A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Computer Science, University College London

Numerical Optimisation Problems in Finance

Yiran Cui

Department of Computer Science, University College London

supervised by

Dr Guido Germano

Department of Computer Science, University College London; Systemic Risk Centre, London School of Economics & Political Science

Dr Sebastian del Baño Rollin

School of Mathematical Science, Queen Mary University of London

11 November 2016

To my mother and father.

献给我的爸爸妈妈。

Acknowledgements

I am not talented but fortunate.

I am grateful to my supervisor Dr Guido Germano for his careful guidance, consistent support and understanding. We spent a typical academic time together: chose every word with care, revised countless times, constantly improved and perfected. He has always treated me in a patient manner and helped with the bureaucracy. As he pointed out, I was his first Chinese PhD student.

Many thanks to Dr Sebastian del Baño Rollin, both a mentor and a friend. Thank him for his generous sharing of the interesting ideas, inspiring talks and valuable advice based on his experience and horizon.

Thank Prof. Tomaso Aste for having faith in me and offering me the opportunity to begin my PhD study in the Financial Computing and Analytics group at UCL.

My great gratitude goes to Ken Hayami Sensei, for all the help he has kindly offered me: took in me to study with his distinguished research group, patiently taught me how to conduct academic research, provided me many opportunities to practice my presentation skill, supported me to "look out into the world". He trained me from someone who knows superficially about linear algebra and numerical computing, to someone who can understand, explain, discuss and implement.

During my study in Japan, Hayami sensei also introduced me to Takashi Tsuchiya Sensei and Dr Keiichi Morikuni. From their figures, I saw the rigorous and diligent attitude of pure scholars. Life has a limit while knowledge does not. Every step accumulates to thousands of miles, every stream flows to an ocean.

My mother deserves the most thanks. She has been leading me to confront difficulties bravely and firmly, treat people gently and honestly, be tolerant with baselines, and be a lady at any move. She has infinite faith to her foresight that I would bring her happiness. Thank my father for telling me in my very young and ignorant days that study is a lonely thing, and aloneness is the best way to concentrate. He planted me with the genes of self-knowledge and studiousness. If he was to see me today, he would be no doubt very proud. I would like to dedicate this thesis to them, and hence attach the Chinese version of acknowledgements.

Lots of thanks to my family and friends, for their spiritual encouragement and financial support; to every teacher of mine, the education spanning on different subjects makes my life never monotonous; to my fellow students who come from various countries for enriching me with different perspectives.

Seeking knowledge abroad, I appreciate every kindness. No matter how effortless

or slight, it must have brought me some warm. I also appreciate every harm out of malice, it taught me the dark side of humanity, and hence helped me shed off childishness and grow up.

I have left my native land for more than four years, travelling widely and reading constantly. I become stronger while the world becomes more complicated. Sometimes I had to mediate between budget and pride, learn to allocate the time and energy that come from loneliness, keep direction and assume responsibility. Very different from what the childhood me imagined it to be, the PhD me does not become highly knowledgeable, and even have countless blind spots. Fortunately, my curiosity towards knowledge and love for life have not been worn down a bit, but been growing with each passing day. Leaving school does not halt study, but lights up the front for me to discover a further part of the world.

Yiran

London, November 2016

致 谢

我并不天资聪颖，但却十分幸运。

感谢我的导师Guido的悉心教导，以及对我想做之事的一贯理解支持。我们共度了一段典型的学术时光：字斟句酌，千锤百炼，精益求精。

感谢Sebastian，我的良师益友。谢谢他慷慨地分享给我那些有趣的点子，发人思考的讨论，基于经验的宝贵建议。

感谢Tomaso充分信任我，给了我在伦敦大学学院金融计算分析研究组攻读博士的机会。

感激速水谦先生为我提供的一切帮助：收我入其名门学习，耐心地教导我怎样进行学术研究，提供给我锻炼演讲能力的机会和去“开眼看世界”的资助。他帮助我从一个对线性代数和数值计算只局限于浮皮潦草、浅尝辄止的人，变成一个可以懂得、解释、讨论和实施的人。我在日本求学期间还结识了土谷隆先生和保国惠一博士，在他们身上，我看到纯粹研究者的严谨勤奋的致知态度。生者有涯，知也无涯。每一个成果，都是积成江水的小流，至成千里的跬步。

我的妈妈值得最多的感谢。她以身作则地教我对事坚定勇敢，待人温柔真诚，宽容且保有原则，时刻保持优雅。她像未卜先知一样笃定地相信着我能带给她幸福。感谢我的爸爸，在我年幼懵懂之时就告诉我学习是孤独的，孤独是最好的学习环境。他给我种下了平凡却自知要刻苦的基因。如果他看到了今天的我，一定会引以为豪。我希望将这篇文章的一部分专门献给他们，因此附上这篇致谢的中文版。

感谢来自家人朋友的精神鼓励和物质支持。感谢我的每一位老师，涵盖多个学科的教育使我的生活从不单调。感谢来自世界各个角落的同学们丰富了我的视角。

独自在异国他乡求学，我感激每一个善意的表达，哪怕举手之劳，哪怕千里鹅毛，都给了我莫大的温暖。也感激每一个恶意的算计，使我了解到人性的暗面，从而褪去稚气，真正长大。

离开祖国已经四年有余，读万卷书，行万里路。虽然自己变强大了，世界也变得更加复杂了。有些时候不得不在拮据与自尊心之间斡旋，学着分配那些来源于孤独的时间和精力，保持方向，承担责任。和少时想象的大不相同，到达博士阶段的我并没有变得渊博富识，甚至还有太多的盲点。所幸我对知识的好奇心、对生活的热情不但没有丝毫减少，而且与日俱增。离开学校并不意味着停止学习，而是照亮了前路，让我去发现更远一点的世界。

崔怡然

2016年11月于伦敦

Declaration

I, Yiran Cui, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Signature _____

Date _____

Abstract

This thesis consists of four projects regarding numerical optimisation and financial derivative pricing.

The first project deals with the calibration of the Heston stochastic volatility model. A method using the Levenberg-Marquardt algorithm with the analytical gradient is developed. It is so far the fastest Heston model calibrator and meets the speed requirement of practical trading.

In the second project, a triply-nested iterative method for the implementation of interior-point methods for linear programs is proposed. It is the first time that an interior-point method entirely based on iterative solvers succeeds in solving a fairly large number of linear programming instances from benchmark libraries under the standard stopping criteria.

The third project extends the Black-Scholes valuation to a complex volatility parameter and presents its singularities at zero and infinity. Fractals that describe the chaotic nature of the Newton-Raphson calculation of the implied volatility are shown for different moneyness values. Among other things, these fractals visualise dramatically the effect of an existing modification for improving the stability and convergence of the search. The project studies scientifically an interesting problem widespread in the financial industry, while revealing artistic values stemming from mathematics.

The fourth project investigates the consistency of a class of stochastic volatility models under spot rate inversion, and hence their suitability in the foreign exchange market. The general formula of the model parameters for the inversion rate is given, which provides basis for further investigation. The result is further extended to the affine stochastic volatility model. The Heston model, among the other members in the stochastic volatility family, is the only one that we found to be consistent under the spot inversion. The conclusion on the Heston model verifies the arbitrage opportunity in the variance swap.

Impact Statement

By the time of completing this thesis, the project on the calibration of the Heston model has received a lot of interests from the financial industry; the project on the implementation of interior-point method has been presented at many workshops and conferences, and received interests from numerical optimisation practitioners.

Contents

List of Tables

List of Figures

Chapter 1

General Introduction

This thesis deals with four projects regarding numerical optimisation and financial derivative pricing. In this chapter, we briefly describe each project, give directions to available preprints and programmes, and outline the thesis.

1.1 Content in brief

Project 1. The first topic is the calibration of the Heston stochastic volatility model. Although the Heston model has been widely used in the financial industry, there is no universal way of calibrating it: the current approaches are either computationally expensive or ad hoc with many assumptions on the parameters. In this work, we express the calibration as a nonlinear least squares problem. We originally derive the explicit form of the gradient which is crucial in the computation of the descent direction. We exploit a suitable representation of the Heston characteristic function and modify it to avoid discontinuities caused by branch switching of complex functions. Using this representation, we obtain the analytical gradient of the price of a vanilla option with respect to the model parameters, which is the key element of all variants of the objective function. The interdependency between the components of the gradient enables an efficient implementation. The explicit gradient avoids the difficulty in deciding the finite difference that is always encountered when using the numerical gradient and is around ten times faster than a numerical gradient. We choose the Levenberg-Marquardt method to calibrate the model and do not observe multiple local minima reported in previous research. Two-dimensional sections show that the objective function is shaped as a narrow valley with a flat bottom. Our method is the fastest calibration of the Heston model developed so far and meets the speed requirement of practical trading.

Project 2. We develop an implementation of interior-point method for linear programming (LP) problems that has a wider application and is not restricted to computational finance. Many financial problems such as static hedging and portfolio optimisation can be formulated as large scale LP problems and thus require fast and

accurate solution. However, the commonly-used direct methods such as Cholesky decomposition tend to break down for large sparse problems even with modifications.

We apply novel inner-iteration preconditioned Krylov subspace methods to the interior-point algorithm for LP. Inner-iteration preconditioners recently proposed by Morikuni and Hayami enable us to overcome the severe ill-conditioning of linear equations solved in the final phase of interior-point iterations. The employed Krylov subspace methods do not suffer from rank-deficiency and therefore no preprocessing is necessary even if rows of the constraint matrix are not linearly independent. Extensive numerical experiments are conducted over diverse instances of 125 LP problems including NETLIB, QAPLIB, and MITTELMANN collections. The largest problem has 434,580 variables. It turns out that our implementation is more stable and robust than the standard public domain solvers SeDuMi (Self-Dual Minimization) and SDPT3 (Semidefinite Programming Toh-Todd-Tütüncü) without increasing CPU time. As far as we know, this is the first time that an interior-point method entirely based on iterative solvers succeeds in solving a fairly large number of LP instances from benchmark libraries under standard stopping criteria.

Project 3. Usually, in the Black-Scholes-Merton pricing theory the volatility is a positive real parameter. In this project we explore what happens if it is allowed to be a complex number. The function for pricing a European option with a complex volatility has essential singularities at zero and infinity. The singularity at zero reflects the put-call parity. Solving for the implied volatility that reproduces a given market price yields not only a real root, but also infinitely many complex roots in a neighbourhood of the origin. The Newton-Raphson calculation of the complex implied volatility has a chaotic nature described by fractals.

Project 4. We investigate the consistency of a class of stochastic volatility models under spot inversions, and hence their applicability in the foreign exchange market. We give the general result for the parameters of the model for the inverse rate. The Heston model, among the other members in the stochastic volatility family, is the only one that we found to be consistent under the spot inversion. The conclusion on the Heston model verifies the arbitrage opportunity in the variance swap. The result is further extended to the affine stochastic volatility model.

1.2 Preprints and programmes

We provide the following information on the preprints and programmes for the projects discussed in the thesis.

The manuscript of Project 1 was accepted with minor revisions by *European Journal of Operational Research*. A preprint can be found on arXiv [26].

Talks on this work was given at

- *12th Joint Meeting of Special Interest Groups, Session on Mathematical Finance, JSIAM (The Japan Society for Industrial and Applied Mathematics), University of Kobe, 4th March 2016,*

- *QFW2017, the XVIII Workshop on Quantitative Finance, University of Milano-Bicocca, Milano, 25th January 2017.*

Since we received many requests on the code for the calibrator from both industry and academia, we distributed under the terms of the GNU General Public License our programme, coded in `C++`, which contains:

- a pricer for vanilla options under the Heston model,

- a calibrator of the Heston model,

- an example of using the calibrator to find the presumed optimal parameter.

The manuscript of Project 2 was submitted to *SIAM Journal on Scientific Computing*. A preprint can be found on arXiv [27].

Talks on this work were given at

- *JSIAM Joint Meeting of Special Interest Groups, Session on Solution of Matrix and Eigenvalue Problems and Applications, University of Tokyo, 25 December 2014 and University of Kobe, 4th March 2016,*

- *22nd ISMP (International Symposium on Mathematical Programming), Session on Advances and Applications in Conic Optimisation, Pittsburgh, 17th June 2015,*

- *GRIPS (National Graduate Institute for Policy Studies) Research Meeting: "Optimisation: Modelling and Algorithms", GRIPS, Tokyo, 22nd March 2016,*

- *Workshop on Advances in Optimisation, TKP Shinagawa Conference Centre, 12th August 2016,*

- *5th IMA (Institute of Mathematics and its Applications) Conference on Numerical Linear Algebra and Optimisation, University of Birmingham, 7th September 2016.*

We plan to publicise the rest of our code when the paper is published. The implementations of AB- and BA-GMRES with inner iterations, coded in Fortran 90/95 and in C for MATLAB-MEX can be found on the co-author Dr Morikuni's homepage `http://researchmap.jp/KeiichiMorikuni/Implementations/`.

The manuscript of Projects 3 is going to be submitted in 2017. The journal is yet to be decided. We distributed under the terms of the GNU General Public License our programme, coded in MATLAB, which contains:

- a Newton-Raphson calibrator of the Black-Scholes-Merton model,

- a script for plotting the Newton-Raphson fractal of complex Black-Scholes-Merton implied volatility.

The manuscript of Project 4 is going to be submitted in 2017. The journal is yet to be decided.

1.3 Outline

To aid the reader, we finish the introduction with an outline of the thesis.

Chapter 1. An introduction to the thesis and a list of URLs for available preprints and programmes.

Chapter 2. We introduce the stochastic volatility model and review the previous calibration methods. We formulate the calibration problem as a nonlinear least squares problem and derive the explicit form of the gradients. We present the algorithm based on Levenberg-Marquardt method and the numerical results of the proposed calibrator.

Chapter 3. We introduce the interior-point method for LP and propose the methods for computing the interior-point steps. Extensive numerical experimental results are given in the end.

Chapter 4. We present an extension of the Black-Scholes-Merton implied volatility to the complex plane by the means of basin of attraction and fractals.

Chapter 5. We give the definition of model consistency under spot inversion which is of importance in the foreign exchange market. We examine the consistency of several stochastic volatility models and the results are presented as theorem and corollaries. The result is extended to the affine-jump diffusion model.

Chapter 6. We conclude the paper and summarize the contributions. A few remarks and future work are stated.

Chapter 2

Full and Fast Calibration of the Heston Stochastic Volatility Model

2.1 Stochastic volatility models

A sophisticated model may reflect the reality better than a simple one, but usually is more challenging to implement and calibrate. This is especially true with mathematical models for the pricing of derivatives and the estimation of risk. The most basic model, introduced by Black and Scholes [12] and Merton [84] (BSM), assumes that the underlying price follows a geometric Brownian motion with constant drift and volatility. The price of a vanilla option is then given as a function of a single parameter, the volatility. However, the BSM model does not adequately take into account essential characteristics of market dynamics such as fat tails, skewness and the correlation between the value of the underlying and its volatility. It has also been observed that the volatility starts to fluctuate when the market reacts to new information [65]. Thus, several extensions of the BSM model were suggested thereafter, including the family of stochastic volatility (SV) models, which introduces a second Brownian motion to describe the fluctuation of the volatility. We study one of the most important SV models; it was proposed by Heston [63] and is defined by the system of stochastic differential equations

$$\mathrm{d}S_t = \mu S_t \mathrm{d}t + \sqrt{v_t} S_t \mathrm{d}W_t^{(1)}, \tag{2.1a}$$

$$\mathrm{d}v_t = \kappa(\bar{v} - v_t)\mathrm{d}t + \sigma\sqrt{v_t}\mathrm{d}W_t^{(2)}, \tag{2.1b}$$

with

$$\mathrm{d}W_t^{(1)}\mathrm{d}W_t^{(2)} = \rho \mathrm{d}t, \tag{2.1c}$$

where S_t is the underlying price and v_t its variance; the parameters $\kappa, \bar{v}, \sigma, \rho$ are respectively called the mean-reversion rate, the long-term variance, the volatility of volatility, and the correlation between the Brownian motions $W_t^{(1)}$ and $W_t^{(2)}$ that

drive the underlying and its variance; moreover there is a fifth parameter v_0, the initial value of the variance.

Model calibration is as crucial as the model itself. Calibration consists in determining the parameter values so that the model reproduces market prices as accurately as possible. Both the accuracy and the speed of calibration are important because practitioners use the calibrated parameters to price a large number of complicated derivative contracts and to develop high-frequency trading strategies.

Throughout this chapter, we use bold uppercase letters for matrices, e.g. $\boldsymbol{J}$, and bold lowercase letters for column vectors, e.g. $\boldsymbol{\theta}$; a superscript $^\intercal$ for the transpose of a matrix or vector; $\boldsymbol{e}$ for a vector of all ones $[1, \ldots, 1]^\intercal$; $\mathrm{E}[\cdot]$ for expectations; $\mathbf{1}_A(\cdot)$ for the indicator function of the set A; $\mathrm{Re}(\cdot)$ for the real part and $\mathrm{Im}(\cdot)$ for the imaginary part of a complex number; $\|\cdot\|$ for the l_2-norm; $\|\cdot\|_\infty$ for the l_∞-norm; log for the natural logarithm.

2.2 Previous research on Heston model calibration

In the literature, there are two main approaches to calibrate the Heston model: *historical* and *implied*. The first fits historical time series of the prices of an option with a fixed strike and maturity, typically by the maximum likelihood method or the efficient method of moments [3, 40, 66]. The second fits the volatility surface of an underlying at a fixed time, i.e., options with several strikes and maturities, to obtain the implied parameter set. Our work follows the second approach, as that is what is used in real-time pricing and risk management. In the following, we survey obstacles and existing methods for the Heston model calibration related to the second approach.

2.2.1 Recognised difficulties

Firstly, the calibration is in a five-dimensional space. There is no consensus among researchers on whether the objective function for the Heston model calibration is convex or irregular. The results of some proposed methods [16, 51, 85] depend on the initial point, which was attributed to a non-convex objective function, but might simply reflect on the inadequacy of the methods. To find a reasonable initial guess, short-term or long-term asymptotic rules are used; see Jacquier and Martini [67] for a detailed review. However, recently Gerlich et al. [49] claimed a convergence to the unique solution independent of the initial guess and suggested that the Heston calibration problem may have some inherent structure leading to a single stationary point. On the other hand, dependencies among the parameters do exist. For example, it is known that σ and κ offset each other: the long-term distribution of the volatility depends only on the ratio σ^2/κ, so that a parameter set with large values of σ and κ gives a fit comparable to a set with small values of σ and κ. Intuitively, the fact that different parameter combinations yield similar values of the objective function could be due to the objective function being flat close to the optimum; see

Section 2.5 in this paper.

Secondly, the analytical gradient for the Heston calibration problem is hard to find and has not been available so far because it was believed that the expression of the Heston characteristic function is overly complicated to provide an insightful analytical gradient: of course, a gradient can be obtained with symbolic algebra packages, but the resulting expressions are intractable. Instead, numerical gradients obtained by finite difference methods have been used in gradient-based optimisation methods; however, numerical gradients have a larger computational cost and a lower accuracy.

2.2.2 Existing methods

We review some heuristics to reduce the dimension of the calibration and then the optimisation methods that have been applied so far.

2.2.2.1 Heuristics for dimension reduction

Since the Heston parameters are closely related to the shape of the implied volatility surface [21, 48, 51, 69] (v_0 controls the position of the volatility smile, ρ the skewness, κ and σ the convexity, and κ times the difference between v_0 and $\bar{v}$ the term structure of implied volatility), efforts have been made to simplify the calibration to a lower dimension by presuming some of the parameter values based on knowledge available for the specific volatility surface. The initial variance v_0 is usually set to the short-term at-the-money (ATM) BSM implied variance, which is based on the term structure of the BSM implied volatility in the Heston model [48, p. 34-35]. A practical calibration experiment [16, p. 29-30] verified the linearity between the initial variance and the BSM implied variance for maturities in the range of 1 to 2 months. Clark [21, Eq. (7.3)] suggested the heuristic assumption $\kappa = 2.75/\tau$ and $\bar{v} = \sigma_{\text{ATM}}(\tau)$, where $\sigma_{\text{ATM}}(\tau)$ is the ATM BS implied volatility with time to maturity τ. Chen [16] proposed a fast intraday recalibration

by fixing κ to the same as yesterday's and v_0 to the 2-month ATM implied volatility, which are heuristics actually adopted in the industry. These assumptions help with an incomplete calibration, but may misguide the iterate to a limited domain and thus to a wrong convergence point.

2.2.2.2 Stochastic optimisation methods

Researchers who believed that a descent direction is unavailable have devoted their attention to stochastic optimisation methods, including Wang-Landau [16], differential evolution and particle swarm [52], simulated annealing [90], etc. To increase the robustness, a deterministic search such as Nelder and Mead using the MATLAB function `fminsearch` is often combined with these stochastic optimisation algorithms. Almost all research using stochastic techniques reports issues with performance. GPU technology has been applied with simulated annealing to speed up

the calibration of the SABR model, a member of the family of SV models. Using 2 nVIDIA Geforce GTX470 GPUs it took 421.72 seconds to calibrate 12 instruments achieving a 10^{-2} maximum error [41], which is still too slow for real-time use.

2.2.2.3 Deterministic optimisation methods

Deterministic optimisation solvers available with commercial packages have been proved to be unstable as the performance largely depends on the quality of the initial guess: this applies to the Excel built-in solver [85] and to the MATLAB solver `lsqnonlin` [9, 47, 90]. Gerlich et al. [49] adopted a Gauss-Newton framework and kept the feasibility of the iterates by projecting to a cone determined by the constraints. The gradient of the objective function was calculated by finite differences and thus costs a large number of function evaluations.

To sum up, existing calibration algorithms are either based on ad hoc assumptions or not fast or stable enough for practical use. In this work, we will focus on deterministic optimisation methods without any presumption on the values of the parameters.

2.3 Problem formulation and gradient calculation

The idea of calibrating a volatility model is to minimise the difference between the vanilla option price calculated with the model and the one observed in the market. In this section, we first formulate the calibration problem in a least squares form. Then, we present the pricing formula of a vanilla option under the Heston model with four algebraically equivalent representations of the characteristic function, discussing their numerical stability and suitability for analytical derivation. We calculate the analytical gradient of the objective function which can be used in any gradient-based optimisation algorithm.

2.3.1 The inverse problem formulation

Denote by $C^*(K_i, \tau_i)$ the market price of a vanilla call option with strike K_i and time to maturity $\tau_i := T_i - t$, $C(\boldsymbol{\theta}; K_i, \tau_i)$ the price computed via the Heston analytical formula (2.9) with the parameter vector $\boldsymbol{\theta} := [v_0, \bar{v}, \rho, \kappa, \sigma]^\intercal$. We assemble the residuals for the n options to be calibrated

$$r_i(\boldsymbol{\theta}) := C(\boldsymbol{\theta}; K_i, \tau_i) - C^*(K_i, \tau_i), \qquad i = 1, \ldots, n \tag{2.2}$$

in the residual vector $\boldsymbol{r}(\boldsymbol{\theta}) \in \mathbb{R}^n$, i.e.,

$$\boldsymbol{r}(\boldsymbol{\theta}) := [r_1(\boldsymbol{\theta}), r_2(\boldsymbol{\theta}), \ldots, r_n(\boldsymbol{\theta})]^\intercal . \tag{2.3}$$

We treat the calibration of the Heston model as an inverse problem in the non-

linear least squares form

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^m} f(\boldsymbol{\theta}), \tag{2.4}$$

where $m = 5$ indicates the dimension, and

$$f(\boldsymbol{\theta}) := \frac{1}{2}\|\boldsymbol{r}(\boldsymbol{\theta})\|^2 = \frac{1}{2}\boldsymbol{r}^\top(\boldsymbol{\theta})\boldsymbol{r}(\boldsymbol{\theta}). \tag{2.5}$$

Since there are many more market observations than parameters to be found, i.e., $n \gg m$, the calibration problem is overdetermined.

