relevant contracts. Moreover, one needs additional iterative Newton steps to achieve a high accuracy close to machine precision for the methods of Li (2008) and Salazar Celis (2017).

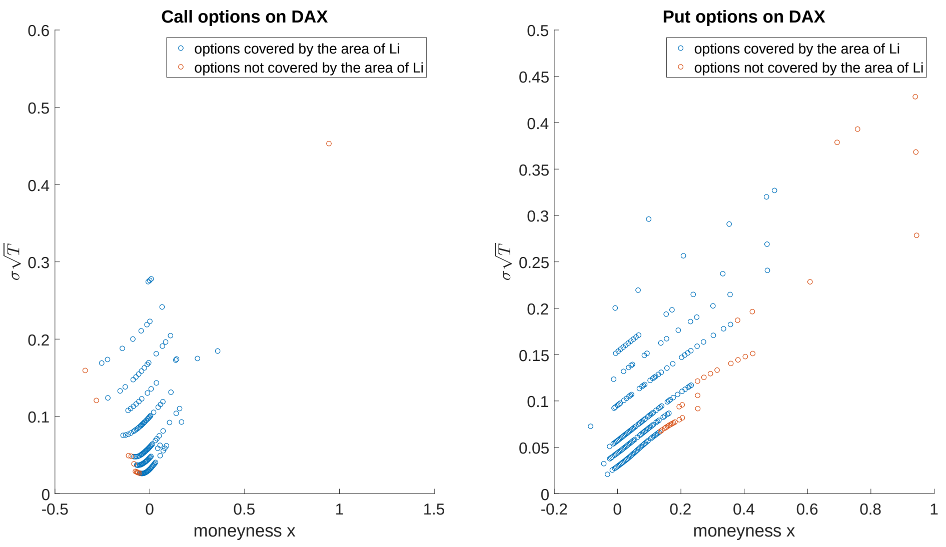


Figure 1.1: Moneyness x and time-scaled volatility $\sigma\sqrt{T}$ of DAX options on 6/20/2017. We only considered options with positive trading volume.

In this paper, we propose polynomial approximation to the implied volatilities surface choosing Chebyshev interpolation. The approximation of the implied volatility thus inherits the appealing properties of Chebyshev interpolation, namely the fact that the approximation is highly efficient, stable and easy to implement. It is sufficient to invert the normalized call price, which reduces the dimensionality of the approximation to a bivariate Chebyshev interpolation. For this we use the algorithm provided in the *MATLAB* package *chebfun* (www.chebfun.org) that exploits the low-rank structure of the problem. Hence, the method enables a fast computation of implied volatilities at a high accuracy. In order to cover the whole range of relevant options, one has to investigate the shape of the call price surface further. We observe areas where the call price is almost linear as well as areas where the call price is extremely flat in the volatility. For an optimal treatment of the different areas we introduce a domain splitting. In the flat areas, we exploit the limit behaviour by introducing appropriate transformations. We show that the error of the interpolation decays subexponentially fast and we provide an explicit error bound. It is straightforward to adjust the method to any pre-set accuracy to obtain an optimal efficiency.

Furthermore, the implied volatility function is represented by a polynomial and hence very easy to handle. Let us emphasize that this procedure is more general and can be applied to similar problems as well. To illustrate this, we approximate the implied volatility in a market model based on a Laplacian density function instead of a normal distribution introduced by Madan (2016).

The remainder of the article is as follows. In Section 2, we recall the normalized call price and the Chebyshev function on which our approach relies. In Section 3, we introduce a simple, bivariate interpolation of the implied volatility based on a low-rank interpolation in Chebyshev nodes. We highlight the potential of the method and show that we reach a maximal error close to machine precision with a low number of interpolation points. In Section 4, we introduce the bivariate interpolation on a larger domain which includes very low and very high volatilities as well as deep in the money and far out of the money options. In Section 7, we show that the method is both, fast and accurate and compare it to the methods of Newton-Raphson, Li (2008) and Jäckel (2015). We devote the last section to the approximation of the implied volatility in the Laplacian market model.

2 Preliminaries

2.1 The normalized Black-Scholes price

As stated, the implied volatility depends on the parameters S_0,K, T, r and the option premium C. The computational effort to interpolate a function depending on five variables is challenging. Fortunately, we can reduce the dimensionality as stated in Jäckel (2015) amongst others using the normalized call price given as

$$c(x,v) = e^{\frac{x}{2}}\Phi\left(\frac{x}{v}+\frac{v}{2}\right) - e^{-\frac{x}{2}}\Phi\left(\frac{x}{v}-\frac{v}{2}\right) \text{ with} \tag{2.1}$$

$$\begin{aligned} x &= \log(S_0 e^{rT}/K) = rT + \log(S_0/K) \\ v &= \sigma\sqrt{T}. \end{aligned}$$

In this context x measures the *moneyness* (the option is out of the money if $x < 0$, at the money if $x \approx 0$ and in the money if $x > 0$), v corresponds to the *time-scaled volatility.* We have

$$c(x,v) = \frac{C(S_0,K,T,r,\sigma)}{\sqrt{S_0 e^{-rT} K}} \tag{2.2}$$

Furthermore, call prices of in the money options can be expressed by those of out of the money options, namely

$$c(-x,v) = c(x,v) + e^{-\frac{x}{2}} - e^{\frac{x}{2}}. \tag{2.3}$$

Hence the domain can be reduced to $x \leq 0$ and consequently the call price is normalized to values in $[0,1]$. To calculate the implied volatility σ for a call price C it is thus sufficient to solve Equation (2.1) for v using the normalized call price c.

2.2 Chebyshev Interpolation

The polynomial interpolation of a function f on $[-1,1]$ in the *Chebyshev points* $x_k = \cos(k\pi/N)$ is given by

$$f(x) \approx I_N(x) := \sum_{j=0}^{N} a_j T_j(x) \qquad \text{with} \qquad a_j = \frac{2^{\mathbb{1}_{0<j<N}}}{N} \sum_{k=0}^{N}{}'' f(x_k) T_j(x_k), \tag{2.4}$$

where $T_j(x) = \cos(j\cos^{-1} 2(x))$ and $\sum''$ indicates that the first and the last summand are halved. If the function has an analytic extension to a Bernstein ellipse E_ρ, the error decays exponentially, see Theorem 8.2 of Trefethen (2013). In practice, this often yields an approximation close to machine precision with a low interpolation order. Together with a stable implementation being available, see Higham (2004), these are the key advantages of the Chebyshev interpolation that we will exploit.

The univariate Chebyshev interpolation admits a two-dimensional tensor based extension. A function $f : [-1,1]^2 \to \mathbb{R}$ can be approximated by the interpolation

$$f(x,y) \approx I^{N_1,N_2}(x,y) := \sum_{i=0}^{N_1-1} \sum_{j=0}^{N_2-1} a_{ij} T_i(x) T_j(y). \tag{2.5}$$

with two-dimensional coefficients given by

$$a_{ij} = \frac{2^{\mathbb{1}_{0<i<N_1}}}{N_1} \frac{2^{\mathbb{1}_{0<j<N_2}}}{N_2} \sum_{k_1=0}^{N_1}{}'' \sum_{k_2=0}^{N_2}{}'' f(x_{k_1}, y_{k_2}) T_i(x_{k_1}) T_j(y_{k_2}).$$

Again, we obtain an subexponential error decay if the function has an analytic extension to a two-dimensional Bernstein ellipse, see Sauter and Schwab (2010). The tensor approach of (2.5) suffers from the curse of dimension: To decrease the error in the same proportion as in the univariate case, the number of summands and thus the complexity increases quadratically. Therefore more efficient bivariate Chebyshev interpolations have been developed. In particular, the algorithm of Townsend and Trefethen (2013) implemented in *chebfun2* reconciles the opposed aims of high accuracy and high efficiency for bivariate functions. It relies on a Gauss elimination with complete pivoting to find an optimal low rank k approximation. This leads to

$$f(x,y) \approx f_k(x,y) := \sum_{j=1}^{k} d_j c_j(y) r_j(x)$$

where c_j and r_j are one-dimensional Chebyshev interpolations of degree N_1 and N_2. This enables a matrix representation of the resulting interpolation.

3 Introduction of the approximation method

We introduce a direct interpolation of the implied volatility function using Chebyshev nodes. The two-dimensional Chebyshev interpolation requires the function to be defined on the rectangle $[-1,1] \times [-1,1]$. For the implied volatility $v(x,c)$ this is not given a priori. The variable x can easily be restricted to some interval $x \in [x_{min}, x_{max}]$ which can be transformed to $[-1,1]$ by a linear transformation φ,

$$\varphi : [x_{min}, x_{max}] \to [-1,1] \qquad \text{with} \qquad \varphi(x) := 1 - 2 \cdot \frac{x_{max} - x}{x_{max} - x_{min}}. \tag{3.1}$$

The maximal domain of c, on the contrary, does depend on x as for $x < 0$ the upper limit is given by $e^{\frac{x}{2}}$.

The intuitive approach is to choose $\xi \in [\xi_{min}, \xi_{max}]$ with $c = \xi e^{\frac{x}{2}}$ for a given moneyness $x \in [x_{min}, x_{max}]$ and scale the resulting interval $[\xi_{min} e^{\frac{x}{2}}, \xi_{max} e^{\frac{x}{2}}]$ to $[-1,1]$ by a linear transformation. If ξ_{min} is not chosen to close to 0, a two-dimensional Chebyshev interpolation on this domain provides promising results.

For a first numerical example, we fix $x_{min} = -5$, $x_{max} = 0$, $\xi_{min} = 0.05$ and $\xi_{max} = 0.8$. Then we choose a 50×50 Chebyshev grid $(\tilde{x}_{ij}, \tilde{c}_{ij}) \in [-1,1]^2$ and transform the points to the domain by setting $x_{ij} := x_{min} + \frac{1}{2}(\tilde{x}_{ij}+1)(x_{max}-x_{min})$ and $c_{ij} := \xi_{min}e^{\frac{x_{ij}}{2}} + \frac{1}{2}(\tilde{c}_{ij}+1)(\xi_{max}e^{\frac{x_{ij}}{2}} - \xi_{min}e^{\frac{x_{ij}}{2}})$. On these points we compute the implied volatilities using the method of Jäckel (2015) and apply the *chebfun2*-algorithm.

To determine the interpolation error we define an equidistant grid of 100 points in the interval $[x_{min}, x_{max}]$. For fixed x, the interval bounds in v are defined as $v_{min}(x) = v\left(\xi_{min}e^{\frac{x}{2}}, x\right)$ and $v_{max}(x) = v\left(\xi_{max}e^{\frac{x}{2}}, x\right)$. For each x-value in the fixed equidistant grid, 100 points distributed equidistantly in $[v_{min}(x), v_{max}(x)]$ are determined. This leads to 100×100 points in the (x, v) space as reference points for which we compute normalized call prices $c(x, v)$. For each reference call price we compute the implied volatility using the bivariate Chebyshev method.
Figure 3.1 shows that this approach performs very well. The maximal error lies below a level of 10^{-7} for $N = N_1 = N_2 = 50$ and decreases exponentially fast in N.