Our objective function is the sum of squared differences of price on a set of strikes and maturities. The precise choice of objective function is a non-trivial matter and, in an accounting sense, ultimately depends on the trade population on an actual trading book. The choice of strikes and maturities is also non-trivial. For example, choosing the same strikes (even as a percentage of spot as in [19, Table 1]) will lead to these strikes being more OTM for shorter than for longer dated maturities which could lead to miscalibration of the longer dated smile and computational underflow problems in the calculation of shorter ended options. To remedy this we use the standard approach in foreign exchange (FX) which is to use strikes defined by given Deltas, namely ATM, and 25 and 10 Delta[1] call and put options (see [21, Section 3.3]). With regards to the actual objective function, other possibilities have been considered (see [20] for a review) which according to loc. cit. can lead to different solutions. One such function used in industry is the sum of squared differences of implied volatilities [22, Section 13.2]. As stated in [22] this quantity can be approximated by dividing the difference in price by the Vega[2] and it is relevant when the price bid/offer spread in volatility terms is independent of strike. Note that this is not always the case as less liquid OTM options will be quoted with a wider bid/offer spread in volatilities. Calibrating to implied volatility will cause OTM options, with lower Vega, to weigh more in the objective function and so the resulting calibration, compared to the one based on the price, will privilege OTM over ATM options. As our algorithm also approximates Vega it might also help exploring the optimization problem in terms of implied volatility as a subject of future work. Since the gradient will be derived from the pricing formula, for us it is most straightforward and consistent to minimise the pricing error in the current work.

Before applying any technique to solve the problem (2.4)–(2.5), one needs to bear in mind that the evaluation of $C(\boldsymbol{\theta}; K_i, \tau_i)$ is expensive for the purpose of calibration;

[1] The sensitivity of BSM European option price with respect to spot price.

[2] The sensitivity of European option price with respect to volatility. Note that there are several possible notions of Vega. For instance, Vega used in hedging positions, or "trader Vega", is the result of bumping actual market data prior to calibration and pricing. In FX, the market practice is to quote OTM options as *Risk Reversal* and *Strangle* volatilities, sensitivity to these quantities is often called Rega and Sega. For the purposes of calibration to implied volatilities, the relevant Vega is what is sometimes referred in industry as the "fenics Vega" defined by the BSM expression for Vega evaluated on the implied volatility. Another Vega that occurs is the sensitivity of prices to model parameters that reflect the overall level of the smile, such as v_0 in the Heston model, this is sometimes alluded to as "model Vega". The hope is that all these different measures are roughly comparable.

hence, one would like to minimise the number of computations of Eq. (2.9) when designing the algorithm. Moreover, the explicit gradient of $C(\boldsymbol{\theta}; K_i, \tau_i)$ with respect to $\boldsymbol{\theta}$ is not available in the literature as it is deemed to be overly complicated. This is indeed true if one starts from the commonly used expressions for the characteristic function by Heston [63, Eq. (17)] or Schoutens et al. [105, Eq. (17)]. However, as shown in the next section, a more convenient choice of the functional form of the characteristic function by del Baño Rollin et al. [35, Eq. (6)] eases the derivation of its analytical gradient.

2.3.2 Pricing formula of a vanilla option and representations of the characteristic function

For a spot price S_t, an interest rate r and a dividend rate q, the price of a vanilla call option with strike K and time to maturity $\tau := T - t$ is

$$C(\boldsymbol{\theta}; K, \tau) = e^{-r\tau}\mathrm{E}[(S_T - K)\mathbf{1}_{\{S_T \geq K\}}(S_T)] \tag{2.6a}$$

$$= e^{-r\tau}\left(\mathrm{E}[S_T\mathbf{1}_{\{S_T \geq K\}}(S_T)] - K\mathrm{E}[\mathbf{1}_{\{S_T \geq K\}}(S_T)]\right) \tag{2.6b}$$

$$= S_t e^{-q\tau} P_1(\boldsymbol{\theta}; K, \tau) - e^{-r\tau} K P_2(\boldsymbol{\theta}; K, \tau). \tag{2.6c}$$

In the Heston model, $P_1(\boldsymbol{\theta}; K, \tau)$ and $P_2(\boldsymbol{\theta}; K, \tau)$ are solutions to certain pricing PDEs [63, Eq. (12)] and are given as

$$P_1(\boldsymbol{\theta}; K, \tau) = \frac{1}{2} + \frac{1}{\pi}\int_0^\infty \mathrm{Re}\left(\frac{e^{-iu\log\frac{K}{S_0}}}{iu}\frac{\phi(\boldsymbol{\theta}; u - i, \tau)}{\phi(\boldsymbol{\theta}; -i, \tau)}\right)\mathrm{d}u, \tag{2.7}$$

$$P_2(\boldsymbol{\theta}; K, \tau) = \frac{1}{2} + \frac{1}{\pi}\int_0^\infty \mathrm{Re}\left(\frac{e^{-iu\log\frac{K}{S_0}}}{iu}\phi(\boldsymbol{\theta}; u, \tau)\right)\mathrm{d}u, \tag{2.8}$$

where i is the imaginary unit, $\phi(\boldsymbol{\theta}; -i, \tau) = S_t e^{(r-q)\tau} =: F$ is the forward price and $\phi(\boldsymbol{\theta}; u, \tau)$ is the characteristic function of the logarithm of the stock price process with u the counterpart of logarithm of price in the Fourier space. Thus, the formula for pricing a vanilla call option becomes

$$\begin{aligned} C(\boldsymbol{\theta}; K, \tau) = \frac{1}{2}S_t e^{-q\tau} - \frac{1}{2}e^{-r\tau}K \\ + \frac{e^{-r\tau}}{\pi}\left[\int_0^\infty \mathrm{Re}\left(\frac{e^{-iu\log\frac{K}{S_0}}}{iu}\phi(\boldsymbol{\theta}; u - i, \tau)\right)\mathrm{d}u\right. \\ \left. - K\int_0^\infty \mathrm{Re}\left(\frac{e^{-iu\log\frac{K}{S_0}}}{iu}\phi(\boldsymbol{\theta}; u, \tau)\right)\mathrm{d}u\right]. \end{aligned} \tag{2.9}$$

The characteristic function was originally given by Heston as [63, Eq. (17)]

$$\phi(\boldsymbol{\theta}; u, \tau) := \mathrm{E}\left[\exp\left(iu\log\frac{S_t}{S_0}\right)\right] = \exp\left\{iu\log\frac{F}{S_0}\right.$$

$$+\frac{\kappa\bar{v}}{\sigma^2}\left[(\xi+d)\tau-2\log\frac{1-g_1e^{d\tau}}{1-g_1}\right]+\frac{v_0}{\sigma^2}(\xi+d)\frac{1-e^{d\tau}}{1-g_1e^{d\tau}}\Bigg\}, \quad (2.10)$$

where

$$\xi := \kappa - \sigma\rho i u, \quad (2.11a)$$

$$d := \sqrt{\xi^2+\sigma^2(u^2+iu)}, \quad (2.11b)$$

$$g_1 := \frac{\xi+d}{\xi-d}. \quad (2.11c)$$

Kahl and Jäckel [72] pointed out that when evaluating this form as a function of u for moderate to long maturities, discontinuities appear because of the branch switching of the complex power function $G^\alpha(u) = \exp(\alpha \log G(u))$ with $G(u) := (1-g_1e^{d\tau})/(1-g_1)$ and $\alpha := \kappa\bar{v}/\sigma^2$, which appears in Eq. (2.10) as a multivalued complex logarithm. This depends on the fact that $G(u)$ has a shape of a spiral as u increases, and when it repeatedly crosses the negative real axis, the phase of $G(u)$ jumps from $-\pi$ to π. Then the phase of $G^\alpha(u)$ changes from $-\alpha\pi$ to $\alpha\pi$, causing a discontinuity when α is not a natural number.

Albrecher et al. [4] found that this happens when the principal value of the complex square root d is selected, as most numerical implementations of these functions do, but can be avoided if the second value is used instead. They proved that this alternative representation, originally proposed by Schoutens et al. [105, Eq. (17)], is continuous and gives numerically stable prices in the full-dimensional and unrestricted parameter space:

$$\phi(\boldsymbol{\theta};u,\tau) = \exp\Bigg\{iu\log\frac{F}{S_0} + \frac{\kappa\bar{v}}{\sigma^2}\left[(\xi-d)\tau-2\log\frac{1-g_2e^{-d\tau}}{1-g_2}\right]+\frac{v_0}{\sigma^2}(\xi-d)\frac{1-e^{-d\tau}}{1-g_2e^{-d\tau}}\Bigg\}, \quad (2.12)$$

where

$$g_2 := \frac{\xi-d}{\xi+d} = \frac{1}{g_1}. \quad (2.13)$$

Another equivalent form of the characteristic function was proposed later by del Baño Rollin et al. [35, Eq. (6)]. We correct the expression in that paper by adding the term $-\kappa\bar{v}\rho\tau iu/\sigma$ to the exponent, resulting in

$$\phi(\boldsymbol{\theta};u,\tau) = \exp\left(iu\log\frac{F}{S_0} - \frac{\kappa\bar{v}\rho\tau iu}{\sigma} - A\right)B^{2\kappa\bar{v}/\sigma^2}, \quad (2.14)$$

where

$$A := \frac{A_1}{A_2}, \quad (2.15a)$$

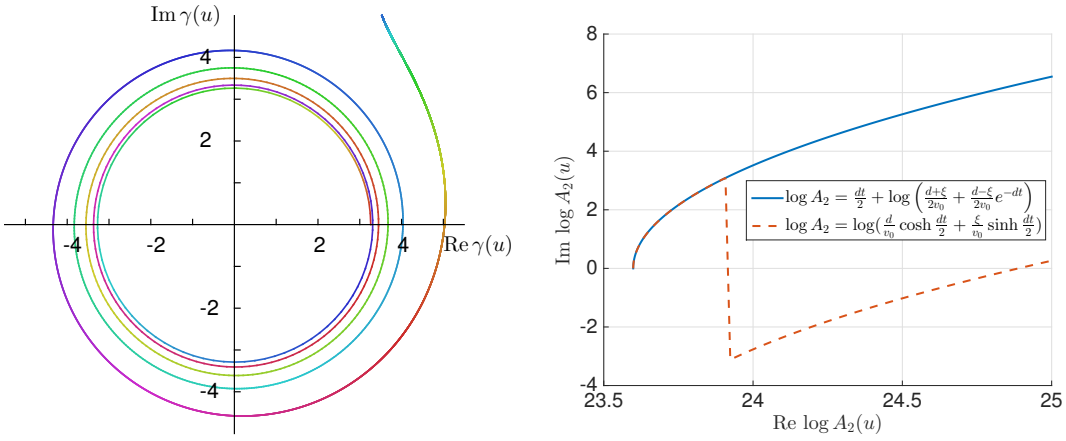
$$A_1 := (u^2+iu)\sinh\frac{d\tau}{2}, \quad (2.15b)$$

$$A_2 := \frac{d}{v_0}\cosh\frac{d\tau}{2} + \frac{\xi}{v_0}\sinh\frac{d\tau}{2}, \quad (2.15c)$$

$$B := \frac{de^{\kappa\tau/2}}{v_0 A_2}. \tag{2.15d}$$

Del Baño Rollin et al. introduced their expression to analyse the log-spot density, and since then it has not been used for any other purpose. It was obtained by manipulating the complex moment generating function; besides being more compact, it replaces the exponential functions in the exponent with hyperbolic functions, which makes the derivatives easier. Therefore, we will use this expression to obtain the analytical gradient.

However, the same discontinuity problem pointed out by Kahl and Jäckel appears here too. It comes from the factor $B^{2\kappa\bar{v}/\sigma^2}$, or more specifically from the denominator of B, i.e., A_2. Fig. 2.1a shows a trajectory of $\gamma(u) := (A_2(u) \log\log|A_2(u)|)/|A_2(u)|$. The double-logarithmic scaling of the radius compensates the rapid outward movement of the spiralling trajectory of $A_2(u)$ [4, 72]. For the curve we adopt the same hue $h \in [0, 1)$ as Kahl and Jäckel [72], $h := \log_{10}(u + 1) \mod 1$, which means that segments of slowly varying colour represent rapid movements of $A_2(u)$ as a function of u.



(a) $\gamma(u)$, $u \in [0, 500]$. (b) $\log A_2(u)$, $u \in [0, 4]$.

FIG. 2.1: Trajectories of $\gamma(u)$ and two equivalent forms of $\log A_2(u)$ in the complex plane. The curves were generated using the parameters in Table 2.1 with time to maturity $\tau = 15$.

TABLE 2.1: Parameters specification.

Model parameters		Market parameters	
κ	3.00	S_0	1.00
$\bar{v}$	0.10	K	1.10
σ	0.25	r	0.02
ρ	−0.80	q	0
v_0	0.08		

We thus modify the representation by rearranging $\log A_2$ to

$$\log A_2 = \log\left(\frac{d}{v_0}\cosh\frac{d\tau}{2} + \frac{\xi}{v_0}\sinh\frac{d\tau}{2}\right) \tag{2.16a}$$

$$= \log\frac{d(e^{d\tau/2}+e^{-d\tau/2}) + \xi(e^{d\tau/2}-e^{-d\tau/2})}{2v_0} \tag{2.16b}$$

$$= \log\frac{(d+\xi)e^{d\tau/2} + (d-\xi)e^{-d\tau/2}}{2v_0} \tag{2.16c}$$

$$= \log\left[e^{d\tau/2}\left(\frac{d+\xi}{2v_0} + \frac{d-\xi}{2v_0}e^{-d\tau}\right)\right] \tag{2.16d}$$

$$= \frac{d\tau}{2} + \log\left(\frac{d+\xi}{2v_0} + \frac{d-\xi}{2v_0}e^{-d\tau}\right). \tag{2.16e}$$

Fig. 2.1b shows the trajectories of the two equivalent formulations of $\log A_2$. The rearrangement (2.16e) resolves the discontinuities arising from the logarithm with Eq. (2.15c) as an argument. Then we insert Eq. (2.16e) into $\log B$ and denote the final expression as D:

$$\log B = \log\frac{d}{v_0} + \frac{\kappa\tau}{2} - \log A_2 \tag{2.17a}$$

$$= \log\frac{d}{v_0} + \frac{(\kappa-d)\tau}{2} - \log\left(\frac{d+\xi}{2v_0} + \frac{d-\xi}{2v_0}e^{-d\tau}\right) =: D. \tag{2.17b}$$

So we propose a new representation of the characteristic function which is algebraically equivalent to all the previous expressions and does not show the discontinuities of Eqs. (2.10) and (2.14) for large maturities:

$$\phi(\boldsymbol{\theta}; u, \tau) = \exp\left(iu\log\frac{F}{S_0} - \frac{\kappa\bar{v}\rho\tau iu}{\sigma} - A + \frac{2\kappa\bar{v}}{\sigma^2}D\right). \tag{2.18}$$

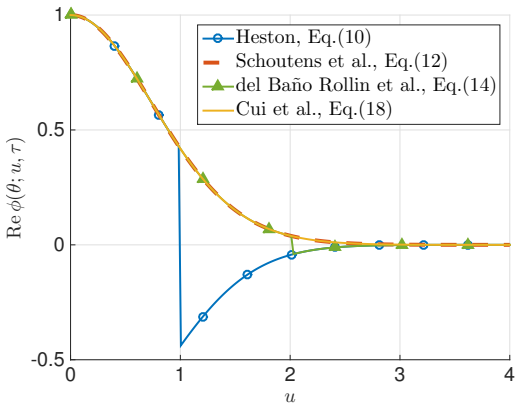


FIG. 2.2: Four equivalent representations of the Heston characteristic function. The curves were generated using the parameters in Table 2.1 with maturity $T = 15$. Eq. (2.10) jumps at $u = 1$, Eq. (2.14) jumps at $u = 2$, while Eqs. (2.12) and (2.18) are continuous.

We have discussed four equivalent representations of the Heston characteristic function, three from previous research and one newly proposed here by us. We compare them in Fig. 2.2: the plot of our expression is continuous and overlaps

TABLE 2.2: Properties of the four representations of the Heston characteristic function.

	Numerically continuous	Easily derivable
Heston	✗	✗
Schoutens et al.	✓	✗
del Baño Rollin et al.	✗	✓
Cui et al.	✓	✓

Schoutens et al.'s, while the other two exhibit discontinuities due to the multivalued complex functions. Moreover our expression, like the one by del Baño Rollin et al. from which it was obtained, has the advantage of being easily derivable, as shown in the next section. These properties are summarised in Table 2.2.

2.3.3 Analytical gradient

We use $\boldsymbol{\nabla} = \partial/\partial\boldsymbol{\theta}$ for the gradient operator with respect to the parameter vector $\boldsymbol{\theta}$ and $\boldsymbol{\nabla}\boldsymbol{\nabla}^\intercal$ for the Hessian operator. For convenience, we omit to write the dependence of the residual vector $\boldsymbol{r}$ on $\boldsymbol{\theta}$.

2.3.3.1 The basic theorem of the analytical gradient

Let $\boldsymbol{J} = \boldsymbol{\nabla}\boldsymbol{r}^\intercal \in \mathbb{R}^{m\times n}$ be the Jacobian matrix of the residual vector $\boldsymbol{r}$ with elements

$$J_{ji} = \left[\frac{\partial r_i}{\partial \theta_j}\right] = \left[\frac{\partial C(\boldsymbol{\theta}; K_i, \tau_i)}{\partial \theta_j}\right], \tag{2.19}$$

and $\boldsymbol{H}(r_i) := \boldsymbol{\nabla}\boldsymbol{\nabla}^\intercal r_i \in \mathbb{R}^{m\times m}$ be the Hessian matrix of each residual r_i with elements

$$H_{jk}(r_i) = \left[\frac{\partial^2 r_i}{\partial\theta_j \partial\theta_k}\right]. \tag{2.20}$$

Following the nonlinear least squares formulation (2.4)–(2.5), one can easily write the gradient and Hessian of the objective function f as

$$\boldsymbol{\nabla} f = \boldsymbol{J}\boldsymbol{r}, \tag{2.21a}$$

$$\boldsymbol{\nabla}\boldsymbol{\nabla}^\intercal f = \boldsymbol{J}\boldsymbol{J}^\intercal + \sum_{i=1}^{n} r_i \boldsymbol{H}(r_i). \tag{2.21b}$$

Theorem 2.1. *Assume that an underlying asset S follows the Heston process (2.1). Let $\boldsymbol{\theta} := [v_0, \bar{v}, \rho, \kappa, \sigma]^\intercal$ be the parameters in the Heston model, $C(\boldsymbol{\theta}; K, \tau)$ be the price of a vanilla call option on S with strike K and time to maturity τ. Then the gradient of $C(\boldsymbol{\theta}; K, \tau)$ with respect to $\boldsymbol{\theta}$ is*

$$\boldsymbol{\nabla} C(\boldsymbol{\theta}; K, \tau) = \frac{e^{-r\tau}}{\pi}\left[\int_0^\infty \mathrm{Re}\left(\frac{e^{-iu\log\frac{K}{S_0}}}{iu}\boldsymbol{\nabla}\phi(\boldsymbol{\theta}; u-i, \tau)\right)\mathrm{d}u\right.$$

$$- K \int_0^\infty \mathrm{Re}\left(\frac{e^{-iu \log \frac{K}{S_0}}}{iu} \nabla \phi(\boldsymbol{\theta}; u, \tau) \right) \mathrm{d}u \Bigg], \quad (2.22)$$

where $\nabla \phi(\boldsymbol{\theta}; u, \tau) = \phi(\boldsymbol{\theta}; u, \tau) \boldsymbol{h}(u)$, $\boldsymbol{h}(u) := [h_1(u), h_2(u), \ldots, h_5(u)]^\intercal$ *with elements*

$$h_1(u) = -\frac{A}{v_0}, \quad (2.23a)$$

$$h_2(u) = \frac{2\kappa}{\sigma^2} D - \frac{\kappa \rho \tau i u}{\sigma}, \quad (2.23b)$$

$$h_3(u) = -\frac{\partial A}{\partial \rho} + \frac{2\kappa \bar{v}}{\sigma^2 d} \left(\frac{\partial d}{\partial \rho} - \frac{d}{A_2} \frac{\partial A_2}{\partial \rho} \right) - \frac{\kappa \bar{v} \tau i u}{\sigma}, \quad (2.23c)$$

$$h_4(u) = \frac{1}{\sigma i u} \frac{\partial A}{\partial \rho} + \frac{2\bar{v}}{\sigma^2} D + \frac{2\kappa \bar{v}}{\sigma^2 B} \frac{\partial B}{\partial \kappa} - \frac{\bar{v} \rho \tau i u}{\sigma}, \quad (2.23d)$$

$$h_5(u) = -\frac{\partial A}{\partial \sigma} - \frac{4\kappa \bar{v}}{\sigma^3} D + \frac{2\kappa \bar{v}}{\sigma^2 d} \left(\frac{\partial d}{\partial \sigma} - \frac{d}{A_2} \frac{\partial A_2}{\partial \sigma} \right) + \frac{\kappa \bar{v} \rho \tau i u}{\sigma^2}; \quad (2.23e)$$

$\xi, d, A, A_1, A_2, B, D, \phi(\boldsymbol{\theta}; u, \tau)$ *are defined in Eqs.* (2.11a), (2.11b), (2.15), (2.17b) *and* (2.18), *respectively.*

Proof. Eq. (2.22) is a direct result from the vanilla option pricing function (2.9). Then the problem reduces to the derivation of the gradient of the characteristic function $\phi(\boldsymbol{\theta}; u, \tau)$. Starting from Eq. (2.14) and following the chain rule, one can get $\nabla \phi(\boldsymbol{\theta}; u, \tau)$ as discussed below.

Since v_0 and $\bar{v}$ are only in the exponent and are not involved with the definition of A or B, we directly obtain

$$\frac{\partial \phi(\boldsymbol{\theta}; u, \tau)}{\partial v_0} = -\frac{A}{v_0} \phi(\boldsymbol{\theta}; u, \tau), \quad (2.24)$$

$$\frac{\partial \phi(\boldsymbol{\theta}; u, \tau)}{\partial \bar{v}} = \frac{2\kappa \log B \phi(\boldsymbol{\theta}; u, \tau)}{\sigma^2}. \quad (2.25)$$

Next we derive the partial derivative with respect to ρ, since it provides some terms that can be reused for the rest. We have

$$\frac{\partial \phi(\boldsymbol{\theta}; u, \tau)}{\partial \rho} = \phi(\boldsymbol{\theta}; u, \tau) \left(-\frac{\kappa \bar{v} \tau i u}{\sigma} - \frac{\partial A}{\partial \rho} \right) + \phi(\boldsymbol{\theta}; u, T) \frac{2\kappa \bar{v}}{\sigma^2} \frac{1}{B} \frac{\partial B}{\partial \rho} \quad (2.26a)$$

$$= \phi(\boldsymbol{\theta}; u, \tau) \left[-\frac{\kappa \bar{v} \tau i u}{\sigma} - \frac{\partial A}{\partial \rho} + \frac{2\kappa \bar{v}}{\sigma^2 d} \left(\frac{\partial d}{\partial \rho} - \frac{d}{A_2} \frac{\partial A_2}{\partial \rho} \right) \right] \quad (2.26b)$$

$$= \phi(\boldsymbol{\theta}; u, \tau) \left[-\frac{\partial A}{\partial \rho} + \frac{2\kappa \bar{v}}{\sigma^2 d} \left(\frac{\partial d}{\partial \rho} - \frac{d}{A_2} \frac{\partial A_2}{\partial \rho} \right) - \frac{\kappa \bar{v} \tau i u}{\sigma} \right], \quad (2.26c)$$

where

$$\frac{\partial d}{\partial \rho} = -\frac{\xi \sigma i u}{d}, \quad (2.27a)$$

$$\frac{\partial A_2}{\partial \rho} = -\frac{\sigma i u (2 + t\xi)}{2d} \left(\xi \cosh \frac{dt}{2} + d \sinh \frac{dt}{2} \right), \quad (2.27b)$$

$$\frac{\partial B}{\partial \rho} = e^{\kappa t/2} \left(\frac{1}{A_2} \frac{\partial d}{\partial \rho} - \frac{d}{A_2^2} \frac{\partial A_2}{\partial \rho} \right), \quad (2.27c)$$

$$\frac{\partial A_1}{\partial \rho} = -\frac{iu(u^2+iu)t\xi\sigma}{2d}\cosh\frac{dt}{2}, \tag{2.27d}$$

$$\frac{\partial A}{\partial \rho} = \frac{1}{A_2}\frac{\partial A_1}{\partial \rho} - \frac{A}{A_2}\frac{\partial A_2}{\partial \rho}. \tag{2.27e}$$

By merging and rearranging terms, we find that

$$\frac{\partial A}{\partial \kappa} = \frac{i}{\sigma u}\frac{\partial A}{\partial \rho}, \tag{2.28a}$$

$$\frac{\partial B}{\partial \kappa} = \frac{i}{\sigma u}\frac{\partial B}{\partial \rho} + \frac{B\tau}{2}, \tag{2.28b}$$

which are inserted into

$$\frac{\partial \phi(\boldsymbol{\theta};u,\tau)}{\partial \kappa} = \phi(\boldsymbol{\theta};u,\tau)\left(-\frac{\partial A}{\partial \kappa} + \frac{2\bar{v}}{\sigma^2}\log B + \frac{2\kappa\bar{v}}{\sigma^2 B}\frac{\partial B}{\partial \kappa} - \frac{\bar{v}\rho\tau iu}{\sigma}\right) \tag{2.29}$$

to reach the expression (2.23d). Similarly, Eq. (2.23e) can be obtained by applying the chain rule to Eq. (2.14), and the intermediate terms for $\partial\phi(\boldsymbol{\theta};u,\tau)/\partial\sigma$ can be written in terms of those for $\partial\phi(\boldsymbol{\theta};u,\tau)/\partial\rho$, that is

$$\frac{\partial d}{\partial \sigma} = \left(\frac{\rho}{\sigma} - \frac{1}{\xi}\right)\frac{\partial d}{\partial \rho} + \frac{\sigma u^2}{d}, \tag{2.30a}$$

$$\frac{\partial A_1}{\partial \sigma} = \frac{(u^2+iu)\tau}{2}\frac{\partial d}{\partial \sigma}\cosh\frac{d\tau}{2}, \tag{2.30b}$$

$$\frac{\partial A_2}{\partial \sigma} = \frac{\rho}{\sigma}\frac{\partial A_2}{\partial \rho} - \frac{2+\tau\xi}{v_0\tau\xi iu}\frac{\partial A_1}{\partial \rho} + \frac{\sigma\tau A_1}{2v_0}, \tag{2.30c}$$

$$\frac{\partial A}{\partial \sigma} = \frac{1}{A_2}\frac{\partial A_1}{\partial \sigma} - \frac{A}{A_2}\frac{\partial A_2}{\partial \sigma}. \tag{2.30d}$$

In the end, we replace $\log B$ appearing in Eqs. (2.25) and (2.29) with D, defined in Eq. (2.17b), to ensure the numerical continuity of the implementation. □

Next we discuss the computation of the integrands in Eq. (2.22) and their convergence.

2.3.3.2 Efficient calculation and convergence of the integrands

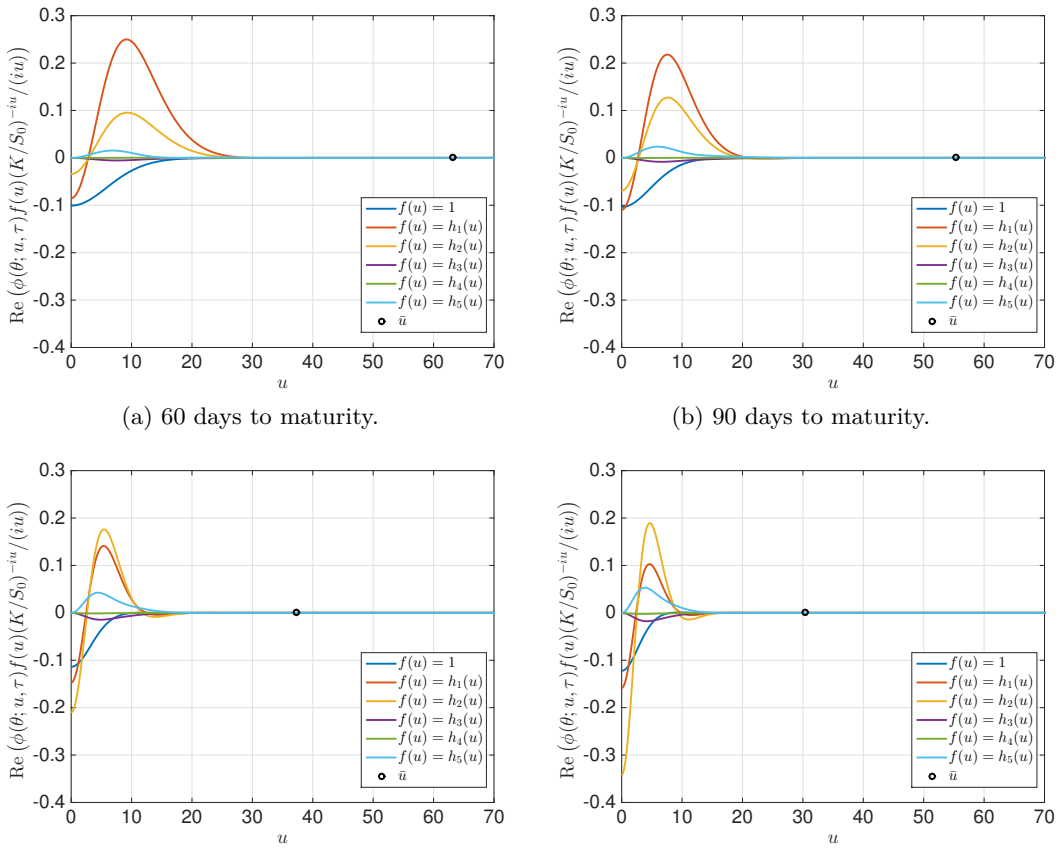
All integrands have the form $\mathrm{Re}\left(\phi(\boldsymbol{\theta};u,\tau)h_j(u)(K/S_0)^{-iu}/(iu)\right)$ and $h_j(u)$ is a product of elementary functions depending on which parameter is under consideration. It has been pointed out in the original paper by Heston [63] that the term $\mathrm{Re}\left(\phi(\boldsymbol{\theta};u,\tau)(K/S_0)^{-iu}/(iu)\right)$ is a smooth function that decays rapidly and presents no difficulties; its product with elementary functions decreases fast too. A visual example is shown in Fig. 2.3, with parameters given in Table 2.1. In our time units, $\tau = 1$ is a trading year made of 252 days.

Denote as $\bar{u}$ the value of u for which all integrands are not larger than 10^{-8}. For our testing parameter set, we observe in Figs. 2.3 and 2.4 that $\bar{u}$ decreases when τ increases. This is due to the fact that the more spread-out a function is, the more localised its Fourier transform is (see the uncertainty principle in physics): as

τ increases, the probability density of S_T stretches out, while its Fourier transform $\phi(\boldsymbol{\theta}; u, \tau)$ squeezes. More specifically, if X and U are random variables whose probability density functions are, apart of a constant, Fourier pairs of each other, the product of their variances is a constant, i.e., $\mathrm{Var}(X)\mathrm{Var}(U) \geq 1$.

Denote as $\bar{u}$ the value of u for which all integrands are not larger than 10^{-8}. For our testing parameter set, we observe in Figs. 2.3 and 2.4 that $\bar{u}$ decreases when T increases. This is due to the fact that the more spread-out a function is, the more localised its Fourier transform is (see the uncertainty principle in physics): as T increases, the probability density of S_T stretches out, while its Fourier transform $\phi(\boldsymbol{\theta}; u, T)$ squeezes. More specifically, if X and U are random variables whose probability density functions are, apart of a constant, Fourier pairs of each other, the product of their variances is a constant, i.e., $\mathrm{Var}(X)\mathrm{Var}(U) \geq 1$. Based on this observation, one can adjust the truncation according to the maturity of the option and hence do fewer integrand evaluations for options with longer maturities.

In order to obtain the integrands in Eq. (2.22), one only needs to compute $\phi(\boldsymbol{\theta}; u, \tau)$ and $\boldsymbol{h}(u)$. After rearranging and merging terms, we find that calculating



(a) 60 days to maturity.

(b) 90 days to maturity.

(c) 180 days to maturity.

(d) 252 days to maturity.

FIG. 2.3: Convergence of the integrands: $\mathrm{Re}\left(\phi(\boldsymbol{\theta}; u, \tau)(K/S_0)^{-iu}/(iu)\right)$ in the Heston pricing formula for $C(\boldsymbol{\theta}; K, \tau)$ (dark blue) and $\mathrm{Re}\left(\phi(\boldsymbol{\theta}; u, \tau)\boldsymbol{h}(u)(K/S_0)^{-iu}/(iu)\right)$ in the components of its gradient $\partial C/\partial \boldsymbol{\theta}$ (other colors); $h_j(u), j = 1, \ldots, 5$ are respectively relevant for $\partial C/\partial \theta_j$. The black circle indicates the value $\bar{u}$ where all integrands are below 10^{-8}.

$\boldsymbol{h}(u)$ can be boiled down to obtaining the intermediate terms (2.27), (2.28) and (2.30).

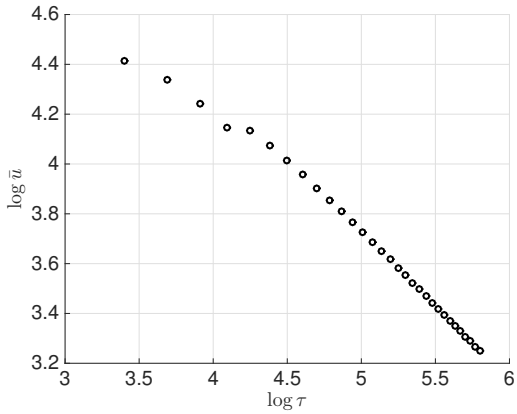


FIG. 2.4: As the time to maturity τ increases, the value $\bar{u}$ for which all integrands evaluate to 10^{-8} or less decreases.

ALGORITHM 2.3.1. Vectorised integration in the Heston gradient.

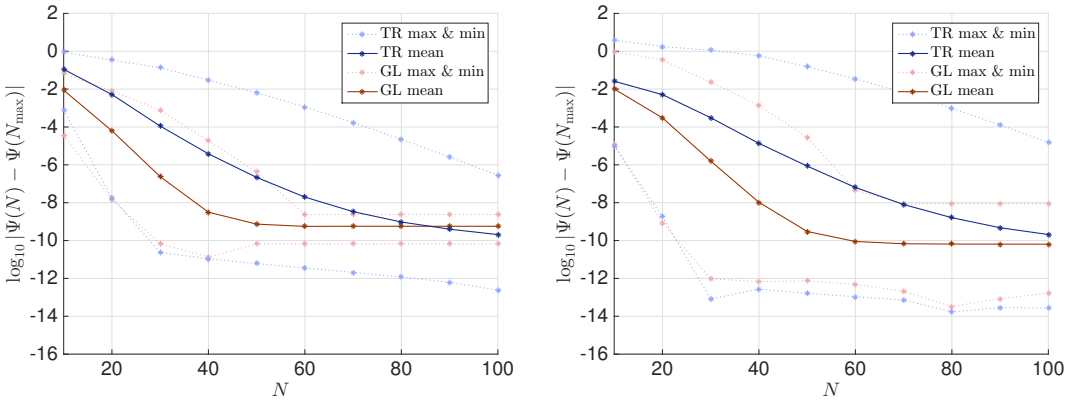
1 Specify N grid nodes $\{u_k\}_{k=1}^N$ and N corresponding weights $\{w_k\}_{k=1}^N$.
2 **for** $k = 1, 2, \ldots, N$ **do**
3 Compute $\boldsymbol{h}(u_k)$.
4 **end**
5 **for** $j = 1, 2, \ldots, 5$ **do**
6 Compute $\int_0^\infty \frac{K^{-iu}}{iu}\phi(\boldsymbol{\theta}; u, \tau)h_j(u)\mathrm{d}u \approx \sum_{k=1}^N \frac{K^{-iu_k}}{iu_k}\phi(\boldsymbol{\theta}; u_k, \tau)h_j(u_k)w_k$.
7 **end**

It is a favorable result that the components of $\boldsymbol{h}(u)$ share these common terms because then the gradient $\boldsymbol{\nabla} C(\boldsymbol{\theta}; K, \tau)$ can be obtained by vectorizing the quadrature for all the integrands as illustrated in Algorithm 2.3.1. Due to the interdependence among components of $\boldsymbol{h}(u)$, this scheme is faster than computing and integrating each component $h_j(u)$ individually. Next, we discuss the choice of the numerical integration method and of the key parameters N, u_k and w_k, but we point out that this vectorised quadrature is compatible with any numerical integration method.

2.3.3.3 Integration scheme

The computation of the integrals in the pricing function (2.9) and the gradient function (2.22) dominates the cost of calibration. Thus, we discuss the proper choice of the numerical integration scheme. Specifically, we compare the trapezoidal rule (TR) and the Gauss-Legendre rule (GL). In Figs. 2.5a and 2.5b, we plot the error of the integral evaluation respectively in the pricing formula and its gradient. The horizontal axis is the number of quadrature nodes N and the vertical axis is the $\log_{10}$ scale of the absolute error of integration, which is defined as

$$\varepsilon_{\text{integration}} := |\Phi(N) - \Phi(N_{\max})|, \tag{2.31}$$



(a) for the pricing function

(b) for the gradient function

FIG. 2.5: Comparison between TR (full blue for mean and dotted purple for maximum and minimum) and GL (full red for mean and dotted pink for maximum and minimum) for the error of the integral evaluation under the Heston model.

where $\Phi(N)$ is the value of the integration with N nodes, N is selected equidistantly in the range $[10, 100]$, $N_{\max}$ should be ∞ and is chosen as 1000 in our case. For the plots we use 40 options with different strikes and maturities. More details on these options are given in Section 2.5.

The error converges faster for GL than TR and has always a smaller variation when more options are involved. In order to achieve an average accuracy of 10^{-8}, GL requires 40 nodes and TR requires 70. In order for the integrations for all the options to achieve an accuracy at 10^{-8}, GL requires 60 nodes and TR requires much more than 100.

Besides the fast convergence of the integral error, GL is advantageous in its selection of nodes. GL rescales the domain of integration to $[-1, +1]$, selects nodes that are symmetric around the origin, and assigns the same weight to each symmetric pair of nodes. Thus, a further reduction in computation can be achieved by making use of the common terms of a node and its opposite. Based on these benefits, we choose the GL integration scheme with about 60 nodes to calibrate the Heston model.

2.3.3.4 Comparison with numerical gradient

Previous calibration methods approximate the gradient by a finite difference scheme. A central difference scheme is the approximation

$$\nabla C(\boldsymbol{\theta}; K, \tau) \approx \frac{C(\boldsymbol{\theta} + \boldsymbol{\epsilon}; K, \tau) - C(\boldsymbol{\theta} - \boldsymbol{\epsilon}; K, \tau)}{2\epsilon}, \tag{2.32}$$

where $\boldsymbol{\epsilon} := \epsilon \boldsymbol{e}$ and ϵ is small. Different values of the increment ϵ could be chosen for each component θ_j; for simplicity we have taken it constant. The size of the difference, ϵ, has a non-trivial effect on the approximation. An excessively small value of ϵ is not able to reflect the overall function behaviour at the point and may lead to a wrong moving direction. Moreover the numerical gradient naturally has an

error and one cannot expect to find a solution with a better accuracy than that of the gradient. In most cases, the iteration stagnates when the error of the objective function is roughly the same size as the error of the gradient.

Besides the instability caused by an inappropriate choice of ϵ, a numerical gradient has a higher computational cost than an analytical gradient. Recall that the evaluation of one option price $C(\boldsymbol{\theta}; K, \tau)$ requires the evaluation of two integrals as in Eq. (2.9). Let n be the number of options to be calibrated. At each iteration, one needs to compute $20n$ integrals if using the finite difference scheme while only $2n$ integrals if using the analytical form with the vectorised integration scheme. To give a more intuitive comparison between the two methods, we perform a preliminary experiment with $\epsilon = 10^{-4}$ and $n = 40$ using the MATLAB function `quadv` with an adaptive Simpson rule for the numerical integration.

In Table 2.3, we report the CPU time as an average of 500 runs and the number of calls of the integral function for each method. In order to give a relative sense of speed that is independent of the machine, the CPU time for analytical gradient is scaled to unity, and that for numerical gradient results about 16 times longer. Considering the 94% of saving in computational time and the exempt from deciding ϵ, we propose to use the analytical Heston gradient with vectorised quadrature in a gradient-based optimisation algorithm to calibrate the model.

2.4 Calibration using the Levenberg-Marquardt method

In this section, we present the algorithm for a complete and fast calibration of the Heston model using the LM method [91].

The LM method is a typical tool to solve a nonlinear least squares problem like Eq. (2.4). The search direction is given by

$$\Delta\boldsymbol{\theta} = (\boldsymbol{J}\boldsymbol{J}^\intercal + \mu\boldsymbol{I})^{-1}\boldsymbol{\nabla} f, \tag{2.33}$$

where $\boldsymbol{I}$ is the identity matrix and μ is a damping factor. By adaptively adjusting μ, the method alternates between the steepest descent method and the Gauss-Newton method. That is, when the iterate is far from the optimum, μ is given a large value so that the Hessian matrix is dominated by the scaled identity matrix

$$\boldsymbol{\nabla}\boldsymbol{\nabla}^\intercal f \approx \mu\boldsymbol{I}; \tag{2.34}$$

TABLE 2.3: A comparison between numerical and analytical gradients for $n = 40$ options.

Computational cost	Numerical gradient	Analytical gradient
CPU time (arbitrary units)	15.8	1.0
Number of integral evaluations	800	80

when the iterate is close to the optimum, μ is assigned a small value so that the Hessian matrix is dominated by the Gauss-Newton approximation

$$\boldsymbol{\nabla}\boldsymbol{\nabla}^{\intercal} f \approx \boldsymbol{J}\boldsymbol{J}^{\intercal}, \tag{2.35}$$

which omits the second term $\sum_{i=1}^{n} r_i \boldsymbol{H}(r_i)$ in Eq. (2.21b). The approximation (2.35) is reliable when either r_i or $\boldsymbol{H}(r_i)$ is small. The former happens when the problem is a so-called *small residual problem* and the latter happens when f is nearly linear. The viewpoint is that the model should yield small residuals around the optimum because otherwise it is an inappropriate model. The Heston model has been known to be able to explain the smile and skew of the volatility surface. Therefore, we conjecture it to be a small residual problem and adopt the approximation of the Hessian in Eq. (2.35) as converging to the optimum. There are various implementations of the LM method, such as `MINPACK` [36], `LEVMAR` [76], `sparseLM` [77] etc. We adopt the `LEVMAR` package which is a robust and stable implementation in C/C++ distributed under GNU. Although it has never been used in computational finance, `LEVMAR` has been integrated into many open source and commercial products in other applications such as astrometric calibration and image processing. See Algorithm 2.4.1.

ALGORITHM 2.4.1. Levenberg-Marquardt algorithm to calibrate the Heston model.

1 Given the initial guess $\boldsymbol{\theta}_0$, compute $\|\boldsymbol{r}(\boldsymbol{\theta}_0)\|$ and $\boldsymbol{J}_0$.
2 Choose the initial damping factor $\mu_0 = \tau \max\{\text{diag}(\boldsymbol{J}_0)\}$ and $\nu_0 = 2$.
3 **for** $k = 0, 1, 2, \ldots$ **do**
4 Solve the normal equations (2.33) for $\Delta\boldsymbol{\theta}_k$.
5 Compute $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \Delta\boldsymbol{\theta}_k$ and $\|\boldsymbol{r}(\boldsymbol{\theta}_{k+1})\|$.
6 Compute $\delta_L = \Delta\boldsymbol{\theta}_k{}^{\intercal}(\mu_k \Delta\boldsymbol{\theta}_k + \boldsymbol{J}_k \boldsymbol{r}(\boldsymbol{\theta}_k))$ and $\delta_F = \|\boldsymbol{r}(\boldsymbol{\theta}_k)\| - \|\boldsymbol{r}(\boldsymbol{\theta}_{k+1})\|$.
7 **if** $\delta_L > 0$ *and* $\delta_F > 0$ **then**
8 Accept the step: compute $\boldsymbol{J}_{k+1}$, $\mu_{k+1} = \mu_k$, $\nu_{k+1} = \nu_k$.
9 **else**
10 Recalculate the step: set $\mu_k = \mu_k \nu_k$, $\nu_k = 2\nu_k$ and repeat from line 4.
11 **end**
12 **if** *the stopping criterion* (2.36) *is met* **then**
13 Break.
14 **end**
15 **end**

In lines 1 and 5 of Algorithm 2.4.1, the option pricing function is evaluated. In lines 1 and 8, the gradient function is evaluated. The values of μ_0 and ν_0 in line 2 are the default choice of `LEVMAR`. In line 4, a 5×5 linear system is solved; in `LEVMAR` this is done by an LDLT factorization with the pivoting strategy of Bunch and Kaufman [13] using the `LAPACK` [8] routine.