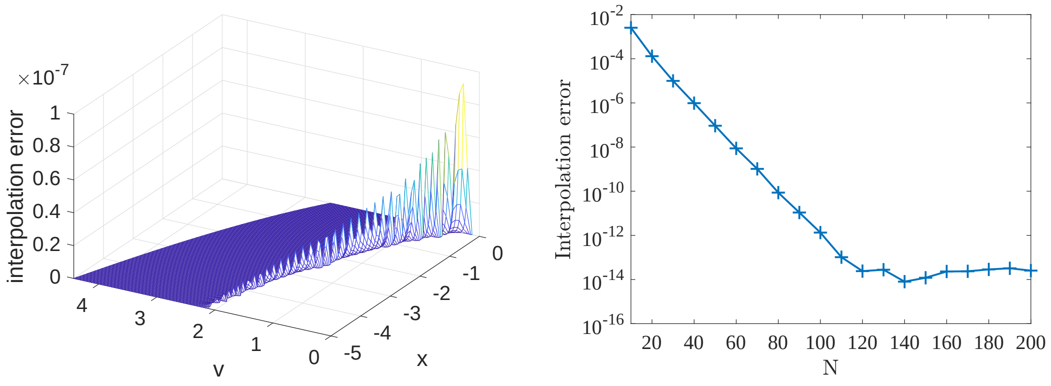


Figure 3.1: Interpolation error (left) and exponential error decay (right) using linear transformations for $x \in [x_{min}, x_{max}]$, $c \in [\xi_{min}e^{\frac{x}{2}}, \xi_{max}e^{\frac{x}{2}}]$ with $x_{min} = -5$, $x_{max} = 0$, $\xi_{min} = 0.05$ and $\xi_{max} = 0.8$.

As in the approximation methods mentioned above, we have pre-fixed a domain that is convenient for the approach. Naturally, the question arises as to which domain is appropriate to cover the relevant option data.

3.1 Investigation of the interpolation domain by market data

To find an appropriate interpolation domain, we investigate option data of the DAX, the EURO STOXX 50, the S&P 500 and the VIX index from *Thomson Reuters Eikon*. For all options with non-zero trading volume we compute the forward moneyness x and the time-scaled volatility $\sigma\sqrt{T}$. Then we check if the resulting parameters are covered by the domain of Li. Figure 3.2 illustrates the option parameters for all four indices. For all indices we observe that a relevant part of the options is not covered by the domain of Li. We observe moneyness between -1.5 and 2 as well as time-scaled volatilities up to 1. In different markets or under different market conditions one can expect to observe even more extreme option parameters. Volatilities become considerably higher during a financial crisis. This motivates us to set up a Chebyshev interpolation of the implied volatility on a significantly larger domain which covers all relevant option data. To do this in the most efficient way we need to enhance the intuitive approach introduced above with a splitting of the domain and tailored scaling functions.

4 Domain splitting and scaling

To derive an approximation of the implied volatility on a sufficiently large domain, we further inspect the normalized call price. The implied volatility is not analytic at $c(x) = 0$ and $c(x) = e^{\frac{x}{2}}$. Therefore the maximal possible interval needs to be restricted to $0 < v_{min}(x) < v_{max}(x) < \infty$

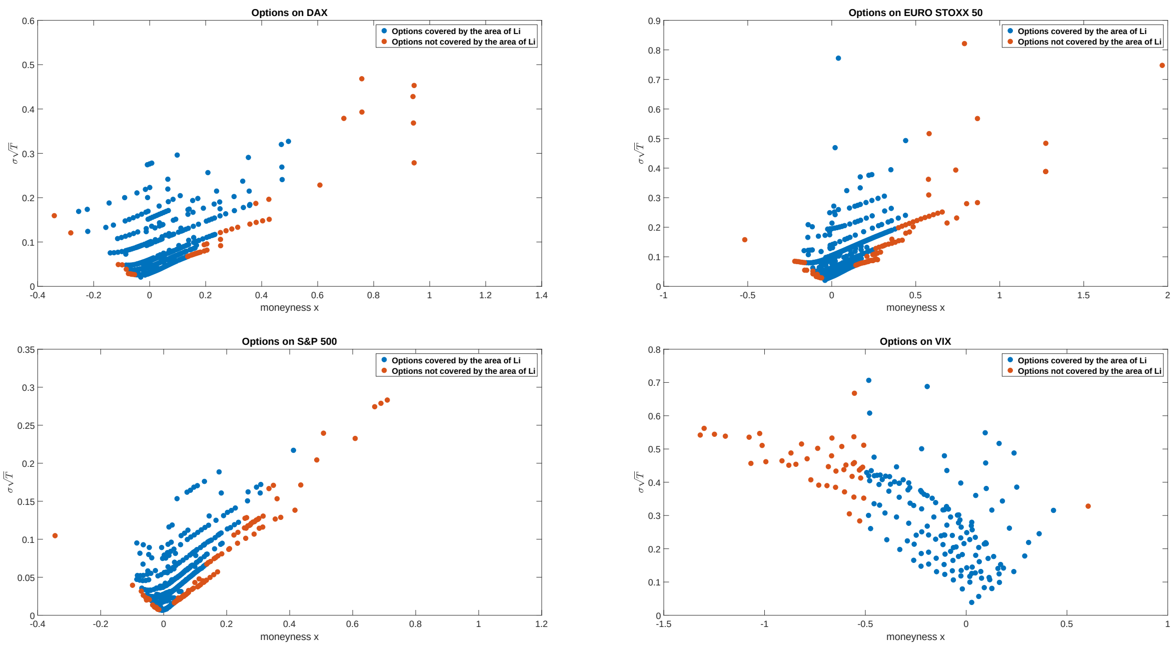


Figure 3.2: Moneyness x and time-scaled volatility $\sigma\sqrt{T}$ of options on four different indices. We only considered options with positive trading volume.

with call prices $0 < c_{min} < c_{max} < e^{\frac{x}{2}}$, which excludes these points. This assumption is not restrictive if the chosen v_{min} is small enough. Extending the domain towards the maximal interval decreases the rate of convergence. To reduce this impact, we exploit the limit behaviour of the call price. Graph 4.1 shows for a fixed moneyness x the normalized call price as a function of the volatility. We observe that the call price is flat for very low as well as very high volatilities and almost linear around the point of inflection. This motivates us to split the domain into three parts.

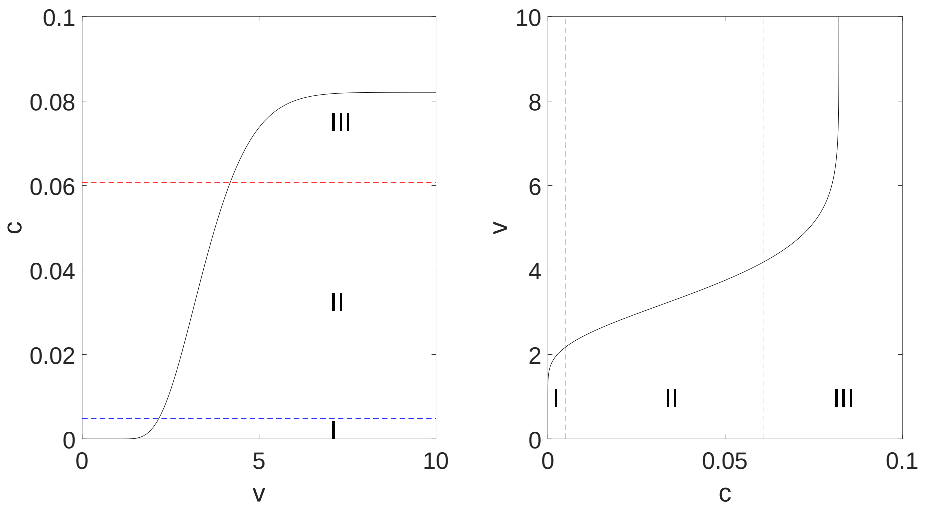


Figure 4.1: Splitting of the normalized call price (c) depending on the time-scaled volatility (v) and its inversion for $x = -5$ into three parts.

$$D_1 := [c_{min}(x), c_1(x)], \quad D_2 := [c_1(x), c_2(x)], \quad D_3 := [c_2(x), c_{max}(x)] \tag{4.1}$$

with corresponding volatilities $0 < v_{min}(x) < v_1(x) < v_2(x) < v_{max}(x) < 1$. The idea of splitting the domain is based on the method of Jäckel (2015).

For each domain we will tailor a bivariate Chebyshev interpolation. Where call prices are flat its inverse becomes very steep. Hence, a direct polynomial interpolation is not well-suited. Fortunately, by exploiting the asymptotic behaviour of the call price function, we resolve the problem. On each interval, we define a scaling function $\phi_{i,x} : D_i \to [-1,1]$ for $i \in \{1,2,3\}$ which transforms the call price to $[-1,1]$ for each $x \in [x_{min}, x_{max}]$. For the resulting functions $\tilde{v} : [-1,1]^2 \to \mathbb{R}$, $(\tilde{c}, \tilde{x}) \mapsto v(c,x)$ with $x = \varphi^{-1}(\tilde{x})$ and $c = \phi_{i,x}^{-1}(\tilde{c})$ for $i \in \{1,2,3\}$ where φ is the linear scaling of (3.1). For a given call price c and moneyness $x \leq 0$ the implied volatility can then be approximated by

$$v(c,x) \approx I_i^{N_1^i, N_2^i}(\phi_{i,x}(c), \varphi(x)) \text{ where } i \text{ satisfies } c \in D_i.$$

4.1 Scaling functions

In the following, we introduce the appropriate scaling functions for each of the areas.