The stopping criterion for the LM algorithm is when one of the following is

satisfied:

$$\|\boldsymbol{r}(\boldsymbol{\theta}_k)\| \leq \varepsilon_1, \tag{2.36a}$$

$$\|\boldsymbol{J}_k\boldsymbol{e}\|_\infty \leq \varepsilon_2, \tag{2.36b}$$

$$\frac{\|\Delta\boldsymbol{\theta}_k\|}{\|\boldsymbol{\theta}_k\|} \leq \varepsilon_3, \tag{2.36c}$$

where $\varepsilon_1, \varepsilon_2$ and ε_3 are tolerance levels. The first condition (2.36a) indicates that the iteration is stopped by a desired value of the objective function (2.4)-(2.5). The second condition (2.36b) indicates that the iteration is stopped by a small gradient. The third condition (2.36c) indicates that the iteration is stopped by a stagnating update.

2.5 Numerical results

In this section, we present our experimental results for the calibration of the Heston model. We first describe the data and then report the performance of our calibration method in comparison with the fastest previous method. We examine the Hessian matrix at the optimal solution which reveals the reason of the multiple optima observed in previous research. In the end, we test on three parameterisations that are typical for certain options. The result justifies the computational efficiency and robustness of our method for practical problems.

2.5.1 Data

In order to check whether the optimal parameter set found by the algorithm is the global optimum, we first presume a parameter set $\boldsymbol{\theta}^*$ specified in Table 2.1, and then use it to generate a volatility surface that is typically characterised by these options: the Δ_{10} call and put options, Δ_{25} call and put options, and Δ_{50} (i.e., ATM) call options with maturity from 30 to 360 days. Here $\Delta := \partial C(\boldsymbol{\theta}; K, \tau)/\partial S$ is the BS greek, i.e., the sensitivity of the option price with respect to the movement of its underlying spot. In Table 2.4, we give the BSM implied volatilities of 40 options

TABLE 2.4: Implied volatility surface for calibration.

Maturity in days	Δ_{10}^{put}	Δ_{25}^{put}	$\Delta_{50}^{\text{call}}$	$\Delta_{25}^{\text{call}}$	$\Delta_{10}^{\text{call}}$
30	2.5096	1.4359	0.2808	0.2540	0.2369
60	2.4351	1.3216	0.2847	0.2606	0.2417
90	2.3823	1.2955	0.2878	0.2660	0.2489
120	2.3383	1.2677	0.2904	0.2699	0.2548
150	2.2996	1.2407	0.2925	0.2745	0.2598
180	2.2619	1.2166	0.2943	0.2777	0.2641
252	2.1767	1.1671	0.2975	0.2837	0.2722
360	2.0618	1.1136	0.3007	0.2897	0.2803

that are generated by $\boldsymbol{\theta}^*$. This is a quoting style of the volatility surface commonly used in the financial industry, where Δ is a measure equivalent to the strike K. We denote call and put options using superscripts, respectively as Δ^{call} and Δ^{put}. The target is thus to find a parameter set $\boldsymbol{\theta}^\dagger$ that can replicate the volatility surface in Table 2.4. If $\boldsymbol{\theta}^\dagger$ is far from $\boldsymbol{\theta}^*$ or in other words, depends on the initial guess $\boldsymbol{\theta}_0$, then one concludes that local optimal parameter sets exist. Otherwise the problem presents only a global optimum.

We validated our method using different optimal parameters and initial guesses in a reasonable range given in Table 2.5. The procedure is described in Algorithm 2.5.1.

TABLE 2.5: Reasonable ranges to randomly generate Heston model parameters and the average absolute distance between the initial guess $\boldsymbol{\theta}_0$ and the optimum $\boldsymbol{\theta}^*$.

Range for model parameters		Absolute deviation from $\boldsymbol{\theta}^*$	
κ	$(0.50, 5.00)$	$\lvert\kappa_0 - \kappa^*\rvert$	1.5097
$\bar{v}$	$(0.05, 0.95)$	$\lvert\bar{v}_0 - \bar{v}^*\rvert$	0.2889
σ	$(0.05, 0.95)$	$\lvert\sigma_0 - \sigma^*\rvert$	0.2875
ρ	$(-0.90, -0.10)$	$\lvert\rho_0 - \rho^*\rvert$	0.2557
v_0	$(0.05, 0.95)$	$\lvert(v_0)_0 - v_0^*\rvert$	0.3063

ALGORITHM 2.5.1. Validation procedure.

for $i = 1, 2, \ldots, 100$ **do**
  Generate a vector of optimal parameters $\boldsymbol{\theta}_i^*$, each component of which is an independent uniformly distributed random number in the interval specified in Table 2.5.
  for $j = 1, 2, \ldots, 100$ **do**
    Generate an initial guess $\boldsymbol{\theta}_{0j}$, each component of which is an independent uniformly distributed random number in the interval specified in Table 2.5.
    Validate Algorithm 2.4.1 using the initial guess $\boldsymbol{\theta}_{0j}$ to find $\boldsymbol{\theta}_i^*$.
  end
end

Following this procedure, we validated Algorithm 2.4.1 with 10 000 test cases. An average of the distances between the initial guesses $\boldsymbol{\theta}_0$ and the optima $\boldsymbol{\theta}^*$ is given in Table 2.5. The results of the tests are discussed in the next section.

2.5.2 Performance

The computations were performed on a MacBook Pro with a 2.6 GHz Intel Core i5 processor, 8 GB of RAM and OS X Yosemite version 10.10.5. The pricing and gradient functions for the Heston model were coded in C++ using Xcode version 7.3.1. We use `LEVMAR` version 2.6 [76] as the LM solver and the tolerance in (2.36) is specified as $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 10^{-10}$. However, in our experiments the LM iteration

was always stopped by meeting the condition on the objective function (2.36a). We use GL integration with $N = 64$ nodes and for simplicity we truncate the upper limit of the integration in Eq. (2.22) at $\bar{u} = 200$ which shall be enough for pricing and calibrating in all cases.

The proposed method succeeds in finding the presumed parameter set in 9 843 cases out of 10 000 without any constraints on the search space and in 9 856 cases restraining the search to the intervals specified in Table 2.5. The average CPU time for the whole calibration process is less than 0.3 seconds. See Table 2.6 for detailed information on the whole validation set. In Table 2.7 and in the rest of this section we specify the information for a representative example with the optimal parameter set $\boldsymbol{\theta}^*$ specified in Table 2.1 and the initial guess $\boldsymbol{\theta}_0 = [1.20, 0.20, 0.30, -0.60, 0.20]^\intercal$.

The convergence of the residual $\boldsymbol{r}_k$ and the relative distance of each parameter towards the optimum is plotted in Fig. 2.6. In Figs. 2.7a and 2.7b, we plot the pricing error on the implied volatility surface at the initial point $\boldsymbol{\theta}_0$ and the optimal point $\boldsymbol{\theta}^\dagger$, respectively. As can be seen, the pricing error decreases from 10^{-2} to 10^{-7} after 13 steps.

This result contrasts the conclusion of previous research: local optimal parameters are not intrinsically embedded in the Heston calibration problem, but rather caused by an objective function shaped as a narrow valley with a flat bottom and a premature stopping criterion.

We plot the contours for $\|\boldsymbol{r}\|$ when varying 2 out of 5 parameters. Starting from $\boldsymbol{\theta}_0$, the iteration path is shown with contour plots in Fig. 2.8. The initial point $\boldsymbol{\theta}_0$ is marked with a black circle and the true solution $\boldsymbol{\theta}^*$ is marked with a black plus symbol. The red lines with asterisks are the iteration paths of $\boldsymbol{\theta}_k$, $k = 1, \ldots, 13$.

TABLE 2.6: Information about the optimisation: average over 10 000 testing cases.

Absolute deviation from $\boldsymbol{\theta}^*$		Error measure		Computational cost	
$\lvert\kappa^\dagger - \kappa^*\rvert$	1.54×10^{-3}	$\|\boldsymbol{r}_0\|$	1.39×10^{-1}	CPU time (seconds)	0.29
$\lvert\bar{v}^\dagger - \bar{v}^*\rvert$	2.40×10^{-5}	$\|\boldsymbol{r}^\dagger\|$	2.94×10^{-11}	LM iterations	12.82
$\lvert\sigma^\dagger - \sigma^*\rvert$	3.79×10^{-3}	$\|\boldsymbol{J}^\dagger \boldsymbol{e}\|_\infty$	1.47×10^{-5}	price evaluations	14.57
$\lvert\rho^\dagger - \rho^*\rvert$	1.52×10^{-2}	$\|\Delta\boldsymbol{\theta}^\dagger\|$	3.21×10^{-4}	gradient evaluations	12.82
$\lvert v_0^\dagger - v_0^*\rvert$	6.98×10^{-6}			linear systems solved	13.57

TABLE 2.7: Information about the optimisation of a representative example.

Absolute deviation from $\boldsymbol{\theta}^*$		Error measure		Computational cost	
$\lvert\kappa^\dagger - \kappa^*\rvert$	1.09×10^{-3}	$\|\boldsymbol{r}_0\|$	4.73×10^{-2}	CPU time (seconds)	0.29
$\lvert\bar{v}^\dagger - \bar{v}^*\rvert$	2.18×10^{-6}	$\|\boldsymbol{r}^\dagger\|$	1.00×10^{-12}	LM iterations	13
$\lvert\sigma^\dagger - \sigma^*\rvert$	4.70×10^{-5}	$\|\boldsymbol{J}^\dagger \boldsymbol{e}\|_\infty$	1.21×10^{-5}	price evaluations	14
$\lvert\rho^\dagger - \rho^*\rvert$	9.89×10^{-6}	$\|\Delta\boldsymbol{\theta}^\dagger\|$	2.50×10^{-4}	gradient evaluations	13
$\lvert v_0^\dagger - v_0^*\rvert$	1.18×10^{-6}			linear systems solved	13

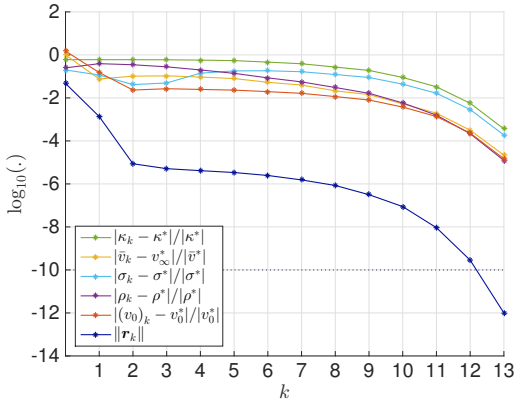
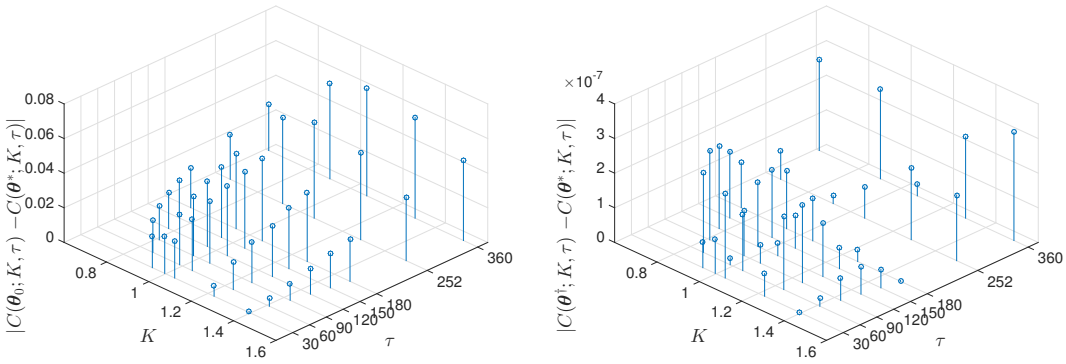


FIG. 2.6: The convergence of the LM method.



(a) at the initial point $\boldsymbol{\theta}^0$ (b) the final point $\boldsymbol{\theta}^\dagger$

FIG. 2.7: Pricing error on the implied volatility surface.

TABLE 2.8: The Hessian matrix $\boldsymbol{\nabla}\boldsymbol{\nabla}^\intercal f(\boldsymbol{\theta}^*)$.

	$\partial\kappa$	$\partial\bar{v}$	$\partial\sigma$	$\partial\rho$	∂v_0
$\partial\kappa$	5.26×10^{-5}				
$\partial\bar{v}$	9.65×10^{-3}	$2.26 \times 10^{+1}$			
$\partial\sigma$	-5.49×10^{-4}	-7.66×10^{-2}	7.46×10^{-3}		
$\partial\rho$	1.61×10^{-4}	2.00×10^{-2}	-2.34×10^{-3}	7.56×10^{-4}	
∂v_0	5.28×10^{-3}	$1.18 \times 10^{+1}$	-3.53×10^{-2}	8.40×10^{-3}	9.69×10^{-1}

For almost all pairs, the first step is a long steepest descent step that is nearly orthogonal to the contour. The rest are relatively cautious steps with the Gauss-Newton approximation of the Hessian. The contour plots do not show evidence for local minima, at least not in 2 dimensional sections.

The Gauss-Newton approximation of the Hessian matrix at the optimal solution is given in Table 2.8.

The Hessian matrix is ill-conditioned with a condition number of 3.978×10^6. The elements $\partial^2 f(\boldsymbol{\theta}^*)/\partial\kappa^2$ and $\partial^2 f(\boldsymbol{\theta}^*)/\partial\rho^2$ are of a much smaller order than the others. This suggests that the objective function, when around the optimum, is *less sensitive* to changes along κ and ρ. The effect of κ on the objective function

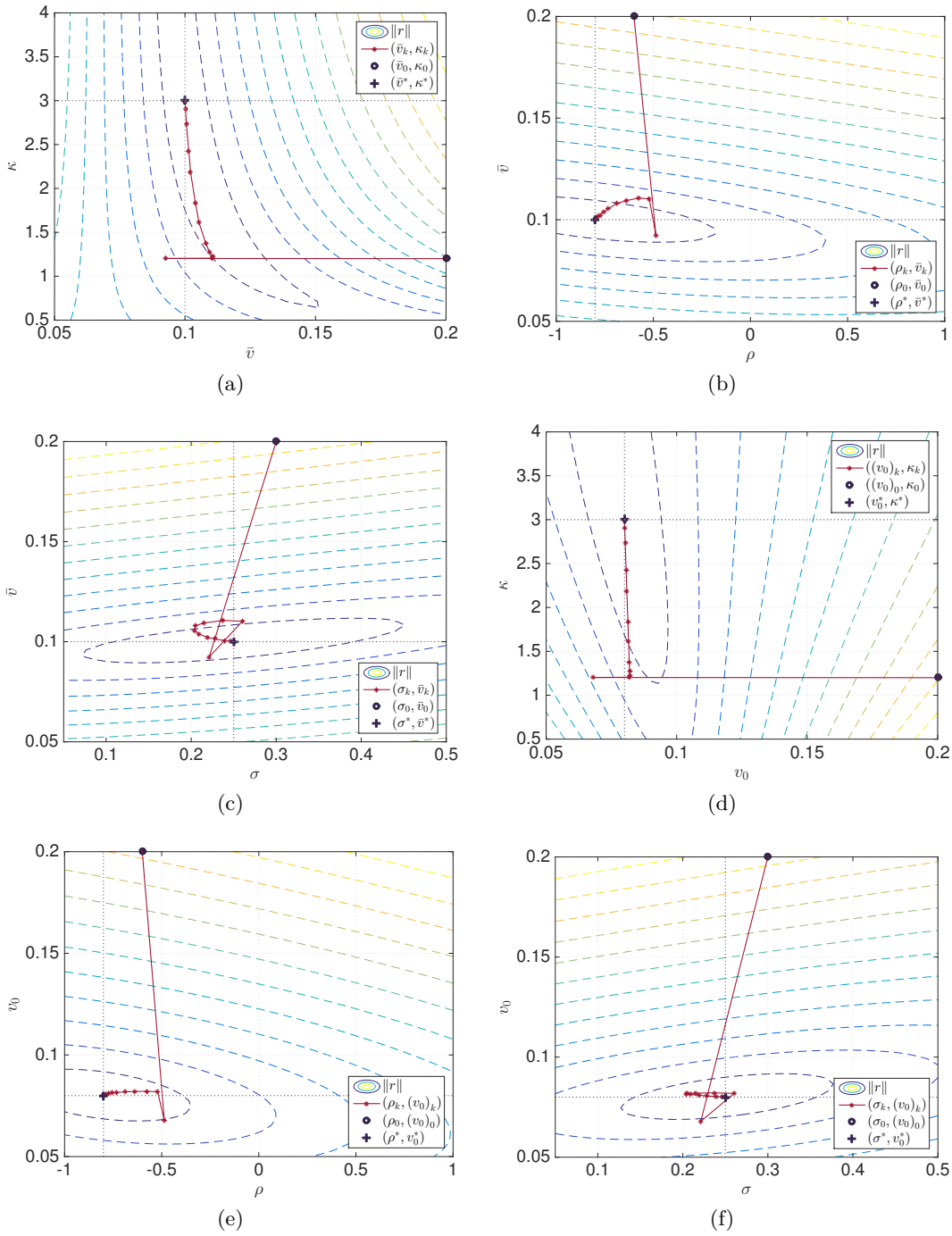


FIG. 2.8: Contours of $\|\boldsymbol{r}\|$ and iteration path for (θ_i, θ_j).

is weak because option prices depend on the integrated volatility, which is little sensitive to the degree of oscillation of volatility; ρ controls the slope of the smile, so this parameter is difficult to identify if a narrow range of moneyness is used for calibration.

In other words, the objective function is more stretched along these two axes as can be verified looking at the contours, for example in Figs. 2.8a and 2.8b. The ratio between $\partial^2 f(\boldsymbol{\theta}^*)/\partial\kappa^2$ and $\partial^2 f(\boldsymbol{\theta}^*)/\partial\bar{v}^2$ is of order 10^{-6}, which indicates a great disparity in sensitivity: changing 1 unit of $\bar{v}$ is comparable to changing 10^6 units of

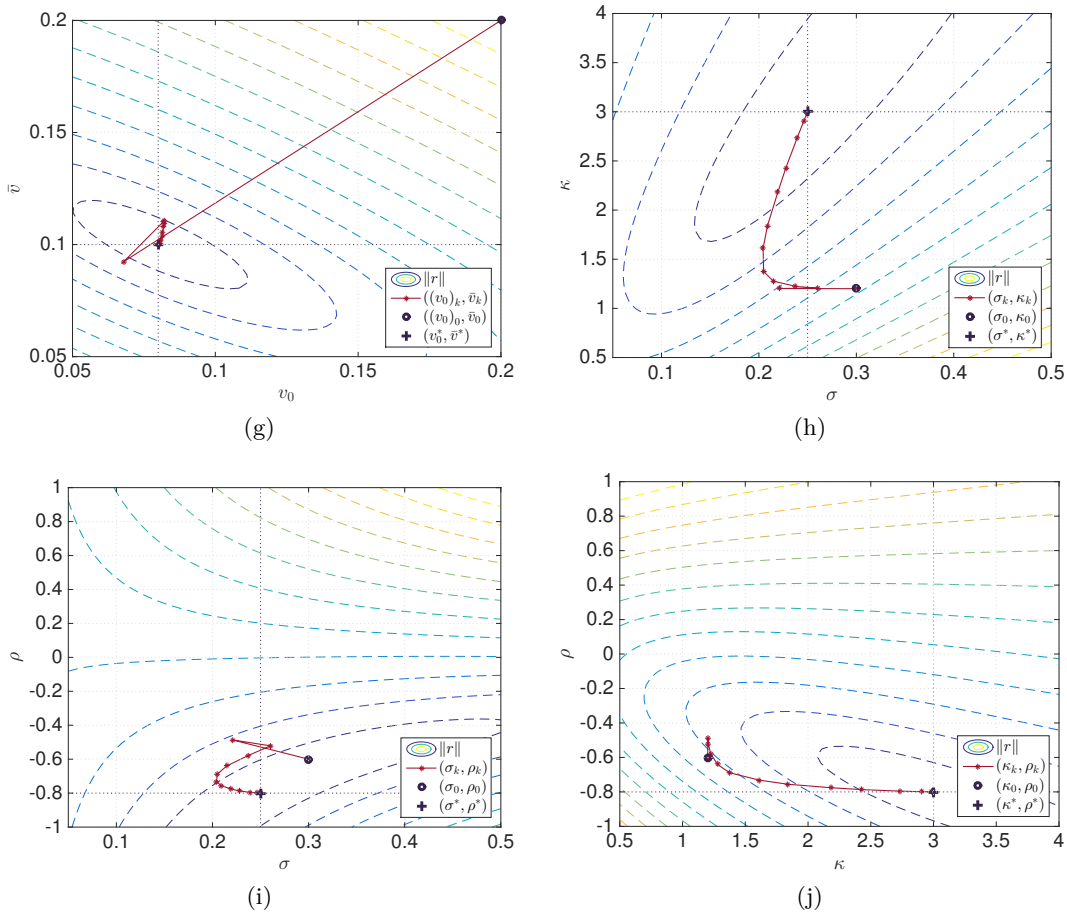


FIG. 2.8: (cont.) Contours of $\|\boldsymbol{r}\|$ and iteration path for (θ_i, θ_j).

κ. On the other hand, this explains the so-called local minima reported in previous research. When one starts from a different initial point and stops the iteration with a high tolerance, it is possible that the iterate lands somewhere in the region where κ and ρ are very different. There are two possible approaches that one can seek to deal with this: the first is to scale the parameters to a similar order and search on a better-scaled objective function; the second is to decrease the tolerance level for the optimisation process, meaning to approach the very bottom of this objective function.

In Table 2.9, we present the performance of the LM method with analytical gradient (LMA), the LM method with numerical gradient (LMN), and a feasibility perturbed sequential quadratic programming method (FPSQP) [49] adopted in UniCredit bank. As the concrete implementation of FPSQP is owned by the bank, we only extract their test results. The computational cost can be compared through the number of evaluations of the pricing function (2.9) per iteration, expressed as a multiple of the number n of options to be calibrated. LMA requires about n pricing function evaluation per step. LMN requires more for the gradient approximation, but the difference is not large since LMN uses a rank-one update for the subsequent Jacobian matrices. FPSQP requires about 5.5 times more than that of LMA and achieves only a lower accuracy for the stopping criterion for the gradient.

We tested our method also on a few realistic model parameterisations. In Table 2.10, we present three test cases that are representative respectively for long-dated FX options, long-dated interest rate options and equity options [7]. They are believed to be prevalent and challenging for the simulation of Heston model [53]. Each component of the initial guess is an independent uniformly distributed random number in the $\pm 10\%$ range of the corresponding optimum. This choice is due to the fact that practitioners usually choose the initial guess as the last available estimation which is expected to be close to the solution if the calibration is frequent enough and the market does not change drastically. We test each case with 100 initial guesses. Our previous test range in Table 2.5 has covered these cases too, but here we would like to focus on the performance of our method when applied to these typical examples and thus justify its computational efficiency and robustness for practical application. The information about the convergence as an average of the 100 initial guesses is given in Table 2.11. For the practical cases with initial guesses in the vicinity, it takes less than or around one second to obtain the optimal solution.

2.6 Conclusion

We proposed a new representation of the Heston characteristic function which is continuous and easily derivable. We derived the analytical form of the gradient of the Heston option pricing function with respect to the model parameters. The result can be applied in any gradient-based algorithm. An algorithm for a full and fast calibration of the Heston model is given. The LM method succeeds in finding the global optimal parameter set within a reasonable number of iterations. The method

TABLE 2.9: Performance comparison between solvers.

	LMA	LMN	FPSQP
Stopping criterion	$\|\boldsymbol{r}(\boldsymbol{\theta}_k)\| \leq 10^{-10}$	$\|\boldsymbol{r}(\boldsymbol{\theta}_k)\| \leq 10^{-10}$	$\|\boldsymbol{\Delta\theta}_k\| \leq 10^{-6}$
Iterations	13	22	-
Price evaluations per iteration	$1.08n$	$1.70n$	$6.00n$

TABLE 2.10: Test cases with realistic Heston model parameters. Case I: long-dated FX options. Case II: long-dated interest rate options. Case III: equity options.

	Case I	Case II	Case III
κ^*	0.50	0.30	1.00
$\bar{v}^*$	0.04	0.04	0.09
σ^*	1.00	0.90	1.00
ρ^*	-0.90	-0.50	-0.30
v_0^*	0.04	0.04	0.09

TABLE 2.11: Calibration results for three typical realistic cases, reporting an average on 100 initial guesses for each of them.

		Case I	Case II	Case III
Absolute deviation from $\boldsymbol{\theta}^*$	$\lvert\kappa^\dagger - \kappa^*\rvert$	2.87×10^{-2}	1.35×10^{-3}	1.20×10^{-3}
	$\lvert\bar{v}^\dagger - \bar{v}^*\rvert$	4.80×10^{-3}	4.52×10^{-5}	2.11×10^{-5}
	$\lvert\sigma^\dagger - \sigma^*\rvert$	5.29×10^{-2}	7.48×10^{-4}	3.94×10^{-4}
	$\lvert\rho^\dagger - \rho^*\rvert$	3.65×10^{-2}	1.69×10^{-5}	1.46×10^{-5}
	$\lvert v_0^\dagger - v_0^*\rvert$	2.14×10^{-3}	1.46×10^{-5}	1.07×10^{-5}
Error measure	$\lVert\boldsymbol{r}_0\rVert$	2.70×10^{-4}	4.51×10^{-5}	1.02×10^{-4}
	$\lVert\boldsymbol{r}^\dagger\rVert$	1.12×10^{-4}	9.24×10^{-11}	3.33×10^{-11}
	$\lVert\boldsymbol{J}^\dagger\boldsymbol{e}\rVert_\infty$	1.77×10^{-1}	4.63×10^{-6}	4.15×10^{-6}
	$\lVert\Delta\boldsymbol{\theta}^\dagger\rVert$	6.88×10^{-21}	1.63×10^{-8}	5.10×10^{-5}
Computational cost	CPU time	0.40	1.11	0.15
	LM iterations	16.83	51.52	6.86
	Price evaluations	23.38	52.60	7.86
	Gradient evaluations	16.83	51.52	6.86
	Linear systems solved	23.38	51.52	6.86

is validated by randomly generated parameterisations as well as three typical cases of Heston model parameterisations for long-dated FX options, long-dated interest rate options and equity options. The resulting parameters can replicate the volatility surface with an l_2-norm error of 10^{-10} and an l_1-norm error around 10^{-7}. The cheap computational cost and the stable performance for different initial guesses make the proposed method suitable for the purpose of high-frequency trading. Several numerical issues are discussed. We also present the final Hessian matrix and contours of the objective function. We point out that either a rescaling of the parameters or a low tolerance level is needed to find the global optimum.

Chapter 3

Implementation of Interior-point Methods for LP based on Krylov Subspace Iterative Solvers with Inner-iteration Preconditioning

Linear programming (LP) is one of the most fundamental optimisation problems. It is defined as a problem formulated by linear objective function and linear constraints. Many real-life problems can be modelled to own this form, and solved by an LP solver.

An example in finance is to solve a basic problem in financial mathematics. The problem was originally described in a note by [30].

In this short problem description, these notations are used:

- m: number of states (scenarios),
- n: number of assets,
- $P \in \mathcal{R}^{m \times n}$: payoff matrix, where P_{ij} represents the payoff of the j^{th} asset under the i^{th} state,
- $\boldsymbol{h} \in \mathcal{R}^n$: trading strategy, hold h_j units of asset j,
- $\boldsymbol{y} \in \mathcal{R}^m$: defined as $\boldsymbol{y} := P \times \boldsymbol{h}$, represents the return of the portfolio under different states given the strategy $\boldsymbol{h}$. Due to the obvious fact that under no states shall we lose money, $\boldsymbol{y} \geq \boldsymbol{0}$ needs to be satisfied component-wise,
- $\boldsymbol{q} \in \mathcal{R}^m$: risk-neutral probability measure. By this definition in the context of financial mathematics, the conditions shall hold:
  1. $\boldsymbol{q} \geq \boldsymbol{0}$, $\sum q_i = 1$, the definition of probability measure,

2. $\langle \boldsymbol{q},\ P(:,j)\rangle = 0$, the expected return of the j^{th} assets should be zero under no arbitrage condition[3],

- $\boldsymbol{e} := [1, 1, \ldots, 1]^\mathsf{T}$: an all-one vector of proper size.

The problem is to maximize the payoff of the portfolio under all states

$$\max \boldsymbol{e}^\mathsf{T}\boldsymbol{y} \qquad \text{subject to} \qquad \boldsymbol{y} = P\boldsymbol{h},\ \ \boldsymbol{y} \geq \boldsymbol{0}. \tag{3.1}$$

It can be written in the standard dual form

$$\max \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{e} \end{bmatrix}^\mathsf{T} \begin{bmatrix} \boldsymbol{h} \\ \boldsymbol{y} \end{bmatrix}, \qquad \text{subject to} \qquad \begin{bmatrix} P & -I \\ -P & I \\ 0 & -I \end{bmatrix} \begin{bmatrix} \boldsymbol{h} \\ \boldsymbol{y} \end{bmatrix} + \boldsymbol{s} = \boldsymbol{0},\ \ \boldsymbol{s} \geq \boldsymbol{0}, \tag{3.2}$$

and its dual is

$$\min \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}^\mathsf{T} \boldsymbol{x}, \qquad \text{subject to} \qquad \begin{bmatrix} P^\mathsf{T} & -P^\mathsf{T} & \boldsymbol{0} \\ -I & I & -I \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{e} \end{bmatrix},\ \ \boldsymbol{x} \geq \boldsymbol{0}. \tag{3.3}$$

The problems (3.2) and (3.3) constitute the primal-dual form of an LP problem.

Not restricted to finance, LP has arisen from a wide range of applications, and the size of problems to be solved has been increasing to millions. Thus, a good lP solver that achieves speed and robustness is of importance to industrial practice.

As is well known, to solve a LP problem, there are two general classes of methods, simplex and interior-point. Both methods seek for solution by searching from one point to the next until the optimum is achieved. In the simplex method, each step is cheap to compute, but the total number of steps may be large. On the other hand, in the interior-point method, each step is more computationally expensive, but can make a huge progress towards the optimal point. Our focus is on the interior-point method. In this section we present an implementation of the interior-point algorithm for LP based on Krylov subspace methods for least squares problems. We employ an inner-iteration preconditioner recently developed by Morikuni and Hayami [61, 93, 94] to deal with severe ill-conditioning of linear equations in the final stage of iterations. The advantage of our methods is that it does not break down even when previous direct methods do. Also, we save computation time and storage compared to previous preconditioners.

3.1 Introduction

Consider the linear programming (LP) problem in the standard primal-dual formulation

$$\min_{\boldsymbol{x}} \boldsymbol{c}^\mathsf{T}\boldsymbol{x} \qquad \text{subject to} \qquad A\boldsymbol{x} = \boldsymbol{b},\ \ \boldsymbol{x} \geq \boldsymbol{0}, \tag{3.4a}$$

[3] $\langle \boldsymbol{x},\ \boldsymbol{y}\rangle$ denotes the inner product between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$

$$\max_{\boldsymbol{y},\boldsymbol{s}} \boldsymbol{b}^{\mathsf{T}}\boldsymbol{y} \qquad \text{subject to} \qquad A^{\mathsf{T}}\boldsymbol{y} + \boldsymbol{s} = \boldsymbol{c}, \quad \boldsymbol{s} \geq \boldsymbol{0}, \tag{3.4b}$$

where $A \in \mathbb{R}^{m\times n}$, $m \leq n$, and we assume the existence of an optimal solution. In this paper, we describe an implementation of the interior-point method for LP based on iterative solvers. The main computational task in one iteration of the interior-point method is the solution of a system of linear equations to compute the search direction. Although there are two known approaches for this, i.e., direct and iterative methods, the direct method is the primary choice so far and there is very few implementation solely depending on an iterative method, e.g., [17]. This is because the linear system becomes notoriously ill-conditioned toward the end of interior-point iterations and no iterative solver has managed to resolve this difficulty.

Here, we apply novel inner-iteration preconditioned Krylov subspace methods for least squares problems. The inner-iteration preconditioners recently proposed by Morikuni and Hayami [93, 94] enable us to deal with the severe ill-conditioning of the system of linear equations. Furthermore, the proposed Krylov subspace methods do not suffer from singularity and therefore no preprocessing is necessary even if A is rank-deficient. This is another advantage of our approach over the direct solvers.

Extensive numerical experiments were conducted over diverse instances of 125 LP problems taken from the benchmark libraries Netlib, Qaplib, and Mittelmann collections. The largest problem has 434,580 variables. Our implementation proves to be more robust than the public-domain solvers SeDuMi (Self-Dual Minimization) [108] and SDPT3 (Semidefinite Programming Toh-Todd-Tütüncü) [110, 111] without increasing CPU time. As far as the authors know, this is the first time an interior-point method entirely based on iterative solvers succeeds in solving a fairly large number of standard LP instances from the benchmark libraries with standard stopping criteria. Our implementation is considerably slower than the interior-point solver of MOSEK [95], one of the state-of-the-art commercial solvers, though it is competitive in robustness. On the other hand, we observed that our implementation is able to solve ill-conditioned dense problems with severe rank-deficiency which the MOSEK solver can not solve.

We emphasize that there are many interesting topics to be further worked out based on this paper. There is still room for improvement regarding the iterative solvers as well as using more sophisticated methods for the interior-point iterations.

In the following, we introduce the interior-point method and review the iterative solvers previously used. We employ an infeasible primal-dual predictor-corrector interior-point method, one of the methods that evolved from the original primal-dual interior-point method [109, 74, 87, 113] incorporating several innovative ideas, e.g., [115, 80].

The optimal solution $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}$ to problem (3.4) must satisfy the Karush-Kuhn-Tucker (KKT) conditions

$$A^{\mathsf{T}}\boldsymbol{y} + \boldsymbol{s} = \boldsymbol{c}, \tag{3.5a}$$

$$A\boldsymbol{x} = \boldsymbol{b}, \tag{3.5b}$$

$$XS\boldsymbol{e} = \boldsymbol{0}, \tag{3.5c}$$

$$\boldsymbol{x} \geq \boldsymbol{0}, \quad \boldsymbol{s} \geq \boldsymbol{0}, \tag{3.5d}$$

where $X := \mathrm{diag}(x_1, x_2, \ldots, x_n)$, $S := \mathrm{diag}(s_1, s_2, \ldots, s_n)$, and $\boldsymbol{e} := [1, 1, \ldots, 1]^\mathsf{T}$. The complementarity condition (3.5c) implies that at the optimal point, one of the elements x_i or s_i must be zero for $i = 1, 2, \ldots, n$.

The following system is obtained by relaxing (3.5c) to $XS\boldsymbol{e} = \mu\boldsymbol{e}$ with $\mu > 0$:

$$XS\boldsymbol{e} = \mu\boldsymbol{e}, \quad A\boldsymbol{x} = \boldsymbol{b}, \quad A^\mathsf{T}\boldsymbol{y} + \boldsymbol{s} = \boldsymbol{c}, \quad \boldsymbol{x} \geq \boldsymbol{0}, \quad \boldsymbol{s} \geq \boldsymbol{0}. \tag{3.6}$$

The interior-point method solves the problem (3.4) by generating approximate solutions to (3.6), with μ decreasing toward zero, so that (3.5) is satisfied within some tolerance level at the solution point. The search direction at each infeasible interior-point step is obtained by solving the Newton equations

$$\begin{bmatrix} \boldsymbol{0} & A^\mathsf{T} & I \\ A & \boldsymbol{0} & \boldsymbol{0} \\ S & \boldsymbol{0} & X \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{y} \\ \Delta\boldsymbol{s} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_\mathrm{d} \\ \boldsymbol{r}_\mathrm{p} \\ \boldsymbol{r}_\mathrm{c} \end{bmatrix}, \tag{3.7}$$

where $\boldsymbol{r}_\mathrm{d} := \boldsymbol{c} - A^\mathsf{T}\boldsymbol{y} - \boldsymbol{s} \in \mathbb{R}^n$ is the residual of the dual problem, $\boldsymbol{r}_\mathrm{p} := \boldsymbol{b} - A\boldsymbol{x} \in \mathbb{R}^m$ is the residual of the primal problem, $\boldsymbol{r}_\mathrm{c} := -XS\boldsymbol{e} + \sigma\mu\boldsymbol{e}$, $\mu := \boldsymbol{x}^\mathsf{T}\boldsymbol{s}/n$ is the duality measure, and $\sigma \in [0, 1)$ is the centring parameter, which is dynamically chosen to govern the progress of the interior-point method. Once the kth iterate $(\boldsymbol{x}^{(k)}, \boldsymbol{y}^{(k)}, \boldsymbol{s}^{(k)})$ is given and (3.7) is solved, we define the next iterate as $(\boldsymbol{x}^{(k+1)}, \boldsymbol{y}^{(k+1)}, \boldsymbol{s}^{(k+1)}) := (\boldsymbol{x}^{(k)}, \boldsymbol{y}^{(k)}, \boldsymbol{s}^{(k)}) + \alpha(\Delta\boldsymbol{x}, \Delta\boldsymbol{y}, \Delta\boldsymbol{s})$, where $\alpha \in (0, 1]$ is a step length to ensure the positivity of $\boldsymbol{x}$ and $\boldsymbol{s}$, and then reduce μ to $\sigma\mu$ before solving (3.7) again.

At each iteration, the solution of (3.7) dominates the total CPU time. The choice of linear solvers depends on the way of arranging the matrix of (3.7). Aside from solving the $(m + 2n) \times (m + 2n)$ system (3.7), one can solve its reduced equivalent form of size $(m + n) \times (m + n)$

$$\begin{bmatrix} A & \boldsymbol{0} \\ S & -XA^\mathsf{T} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_\mathrm{p} \\ \boldsymbol{r}_\mathrm{c} - X\boldsymbol{r}_\mathrm{d} \end{bmatrix}, \tag{3.8}$$

or a more condensed equivalent form of size $m \times m$

$$AXS^{-1}A^\mathsf{T}\Delta\boldsymbol{y} = \boldsymbol{r}_\mathrm{p} - AS^{-1}(\boldsymbol{r}_\mathrm{c} - X\boldsymbol{r}_\mathrm{d}), \tag{3.9}$$

both of which are obtained by performing block Gaussian eliminations on (3.7). We are concerned in this paper with solving the third equivalent form (3.9).

It is known that the matrix of (3.9) is not positive definite when any of the following cases is encountered. First, when A is rank-deficient, system (3.9) is sin-

gular. There exist presolving techniques that can detect and remove the dependent rows in A, see, e.g., [5, 55]. Second, in late interior-point iterations, the diagonal matrix XS^{-1} has very tiny and very large diagonal values as a result of convergence. Thus, the matrix may become positive semidefinite, or even slightly indefinite, due to rounding error. In particular, the situation becomes severe when primal degeneracy occurs at the optimal solution. One can refer to [58, 116] for more detailed explanations.

Thus, when direct methods such as Cholesky decomposition are applied to (3.9), some diagonal pivots encountered during decomposition can be zero or negative, causing the algorithm to break down. Many direct methods adopt a strategy of replacing the problematic pivot with a very large number. See, e.g., [116] for the Cholesky-Infinity factorization, which is specially designed to solve (3.9) when it is positive semidefinite but not definite. Numerical experience [1, 78, 44, 79, 6, 114, 29] indicates that direct methods provide sufficiently accurate solutions for interior-point methods to converge regardless of the ill-conditioning of the matrix. However, as the LP problems become larger, the significant fill-ins in decompositions make direct methods prohibitively expensive. It is stated in [56] that the fill-ins are observed even for very sparse matrices. Moreover, the matrix can be dense, as in quadratic programs in support vector machine training [42] or linear programs in basis pursuit [17], and even when A is sparse, $AXS^{-1}A^{\mathsf{T}}$ can be dense or have a pattern of nonzero elements that renders the system difficult for direct methods. The expensive solution of the KKT systems is a usual disadvantage of second-order methods including interior-point methods.

These drawbacks of direct methods and the progress in preconditioning techniques motivate researchers to develop stable iterative methods for solving (3.9) or alternatively (3.8). The major problem is that as the interior-point iterations proceed, the condition number of the term XS^{-1} increases, making the system of linear equations intractable. One way to deal with this is to employ suitable preconditioners. Since our main focus is on solving (3.9), we explain preconditioners for (3.9) in detail in the following. We mention [18, 45, 46, 10, 96] as literature related to preconditioners for (3.8).

For the iterative solution of (3.9), the conjugate gradient (CG) method [62] has been applied with diagonal scaling preconditioners [14, 100, 75] or incomplete Cholesky preconditioners [80, 73, 18, 83]. LSQR with a preconditioner was used in [50]. A matrix-free method of using CG for least squares (CGLS) preconditioned by a partial Cholesky decomposition was proposed in [57]. In [24], a preconditioner based on Greville's method [25] for generalized minimal residual (GMRES) method was applied. Suitable preconditioners were also introduced for particular fields such as the minimum-cost network flow problem in [102, 70, 88, 89]. One may refer to [31] for a review on the application of numerical linear algebra algorithms to the solutions of KKT systems in the optimisation context.

In this chapter, we propose to solve (3.9) using Krylov subspace methods precon-

ditioned by stationary inner-iterations recently proposed for least squares problems in [61, 93, 94]. In Section 3.2, we briefly describe the framework of Mehrotra's predictor-corrector interior-point algorithm we implemented and the normal equations arising from this algorithm. In Section 3.3, we specify the application of our method to the normal equations. In Section 3.4, we present numerical results in comparison with a modified sparse Cholesky method and three direct solvers in CVX, a major public package for specifying and solving convex programs [60, 59]. In Section 3.5, we conclude the work.

Throughout this chapter, we use bold lower case letters for column vectors. We denote quantities related to the kth interior-point iteration by using a superscript with round brackets, e.g., $\boldsymbol{x}^{(k)}$, the kth iteration of Krylov subspace methods by using a subscript without brackets, e.g., $\boldsymbol{x}_k$, and the kth inner iteration by using a superscript with angle brackets, e.g., $\boldsymbol{x}^{\langle k \rangle}$. $\mathcal{R}(A)$ denotes the range space of a matrix A. $\kappa(A)$ denotes the condition number $\kappa(A) = \sigma_1(A)/\sigma_r(A)$, where $\sigma_1(A)$ and $\sigma_r(A)$ denote the maximum and minimum nonzero singular values of A, respectively. $\mathcal{K}_k(A, \boldsymbol{b}) = \mathrm{span}\{\boldsymbol{b}, A\boldsymbol{b}, \ldots, A^{k-1}\boldsymbol{b}\}$ denotes the Krylov subspace of order k.

3.2 Interior-point algorithm and the normal equations

We implement an infeasible version of Mehrotra's predictor-corrector method [81], which has been established as a standard in this area [78, 79, 113, 82]. Note that our method can be applied to other interior-point methods (see, e.g., [113] for more interior-point methods) whose directions are computed via the normal equations (3.9).

3.2.1 Mehrotra's predictor-corrector algorithm

In this method, the centring parameter σ is determined by dividing each step into two stages.

In the first stage, we solve for the affine direction $(\Delta\boldsymbol{x}_{\mathrm{af}}, \Delta\boldsymbol{y}_{\mathrm{af}}, \Delta\boldsymbol{s}_{\mathrm{af}})$

$$\begin{bmatrix} \mathbf{0} & A^{\mathsf{T}} & I \\ A & \mathbf{0} & \mathbf{0} \\ S & \mathbf{0} & X \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x}_{\mathrm{af}} \\ \Delta\boldsymbol{y}_{\mathrm{af}} \\ \Delta\boldsymbol{s}_{\mathrm{af}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_{\mathrm{d}} \\ \boldsymbol{r}_{\mathrm{p}} \\ -XS\boldsymbol{e} \end{bmatrix}, \tag{3.10}$$

and measure its progress in reducing μ. If the affine direction makes large enough progress without violating the nonnegative boundary (3.5d), then σ is assigned a small value. Otherwise, σ is assigned a larger value to steer the iterate to be more centred in the strictly positive region.

In the second stage, we solve for the corrector direction $(\Delta\boldsymbol{x}_{\mathrm{cc}}, \Delta\boldsymbol{y}_{\mathrm{cc}}, \Delta\boldsymbol{s}_{\mathrm{cc}})$

$$\begin{bmatrix} \mathbf{0} & A^{\mathsf{T}} & I \\ A & \mathbf{0} & \mathbf{0} \\ S & \mathbf{0} & X \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x}_{\mathrm{cc}} \\ \Delta\boldsymbol{y}_{\mathrm{cc}} \\ \Delta\boldsymbol{s}_{\mathrm{cc}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ -\Delta X_{\mathrm{af}} \Delta S_{\mathrm{af}} \boldsymbol{e} + \sigma\mu\boldsymbol{e} \end{bmatrix}, \tag{3.11}$$

where $\Delta X_{\rm af} = {\rm diag}(\Delta \boldsymbol{x}_{\rm af})$, $\Delta S_{\rm af} = {\rm diag}(\Delta \boldsymbol{s}_{\rm af})$ and σ is determined according to the solution in the first stage. Finally, we update the current iterate along the linear combination of the two directions.

In our implementation of the interior-point method, we adopt Mehrotra's predictor-corrector algorithm as follows.

ALGORITHM 3.2.1. Mehrotra's predictor-corrector algorithm.