4.1.1 Medium volatilities

First consider the middle part of the function. As discussed, for v around the point of inflection, the implied volatility surface is almost linear. Thus, a linear scaling suffices,

$$\phi_{2,x} : [c_1(x), c_2(x)] \to [-1,1], \quad c \mapsto 2\frac{c - c_1(x)}{c_2(x) - c_1(x)} - 1.$$

Clearly, ϕ_2 is analytic and the inverse is given by

$$\phi_{2,x}^{-1}[-1,1] \to [c_1(x), c_2(x)], \quad \tilde{c} \mapsto c_1(x) + \frac{1}{2}(\tilde{c}+1)(c_2(x) - c_1(x)).$$

4.1.2 Low volatilities

For low volatilities the call price function is very flat, and thus the implied volatility function as its inverse is steep. Therefore, a linear scaling will not provide an appropriate transformation prior to a polynomial interpolation. Instead, we propose a suitable scaling function that reduces the steepness of the inverse such that it becomes almost linear. This will increase the efficiency of the resulting approximation considerably, when compared to a linear scaling. To do this, we explore the limit behaviour of the normalized call price. For $v \to 0$ we have by equation (2.8) of Jäckel (2006) that

$$c(x,v) \approx \varphi\left(\frac{x}{v}\right)\left(\frac{v^3}{x^2}\right),$$

where φ is the density of the standard normal distribution. By inverting the function $c(v) = \varphi\left(\frac{x}{v}\right)$, which has the major effect in the limit, we obtain an inverse of the form $v = c_2(-(c_1 + 2 * log(c)/x^2))^{-1/2}$ with constants c_1, c_2 that are not relevant for us. This leads to the following transformation

$$\tilde{\phi}_{1,x} : [0, c_1(x)] \to [-1,1]$$
$$c \mapsto \begin{cases} 2\left(-\frac{2}{(x-\delta)^2}\log(c) + \frac{2}{(x-\delta)^2}\log(c_1(x)) + 1\right)^{-\frac{1}{2}} - 1 & \text{if } c > 0 \\ -1 & \text{else.} \end{cases}$$

The parameter $\delta > 0$ ensures the well-definedness for $x = 0$ and the remaining terms are needed to map the interval $[0, c_1(x)]$ to $[-1, 1]$. The transformation $\tilde{\phi}_{1,x}$ is analytic with inverse

$$\tilde{\phi}_{1,x}^{-1} : [-1,1] \to [0, c_1(x)] : \ \tilde{c} \mapsto \begin{cases} c_1(x) e^{-\frac{2(x-\delta)^2}{(\tilde{c}+1)^2} + \frac{(x-\delta)^2}{2}} & \text{if } \tilde{c} > -1 \\ 0 & \text{else.} \end{cases}$$

Using this transformation the function $v(\tilde{\phi}_{1,x}^{-1}(\tilde{c}), x)$ is approximately linear in $\tilde{c}$.

As already mentioned, to guarantee analyticity we restrict the interval to $[c_{min}(x), c_1(x)]$ for $0 < c_{min}(x) < c_1(x) < e^{\frac{x}{2}}$. Therefore, we define the scaling function for the low volatilities $\phi_{1,x} : [c_{min}(x), c_1(x)] \to [-1, 1]$ as $\phi_{1,x}(c) := l(\tilde{\phi}_{1,x}(c))$, where l is the linear transformation

$$l : [\tilde{\phi}_{1,x}(c_{min}(x)), 1] \to [-1, 1] : c \mapsto 2 \cdot \frac{c - \tilde{\phi}_{1,x}(c_{min}(x))}{1 - \tilde{\phi}_{1,x}(c_{min}(x))} - 1.$$

The function $c \mapsto \phi_{1,x}(c)$ is analytic in the interval $[c_{min}(x), c_1(x)]$ as it is a composition of two analytic functions. The inverse of $\phi_{1,x}$ is given by $\phi_{1,x}^{-1}(\tilde{c}) = \tilde{\phi}_{1,x}^{-1}\left(l^{-1}\left(\tilde{c}\right)\right)$.

4.1.3 High volatilities

Just as for the low volatilities, the call price function is very flat for high volatilities and thus its inverse becomes steep. As $\lim_{c \to e^{\frac{x}{2}}} v(c, x) = \infty$ the implied volatility function is not even bounded. As a first step, the volatility is capped by some v_{max} to guarantee that the slope will not be arbitrarily high. Again, a linear transformation is not the best choice and we propose a different scaling based on the behaviour of the call price. From Jäckel (2006) equation (2.7) we obtain for $v \to \infty$

$$c(x, v) \approx e^{\frac{x}{2}} - \frac{4}{v} \varphi\left(\frac{v}{2}\right).$$

A similar transformation as in the case of low volatilities entails improvement. Assume first that $c_{max}(x) = e^{\frac{x}{2}}$ and define

$$\tilde{\phi}_{3,x} : [c_2(x), e^{\frac{x}{2}}] \to [0, \infty] : c \mapsto \begin{cases} \left(-8 \log\left(\frac{e^{\frac{x}{2}} - c}{e^{\frac{x}{2}} - c_2(x)}\right)\right)^{\frac{1}{2}} & \text{if } c < e^{\frac{x}{2}} \\ \infty & \text{else} \end{cases}$$

with inverse

$$\tilde{\phi}_{3,x}^{-1} : [0, \infty] \to [c_2(x), e^{\frac{x}{2}}] : \tilde{c} \mapsto \begin{cases} e^{\frac{x}{2}} - \left(e^{x/2} - c_2(x)\right) e^{-\frac{\tilde{c}^2}{8}} & \text{if } \tilde{c} < \infty \\ e^{\frac{x}{2}} & \text{else.} \end{cases}$$

Exploiting the limit behaviour of the call price, one can show that for v large enough $\tilde{c} = \tilde{\phi}_{3,x}(c(x, v)) \approx -v$. Hence $v = v(\tilde{\phi}_{3,x}^{-1}(\tilde{c}), x) \approx -\tilde{c}$ which is linear in $\tilde{c}$.

Now for $c_{max}(x) < e^{\frac{x}{2}}$ the transformation is a bijection into a bounded domain which can be normalized to $[-1, 1]$ by the linear transform as in the previous case

$$l : [0, \tilde{\phi}_{3,x}(c_{max}(x))] \to [-1, 1] : c \mapsto \frac{2c}{\tilde{\phi}_{3,x}(c_{max}(x))} - 1.$$

Thus $\phi_{3,x}(c) := l(\tilde{\phi}_{3,x}(x))$ and $\phi_{3,x}^{-1}(\tilde{c}) = \tilde{\phi}_{3,x}^{-1}\left(l^{-1}\left(\tilde{c}\right)\right)$ depending on the choice of c_{max}.

4.2 Splitting

The explicit choice of the boundaries depends on the particular application. In the following we want to set the boundaries in such a way that a very large set of parameters is covered and the rate of convergence is about the same for all areas.

Maximal volatility v_{max}:
We choose as an upper bound for the time scaled volatility $v_{max} = 6$. This allows us to include highly volatile markets and long maturities. At the same time the method can achieve accuracies close to machine precision.

Minimal volatility v_{min}:
We define a lower bound by

$$v_{min}(x) = 0.001 - 0.03x.$$

For this choice the corresponding prices $c_{min}(x)$ can be computed with the standard machine precision. It includes very low volatilities. For instance at $x = \log(e^{rT} S_0/K) = 0$ this choice even allows call options with a time to maturity of one day ($T = 1/365$) and a Black-Scholes volatility of $\sigma \approx 2\%$. The rate of convergence can be increased further if v_{min} is chosen higher.

Splitting volatilities v_1 and v_2:
We choose v_1 and v_2 according to the properties of the call price function. The call price function has a unique inflection point for $v_c(x) = \sqrt{2|x|}$ where the slope is maximal. Jäckel (2015) proposes the lower bound v_1 as the zeros of the tangent line at this point. The upper bound v_2 is set to be the point where the line hits the maximal call price depending on x. See Figure 4.2. The tangent line is given as

$$f(v) = \frac{\partial}{\partial v} c(x, v_c)\,(v - v_c(x)) + c(x, v_c).$$

Thus

$$\tilde{v}_1(x) = v_c(x) - \frac{c(x, v_c)}{\frac{\partial}{\partial v} c(x, v_c)} \qquad \tilde{v}_2(x) = v_c(x) + \frac{e^{\frac{x}{2}} - c(x, v_c)}{\frac{\partial}{\partial v} c(x, v_c)}$$

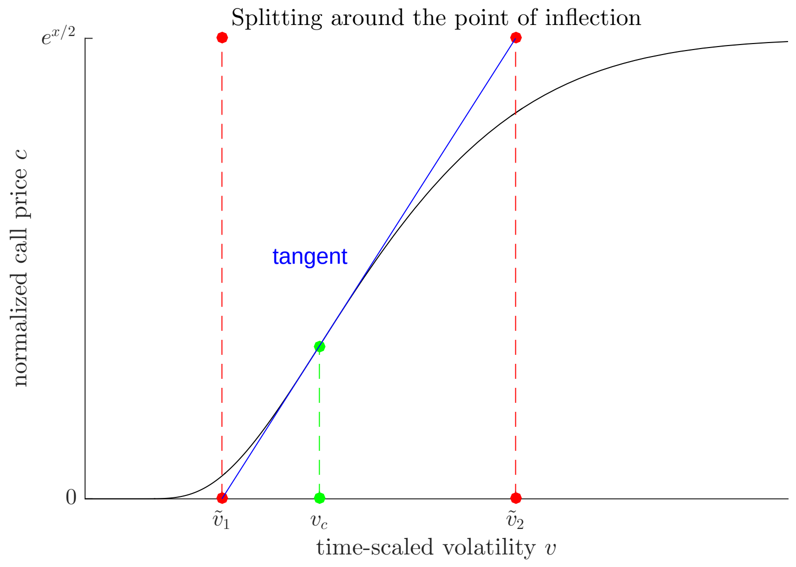


Figure 4.2: Definition of the splitting at $\tilde{v}_1$ and $\tilde{v}_2$ by the zeros of the tangent line at the point of inflection v_c.

However, this choice of boundaries has two serious disadvantages. First, the boundary $\tilde{v}_1$ tends to zero, hence for small values of x we obtain $\tilde{v}_1(x) < v_{min}(x)$. Second, the computation of $\tilde{v}_1(x)$ and $\tilde{v}_2(x)$ requires the evaluation of $c(x, v_c)$ and $\frac{\partial}{\partial v} c(x, v_c)$ for each x. For real-time computation on large data sets, this becomes a computational burden. We solve this problem by replacing $\tilde{v}_1$ and $\tilde{v}_2$ with linear approximations. We propose the boundaries

$$v_1(x) = 0.25 - 0.4x \qquad \text{and} \qquad v_2(x) = 2 - 0.4x.$$

Splitting of the low volatility area
For low volatilities we improve the interpolation by introducing a further splitting in x. The behaviour of the function changes at the point of inflection. As shown before, we need to set $v_1(0) > v_{min}(0) > 0$. Thus at some point the interpolation boundary v_1 will cross this change of behaviour. This can be anticipated by a splitting at the point x where $v_c(x) = v_1(x)$. For the proposed linear splitting these points are given by -11.2152 and -0.0348. The first point is outside the domain $[-5, 0]$ for x, hence we only consider the second point. We divide the area of the low volatilities in an Area I for $x \in [-5, -0.0348]$ and an Area I' for $x \in [-0.0348, 0]$, see Figure 6.1. The empirical results show that this additional splitting further improves the rate of convergence.

5 Error analysis

The following theorem is the theoretical foundation of the high efficiency of the approximation method. Thanks to the analyticity of the Black-Scholes call price and the scaling functions, we gather that the convergence is sub-exponential in the number of nodal points.

Theorem 5.1. *Let $\phi_i^{-1}(\tilde{c}, \tilde{x})$ be analytically continuable to some open region around $[-1, 1]^2$ and let $0 < \phi_i^{-1}([-1, 1], x) < e^{\frac{x}{2}}$ for each $x \in [-1, 1]$. Then there exist constants $\rho_1, \rho_2 > 1, V > 0$ such that $\tilde{v}(\tilde{c}, \tilde{x}) := v(\phi_i^{-1}(\tilde{c}, \tilde{x}), \phi_x^{-1}(\tilde{x}))$ is analytic and for its bivariate Chebshev interpolation $I_i^{N_1^i, N_2^i}(\tilde{c}, \tilde{x}) := \sum_{j=0}^{N_1^i - 1} \sum_{k=0}^{N_2^i - 1} a_{jk} T_j(\tilde{c}) T_k(\tilde{x})$ holds*

$$\max_{(\tilde{c}, \tilde{x}) \in [-1,1]^2} |\tilde{v}(\tilde{c}, \tilde{x}) - I_i^{N_1^i, N_2^i}(\tilde{c}, \tilde{x})| \leq 4V \left(\frac{\rho_1^{-2(N_1 - 1)} + \rho_2^{-2(N_2 - 1)}}{(1 - \rho_1^{-2})(1 - \rho_2^{-2})} \right)^{\frac{1}{2}}.$$

Proof. According to Lemma 7.3.3 of Sauter and Schwab (2010) we need to show that $\tilde{v}(\tilde{c}, \tilde{x}) := v(\phi_i^{-1}(\tilde{c}), \phi_x^{-1}(\tilde{x}))$ is analytically continuable and bounded on $E_{\rho_1} \times E_{\rho_2}$ where E_{ρ_1} and E_{ρ_2} are Bernstein ellipses. Gaß, Glau, Mahlstedt, and Mair (2018) show that the call price is analytic. For fixed $\tilde{x}$ the implied volatility function $\tilde{v}$ is holomorphic in $\tilde{c} \in [-1, 1]$ since the inverse of a bijective holomorphic function is again holomorphic. Next we need to prove analyticity in $\tilde{x} \in [-1, 1]$. Let $\tilde{c} \in [-1, 1]$. Define $F(x, v) := c(x, v) - \phi_i(\tilde{c}, x)$. Then the function $v(\phi_i(\tilde{c}, x), x)$ is implicitly given by the solution of $F(x, v) = 0$. Furthermore, for each $x \in [x_{min}, x_{max}]$, F is holomorphic in some open region with

$$\left| \frac{\partial}{\partial v} F(x, v) \right| = \left| \frac{\partial}{\partial v} c(x, v) \right| = \left| \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2v^2} - \frac{v^2}{8}} \right| > 0$$

as $v > 0$. Thus by the complex implicit function theorem (see Theorem 7.6 of Fritzsche and Grauert (2012)) there exits a unique function $v(\phi_i(\tilde{c}, x), x)$ that is holomorphic in some region around x. Thus $\tilde{v}$ is holomorphic in $G_1 \times G_2$ where G_1 and G_2 are open regions of $[-1, 1]$. Thus there exist $\rho_1, \rho_2 > 1$ such that $E_{\rho_1} \subset G_1$ and $E_{\rho_2} \subset G_2$. The boundedness follows for sufficiently small ρ_1, ρ_2 as $\tilde{v}$ is continuous on $[-1, 1]^2$. □

From the analyticity of the implied volatility function follows that it is infinitely often continuously differentiable in each area and Gaß et al. (2018) showed that in this case the

Chebyshev interpolation also approximates all derivatives. This can be exploited when the implied volatility is used in a gradient-based optimization routine.

We can enhance the efficiency even further by exploiting the low-rank structure of the bivariate functions. To do so, in our implementation we use the *chebfun2* algorithm based on Townsend and Trefethen (2013).

6 Implementation

As a starting point for the approximation of the implied volatility function, we split the interpolation domain into four different areas. For each area, we approximate the implied volatility by a separate bivariate Chebyshev interpolation of the form $v \approx I_i^{N_1^i, N_2^i}(\phi_{i,x}(c), \phi_x(x))$ where ϕ_x is defined as in (3.1) and for each area we have a different scaling $\phi_{i,x}$ in c. For the sake of a lucid presentation, we list the different areas and transformations below. A list of all functions required for the implementation are listed as pseudocode in Algorithm 1 in the appendix.

Area I: For $x \in [-5, -0.0348]$ and $c \in [c_{min}(x), c_1(x)]$ we have

$$\phi_{1,x}(c) := 2 \cdot \frac{\tilde{\phi}_1(c) - \tilde{\phi}_1(c_{min}(x))}{1 - \tilde{\phi}_1(c_{min}(x))} - 1.$$

Area I': For $x \in [-0.0348, 0]$ and $c \in [c_{min}(x), c_1(x)]$ we again use transformation $\phi_{1,x}(c)$.

Area II: For $x \in [-5, 0]$ and $c \in [c_1(x), c_2(x)]$ we have

$$\phi_{2,x}(c) := 2\frac{c - c_1(x)}{c_2(x) - c_1(x)} - 1.$$

Area III: For $x \in [-5, 0]$ and $c \in [c_1(x), c_{max}(x)]$ we have

$$\phi_{3,x}(c) := \frac{2\tilde{\phi}_3(c)}{\tilde{\phi}_3(c_{max}(x))} - 1.$$

The call prices $c_{min}(x), c_1(x), c_2(x)$ and $c_{max}(x)$ correspond to the volatilities

$$v_{min}(x) = 0.001 - 0.03x, \quad v_1(x) = 0.25 - 0.4x, \quad v_2(x) = 2 - 0.4x, \quad v_{max}(x) = 6.$$

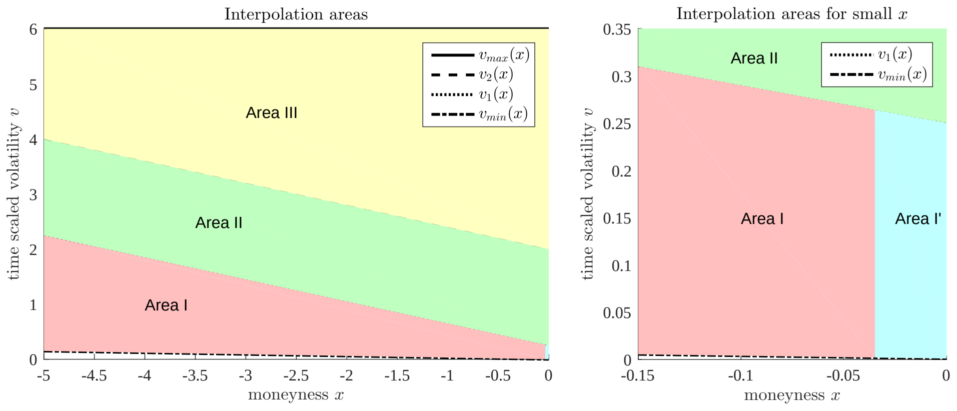


Figure 6.1: The four different interpolation areas of the Chebyshev method.

Moreover, we replace the boundary call prices $c_1(x)$, $c_2(x)$ and $c_{max}(x)$ by univariate interpolations to reduce the runtime further. The evaluation of $c_{min}(x)$, however, is done directly, since for low volatilities the call price is hard to approximate. For this step we can use the implementation of the call price function provided in Jäckel (2015), which is of very high precision.

6.1 Algorithmic structure

Our method allows for an online/offline decomposition:

- *offline-phase* (preparation):
  In each area, we compute the implied volatilities on a $N \times N$ grid of Chebyshev points. Then we apply the *chebfun2* algorithm with pre-specified accuracy and obtain a low-rank approximation. For the calculation of the implied volatility at the grid points one can either use Jäckel's method (see Algorithm 2 in the appendix) or an iterative root finder (see Algorithm 3 in the appendix). For example the bisection method is well suited to compute very accurate implied volatilities at the nodal points. We were able to reach an accuracy in the region of 10^{-13} with this approach.

- *online-phase* (real-time evaluation):
  In the online phase implied volatilities are computed from real-time data, containing a vector of call prices $C \in \mathbb{R}^n$ and the corresponding strikes $K \in \mathbb{R}^n$, spot prices $S_0 \in \mathbb{R}^n$, maturities $T \in \mathbb{R}^n$ and interest rates $r \in \mathbb{R}^n$. Algorithm 4 in the appendix provides a pseudocode version for the online step.

  – *Normalization*: We calculate the normalized call price c and the forward moneyness x from the data. Option prices with $x > 0$ need to be transformed to prices with moneyness $-x$ by Formula (2.3).

  – *Splitting*: For each pair (x, c), we need to find the corresponding area. As the computation of $c_{min}(x)$ requires the most computational effort, we proceed as follows. First, we compute $c_{max}(x)$ and check if $c \leq c_{max}(x)$. Next, we check if $c < c_2(x)$ and eventually $c < c_1(x)$. Only in the latter case, do we compute $c_{min}(x)$ and check whether $c \geq c_{min}(x)$.

  – *Transformation*: We compute the transformed call prices $\phi_{i,x}(c)$ and moneyness $\phi_x(x)$ with the respective transformations.

  – *Evaluation*: We evaluate the bivariate Chebyshev interpolations provided in the *offline-phase* at the transformed call prices and moneyness to obtain the time-scaled implied volatility.

The runtime of the *online*-phase is primarily determined by the *splitting* and the *evalutation*-phase. The evaluation of the bivariate interpolations can be done in different ways and can be performed in very few computational steps depending on the required accuracy.

In the offline the $chebfun2$ algorithm returns in each area a $chebfun2$-object which is a Chebyshev interpolant in low rank form $I^{N_1,N_2} = \sum_{j=1}^{k} d_j c_j(y) r_j(x)$ where $r_j(x)$ and $c_j(y)$ are univariate Chebyshev interpolations of rank N_1 and N_2. More precisely,

$$r_j(x) = \sum_{i=0}^{N_1-1} a_i T_i(x) \text{ and } c_j(y) = \sum_{i=0}^{N_2-1} b_i T_i(y).$$

One can now either store the four $chebfun2$-objects directly and use them in the online evaluation or one can extract the coefficients and evaluate the low rank form manually. The first choice is simpler and for many applications sufficient, the latter can be slightly faster when it is implemented in an efficient way. The Chebyshev polynomials $T_0, T_1, ..., T_{N_1-1}$ can be computed in different ways, for instance by $T_k(x) = \cos(k \cos^{-1}(x))$ or by the iterative formula $T_0(x) = 0$, $T_1(x) = 1$, $T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x)$. It turns out that for large data sets the iterative evaluation of the Chebyshev polynomials is advantageous compared to the cosine formula as only simple additions and multiplications are involved while the evaluation of cos and $\cos^{-1}$ is slightly slower. Therefore we use this approach in our implementation.