1: Given $(\boldsymbol{x}^{(0)}, \boldsymbol{y}^{(0)}, \boldsymbol{s}^{(0)})$ with $(\boldsymbol{x}^{(0)}, \boldsymbol{s}^{(0)}) > \boldsymbol{0}$.
2: **for** $k = 0, 1, 2, \ldots$ until convergence, **do**
3: $\quad \mu^{(k)} := {\boldsymbol{x}^{(k)}}^{\mathsf{T}} \boldsymbol{s}^{(k)}/n$ `// the predictor stage`
4: $\quad$ Solve (3.10) for the affine direction $(\Delta \boldsymbol{x}_{\rm af}, \Delta \boldsymbol{y}_{\rm af}, \Delta \boldsymbol{s}_{\rm af})$.
5: $\quad$ Compute $\alpha_{\rm p},\ \alpha_{\rm d}$.
6: $\quad$ **if** $\min(\alpha_{\rm p},\ \alpha_{\rm d}) \geq 1$ **then**
7: $\quad\quad \sigma := 0,\ \left(\Delta \boldsymbol{x}^{(k)}, \Delta \boldsymbol{y}^{(k)}, \Delta \boldsymbol{s}^{(k)}\right) := (\Delta \boldsymbol{x}_{\rm af}, \Delta \boldsymbol{y}_{\rm af}, \Delta \boldsymbol{s}_{\rm af})$
8: $\quad$ **else**
9: $\quad\quad$ Set $\mu_{\rm af}$ and $\sigma :=$ a small value, e.g., 0.208. `// the corrector stage`
10: $\quad\quad$ Solve (3.11) for the corrector direction $(\Delta \boldsymbol{x}_{\rm cc}, \Delta \boldsymbol{y}_{\rm cc}, \Delta \boldsymbol{s}_{\rm cc})$.
11: $\quad\quad \left(\Delta \boldsymbol{x}^{(k)}, \Delta \boldsymbol{y}^{(k)}, \Delta \boldsymbol{s}^{(k)}\right) := (\Delta \boldsymbol{x}_{\rm af}, \Delta \boldsymbol{y}_{\rm af}, \Delta \boldsymbol{s}_{\rm af}) + (\Delta \boldsymbol{x}_{\rm cc}, \Delta \boldsymbol{y}_{\rm cc}, \Delta \boldsymbol{s}_{\rm cc})$
12: $\quad$ **end if**
13: $\quad$ Compute $\hat{\alpha}_{\rm p},\ \hat{\alpha}_{\rm d}$.
14: $\quad \boldsymbol{x}^{(k+1)} := \boldsymbol{x}^{(k)} + \hat{\alpha}_{\rm p} \Delta \boldsymbol{x}^{(k)},\ \left(\boldsymbol{y}^{(k+1)}, \boldsymbol{s}^{(k+1)}\right) := \left(\boldsymbol{y}^{(k)}, \boldsymbol{s}^{(k)}\right) + \hat{\alpha}_{\rm d} \left(\Delta \boldsymbol{y}^{(k)}, \Delta \boldsymbol{s}^{(k)}\right)$
15: **end for**

In line 5 in Algorithm 3.2.1, the step lengths $\alpha_{\rm p},\ \alpha_{\rm d}$ are computed by

$$\alpha_{\rm p} = \eta \min\left(1, \min_{i:\Delta x_i < 0} \left(-\frac{x_i}{\Delta x_i}\right)\right), \quad \alpha_{\rm d} = \eta \min\left(1, \min_{i:\Delta s_i < 0} \left(-\frac{s_i}{\Delta s_i}\right)\right), \tag{3.12}$$

where $(\Delta \boldsymbol{x}, \Delta \boldsymbol{s}) = (\Delta \boldsymbol{x}_{\rm af}, \Delta \boldsymbol{s}_{\rm af}), \eta \in [0.9, 1)$.

In line 9, the quantity $\mu_{\rm af}$ is computed by

$$\mu_{\rm af} = (\boldsymbol{x}^{(k)} + \alpha_{\rm p} \Delta \boldsymbol{x}_{\rm af})^{\mathsf{T}} (\boldsymbol{s}^{(k)} + \alpha_{\rm d} \Delta \boldsymbol{s}_{\rm af})/n.$$

In the same line, the parameter σ is chosen as $\sigma = \min(0.208, (\mu_{\rm af}/\mu^{(k)})^2)$ in the early phase of the interior-point iterations, where the value 0.208 is adopted from the LIPSOL package [116]. In the late phase of the interior-point iterations, σ is chosen as approximately 10 times the error measure Γ defined in (3.31). Here the distinction between *early* and *late* phases is when Γ is more or less than 10^{-3}.

In line 13, we first compute trial step lengths $\alpha_{\rm p}, \alpha_{\rm d}$ using Eqs. (3.12) with $(\Delta \boldsymbol{x}, \Delta \boldsymbol{s}) = (\Delta \boldsymbol{x}^{(k)}, \Delta \boldsymbol{s}^{(k)})$. Then, we gradually reduce $\alpha_{\rm p}, \alpha_{\rm d}$ to find the largest step lengths that can ensure the centrality of the updated iterates, i.e., to find the maximum $\hat{\alpha}_{\rm p},\ \hat{\alpha}_{\rm d}$ that satisfy

$$\min_i (x_i + \hat{\alpha}_{\rm p} \Delta x_i)(s_i + \hat{\alpha}_{\rm d} \Delta s_i) \geq \phi (\boldsymbol{x} + \hat{\alpha}_{\rm p} \Delta \boldsymbol{x})^{\mathsf{T}} (\boldsymbol{s} + \hat{\alpha}_{\rm d} \Delta \boldsymbol{s})/n,$$

where ϕ is typically chosen as 10^{-5}.

3.2.2 The normal equations in the interior-point algorithm

We consider modifying Algorithm 3.2.1 so that it is not necessary to update $\boldsymbol{y}^{(k)}$. Since we assume the existence of an optimal solution to problem (3.4), we have $\boldsymbol{b} \in \mathcal{R}(A)$. Let $D := S^{-1/2}X^{1/2}$ and $\mathcal{A} := AD$. Problem (3.9) with $\Delta\boldsymbol{w} = \mathcal{A}^{\mathsf{T}}\Delta\boldsymbol{y}$ (the normal equations of the second kind) is equivalent to

$$\min \|\Delta\boldsymbol{w}\|_2 \quad \text{subject to} \quad \mathcal{A}\Delta\boldsymbol{w} = \boldsymbol{f}, \tag{3.13}$$

where $\boldsymbol{f} := \boldsymbol{r}_{\mathrm{p}} - AS^{-1}(\boldsymbol{r}_{\mathrm{c}} - X\boldsymbol{r}_{\mathrm{d}})$.

In the predictor stage, the problem (3.10) is equivalent to first solving (3.13) for $\Delta\boldsymbol{w}_{\mathrm{af}}$ with $\Delta\boldsymbol{w} = \Delta\boldsymbol{w}_{\mathrm{af}}$, $\boldsymbol{f} = \boldsymbol{f}_{\mathrm{af}} := \boldsymbol{b} + AS^{-1}X\boldsymbol{r}_{\mathrm{d}}$, and then updating the others by

$$\Delta\boldsymbol{s}_{\mathrm{af}} = \boldsymbol{r}_{\mathrm{d}} - D^{-1}\Delta\boldsymbol{w}_{\mathrm{af}}, \tag{3.14a}$$

$$\Delta\boldsymbol{x}_{\mathrm{af}} = -D^2\Delta\boldsymbol{s}_{\mathrm{af}} - \boldsymbol{x}. \tag{3.14b}$$

In the corrector stage, the problem (3.11) is equivalent to first solving (3.13) for $\Delta\boldsymbol{w}_{\mathrm{cc}}$ with $\Delta\boldsymbol{w} = \Delta\boldsymbol{w}_{\mathrm{cc}}$, $\boldsymbol{f} = \boldsymbol{f}_{\mathrm{cc}} := AS^{-1}\Delta X_{\mathrm{af}}\Delta S_{\mathrm{af}}\boldsymbol{e} - \sigma\mu AS^{-1}\boldsymbol{e}$, and then updating the others by

$$\Delta\boldsymbol{s}_{\mathrm{cc}} = -D^{-1}\Delta\boldsymbol{w}_{\mathrm{cc}}, \tag{3.15a}$$

$$\Delta\boldsymbol{x}_{\mathrm{cc}} = -D^2\Delta\boldsymbol{s}_{\mathrm{cc}} - S^{-1}\Delta X_{\mathrm{af}}\Delta\boldsymbol{s}_{\mathrm{af}} + \sigma\mu S^{-1}\boldsymbol{e}. \tag{3.15b}$$

By solving (3.13) for $\Delta\boldsymbol{w}$ instead of solving (3.9) for $\Delta\boldsymbol{y}$, we can compute $\Delta\boldsymbol{s}_{\mathrm{af}}$, $\Delta\boldsymbol{x}_{\mathrm{af}}$, $\Delta\boldsymbol{s}_{\mathrm{cc}}$, and $\Delta\boldsymbol{x}_{\mathrm{cc}}$ and can save 1MV[4] in (3.14a) and another in (3.15a) if a predictor step is performed per interior-point iteration.

Remark 3.1. For solving an interior-point step from the condensed step equation (3.9) using a suited Krylov subspace method, updating $(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{s})$ rather than $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s})$ can save 1MV each interior-point iteration.

Note that in the predictor and corrector stages, problem (3.13) has the same matrix but different right-hand sides. We introduce methods for solving it in the next section.

3.3 Application of inner-iteration preconditioned Krylov subspace methods

In lines 4 and 10 of Algorithm 3.2.1, the linear system (3.13) needs to be solved, with its matrix becoming increasingly ill-conditioned as the interior-point iterations proceed. In this section, we focus on applying inner-iteration preconditioned Krylov subspace methods to (3.13) because they are advantageous in dealing with ill-conditioned sparse matrices. The methods to be discussed are the preconditioned

[4] "MV" denotes the computational cost required for one matrix-vector multiplication.

CG and MINRES methods [62, 97] applied to the normal equations of the second kind ((P)CGNE and (P)MRNE, respectively) [23, 94], and the right-preconditioned generalized minimal residual method (AB-GMRES) [61, 94].

First, the conjugate gradient (CG) method [62] is an iterative method for solving linear systems $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A} \in \mathbf{R}^{n\times n}$ is a symmetric and positive (semi)definite matrix and $\mathbf{b} \in \mathcal{R}(\mathbf{A})$. CG starts with an initial approximate solution $\mathbf{x}_0 \in \mathbb{R}^n$ and determines the kth iterate $\mathbf{x}_k \in \mathbb{R}^n$ by minimising $\|\mathbf{x}_k - \mathbf{x}_*\|_{\mathbf{A}}^2$ over the space $\mathbf{x}_0 + \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$, where $\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$, $\mathbf{x}_*$ is a solution of $\mathbf{Ax} = \mathbf{b}$, and $\|\mathbf{x}_k - \mathbf{x}_*\|_{\mathbf{A}}^2 := (\mathbf{x}_k - \mathbf{x}_*)^\mathsf{T}\mathbf{A}(\mathbf{x}_k - \mathbf{x}_*)$.

Second, MINRES [97] is another iterative method for solving linear systems $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A} \in \mathbf{R}^{n\times n}$ is symmetric. MINRES with $\mathbf{x}_0$ determines the kth iterate $\mathbf{x}_k$ by minimising $\|\mathbf{b} - \mathbf{Ax}\|_2$ over the same space as CG.

Third, the generalized minimal residual (GMRES) method [104] is an iterative method for solving linear systems $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A} \in \mathbf{R}^{n\times n}$. GMRES with $\mathbf{x}_0$ determines the kth iterate $\mathbf{x}_k$ by minimising $\|\mathbf{b} - \mathbf{Ax}\|_2$ over $\boldsymbol{x}_0 + \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$.

3.3.1 Application of inner-iteration preconditioned CGNE and MRNE methods

We first introduce CGNE and MRNE. Let $\mathbf{A} = \mathcal{A}\mathcal{A}^\mathsf{T}$, $\mathbf{x} = \Delta\boldsymbol{y}_{\rm af}$, $\mathbf{b} = \boldsymbol{f}_{\rm af}$, and $\Delta\boldsymbol{w}_{\rm af} = \mathcal{A}^\mathsf{T}\Delta\boldsymbol{y}_{\rm af}$ for the predictor stage, and similarly, let $\mathbf{A} = \mathcal{A}\mathcal{A}^\mathsf{T}$, $\mathbf{x} = \Delta\boldsymbol{y}_{\rm cc}$, $\mathbf{b} = \boldsymbol{f}_{\rm cc}$, and $\Delta\boldsymbol{w}_{\rm cc} = \mathcal{A}^\mathsf{T}\Delta\boldsymbol{y}_{\rm cc}$ for the corrector stage. CG and MINRES applied to systems $\mathbf{Ax} = \mathbf{b}$ are CGNE and MRNE, respectively. With these settings, let the initial solution $\Delta\boldsymbol{w}_0 \in \mathcal{R}(\mathcal{A}^\mathsf{T})$ in both stages, and denote the initial residual by $\boldsymbol{g}_0 := \boldsymbol{f} - \mathcal{A}\Delta\boldsymbol{w}_0$. CGNE and MRNE can solve (3.13) without forming $\mathcal{A}\mathcal{A}^\mathsf{T}$ explicitly.

Concretely, CGNE gives the kth iterate $\Delta\boldsymbol{w}_k$ such that $\|\Delta\boldsymbol{w}_k - \Delta\boldsymbol{w}_*\|_2 = \min_{\Delta\boldsymbol{w}\in\Delta\boldsymbol{w}_0+\mathcal{K}_k(\mathcal{A}^\mathsf{T}\mathcal{A},\mathcal{A}^\mathsf{T}\boldsymbol{g}_0)} \|\Delta\boldsymbol{w} - \Delta\boldsymbol{w}_*\|_2$, where $\Delta\boldsymbol{w}_*$ is the minimum-norm solution of $\mathcal{A}\Delta\boldsymbol{w} = \boldsymbol{f}$ for $\Delta\boldsymbol{w}_0 \in \mathcal{R}(\mathcal{A}^\mathsf{T})$ and $\boldsymbol{f} \in \mathcal{R}(\mathcal{A})$. MRNE gives the kth iterate $\Delta\boldsymbol{w}_k$ such that $\|\boldsymbol{f} - \mathcal{A}\Delta\boldsymbol{w}_k\|_2 = \min_{\Delta\boldsymbol{w}\in\Delta\boldsymbol{w}_0+\mathcal{K}_k(\mathcal{A}^\mathsf{T}\mathcal{A},\mathcal{A}^\mathsf{T}\boldsymbol{g}_0)} \|\boldsymbol{f} - \mathcal{A}\Delta\boldsymbol{w}\|_2$.

We use inner-iteration preconditioning for CGNE and MRNE methods. The following is a brief summary of the part of [94] where the inner-outer iteration method is analysed. We give the expressions for the inner-iteration preconditioning and preconditioned matrices to state the conditions under which the former is SPD. Let M be a symmetric nonsingular splitting matrix of $\mathcal{A}\mathcal{A}^\mathsf{T}$ such that $\mathcal{A}\mathcal{A}^\mathsf{T} = M - N$. Denote the inner-iteration matrix by $H = M^{-1}N$. The inner-iteration preconditioning and preconditioned matrices are $C^{(\ell)} = \sum_{i=0}^{\ell-1} H^i M^{-1}$ and $\mathcal{A}\mathcal{A}^\mathsf{T}C^{(\ell)} = M\sum_{i=0}^{\ell-1}(I - H)H^iM^{-1} = M(I - H^\ell)M^{-1}$, respectively. If $C^{(\ell)}$ is nonsingular, then $\mathcal{A}\mathcal{A}^\mathsf{T}C^{(\ell)}\boldsymbol{u} = \boldsymbol{f}$, $\boldsymbol{z} = C^{(\ell)}\boldsymbol{u}$ is equivalent to $\mathcal{A}\mathcal{A}^\mathsf{T}\boldsymbol{z} = \boldsymbol{f}$ for all $\boldsymbol{f} \in \mathcal{R}(\mathcal{A})$. For ℓ odd, $C^{(\ell)}$ is symmetric and positive definite (SPD) if and only if the inner-iteration splitting matrix M is SPD [92, Theorem 2.8]. For ℓ even, $C^{(\ell)}$ is SPD if and only if the inner-iteration splitting matrix $M + N$ is SPD [92, Theorem 2.8]. We give Algorithms 3.3.1–3.3.2 for CGNE and MRNE preconditioned by inner

iterations [94, Algorithms E.3, E.4].

ALGORITHM 3.3.1. CGNE method preconditioned by inner iterations.

1: Let $\Delta \boldsymbol{w}_0$ be the initial approximate solution, and $\boldsymbol{g}_0 := \boldsymbol{f} - \mathcal{A}\Delta \boldsymbol{w}_0$.
2: Apply ℓ steps of a stationary iterative method to $\mathcal{A}\mathcal{A}^\mathsf{T}\boldsymbol{z} = \boldsymbol{g}_0$, $\boldsymbol{u} = \mathcal{A}^\mathsf{T}\boldsymbol{z}$ to obtain $\boldsymbol{z}_0 := \mathcal{C}^{\langle \ell \rangle}\boldsymbol{g}_0$ and $\boldsymbol{u}_0 := \mathcal{A}^\mathsf{T}\boldsymbol{z}_0$.
3: $\boldsymbol{s}_0 := \boldsymbol{u}_0, \gamma_0 := (\boldsymbol{g}_0, \boldsymbol{z}_0)$
4: **for** $k = 0, 1, 2, \ldots$ until convergence, **do**
5: $\alpha_k := \gamma_k/(\boldsymbol{s}_k, \boldsymbol{s}_k)$, $\Delta \boldsymbol{w}_{k+1} := \Delta \boldsymbol{w}_k + \alpha \boldsymbol{s}_k$, $\boldsymbol{g}_{k+1} := \boldsymbol{g}_k - \alpha_k \mathcal{A}\boldsymbol{s}_k$
6: Apply ℓ steps of a stationary iterative method to $\mathcal{A}\mathcal{A}^\mathsf{T}\boldsymbol{z} = \boldsymbol{g}_{k+1}$ to obtain $\boldsymbol{z}_{k+1} := \mathcal{C}^{\langle \ell \rangle}\boldsymbol{g}_{k+1}$ and $\boldsymbol{u}_{k+1} := \mathcal{A}^\mathsf{T}\boldsymbol{z}_{k+1}$.
7: $\gamma_{k+1} := (\boldsymbol{g}_{k+1}, \boldsymbol{z}_{k+1})$, $\beta_k := \gamma_{k+1}/\gamma_k$, $\boldsymbol{s}_{k+1} := \boldsymbol{u}_{k+1} + \beta_k \boldsymbol{s}_k$
8: **end for**

ALGORITHM 3.3.2. MRNE method preconditioned by inner iterations.

1: Let $\Delta \boldsymbol{w}_0$ be the initial approximate solution, and $\boldsymbol{g}_0 := \boldsymbol{f} - \mathcal{A}\Delta \boldsymbol{w}_0$.
2: Apply ℓ steps of a stationary iterative method to $\mathcal{A}\mathcal{A}^\mathsf{T}\boldsymbol{u} = \boldsymbol{g}_0$, $\boldsymbol{s} = \mathcal{A}^\mathsf{T}\boldsymbol{u}$ to obtain $\boldsymbol{s}_0 := \mathcal{A}^\mathsf{T}\mathcal{C}^{(\ell)}\boldsymbol{g}_0$.
3: $\boldsymbol{p}_0 := \boldsymbol{s}_0$, $\gamma_0 := \|\boldsymbol{s}_0\|_2^2$
4: **for** $k = 1, 2, \ldots$ until convergence, **do**
5: $\boldsymbol{t}_k := \mathcal{A}\boldsymbol{p}_k$
6: Apply ℓ steps of a stationary iterative method to $\mathcal{A}\mathcal{A}^\mathsf{T}\boldsymbol{u} = \boldsymbol{t}_k$, $\boldsymbol{v} = \mathcal{A}^\mathsf{T}\boldsymbol{u}$ to obtain $\boldsymbol{v}_k := \mathcal{A}^\mathsf{T}\mathcal{C}^{(\ell)}\boldsymbol{t}_k$.
7: $\alpha_k := \gamma_k/(\boldsymbol{v}_k, \boldsymbol{p}_k)$, $\Delta \boldsymbol{w}_k := \Delta \boldsymbol{w}_k + \alpha_k \boldsymbol{p}_k$, $\boldsymbol{g}_{k+1} := \boldsymbol{g}_k - \alpha_k \boldsymbol{t}_k$, $\boldsymbol{s}_{k+1} := \boldsymbol{s}_k - \alpha_k \boldsymbol{v}_k$
8: $\gamma_k := \|\boldsymbol{s}_{k+1}\|_2^2$, $\beta_k := \gamma_{k+1}/\gamma_k$, $\boldsymbol{p}_{k+1} := \boldsymbol{s}_k + \beta_k \boldsymbol{p}_k$
9: **end for**

3.3.2 Application of inner-iteration preconditioned AB-GMRES method

Next, we introduce AB-GMRES. GMRES can solve a square linear system transformed from the rectangular system $\mathcal{A}\Delta \boldsymbol{w}_{\mathrm{af}} = \boldsymbol{f}_{\mathrm{af}}$ in the predictor stage and $\mathcal{A}\Delta \boldsymbol{w}_{\mathrm{cc}} = \boldsymbol{f}_{\mathrm{cc}}$ in the corrector stage by using a rectangular right-preconditioning matrix that does not necessarily have to be $\mathcal{A}^\mathsf{T}$. Let $\mathcal{B} \in \mathbb{R}^{n \times m}$ be a preconditioning matrix for $\mathcal{A}$. Then, AB-GMRES corresponds to GMRES [104] applied to

$$\mathcal{A}\mathcal{B}\boldsymbol{z} = \boldsymbol{f}, \quad \Delta \boldsymbol{w} = \mathcal{B}\boldsymbol{z},$$

which is equivalent to the minimum-norm solution to the problem (3.13), for all $\boldsymbol{f} \in \mathcal{R}(\mathcal{A})$ if $\mathcal{R}(\mathcal{B}) = \mathcal{R}(\mathcal{A}^\mathsf{T})$ [94, Theorem 5.2], where $\Delta \boldsymbol{w} = \Delta \boldsymbol{w}_{\mathrm{af}}$ or $\Delta \boldsymbol{w}_{\mathrm{cc}}$, $\boldsymbol{f} = \boldsymbol{f}_{\mathrm{af}}$ or $\boldsymbol{f}_{\mathrm{cc}}$, respectively. AB-GMRES gives the kth iterate $\Delta \boldsymbol{w}_k = \mathcal{B}\boldsymbol{z}_k$ such that $\boldsymbol{z}_k = \mathrm{argmin}_{\boldsymbol{z} \in \boldsymbol{z}_0 + \mathcal{K}_k(\mathcal{A}\mathcal{B}, \boldsymbol{g}_0)} \|\boldsymbol{f} - \mathcal{A}\mathcal{B}\boldsymbol{z}\|_2$, where $\boldsymbol{z}_0$ is the initial iterate and $\boldsymbol{g}_0 = \boldsymbol{f} - \mathcal{A}\mathcal{B}\boldsymbol{z}_0$.