After setting up the Chebyshev method for a pre-specified accuracy we obtain a low-rank approximation for each of the four areas. Table 6.1 displays the ranks k and the grid sizes N_1, N_2 of the low rank interpolation operator for the three specified accuracies 10^{-6} (low accuracy), 10^{-9} (medium accuracy) and 10^{-12} (high accuracy). As expected the ranks and grid sizes are higher for a higher accuracy. Moreover, we observe that we need more interpolation nodes in Area I and Area I' to obtain the same level of accuracy as in Area II and Area III.

Area	low accuracy	medium accuracy	high accuracy
Area I	$k = 10$, $N_1 = 25$, $N_2 = 36$	$k = 16$, $N_1 = 46$, $N_2 = 79$	$k = 22$, $N_1 = 67$, $N_2 = 122$
Area I'	$k = 9$, $N_1 = 27$, $N_2 = 18$	$k = 16$, $N_1 = 51$, $N_2 = 39$	$k = 23$, $N_1 = 77$, $N_2 = 57$
Area II	$k = 6$, $N_1 = 21$, $N_2 = 20$	$k = 11$, $N_1 = 36$, $N_2 = 33$	$k = 14$, $N_1 = 51$, $N_2 = 47$
Area III	$k = 5$, $N_1 = 11$, $N_2 = 9$	$k = 7$, $N_1 = 17$, $N_2 = 14$	$k = 9$, $N_1 = 23$, $N_2 = 19$

Table 6.1: Rank k and grid sizes N_1, N_2 of the low rank Chebyshev interpolation in the different areas for three different levels of pre-specified accuracy.

7 Numerical Results

We compare our approximation method to

- the Jäckel (2015) method,
- the approximation formula given in Li (2008),
- the approximation formula given in Li (2008) with the proposed polishing of two Newton-Raphson iterations,
- the Newton-Raphson algorithm with the starting point given in Manaster and Koehler (1982). The algorithm terminates if $|v_n - v_{n-1}| < 10^{-6}$.

In order to do so, we first choose a domain $\mathcal{D}_1$ on which all methods can be applied and compare the resulting errors and runtimes (Section 7.1). On the complete domain $\mathcal{D}_2$, we compare the proposed method to the Jäckel (2015) method and the Newton-Raphson algorithm as those are the only ones that can also be applied on this set (Section 7.2). Finally, we include actual market data (Section 7.3). All codes are written in *Matlab* R2014a and the experiments are run on a computer with Intel Xeon CPU with 3.10 GHz with 20 MB SmartCache.

7.1 Comparison on Domain $\mathcal{D}_1$

The domain on which all methods work is the domain of Li (2008) bounded below by $v_{min}(x)$, i.e.

$$\mathcal{D}_1 := \left\{ -0.5 \leq x \leq 0.5, 0 \leq v \leq 1, \max\left(\frac{|x|}{2}, v_{min}(-|x|)\right) \leq v \right\}$$

See Figure 7.1 for a comparison of the domain of Li (2008) and the domain of the Chebyshev method. On $\mathcal{D}_1$ we compute normalized call prices on a 1000×1000-grid, where the distribution of the points is determined as in the numerical example of Section 3. We compare the runtimes and errors in the time-scaled volatilities $\Delta v := |v - v^{imp}|$ and the repricing errors $\Delta c := |c(x, v) - c(x, v^{imp})|$ of the methods. Figure 7.2 illustrates the errors Δv of the reference methods. Figure 7.3 displays the errors of the Chebyshev approach for three different pre-specified accuracies.

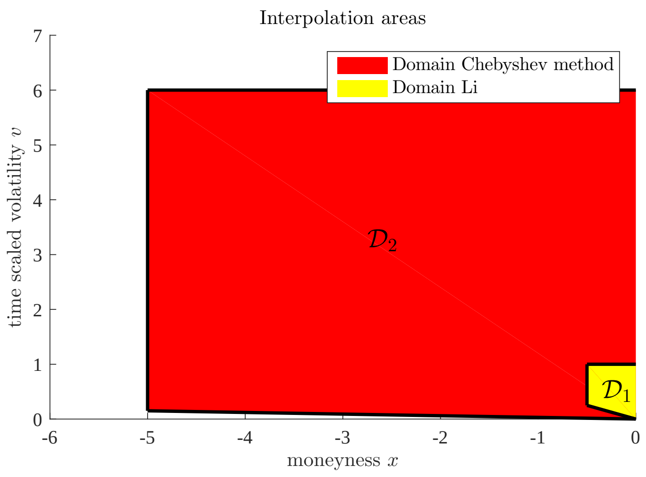


Figure 7.1: Domain $\mathcal{D}_2$ of the Chebyshev interpolation (red), domain of Li (yellow) and domain $\mathcal{D}_1$ as the intersection of both.

The Jäckel (2015) method comes with a solution close to machine precision for all input parameters and thus qualifies as our reference method in the offline-phase of the Chebyshev approximation. Also the Newton-Raphson algorithm reaches very high precision. The approximation of Li (2008), however, is not able to reach the same range of precision. As Table 7.1 shows, the mean error of σ is a factor even 10^{10} higher than Jäckel's approximation. The proposed modification of Li (2008) with two additional Newton-Raphson steps reduces the error. However, for low volatilities the effect is rather small and the maximal error is still in the region of 10^{-5}, see Table 7.1.

Figure 7.3 displays the interpolation error of the Chebyshev method for three different pre-specified accuracies. The error is of the same order for the whole interpolation domain, which shows that a pre-specified accuracy can be reached for all input parameters with the same complexity.

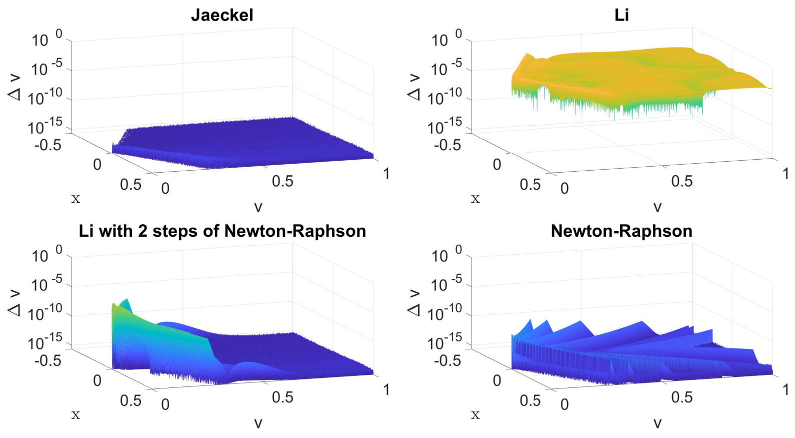


Figure 7.2: Errors $\Delta v := |v - v^{imp}|$ of the reference methods

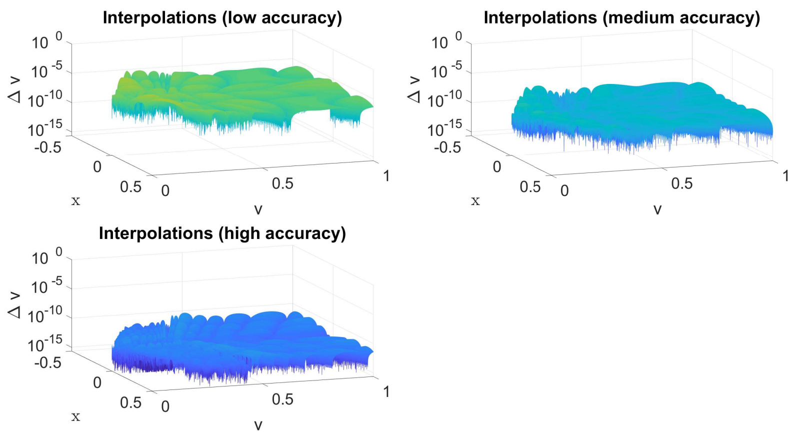


Figure 7.3: Errors $\Delta v := |v - v^{imp}|$ of the Chebyshev approach with three different pre-specified accuracies.

Table 7.1 shows the maximal and the mean error in terms of the time-scaled volatilities and the normalized call prices as well as the runtime as a proportion of the runtime of the Newton-Raphson method, which takes $1.45s$. For the Chebyshev method, the runtime measures the time of the online phase. When comparing the runtimes, the Li method is the fastest. It comes, however, with the lowest precision of a maximal error in σ of $3.26 \cdot 10^{-3}$. For a higher precision in the range of 10^{-5}, the Chebyshev method with low accuracy turns out to be faster than the improved Li method. Comparing the mean, the same holds for the Chebyshev method with medium accuracy. For very high precisions the Chebyshev method with high accuracy is faster than the Newton-Raphson approach. Compared to Jäckel's method, the Chebyshev approach is two times faster but with a maximal error of 10^{-11} instead of 10^{-14}.

Method	max $\lvert\Delta\sigma\rvert$	mean $\lvert\Delta\sigma\rvert$	max $\lvert\Delta c\rvert$	mean $\lvert\Delta c\rvert$	runtime
Jäckel	$2.80 \cdot 10^{-14}$	$4.57 \cdot 10^{-16}$	$1.67 \cdot 10^{-15}$	$9.99 \cdot 10^{-17}$	1.39
Li	$3.26 \cdot 10^{-3}$	$3.42 \cdot 10^{-4}$	$2.15 \cdot 10^{-4}$	$9.43 \cdot 10^{-5}$	0.12
Li with 2 steps of Newton-Raphson	$2.02 \cdot 10^{-5}$	$6.12 \cdot 10^{-9}$	$1.10 \cdot 10^{-6}$	$3.89 \cdot 10^{-10}$	0.63
Newton-Raphson	$2.05 \cdot 10^{-10}$	$6.32 \cdot 10^{-14}$	$2.91 \cdot 10^{-11}$	$1.00 \cdot 10^{-14}$	1
Chebyshev method (low accuracy)	$1.52 \cdot 10^{-5}$	$1.40 \cdot 10^{-6}$	$4.91 \cdot 10^{-6}$	$3.94 \cdot 10^{-7}$	0.40
Chebyshev method (medium accuracy)	$3.20 \cdot 10^{-8}$	$2.17 \cdot 10^{-9}$	$3.52 \cdot 10^{-9}$	$5.92 \cdot 10^{-10}$	0.55
Chebyshev method (high accuracy)	$4.88 \cdot 10^{-11}$	$4.78 \cdot 10^{-12}$	$1.51 \cdot 10^{-11}$	$1.41 \cdot 10^{-12}$	0.67

Table 7.1: Interpolation error and runtimes on domain $\mathcal{D}_1$.

7.2 Comparison on Domain $\mathcal{D}_2$

We compare the Chebyshev method on the large domain $\mathcal{D}_2$ to the Newton-Raphson approach and the algorithm of Jäckel. The errors and runtimes on a 1000×1000 grid, specified as in Section 7.1, are computed. Figure 7.4 and 7.5 illustrate the resulting errors of the reference methods and the Chebyshev approach. The observations of the error behaviour on the larger domain $\mathcal{D}_2$ are consistent with that on the smaller domain $\mathcal{D}_1$, see Figure 7.2 and Figure 7.3.

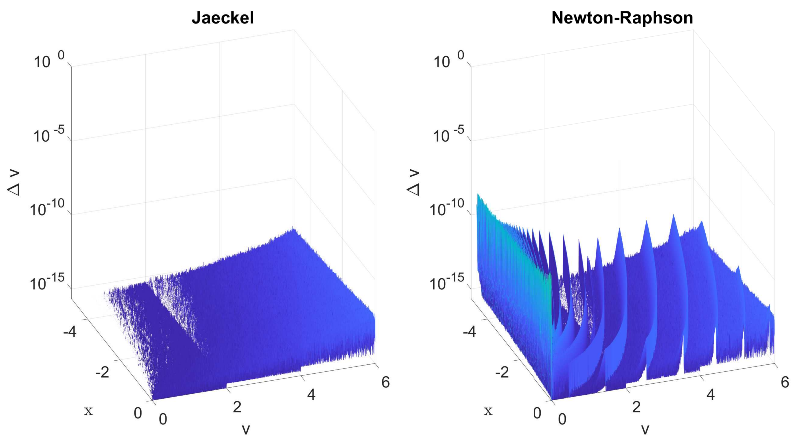


Figure 7.4: Errors $\Delta v := |v - v^{imp}|$ of the reference methods

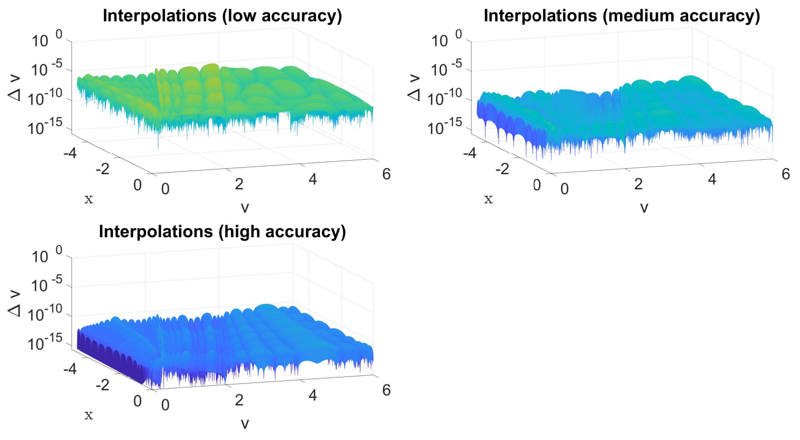


Figure 7.5: Errors $\Delta v := |v - v^{imp}|$ of the reference methods

Table 7.2 shows the maximal and the mean error as well as the runtimes scaled as in 7.1. Here, the Newton-Raphson method takes $4.29s$.

Method	max $\lvert\Delta\sigma\rvert$	mean $\lvert\Delta\sigma\rvert$	max $\lvert\Delta c\rvert$	mean $\lvert\Delta c\rvert$	runtime
Jäckel	$5.30 \cdot 10^{-13}$	$5.35 \cdot 10^{-15}$	$2.55 \cdot 10^{-15}$	$7.10 \cdot 10^{-17}$	0.52
Newton-Raphson	$8.34 \cdot 10^{-8}$	$6.64 \cdot 10^{-12}$	$1.94 \cdot 10^{-11}$	$1.28 \cdot 10^{-15}$	1
Chebyshev method (low accuracy)	$2.55 \cdot 10^{-5}$	$1.85 \cdot 10^{-6}$	$4.63 \cdot 10^{-6}$	$1.42 \cdot 10^{-7}$	0.14
Chebyshev method (medium accuracy)	$4.42 \cdot 10^{-8}$	$2.38 \cdot 10^{-9}$	$4.02 \cdot 10^{-9}$	$1.36 \cdot 10^{-10}$	0.16
Chebyshev method (high accuracy)	$1.66 \cdot 10^{-10}$	$1.32 \cdot 10^{-11}$	$1.52 \cdot 10^{-11}$	$4.83 \cdot 10^{-13}$	0.20

Table 7.2: Interpolation error and runtimes on domain $\mathcal{D}_2$.

To reach a medium accuracy in the maximal error in the range of 10^{-8}, the Chebyshev method is more than six times faster than the Newton-Raphson approach. Moreover, the Chebyshev method is able to reach higher accuracies of 10^{-10} and still needs only 20% of the runtime of Newton-Raphson. Jäckel's method reaches very high precisions and is faster than Newton-Raphson. Compared the Jäckel method, the Chebyshev method allows us to pre-specify accuracies and reduce the runtimes significantly. For example, if accuracies in the region of 10^{-8} are sufficient, the Chebyshev method is more than three times faster than Jäckel's approach.

7.3 Comparison for market data

In Section 3.1 we investigated market data of options and concluded that a significant part of the options is not covered by the domain of Li (2008). This was the motivation to consider a much larger interpolation domain for the Chebyshev method. An empirical investigation confirms that all the options shown in Figure 3.2 lie within our domain.

Next, we compare the Chebyshev method on this market data to the Newton-Raphson approach and the algorithm of Jäckel. The errors and runtimes are computed for options on the S&P 500 index traded on 7/17/2017 (Source Thomson Reuters Eikon). We use the same options as for Figure 3.2. To obtain more reliable results for the runtime comparison we compute the implied volatilities of the options 5000 times.

Table 7.3 shows the maximal and the mean error as well as the runtimes scaled as in Section 7.1. Here, the Newton-Raphson method takes $5.72s$.

Method	max $\lvert\Delta\sigma\rvert$	mean $\lvert\Delta\sigma\rvert$	max $\lvert\Delta c\rvert$	mean $\lvert\Delta c\rvert$	runtime
Jäckel	$8.05 \cdot 10^{-16}$	$1.40 \cdot 10^{-16}$	$2.11 \cdot 10^{-15}$	$2.43 \cdot 10^{-16}$	0.89
Newton-Raphson	$1.78 \cdot 10^{-10}$	$2.91 \cdot 10^{-12}$	$7.72 \cdot 10^{-12}$	$2.22 \cdot 10^{-13}$	1
Chebyshev method (low accuracy)	$1.57 \cdot 10^{-5}$	$2.95 \cdot 10^{-6}$	$4.44 \cdot 10^{-6}$	$4.78 \cdot 10^{-7}$	0.37
Chebyshev method (medium accuracy)	$4.19 \cdot 10^{-8}$	$3.87 \cdot 10^{-9}$	$3.45 \cdot 10^{-9}$	$3.98 \cdot 10^{-10}$	0.48
Chebyshev method (high accuracy)	$1.73 \cdot 10^{-11}$	$2.21 \cdot 10^{-12}$	$2.70 \cdot 10^{-12}$	$2.91 \cdot 10^{-13}$	0.58

Table 7.3: Interpolation error and runtimes for S&P 500 market data.

The results are similar to those of Section 7.2. The Chebyshev method is the fastest of the three methods and reaches the target accuracies. The method is about twice as fast as the Newton-Raphson approach for similar accuracies. Again, Jäckel's method reaches very high precisions but it is significantly slower than the Chebyshev method.

Besides the observed gain in efficiency the Chebyshev method enjoys conceptual advantages. It delivers a closed-form approximation in a simple and easy to use polynomial structure. Moreover, the proposed approach can be applied to other problems of similar structure. The following section illustrates this flexibility.

8 Laplace implied volatility

Besides the Black-Scholes implied volatility there are several other models with implied volatilities. To overcome the problems of thin tails in the Black-Scholes model, Madan (2016) proposes the replacement of the density of the normal distribution with a Laplace density. This leads to a model with fatter tails without adding additional parameters. The stock price process in this model is defined by

$$S_t = S_0 \exp\left((r-q)t + X_t + \log\left(1 - \frac{\sigma^2 t}{2}\right)\right), \tag{8.1}$$

where X_t is distributed according to the time-dependent Laplace density

$$g(x,t) = \frac{1}{\sigma\sqrt{2t}} e^{-\frac{\sqrt{2}|x|}{\sigma\sqrt{t}}}, \qquad x \in \mathbb{R}. \tag{8.2}$$

The call price in the model is given by

$$C(S_0, K, r, q, t) = \\ \begin{cases} e^{-qt} S_0 \frac{e^{-(\sqrt{2}-\sigma\sqrt{t})|d|}}{2}\left(1 + \sigma\sqrt{t/2}\right) - e^{-rt} K \frac{e^{-\sqrt{2}|d|}}{2}, & d > 0 \\ K e^{-rT}\left(\frac{e^{-\sqrt{2}|d|}}{2} - 1\right) - S_0 e^{-qt}\left(\frac{e^{-(\sqrt{2}+\sigma\sqrt{t})|d|}}{2}\left(1 - \sigma\sqrt{t/2}\right) - 1\right), & d < 0 \end{cases}$$

with

$$d = \frac{\log(K/S_0)}{\sigma\sqrt{t}} - \frac{(r-q)\sqrt{t}}{\sigma} - \frac{\log(1-\frac{\sigma^2 t}{2})}{\sigma\sqrt{t}}.$$

Madan (2016) shows that this model can be used for hedging purposes and outperforms classical delta hedging in the Black-Scholes model. Madan and Wang (2016) considered the application of the model to risk management. For both, hedging and risk management, it is necessary to have a fast and accurate formula for the Laplace implied volatility. To this end, we apply the bivariate Chebyshev method to implied volatilities based on the Laplace density.