Specifically, we apply AB-GMRES preconditioned by inner iterations [93, 94] to (3.13). This method was shown to outperform previous methods on ill-conditioned

and rank-deficient problems. We give expressions for the inner-iteration preconditioning and preconditioned matrices. Let M be a nonsingular splitting matrix such that $\mathcal{A}\mathcal{A}^\mathsf{T} = M - N$. Denote the inner-iteration matrix by $H = M^{-1}N$. With $C^{(\ell)} = \sum_{i=0}^{\ell-1} H^i M^{-1}$, the inner-iteration preconditioning and preconditioned matrices are $\mathcal{B}^{(\ell)} = \mathcal{A}^\mathsf{T} C^{(\ell)}$ and $\mathcal{A}\mathcal{B}^{(\ell)} = \sum_{i=0}^{\ell-1}(I-H)H^i = M(I-H^\ell)M^{-1}$, respectively. If the inner-iteration matrix H is semiconvergent, i.e., $\lim_{i\to\infty} H^i$ exists, then AB-GMRES determines the minimum-norm solution of $\mathcal{A}\Delta\boldsymbol{w} = \boldsymbol{f}$ without breakdown for all $\boldsymbol{f} \in \mathcal{R}(\mathcal{A})$ and for all $\Delta\boldsymbol{w}_0 \in \mathcal{R}(\mathcal{A}^\mathsf{T})$ [94, Theorem 5.5]. The inner-iteration preconditioning matrix $\mathcal{B}^{(\ell)}$ works on $\mathcal{A}$ in AB-GMRES as in Algorithm 3.3.3 [94, Algorithm 5.1].

ALGORITHM 3.3.3. AB-GMRES method preconditioned by inner iterations.

1: Let $\Delta\boldsymbol{w}_0 \in \mathbb{R}^n$ be the initial approximate solution, and $\boldsymbol{g}_0 := \boldsymbol{f} - \mathcal{A}\Delta\boldsymbol{w}_0$.
2: $\beta := \|\boldsymbol{g}_0\|_2, \ \boldsymbol{v}_1 := \boldsymbol{r}_0/\beta$
3: **for** $k = 1, 2, \ldots$ until convergence, **do**
4: Apply ℓ steps of a stationary iterative method to $\mathcal{A}\mathcal{A}^\mathsf{T}\boldsymbol{p} = \boldsymbol{v}_k, \ \boldsymbol{z} = \mathcal{A}^\mathsf{T}\boldsymbol{p}$ to obtain $\boldsymbol{z}_k := \mathcal{B}^{\langle\ell\rangle}\boldsymbol{v}_k$.
5: $\boldsymbol{u}_k := \mathcal{A}\boldsymbol{z}_k$
6: **for** $i = 1, 2, \ldots, k$, **do**
7: $h_{i,k} := (\boldsymbol{u}_k, \boldsymbol{v}_i), \ \boldsymbol{u}_k := \boldsymbol{u}_k - h_{i,k}\boldsymbol{v}_i$
8: **end for**
9: $h_{k+1,k} := \|\boldsymbol{u}_k\|_2, \ \boldsymbol{v}_{k+1} := \boldsymbol{u}_k/h_{k+1,k}$
10: **end for**
11: $\boldsymbol{p}_k := \arg\min_{\boldsymbol{p}\in R^k} \|\beta\boldsymbol{e}_1 - \bar{H}_k\boldsymbol{p}\|_2, \ \boldsymbol{q}_k = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k]\boldsymbol{p}_k$
12: Apply ℓ steps of a stationary iterative method to $\mathcal{A}\mathcal{A}^\mathsf{T}\boldsymbol{p} = \boldsymbol{q}_k, \ \boldsymbol{z} = \mathcal{A}^\mathsf{T}\boldsymbol{p}$ to obtain $\boldsymbol{z}' := \mathcal{B}^{\langle\ell\rangle}\boldsymbol{q}_k$.
13: $\Delta\boldsymbol{w}_k := \Delta\boldsymbol{w}_0 + \boldsymbol{z}'$

Here, $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k$ are orthonormal, $\boldsymbol{e}_1$ is the first column of the identity matrix, and $\bar{H}_k = \{h_{i,j}\} \in \mathbb{R}^{(k+1)\times k}$.

Note that the left-preconditioned generalized minimal residual method (BA-GMRES) [61, 93, 94] can be applied to solve the corrector stage problem, which can be written as the normal equations of the first kind

$$\mathcal{A}\mathcal{A}^\mathsf{T}\Delta\boldsymbol{y}_{\rm cc} = \mathcal{A}(SX)^{-1/2}\left(\Delta X_{\rm af}\Delta S_{\rm af}\boldsymbol{e} - \sigma\mu\boldsymbol{e}\right),$$

or equivalently

$$\min_{\Delta\boldsymbol{y}_{\rm cc}} \|\mathcal{A}^\mathsf{T}\Delta\boldsymbol{y}_{\rm cc} - (SX)^{-1/2}\left(\Delta X_{\rm af}\Delta S_{\rm af}\boldsymbol{e} - \sigma\mu\boldsymbol{e}\right)\|_2. \tag{3.16}$$

In fact, this formulation was adopted in [57] and solved by the CGLS method preconditioned by partial Cholesky decomposition that works in m-dimensional space. The BA-GMRES also works in m-dimensional space.

The advantage of the inner-iteration preconditioning methods is that we can avoid explicitly computing and storing the preconditioning matrices for $\mathcal{A}$ in (3.13).

We present efficient algorithms for specific inner iterations in the next section.

3.3.3 SSOR inner iterations for preconditioning the CGNE and MRNE methods

The inner-iteration preconditioned CGNE and MRNE methods require a symmetric preconditioning matrix. This is achieved by the SSOR inner-iteration preconditioning, which works on the normal equations of the second kind $\mathcal{A}\mathcal{A}^\mathsf{T}\boldsymbol{z} = \boldsymbol{g}$, $\boldsymbol{u} = \mathcal{A}^\mathsf{T}\boldsymbol{z}$, and its preconditioning matrix $C^{(\ell)}$ is SPD for ℓ odd for $\omega \in (0, 2)$ [92, Theorem 2.8]. This method exploits a symmetric splitting matrix by the forward updates, $i = 1, 2, \ldots, m$ in lines 3–6 in Algorithm 3.3.5 and the reverse updates, $i = m, m-1, \ldots, 1$, and can be efficiently implemented as the NE-SSOR method [103], [94, Algorithm D.8]. See [11] where SSOR preconditioning for CGNE with $\ell = 1$ is proposed. Let $\boldsymbol{\alpha}_i^\mathsf{T}$ be the ith row vector of $\mathcal{A}$. Algorithm 3.3.4 shows the NE-SSOR method.

ALGORITHM 3.3.4. NE-SSOR method.

1: Let $\boldsymbol{z}^{\langle 0\rangle} = \boldsymbol{0}$ and $\boldsymbol{u}^{\langle 0\rangle} = \boldsymbol{0}$.
2: **for** $k = 1, 2, \ldots, \ell$, **do**
3: **for** $i = 1, 2, \ldots, m$, **do**
4: $d_i^{\langle k-\frac{1}{2}\rangle} := \omega[g_i - (\boldsymbol{\alpha}_i, \boldsymbol{u}^{\langle k-1\rangle})]/\|\boldsymbol{\alpha}_i\|_2^2$
5: $z_i^{\langle k-\frac{1}{2}\rangle} := z_i^{\langle k-1\rangle} + d_i^{\langle k-\frac{1}{2}\rangle}, \boldsymbol{u}^{\langle k-1\rangle} := \boldsymbol{u}^{\langle k-1\rangle} + d_i^{\langle k-\frac{1}{2}\rangle}\boldsymbol{\alpha}_i$
6: **end for**
7: **for** $i = m, m-1, \ldots, 1$, **do**
8: $d_i^{\langle k\rangle} := \omega[g_i - (\boldsymbol{\alpha}_i, \boldsymbol{u}^{\langle k-1\rangle})]/\|\boldsymbol{\alpha}_i\|_2^2$
9: $z_i^{\langle k\rangle} := z_i^{\langle k-\frac{1}{2}\rangle} + d_i^{\langle k\rangle}, \boldsymbol{u}^{\langle k-1\rangle} := \boldsymbol{u}^{\langle k-1\rangle} + d_i^{\langle k\rangle}\boldsymbol{\alpha}_i$
10: **end for**
11: $\boldsymbol{u}^{\langle k\rangle} := \boldsymbol{u}^{\langle k-1\rangle}$
12: **end for**

When Algorithm 3.3.4 is applied to lines 2 and 6 of Algorithm 3.3.1 and lines 2 and 6 of Algorithm 3.3.2, the normal equations of the second kind are solved approximately.

3.3.4 SOR inner iterations for preconditioning the AB-GMRES method

Next, we introduce the successive overrelaxation (SOR) method applied to the normal equations of the second kind $\mathcal{A}\mathcal{A}^\mathsf{T}\boldsymbol{p} = \boldsymbol{g}$, $\boldsymbol{z} = \mathcal{A}^\mathsf{T}\boldsymbol{p}$ with $\boldsymbol{g} = \boldsymbol{v}_k$ or $\boldsymbol{q}_k$ as used in Algorithm 3.3.3. If the relaxation parameter ω satisfies $\omega \in (0, 2)$, then the iteration matrix H of this method is semiconvergent, i.e., $\lim_{i\to\infty} H^i$ exists [34]. An efficient algorithm for this method is called NE-SOR and is given as follows [103], [94, Algorithm D.7].

When Algorithm 3.3.5 is applied to lines 4 and 12 of Algorithm 3.3.3, the normal equations of the second kind are solved approximately.

ALGORITHM 3.3.5. NE-SOR method.

1: Let $\boldsymbol{z}^{\langle 0\rangle} = \boldsymbol{0}$.
2: **for** $k = 1, 2, \ldots, \ell$, **do**
3: **for** $i = 1, 2, \ldots, m$, **do**
4: $d_i^{\langle k\rangle} := \omega[g_i - (\boldsymbol{\alpha}_i,\ \boldsymbol{z}^{\langle k-1\rangle})]/\|\boldsymbol{\alpha}_i\|_2^2,\ \boldsymbol{z}^{\langle k-1\rangle} := \boldsymbol{z}^{\langle k-1\rangle} + d_i^{\langle k\rangle}\boldsymbol{\alpha}_i$
5: **end for**
6: $\boldsymbol{z}^{\langle k\rangle} := \boldsymbol{z}^{\langle k-1\rangle}$
7: **end for**

Since the rows of A are required in the NE-(S)SOR iterations, it would be more efficient if A is stored row-wise.

3.3.5 Row-scaling of $\mathcal{A}$

Let $\mathcal{D}$ be a diagonal matrix whose diagonal elements are positive. Then, problem (3.13) is equivalent to

$$\min \|\Delta \boldsymbol{w}\|_2 \quad \text{subject to} \quad \mathcal{D}^{-1}\mathcal{A}\Delta \boldsymbol{w} = \mathcal{D}^{-1}\boldsymbol{f}. \tag{3.17}$$

Denote $\hat{\mathcal{A}} := \mathcal{D}^{-1}\mathcal{A}$ and $\hat{f} := \mathcal{D}^{-1}\boldsymbol{f}$. Then, the scaled problem (3.17) is

$$\min \|\Delta \boldsymbol{w}\|_2 \quad \text{subject to} \quad \hat{\mathcal{A}}\Delta \boldsymbol{w} = \hat{\boldsymbol{f}}. \tag{3.18}$$

If $\hat{\mathcal{B}} \in \mathbb{R}^{n\times m}$ satisfies $\mathcal{R}(\hat{\mathcal{B}}) = \mathcal{R}(\hat{\mathcal{A}}^{\mathsf{T}})$, then (3.18) is equivalent to

$$\hat{\mathcal{A}}\hat{\mathcal{B}}\hat{\boldsymbol{z}} = \hat{\boldsymbol{f}}, \qquad \Delta \boldsymbol{w} = \hat{\mathcal{B}}\hat{\boldsymbol{z}} \tag{3.19}$$

for all $\hat{\boldsymbol{f}} \in \mathcal{R}(\hat{\mathcal{A}})$. The methods discussed earlier can be applied to (3.19). In the NE-(S)SOR inner iterations, one has to compute $\|\hat{\boldsymbol{\alpha}}_i\|_2$, the norm of the ith row of $\hat{\mathcal{A}}$. However, this can be omitted if the ith diagonal element of $\mathcal{D}$ is chosen as the norm of the ith row of $\mathcal{A}$, that is, $\mathcal{D}(i,i) := \|\boldsymbol{\alpha}_i\|_2,\ i = 1, \ldots, m$. With this choice, the matrix $\hat{\mathcal{A}}$ has unit row norm $\|\hat{\boldsymbol{\alpha}}_i\|_2 = 1,\ i = 1, \ldots, m$. Hence, we do not have to compute the norms $\|\hat{\boldsymbol{\alpha}}_i\|_2$ inside the NE-(S)SOR inner iterations if we compute the norms $\|\boldsymbol{\alpha}_i\|_2$ for the construction of the scaling matrix $\mathcal{D}$. The row-scaling scheme does not incur extra CPU time. We observe in the numerical results that this scheme improves the convergence of the Krylov subspace methods.

CGNE and MRNE preconditioned by inner iterations applied to a scaled linear system $\mathcal{D}^{-1}\mathcal{A}\Delta \boldsymbol{w} = \mathcal{D}^{-1}\boldsymbol{f}$ are equivalent to CG and MINRES applied to $\mathcal{D}^{-1}\mathcal{A}\mathcal{A}^{\mathsf{T}}C^{(\ell)}\mathcal{D}\boldsymbol{v} = \boldsymbol{f}$, $\Delta \boldsymbol{w} = \mathcal{A}^{\mathsf{T}}C^{(\ell)}\mathcal{D}\boldsymbol{v}$, respectively, and hence determine the minimum-norm solution of $\mathcal{A}\Delta \boldsymbol{w} = \boldsymbol{f}$ for all $\boldsymbol{f} \in \mathcal{R}(\mathcal{A})$ and for all $\Delta \boldsymbol{w}_0 \in \mathbb{R}^n$ if C^ℓ is SPD. Now we give conditions under which AB-GMRES preconditioned by inner iterations applied to a scaled linear system $\mathcal{D}^{-1}\mathcal{A}\Delta \boldsymbol{w} = \mathcal{D}^{-1}\boldsymbol{f}$ determines the minimum-norm solution of the unscaled one $\mathcal{A}\Delta \boldsymbol{w} = \boldsymbol{f}$.

Lemma 3.1. *If $\mathcal{R}(\mathcal{B}) = \mathcal{R}(\mathcal{A}^{\mathsf{T}})$ and $\mathcal{D} \in \mathbb{R}^{m\times m}$ is nonsingular, then AB-GMRES applied to $\mathcal{D}^{-1}\mathcal{A}\Delta \boldsymbol{w} = \mathcal{D}^{-1}\boldsymbol{f}$ determines the solution of $\min \|\Delta \boldsymbol{w}\|_2$, subject to*

$\mathcal{A}\Delta\boldsymbol{w} = \boldsymbol{f}$ without breakdown for all $\boldsymbol{f} \in \mathcal{R}(\mathcal{A})$ and for all $\Delta\boldsymbol{w}_0 \in \mathbb{R}^n$ if and only if $\mathcal{N}(\mathcal{B}) \cap \mathcal{R}(\mathcal{D}^{-1}\mathcal{A}) = \{\boldsymbol{0}\}$.

Proof. Since $\mathcal{R}(\mathcal{B}) = \mathcal{R}(\mathcal{A}^\mathsf{T})$ gives $\mathcal{R}(\mathcal{D}^{-1}\mathcal{A}\mathcal{B}) = \mathcal{R}(\mathcal{D}^{-1}\mathcal{A}\mathcal{A}^\mathsf{T}) = \mathcal{R}(\mathcal{D}^{-1}\mathcal{A})$, the equality $\min_{\boldsymbol{u}\in\mathbb{R}^m} \|\mathcal{D}^{-1}(\boldsymbol{f} - \mathcal{A}\mathcal{B}\boldsymbol{u})\|_2 = \min_{\Delta\boldsymbol{w}\in\mathbb{R}^n} \|\mathcal{D}^{-1}(\boldsymbol{f} - \mathcal{A}\Delta\boldsymbol{w})\|_2$ holds for all $\boldsymbol{f} \in \mathbb{R}^m$ [61, Theorem 3.1]. AB-GMRES applied to $\mathcal{D}^{-1}\mathcal{A}\Delta\boldsymbol{w} = \mathcal{D}^{-1}\boldsymbol{f}$ determines the kth iterate $\Delta\boldsymbol{w}_k$ by minimising $\|\mathcal{D}(\boldsymbol{f} - \mathcal{A}\Delta\boldsymbol{w})\|_2$ over the space $\Delta\boldsymbol{w}_0 + \mathcal{K}_k(\mathcal{D}^{-1}\mathcal{A}\mathcal{B}, \mathcal{D}^{-1}\boldsymbol{g}_0)$, and thus determines the solution of $\min \|\Delta\boldsymbol{w}\|_2$, subject to $\mathcal{D}^{-1}\mathcal{A}\Delta\boldsymbol{w} = \mathcal{D}^{-1}\boldsymbol{f}$ without breakdown for all $\boldsymbol{f} \in \mathcal{R}(\mathcal{A})$ and for all $\Delta\boldsymbol{w}_0 \in \mathbb{R}^n$ if and only if $\mathcal{N}(\mathcal{D}^{-1}\mathcal{A}\mathcal{B}) \cap \mathcal{R}(\mathcal{D}^{-1}\mathcal{A}\mathcal{B}) = \{\boldsymbol{0}\}$ [94, Theorem 5.2], which reduces to $\mathcal{R}(\mathcal{D}^{-1}\mathcal{A}) \cap \mathcal{N}(\mathcal{B}) = \{\boldsymbol{0}\}$ from $\mathcal{N}(\mathcal{D}^{-1}\mathcal{A}\mathcal{B}) = \mathcal{R}(\mathcal{B}^\mathsf{T}\mathcal{A}^\mathsf{T}\mathcal{D}^{-\mathsf{T}})^\perp = \mathcal{R}(\mathcal{B}^\mathsf{T}\mathcal{A}^\mathsf{T})^\perp = \mathcal{R}(\mathcal{B}^\mathsf{T}\mathcal{B})^\perp = \mathcal{R}(\mathcal{B}^\mathsf{T})^\perp = \mathcal{N}(\mathcal{B})$. □

Theorem 3.1. *If $\mathcal{D} \in \mathbb{R}^{m\times m}$ is nonsingular and the inner-iteration matrix is semiconvergent, then AB-GMRES preconditioned by the inner iterations applied to $\mathcal{D}^{-1}\mathcal{A}\Delta\boldsymbol{w} = \mathcal{D}^{-1}\boldsymbol{f}$ determines the solution of $\min \|\Delta\boldsymbol{w}\|_2$, subject to $\mathcal{A}\Delta\boldsymbol{w} = \boldsymbol{f}$ without breakdown for all $\boldsymbol{f} \in \mathcal{R}(\mathcal{A})$ and for all $\Delta\boldsymbol{w}_0 \in \mathbb{R}^n$.*

Proof. From Lemma 3.1, it is sufficient to show that $\mathcal{R}(\mathcal{B}) = \mathcal{R}(\mathcal{A}^\mathsf{T})$ and $\mathcal{N}(\mathcal{D}^{-1}\mathcal{A}\mathcal{B}) \cap \mathcal{R}(\mathcal{D}^{-1}\mathcal{A}\mathcal{B}) = \{\boldsymbol{0}\}$. Since $\mathcal{D}^{-1}M\mathcal{D}^{-\mathsf{T}} = \mathcal{D}^{-1}(\mathcal{A}\mathcal{A}^\mathsf{T} - N)\mathcal{D}^{-\mathsf{T}}$ is the splitting matrix of $\mathcal{D}^{-1}\mathcal{A}\mathcal{A}^\mathsf{T}\mathcal{D}^{-\mathsf{T}}$ for the inner iterations, the inner-iteration matrix is $\mathcal{D}^\mathsf{T}H\mathcal{D}^{-\mathsf{T}}$. Hence, the inner-iteration preconditioning matrix $\mathcal{B} = \mathcal{A}^\mathsf{T}C^{(\ell)}\mathcal{D}$ satisfies $\mathcal{R}(\mathcal{B}) = \mathcal{R}(\mathcal{A}^\mathsf{T})$ [94, Lemma 4.5]. On the other hand, $\mathcal{D}^{-1}\mathcal{A}\mathcal{B} = \mathcal{D}^{-1}M(I - H^\ell)(\mathcal{D}^{-1}M)^{-1}$ satisfies $\mathcal{N}(\mathcal{D}^{-1}\mathcal{A}\mathcal{B}) \cap \mathcal{R}(\mathcal{D}^{-1}\mathcal{A}\mathcal{B}) = \{\boldsymbol{0}\}$ [94, Lemmas 4.3, 4.4]. □

3.4 Numerical experiments

In this section, we compare the performance of the interior-point method based on the iterative solvers with the standard interior-point softwares. We also developed an efficient direct solver coded in C to compare with the iterative solvers. For the sake of completeness, we briefly describe our direct solver first.

3.4.1 Direct solver for the normal equations

To deal with the rank-deficiency, we used a strategy that is similar to the Cholesky-Infinity modification scheme introduced in the LIPSOL solver [116]. However, instead of penalizing the elements that are close to zero, we removed them and solved the reduced system. We implemented this modification by an LDLT decomposition as explained below.

The generic sparse Cholesky solver breaks down once a numerically zero pivot is encountered. The Cholesky decomposition of $\mathcal{A}\mathcal{A}^\mathsf{T}$ is

$$\mathcal{A}\mathcal{A}^\mathsf{T} = LL^\mathsf{T}, \tag{3.20}$$

where L is the lower triangular matrix. The breakdown happens when the diagonal entries of the L contain numerically zero elements. To describe this, consider a 2×2

block case. Mathematically, breakdown happens when

$$L = \begin{bmatrix} R_1 & \mathbf{0} \\ R_2 & \mathbf{0} \end{bmatrix}, \tag{3.21}$$

where $R_1 \in \mathbb{R}^{r\times r}$ is a lower triangular and nonsingular matrix, r corresponds to the rank of $\mathcal{A}$, $R_2 \in \mathbb{R}^{(m-r)\times r}$. Recall that the problem to be solved is $\mathcal{A}\mathcal{A}^\mathsf{T}\Delta\boldsymbol{y} = \boldsymbol{f}$. Without loss of generality, assume that the first r rows of $\mathcal{A}$ is linearly independent and let the partitions of $\Delta\boldsymbol{y}$ and $\boldsymbol{f}$ be

$$\Delta\boldsymbol{y} = \begin{bmatrix} \Delta\boldsymbol{y}_1 \\ \Delta\boldsymbol{y}_2 \end{bmatrix}, \qquad \boldsymbol{f} = \begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{bmatrix}, \tag{3.22}$$

respectively, such that $\Delta\boldsymbol{y}_1 \in \mathbb{R}^r$, $\boldsymbol{f}_1 \in \mathbb{R}^r$, $\Delta\boldsymbol{y}_2 \in \mathbb{R}^{m-r}$ and $\boldsymbol{f}_2 \in \mathbb{R}^{m-r}$. Then the problem becomes

$$LL^\mathsf{T}\Delta\boldsymbol{y} = \boldsymbol{f}, \tag{3.23a}$$

$$\begin{bmatrix} R_1 & \mathbf{0} \\ R_2 & \mathbf{0} \end{bmatrix} \begin{bmatrix} R_1^\mathsf{T} & R_2^\mathsf{T} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{y}_1 \\ \Delta\boldsymbol{y}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{bmatrix}, \tag{3.23b}$$

$$\begin{bmatrix} R_1R_1^\mathsf{T} & R_1R_2^\mathsf{T} \\ R_2R_1^\mathsf{T} & R_2R_2^\mathsf{T} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{y}_1 \\ \Delta\boldsymbol{y}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{bmatrix}, \tag{3.23c}$$

which is equivalent to

$$R_1(R_1^\mathsf{T}\Delta\boldsymbol{y}_1 + R_2^\mathsf{T}\Delta\boldsymbol{y}_2) = \boldsymbol{f}_1, \tag{3.24a}$$

$$R_2(R_1^\mathsf{T}\Delta\boldsymbol{y}_1 + R_2^\mathsf{T}\Delta\boldsymbol{y}_2) = \boldsymbol{f}_2. \tag{3.24b}$$

Then, the solution $\Delta\boldsymbol{y}$ is given by the r linear independent equations (3.24a). By setting $\Delta\boldsymbol{y}_2 = \mathbf{0}$, we have

$$\begin{bmatrix} \Delta\boldsymbol{y}_1 \\ \Delta\boldsymbol{y}_2 \end{bmatrix} =: \Delta\boldsymbol{y} = \begin{bmatrix} (R_1R_1^\mathsf{T})^{-1}\boldsymbol{f}_1 \\ \mathbf{0} \end{bmatrix}. \tag{3.25}$$

Thus, the idea to modify the sparse Cholesky solver is to set the degenerate part of $\Delta\boldsymbol{y}$ (the partition of $\Delta\boldsymbol{y}$ that corresponds to linearly dependent rows of $\mathcal{A}$) to be zero and then solve for the rest. We implement this modification by an LDLT decomposition, since the generic sparse Cholesky solver breaks down once a numerically zero pivot is encountered. Some indefinite matrices for which no Cholesky decomposition exists have an LDLT decomposition with negative or zero entries in D.