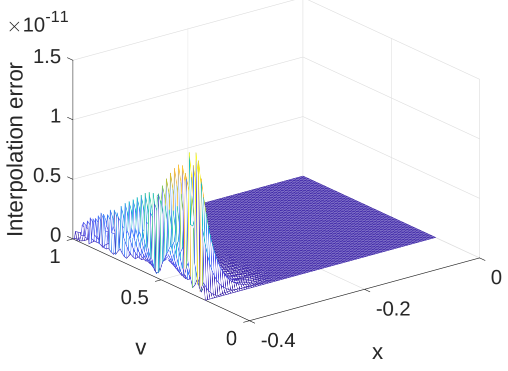
As in the previous case, we normalize the call price by setting $v = \sigma\sqrt{T}$, $x = \log(S_0 e^{(r-q)T}/K)$ and $C(S_0, K, r, q, t) = \sqrt{S_0 e^{-(r+q)T} K} c(x, v)$ to

$$c(x,v) = \begin{cases} \frac{e^{-(\sqrt{2}-v)|d|+x/2}}{2}(1+v/\sqrt{2}) - \frac{e^{\sqrt{2}|d|-x/2}}{2}, & d > 0 \\ e^{-x/2}\left(\frac{e^{-\sqrt{2}|d|}}{2} - 1\right) - e^{x/2}\left(\frac{e^{-(\sqrt{2}+v)|d|}}{2}\left(1 - v/\sqrt{2}\right) - 1\right), & d < 0 \end{cases} \tag{8.3}$$

with

$$d = -\frac{x}{v} - \frac{\log\left(1 - \frac{v^2}{2}\right)}{v}.$$

Similar to Section 2.1, we thus have reduced the approximation problem to a bivariate interpolation. For the domain $0.25 \leq v \leq 1$, $-0.4 \leq x \leq 0$, we perform a bivariate Chebyshev interpolation of the Laplace implied volatility. At the interpolation nodes a Brent-Dekker algorithm is used to compute the implied volatilities. Figure 8.1 shows the exponential error decay of the interpolation on a $N \times N$-Chebyshev grid. The interpolation is already in the region of 10^{-11} for $N = 50$. This shows the high potential of the method in the Laplace model, comparable to the numerical example in Section 3. In order to obtain high efficiency on a larger domain, one can establish a splitting procedure with appropriate scaling functions by exploiting the limit behaviour of the Laplace call price, in the spirit of Section 4.



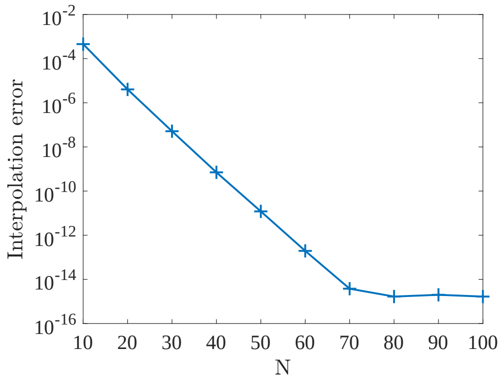


Figure 8.1: Exponential error decay of the bivariate interpolation of the Laplace implied volatility on a $N \times N$-Chebyshev grid for $0.25 \leq v \leq 1$, $-0.4 \leq x \leq 0$. For $N = 50$ the error is already in the region of 10^{-11}.

9 Conclusion

We have introduced a new approximation method to compute the implied volatility. The backbone of the method is a bivariate Chebyshev interpolation. We have set up an interpolation domain, which is able to cover all relevant options based on observed market data. In order to achieve highest efficiency, we have split the domain into different interpolation areas with appropriate scaling functions. A theoretical error analysis shows subexponential convergence and a combination with low-rank techniques allows us to enhance the observed efficiency. Compared to other non-iterative approximation methods, the Chebyshev method is able to cover all relevant option data, including deep in and far out of the money options as well as low and high volatilities, see Figure 3.2 and Figure 7.1. Moreover, numerical experiments show that the Chebyshev method achieves considerably higher accuracies on the common domain $\mathcal{D}_1$. In comparison to the iterative method of Jäckel (2015), the Chebyshev method can reduce the runtimes significantly by pre-specifying the target accuracy. Besides the gain in efficiency, the Chebyshev method exhibits conceptual advantages:

- *Closed form bivariate approximation formula*: The Chebyshev interpolations in all areas have the *polynomial structure*

$$v(c,x) \approx \sum_{j=1}^{k} d_j c_j(\phi_i(c)) r_j(\phi_x(x)) \tag{9.1}$$

  where ϕ_i and ϕ_x are the transformations on the respective area. This structure can be further explored to express derivatives in a simple form. For example the first derivative with respect to the call price is given by

$$\frac{\partial}{\partial c} v(c,x) = \left(\frac{\partial}{\partial v} c(v,x)\right)^{-1} = \sum_{j=1}^{k} d_j r_j(\phi_x(x)) \cdot \frac{\partial}{\partial \phi_i(c)} c_j(\phi_i(c)) \cdot \frac{\partial}{\partial c}\phi_i(c)$$

  We observe that the approximate derivative is again a function of x and c in polynomial structure. In particular, this avoids the computation of the implied volatility itself.

- *Easy Implementation*: Once the interpolation operator is set up in an offline phase, the polynomial structure of the approximation formula 9.1 leads to simple code. This facilitates the transfer of the code to other systems and programming languages as part of the maintenance. The method only requires the coefficients of the bivariate Chebyshev interpolation on each area and the corresponding scaling functions.

- *Adaptability*: The efficiency of the Chebyshev method can be even further improved by incorporating additional knowledge. If the option data of interest lies in a domain smaller than $\mathcal{D}_2$, one can tailor the method to this domain by modifying the splitting.

The Chebyshev method enjoys high flexibility and the approach can be transferred to similar problems. We have illustrated this by approximating the Laplace implied volatility.

Declaration of Interest

The authors report no conflicts of interest. The authors alone are responsible for the content and writing of the paper.

Acknowledgements

Christian Pötz would like to thank the KPMG Center of Excellence in Risk Management for their support.

A Appendix: Algorithm

In this appendix, we present the algorithms of the Chebyshev method for the implied volatility in form of pseudocodes. The implementation is described in detail in Section 6. Algorithm 1 defines the transformations and scaling functions. Algorithm 2 and Algorithm 3 describe two versions of the offline phase of our approach. Algorithm 2 uses Jäckel's method to compute the implied volatilities at the transformed Chebyshev grid points. Algorithm 3 shows the offline phase using a general root finder to compute the implied volatilities at the transformed grid points. This illustrates the flexibility of our approach to use any method to compute the implied volatility in the offline phase which is sufficiently accurate on the domain of interests. In the offline phase, efficiency is not essential since it is only performed once to set up the method and for instance a simple but slow bisection search is also an appropriate choice. Algorithm 4 presents the Chebyshev implied volatility function using $chebfun2$ objects in the four interpolation areas. As stated in Section 6.1, the $chebfun2$ objects can be replaced by other techniques to evaluate or approximate the bivariate Chebyshev polynomials. For example one can exploit the recursive structure of the polynomials.

Algorithm 1 Initilization

```
1: **Define normalized call price:**
2: $c(x,v) \leftarrow e^{\frac{x}{2}}\Phi\left(\frac{x}{v}+\frac{v}{2}\right) - e^{-\frac{x}{2}}\Phi\left(\frac{x}{v}-\frac{v}{2}\right)$
3: **Define splitting call prices:**
4: $c_{min}(x) \leftarrow c(x, 0.001-0.03x)$, $c_1(x) \leftarrow c(x, 0.25-0.4x)$, $c_2(x) \leftarrow c(x, 2-0.4x)$, $c_{max}(x) \leftarrow c(x,6)$
5: **Define scaling functions:**
6: $\varphi(x,\underline{x},\overline{x}) \leftarrow 1 - 2\cdot\frac{\overline{x}-x}{\overline{x}-\underline{x}}$
7: $\tilde{\phi}_1(x,c) \leftarrow 2\left(-\frac{2}{(x-\delta)^2}\log(c) + \frac{2}{(x-\delta)^2}\log(c_1(x)) + 1\right)^{-\frac{1}{2}} - 1$
8: $\phi_1(x,c) \leftarrow 2\frac{\tilde{\phi}_1(x,c)-\tilde{\phi}_1(x,c_{min}(x))}{1-\tilde{\phi}_1(x,c_{min}(x))} - 1$
9: $\phi_2(x,c) \leftarrow 2\frac{c-c_1(x)}{c_2(x)-c_1(x)} - 1$
10: $\tilde{\phi}_3(x,c) \leftarrow \left(-8\log\left(\frac{e^{\frac{x}{2}}-c}{e^{\frac{x}{2}}-c_2(x)}\right)\right)^{\frac{1}{2}}$
11: $\phi_3(x,c) \leftarrow \frac{2\tilde{\phi}_3(x,c)}{\tilde{\phi}_3(x,c_{max}(x))} - 1$
12: **Define inverse scaling functions:**
13: $\varphi^{-1}(y,\underline{x},\overline{x}) \leftarrow \overline{x} - 0.5(\overline{x}-\underline{x})(1-y)$
14: $\tilde{\phi}_1^{-1}(x,\tilde{c}) \leftarrow c_1(x)e^{-\frac{2(x-\delta)^2}{(\tilde{c}+1)^2}+\frac{(x-\delta)^2}{2}}$
15: $\phi_1^{-1}(x,\tilde{c}) \leftarrow \tilde{\phi}_1^{-1}(x, \tilde{\phi}_1(x,c_{min}(x)) + 0.5(1-\tilde{\phi}_1(x,c_{min}(x)))(1+\tilde{c}))$
16: $\phi_2^{-1}(x,\tilde{c}) \leftarrow c_1(x) + \frac{1}{2}(\tilde{c}+1)(c_2(x)-c_1(x))$
17: $\tilde{\phi}_3^{-1}(x,\tilde{c}) \leftarrow e^{\frac{x}{2}} - \left(e^{x/2}-c_2(x)\right)e^{-\frac{\tilde{c}^2}{8}}$
18: $\phi_3^{-1}(x,\tilde{c}) \leftarrow \tilde{\phi}_3^{-1}(x, 0.5\tilde{\phi}_3(x,c_{max}(x))(1+\tilde{c}))$
```

Algorithm 2 Pre-computation step (Using Jäckel's method)

```
Require: $\widehat{v}_{\mathrm{JL}}(x,c)$ (Jäckel's implied vola function), load *chebfun* package, $\varepsilon$ error tolerance
 1: **Area I**
 2: $\tilde{v}_1(\tilde{x},\tilde{c}) \;\leftarrow\; \widehat{v}_{\mathrm{JL}}(\varphi^{-1}(\tilde{x},-5,-0.0348),\phi_1^{-1}(\tilde{x},\tilde{c}))$
 3: $I_1(\tilde{x},\tilde{c}) \;\leftarrow\;$ chebfun2$((\tilde{x},\tilde{c})\mapsto\tilde{v}_1(\tilde{x},\tilde{c}),\ \mathit{'tol'}=\varepsilon)$
 4: **Area I'**
 5: $\tilde{v}_{1+}(\tilde{x},\tilde{c}) \;\leftarrow\; \widehat{v}_{\mathrm{JL}}(\varphi^{-1}(\tilde{x},-0.0348,0),\phi_1^{-1}(\tilde{x},\tilde{c}))$
 6: $I_{1+}(\tilde{x},\tilde{c}) \;\leftarrow\;$ chebfun2$((\tilde{x},\tilde{c})\mapsto\tilde{v}_{1+}(\tilde{x},\tilde{c}),\ \mathit{'tol'}=\varepsilon)$
 7: **Area II**
 8: $\tilde{v}_2(\tilde{x},\tilde{c}) \;\leftarrow\; \widehat{v}_{\mathrm{JL}}(\varphi^{-1}(\tilde{x},-5,0),\phi_2^{-1}(\tilde{x},\tilde{c}))$
 9: $I_2(\tilde{x},\tilde{c}) \;\leftarrow\;$ chebfun2$((\tilde{x},\tilde{c})\mapsto\tilde{v}_2(\tilde{x},\tilde{c}),\ \mathit{'tol'}=\varepsilon)$
10: **Area III**
11: $\tilde{v}_3(\tilde{x},\tilde{c}) \;\leftarrow\; \widehat{v}_{\mathrm{JL}}(\varphi^{-1}(\tilde{x},-5,0),\phi_3^{-1}(\tilde{x},\tilde{c}))$
12: $I_3(\tilde{x},\tilde{c}) \;\leftarrow\;$ chebfun2$((\tilde{x},\tilde{c})\mapsto\tilde{v}_3(\tilde{x},\tilde{c}),\ \mathit{'tol'}=\varepsilon)$
13: **Output:** Four *chebfun2* objects $I_1(\tilde{x},\tilde{c})$, $I_{1+}(\tilde{x},\tilde{c})$, $I_2(\tilde{x},\tilde{c})$, $I_3(\tilde{x},\tilde{c})$
```

Algorithm 3 Pre-computation step (Using a rootfinder)

```
Require: $\mathit{root}(f,\mathit{tol})$ (rootfinder which takes a function $f$ and error tolerance $\mathit{tol}$), load *chebfun*
    package, $\varepsilon$ error tolerance
 1: **Area I**
 2: $\tilde{v}_1(\tilde{x},\tilde{c}) \;\leftarrow\; \mathit{root}(v\mapsto\phi_1^{-1}(\tilde{x},\tilde{c})-c(\varphi^{-1}(\tilde{x},-5,-0.0348),v),\varepsilon)$
 3: $I_1(\tilde{x},\tilde{c}) \;\leftarrow\;$ chebfun2$((\tilde{x},\tilde{c})\mapsto\tilde{v}_1(\tilde{x},\tilde{c}),\ \mathit{'tol'}=\varepsilon)$
 4: **Area I'**
 5: $\tilde{v}_{1+}(\tilde{x},\tilde{c}) \;\leftarrow\; \mathit{root}(v\mapsto\phi_1^{-1}(\tilde{x},\tilde{c})-c(\varphi^{-1}(\tilde{x},-0.0348,0),v),\varepsilon)$
 6: $I_{1+}(\tilde{x},\tilde{c}) \;\leftarrow\;$ chebfun2$((\tilde{x},\tilde{c})\mapsto\tilde{v}_{1+}(\tilde{x},\tilde{c}),\ \mathit{'tol'}=\varepsilon)$
 7: **Area II**
 8: $\tilde{v}_2(\tilde{x},\tilde{c}) \;\leftarrow\; \mathit{root}(v\mapsto\phi_2^{-1}(\tilde{x},\tilde{c})-c(\varphi^{-1}(\tilde{x},-5,0),v),\varepsilon)$
 9: $I_2(\tilde{x},\tilde{c}) \;\leftarrow\;$ chebfun2$((\tilde{x},\tilde{c})\mapsto\tilde{v}_2(\tilde{x},\tilde{c}),\ \mathit{'tol'}=\varepsilon)$
10: **Area III**
11: $\tilde{v}_3(\tilde{x},\tilde{c}) \;\leftarrow\; \mathit{root}(v\mapsto\phi_3^{-1}(\tilde{x},\tilde{c})-c(\varphi^{-1}(\tilde{x},-5,0),v),\varepsilon)$
12: $I_3(\tilde{x},\tilde{c}) \;\leftarrow\;$ chebfun2$((\tilde{x},\tilde{c})\mapsto\tilde{v}_3(\tilde{x},\tilde{c}),\ \mathit{'tol'}=\varepsilon)$
13: **Output:** Four *chebfun2* objects $I_1(\tilde{x},\tilde{c})$, $I_{1+}(\tilde{x},\tilde{c})$, $I_2(\tilde{x},\tilde{c})$, $I_3(\tilde{x},\tilde{c})$
```

Algorithm 4 The Chebyshev implied volatility function

```
Require: Four chebfun2 objects $I_1(\tilde{x},\tilde{c})$, $I_{1+}(\tilde{x},\tilde{c})$, $I_2(\tilde{x},\tilde{c})$, $I_3(\tilde{x},\tilde{c})$
 1: Input call price $C$, strike $K$, spot $S_0$, maturity $T$ and rate $r$
 2: Normalization
 3: $x \leftarrow rT + \log(S_0/K)$,   $c \leftarrow C/\sqrt{S_0 e^{-rT} K}$
 4: if $x < 0$ then $c \leftarrow c + e^{-x/2} - e^{x/2}$,   $x \leftarrow -x$
 5: Splitting
 6: if $c \le c_{max}(x)$ and $x \le 5$ then
 7:     if $c < c_2(x)$ then
 8:         if $c < c_1(x)$ and $c \ge c_{min}(x)$ then
 9:             if $x < -0.0348$ then $Area \leftarrow$ Area I'
10:             else $Area \leftarrow$ Area I
11:         else $Area \leftarrow$ Area II
12:     else $Area \leftarrow$ Area III
13: else Return $c$ or $x$ is too high
14: Transformation and Evaluation
15: Switch cases '$Area$'
16: Case Area I: $\tilde{x} \leftarrow \varphi(x, -5, -0.0348)$,   $\tilde{c} \leftarrow \phi_1(x,c)$ and $v \leftarrow I_1(\tilde{x},\tilde{c})$
17: Case Area I': $\tilde{x} \leftarrow \varphi(x, -0.0348, 0)$,   $\tilde{c} \leftarrow \phi_1(x,c)$ and $v \leftarrow I_{1+}(\tilde{x},\tilde{c})$
18: Case Area II: $\tilde{x} \leftarrow \varphi(x, -5, 0)$,   $\tilde{c} \leftarrow \phi_2(x,c)$ and $v \leftarrow I_2(\tilde{x},\tilde{c})$
19: Case Area III: $\tilde{x} \leftarrow \varphi(x, -5, 0)$,   $\tilde{c} \leftarrow \phi_1(x,c)$ and $v \leftarrow I_3(\tilde{x},\tilde{c})$
20: Return: $v$
```

References

Baumeister, J. (2013). Inverse problems in finance. In *Recent developments in computational finance: Foundations, algorithms and applications* (pp. 81–157). doi:10.1142/9789814436434_0003

Black, F., & Scholes, M. (1973). The pricing of options and other liabilities. *Journal of Political Economy*, *81*, 637–654. doi:10.1086/260062

Brenner, M., & Subrahmanyan, M. G. (1988). A simple formula to compute the implied standard deviation. *Financial Analysts Journal*, *44*(5), 80–83. doi:10.2469/faj.v44.n5.80

Chambers, D. R., & Nawalkha, S. K. (2001). An improved approach to computing implied volatility. *Financial Review*, *36*(3), 89–100. doi:10.1111/j.1540-6288.2001.tb00021.x

Chance, D. M. (1996). A generalized simple formula to compute the implied volatility. *Financial Review*, *31*(4), 859–867. doi:10.1111/j.1540-6288.1996.tb00900.x

Corrado, C. J., & Miller, T. W. (1996). A note on a simple, accurate formula to compute implied standard deviations. *Journal of Banking & Finance*, *20*(3), 595–603. doi:10.1016/0378-4266(95)00014-3

Delbourgo, R., & Gregory, J. A. (1985). Shape preserving piecewise rational interpolation. *SIAM journal on scientific and statistical computing*, *6*(4), 967–976. doi:10.1137/0906065

Fritzsche, K., & Grauert, H. (2012). *From holomorphic functions to complex manifolds*. doi:10.1007/978-1-4684-9273-6

Gaß, M., Glau, K., Mahlstedt, M., & Mair, M. (2018). Chebyshev interpolation for parametric option pricing. *Finance and Stochastics*, *22*(3), 701–731. doi:10.1007/s00780-018-0361-y

Higham, N. J. (2004). The numerical stability of barycentric lagrange interpolation. *IMA Journal of Numerical Analysis*, *24*(4), 547–556. doi:10.1093/imanum/24.4.547

Jäckel, P. (2006). By implication. *Wilmott*, *26*, 60–66.

Jäckel, P. (2015). Let's be rational. *Wilmott*, *2015*(75), 40–53. doi:10.1002/wilm.10395

Li, M. (2008). Approximate inversion of the black–scholes formula using rational functions. *European Journal of Operational Research*, *185*(2), 743–759. doi:10.1016/j.ejor.2006.12.028

Lorig, M., Pagliarani, S., & Pascucci, A. (2014). A taylor series approach to pricing and implied volatility for local-stochastic volatility models. *The Journal of Risk*, *17*(2), 3. doi:10.21314/JOR.2014.297

Madan, D. B. (2016). Adapted hedging. *Annals of Finance*, *12*(3-4), 305–334. doi:10.1007/s10436-016-0282-8

Madan, D. B., & Wang, K. (2016). Laplacian risk management. *Finance Research Letters*. doi:10.1016/j.frl.2016.12.013

Manaster, S., & Koehler, G. (1982). The calculation of implied variances from the black-scholes model: A note. *The Journal of Finance*, *37*(1), 227–230. doi:10.2307/2327127

Merton, R. C. (1973). Theory of rational option pricing. *The Bell Journal of economics and management science*, 141–183. doi:10.2307/3003143

Pistorius, M., & Stolte, J. (2012). Fast computation of vanilla prices in time-changed models and implied volatilities using rational approximations. *International Journal of Theoretical and Applied Finance*, *15*(04), 1250031. doi:10.1142/S0219024912500318

Salazar Celis, O. (2017). A parametrized barycentric approximation for inverse problems with application to the Black–Scholes formula. *IMA Journal of Numerical Analysis*. doi:10.1093/imanum/drx020

Sauter, S., & Schwab, C. (2010). *Boundary element methods, translated and expanded from the 2004 german original*. doi:10.1007/978-3-540-68093-2

Townsend, A., & Trefethen, L. N. (2013). An extension of chebfun to two dimensions. *SIAM Journal on Scientific Computing*, *35*(6), C495–C518. doi:10.1137/130908002

Trefethen, L. N. (2013). *Approximation Theory and Approximation Practice*. SIAM books.