We explain the implementation by an example where $\mathcal{A}\mathcal{A}^\mathsf{T} \in \mathbb{R}^{3\times 3}$. For matrix $\mathcal{A}\mathcal{A}^\mathsf{T}$, LDLT decomposition gives

$$\mathcal{A}\mathcal{A}^\mathsf{T} = LDL^\mathsf{T} \tag{3.26a}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix} \begin{bmatrix} 1 & l_{21} & l_{31} \\ 0 & 1 & l_{32} \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.26b}$$

Correspondingly, we partition the $\Delta \boldsymbol{y}$ and $\boldsymbol{f}$ as

$$\Delta \boldsymbol{y} = \begin{bmatrix} \Delta y_1 \\ \Delta y_2 \\ \Delta y_3 \end{bmatrix}, \qquad \boldsymbol{f} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}. \tag{3.27}$$

Assuming that the diagonal element d_2 is close to zero, we let the corresponding $\Delta y_2 = 0$, and delete f_2 and the corresponding row and column in the decomposition $\left\{L, D, L^{\mathsf{T}}\right\}$. Then, we set

$$\tilde{L} = \begin{bmatrix} 1 & 0 \\ l_{31} & 1 \end{bmatrix}, \qquad \tilde{D} = \begin{bmatrix} d_1 & 0 \\ 0 & d_3 \end{bmatrix}, \qquad \tilde{\boldsymbol{f}} = \begin{bmatrix} f_1 \\ f_3 \end{bmatrix}, \qquad \tilde{\Delta \boldsymbol{y}} = \begin{bmatrix} \Delta y_1 \\ \Delta y_3 \end{bmatrix}. \tag{3.28}$$

and solve

$$\tilde{L}\tilde{D}^{1/2}\left((\tilde{L}\tilde{D}^{1/2})^{\mathsf{T}}\tilde{\Delta \boldsymbol{y}}\right) = \tilde{\boldsymbol{f}}, \tag{3.29}$$

using forward and backward substitution. The solution is then given by

$$\Delta \boldsymbol{y} = \begin{bmatrix} \Delta y_1 \\ 0 \\ \Delta y_3 \end{bmatrix}. \tag{3.30}$$

Concretely, we used the MATLAB built-in function `chol` to detect whether the matrix is symmetric positive definite. We used the `ldlchol` from CSPARSE package version 3.1.0 [32] when the matrix was symmetric positive definite, and we turned to the MATLAB built-in solver `ldl` for the semidefinite cases which uses MA57 [38].

3.4.2 Implementation specifications

In this section, we describe our numerical experiments.

The initial solution for the interior-point method was set using the method described in LIPSOL solver [116]. The initial solution for the Krylov subspace iterations and the inner iterations was set to zero.

We set the maximum number of the interior-point iterations as 99 and the stopping criterion regarding the error measure as

$$\Gamma \leq \epsilon_{\text{out}} = 10^{-8}, \quad \Gamma := \max\left\{\mu^{(k)}, \frac{\|\boldsymbol{b} - A\boldsymbol{x}^{(k)}\|_2}{\max\{\|\boldsymbol{b}\|_2, 1\}}, \frac{\|\boldsymbol{c} - \boldsymbol{s}^{(k)} - A^{\mathsf{T}}\boldsymbol{y}^{(k)}\|_2}{\max\{\|\boldsymbol{c}\|_2, 1\}}\right\}. \tag{3.31}$$

For the iterative solver for the linear system (3.13), we set the maximum number of iterations for CGNE, MRNE and AB-GMRES as m, and relaxed it to a larger number for some difficult problems for CGNE and MRNE. We set the stopping

criterion for the scaled residual as

$$\|\hat{\boldsymbol{f}} - \hat{\mathcal{A}}\Delta\boldsymbol{w}^{(k)}\|_2 \leq \epsilon_{\text{in}}\|\hat{\boldsymbol{f}}\|_2,$$

where ϵ_{in} is initially 10^{-6} and is kept in the range $[10^{-14}, 10^{-4}]$ during the process. We adjusted ϵ_{in} according to the progress of the interior-point iterations. We truncated the iterative solving prematurely in the early interior-point iterations, and pursued a more precise direction as the LP solution was approached. The progress was measured by the error measure Γ. Concretely, we adjusted ϵ_{in} as

$$\epsilon_{\text{in}}^{(k)} = \begin{cases} \epsilon_{\text{in}}^{(k-1)} \times 0.75 & \text{if } \log_{10}\Gamma^{(k)} \in (-3, 1], \\ \epsilon_{\text{in}}^{(k-1)} \times 0.375 & \text{if } \log_{10}\Gamma^{(k)} \in (-\infty, -3]. \end{cases}$$

For steps where iterative solvers failed to converge within the maximum number of iterations, we slightly increased the value of ϵ_{in} by multiplying by 1.5.

We adopt the implementation of AB-GMRES preconditioned by NE-SOR inner-iterations [71] with the additional row-scaling scheme (Section 3.3.5). No restarts were used for the AB-GMRES method. The non-breakdown conditions discussed in Sections 3.3.1 and 3.3.2 are satisfied.

For the direct solver, the tolerance for dropping pivot elements close to zero was 10^{-16} for most of the problems; for some problems this tolerance has to be increased to 10^{-6} to overcome breakdown.

The experiment was conducted on a MacBook Pro with a 2.6 GHz Intel Core i5 processor with 8 GB of random-access memory, OS X El Capitan version 10.11.2. The interior-point method was coded in MATLAB R2014b and the iterative solvers including AB-GMRES (NE-SOR), CGNE (NE-SSOR), and MRNE (NE-SSOR), were coded in C and compiled as MATLAB Executable (MEX) files accelerated with Basic Linear Algebra Subprograms (BLAS).

We compared our implementation with the standard solvers available in CVX [60, 59]: SDPT3 version 4.0 [110, 111], SeDuMi version 1.34 [108], and MOSEK version 7.1.0.12 [95], with the default interior-point stopping criterion (3.31). Note that SDPT3 and SeDuMi are non-commercial public-domain solvers, whereas MOSEK is a commercial solver known as one of the state-of-the-art solvers. These solvers were implemented with the CVX MATLAB interface, and we recorded the CPU time reported in the screen output of each solver. However, it usually took a longer time for the CVX to finish the whole process. The larger the problem was, the more apparent this extra CPU time became. For example, for problem ken_18, the screen output of SeDuMi was 765.3 seconds while the total processing time was 7,615.2 seconds.

3.4.3 Typical LP problems: sparse and full-rank problems

We tested 125 typical LP problems from the NETLIB, QAPLIB and MITTELMANN collections in [33]. Most problems usually have sparse and full-rank constraint matrix

TABLE 3.1: Overall performance of the solvers on 125 testing problems.

Status	Solved	Solved†	Unsolved	Expensive
AB-GMRES (NE-SOR)	123	0	0	2
CGNE (NE-SSOR)	124	1	0	0
MRNE (NE-SSOR)	125	0	0	0
Modified Cholesky	117	2	6	0
SDPT3	76	19	25	5
SeDuMi	103	16	6	0
MOSEK	125	0	0	0

A (except problems `bore3d` and `cycle`). For the problems with $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$, $\boldsymbol{l} \neq \boldsymbol{0}$, $\boldsymbol{u} \neq \boldsymbol{\infty}$, we transform them using the approach in LIPSOL [116].

The overall numerical experience is summarised in Table 3.1. MRNE (NE-SSOR) and MOSEK were most stable in the sense that they solved all 125 problems. CGNE (NE-SSOR) method solved all problems except for the largest QAPLIB problem, which was solved to a slightly larger tolerance level of 10^{-7}. AB-GMRES (NE-SOR) was also very stable and solved the problems accurately enough. However, it took longer than 20 hours for two problems that have 154,699 and 23,541 unknowns, respectively, although it succeeded in solving larger problems such as `pds-80`. The other solvers were less stable. The modified Cholesky solver solved only 93% of the problems, although it was fast for the problems that it could successfully solve. SDPT3 solved 61% and SeDuMi 82% of the problems. Here we should mention that SeDuMi and SDPT3 are designed for LP, semidefinite programming (SDP), and second-order cone programming (SOCP), while our code is (currently) tuned solely for LP.

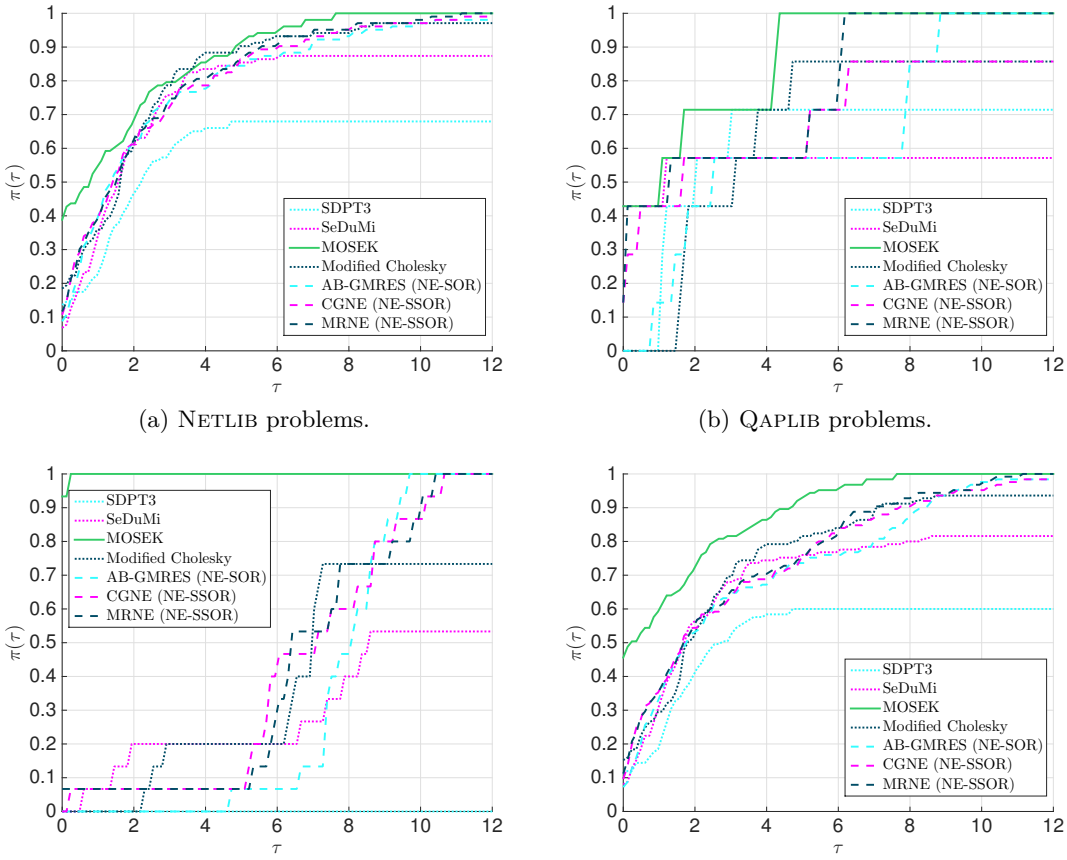
Note that MOSEK solver uses a multi-corrector interior-point method [54] while our implementation is a single corrector (i.e., predictor-corrector) method. This led to different numbers of interior-point iterations as given in the tables. Thus, there is still room for improvement in the efficiency of our solver based on iterative solvers if a more elaborately tuned interior-point framework such as the one in MOSEK is adopted.

In order to show the trends of performance, we use the Dolan-Moré performance profiles [37] in Figs. 3.1–3.2, with $\pi(\tau) := P(\log_2 r_{ps} \leq \tau)$ the proportion of problems for which $\log_2$-scaled performance ratio is at most τ, where $r_{ps} := t_{ps}/t_{\mathrm{p}}^*$, t_{ps} is the CPU time for solver s to solve problem p, and t_{p}^* is the minimal CPU time for problem p. The comparison indicates that the iterative solvers, although slower than the commercial solver MOSEK in some cases, were often able to solve the problems to the designated accuracy.

In Tables 3.2–3.4, we give the following information:

1. the name of the problem and the size (m, n) of the constraint matrix,

2. the number of interior-point iterations required for convergence,



(a) NETLIB problems. (b) QAPLIB problems.

(c) MITTELMANN problems. (d) All the problems.

FIG. 3.1: Dolan-Moré profiles comparing the CPU time costs for the proposed solvers, public-domain and commercial solvers.

3. CPU time for the entire computation in seconds. For the cases shorter than 3,000 seconds, CPU time is taken as an average over 10 measurements. In each row, we indicate in red boldface and blue underline the fastest and second fastest solvers in CPU time, respectively.

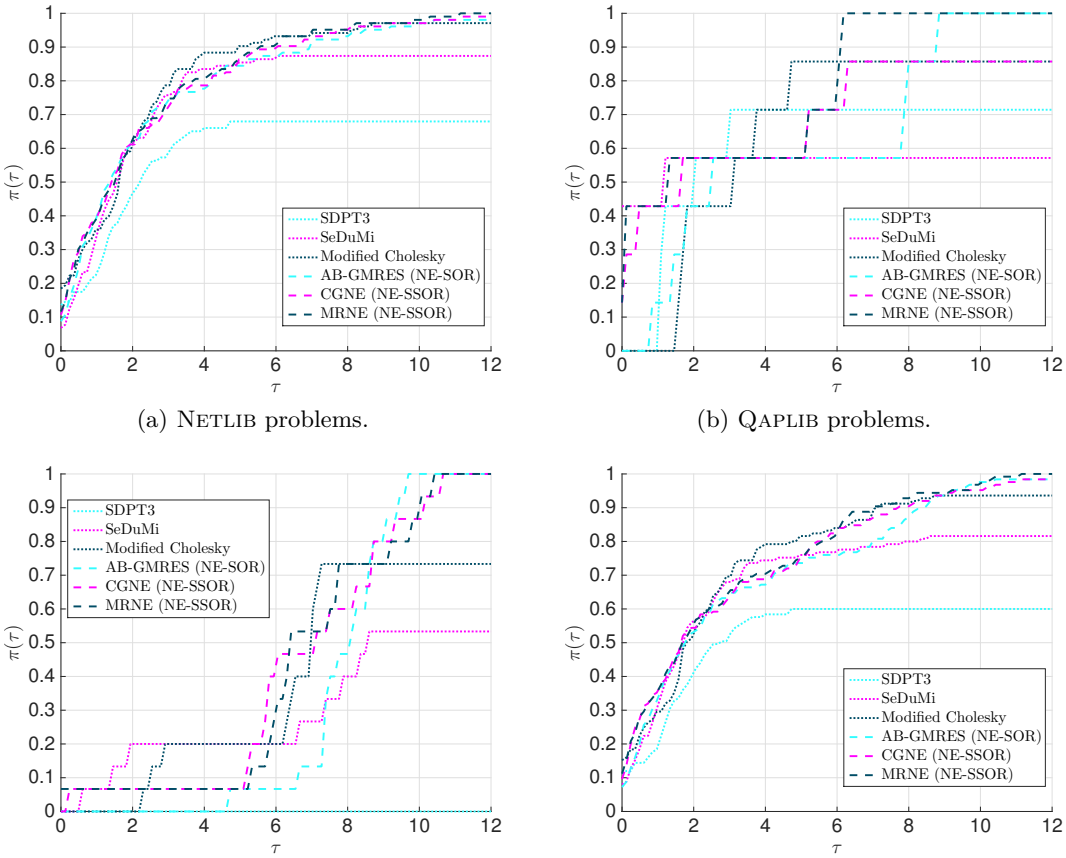
Besides the statistics, we also use the following notation:

† inaccurately solved, i.e., the value of ϵ_{out} was relaxed to a larger level. For our solvers, we provide extra information at the stopping point: $\dagger_a$, $a = \lfloor \log_{10} \Gamma \rfloor$ in the iter column, and $\dagger_b$, $b = \lfloor \log_{10} \kappa(\mathcal{A}\mathcal{A}^{\mathsf{T}}) \rfloor$ in the time column, where $\lfloor \cdot \rfloor$ is the floor function; the CVX solvers do not provide the condition number but only the relative duality gap,

\- the iterations diverged due to numerical instabilities,

$\diamond$ the iterations took longer than 20 hours.

Note that all zero rows and columns of the constraint matrix A were removed beforehand. The problems marked with # are with rank-deficient A even after this preprocessing. For these problems we put rank(A) in brackets after m, which is computed using the MATLAB function `sprank`.



(a) NETLIB problems. (b) QAPLIB problems.

(c) MITTELMANN problems. (d) All the problems.

FIG. 3.2: Dolan-Moré profiles comparing the CPU time costs for the proposed solvers and public-domain solvers.

In order to give an idea of the typical differences between methods, we present the interior-point convergence curves for problem ken_13. The problem has a constraint matrix $A \in \mathbb{R}^{28,632 \times 42,659}$ with full row rank and $97,246$ nonzero elements.

Different aspects of the performance of the four solvers are displayed in Fig. 3.3. The red dotted line with diamond markers represents the quantity related to AB-GMRES (NE-SOR), the blue with downward-pointing triangle CGNE (NE-SSOR), the yellow with asterisk MRNE (NE-SSOR), and the dark green with plus sign the modified Cholesky solver. Note that for this problem ken_13, the modified Cholesky solver became numerically inaccurate at the last step and it broke down if the default dropping tolerance was used. Thus, we increased it to 10^{-6}.

Fig. 3.3a shows $\kappa(\mathcal{A}\mathcal{A}^{\mathsf{T}})$ in $\log_{10}$ scale. It verifies the claim that the least squares problem becomes increasingly ill-conditioned at the final steps in the interior-point process: $\kappa(\mathcal{A}\mathcal{A}^{\mathsf{T}})$ started from around 10^{20} and increased to 10^{80} at the last 3-5 steps. Fig. 3.3b shows the convergence curve of the duality measure μ in $\log_{10}$ scale. The μ drops below the tolerance and the stopping criterion is satisfied. Although it is not shown in the figure, we found that the interior-point method with modified Cholesky with the default value of the dropping tolerance 10^{-16} stagnated for $\mu \simeq 10^{-4}$. Comparing with Fig. 3.3a, it is observed that the solvers started to behave differently

TABLE 3.2: Experiments on NETLIB problems. In each row, red boldface and blue underline denote the fastest and second fastest solvers in CPU time (seconds), respectively.

			Inner-outer iteration solvers						Direct solver		Public-domain solvers				Commercial solver	
			AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		SDPT3		SeDuMi		MOSEK	
Problem	m	n	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
25fv47	821	1,876	25	4.62	25	5.00	25	4.60	25	3.67	59	2.50	29	**2.30**	26	3.90
adlittle	56	138	12	**0.03**	13	**0.03**	13	0.05	12	0.09	16	0.16	14	0.10	10	1.98
afiro	27	51	8	0.02	8	**0.01**	8	**0.01**	8	0.03	11	0.11	7	0.10	9	1.91
agg	488	615	21	0.72	21	0.88	24	0.79	21	1.49	34	**0.61**	32	0.90	18	2.24
agg2	516	758	21	0.64	21	0.56	23	**0.53**	21	1.55	32	1.28	23	1.00	13	2.12
agg3	516	758	19	0.68	19	**0.52**	21	0.58	19	1.38	32	1.24	22	1.10	12	2.06
bandm	305	472	18	0.73	19	0.62	19	0.74	17	0.90	42	1.52	20	**0.50**	15	2.17
beaconfd	173	295	13	**0.07**	13	**0.07**	13	**0.07**	12	0.41	15	0.22	10	0.20	8	1.97
blend	74	114	12	**0.06**	14	0.07	13	0.08	12	0.11	15	0.16	11	0.10	9	1.98
bnl1	643	1,586	25	2.53	25	4.66	25	4.92	25	**1.95**	$\dagger_{-5}$	†	64	2.50	20	2.51
bnl2	2,324	4,486	32	44.98	32	23.37	32	27.63	32	12.41	$\dagger_{-4}$	†	38	5.80	25	**2.66**
bore3d#	233 (232)	334	19	0.35	19	0.23	19	**0.21**	19	0.63	35	1.92	18	1.50	19	3.00
brandy	220	303	17	0.43	18	0.86	18	0.86	17	0.59	46	1.02	19	**0.40**	12	2.04
capri	271	482	19	**0.80**	19	0.88	19	0.91	19	1.04	47	3.22	33	1.60	14	2.63
cre_a	3,516	7,248	30	186.77	30	48.43	31	35.79	31	105.60	$\dagger_{-7}$	†	28	8.70	20	**2.69**
cre_b	9,648	77,137	43	787.95	42	611.11	42	455.04	53	1,143.90	$\dagger_{-6}$	†	$\dagger_{-7}$	†	19	**3.63**
cre_c	3,068	6,411	30	268.84	32	47.92	33	46.12	33	79.67	-	-	28	7.70	17	**2.56**
cre_d	8,926	73,948	37	387.17	37	316.81	37	213.69	37	847.00	-	-	34	42.10	16	**3.06**
cycle#	1,903 (1,875)	3,371	30	61.87	31	50.44	61	185.12	-	-	$\dagger_{-6}$	†	30	5.30	20	**2.76**
czprob	929	3,562	39	**1.51**	38	1.60	39	1.73	39	10.45	$\dagger_{-5}$	†	39	2.80	27	2.91
d2q06c	2,171	5,831	32	132.75	33	581.83	36	750.06	32	24.09	84	6.43	29	4.10	21	**2.85**
d6cube	415	6,184	23	3.77	24	7.41	23	7.12	26	2.68	34	**1.65**	-	-	11	2.50
degen2	444	757	15	1.26	16	1.13	16	1.18	21	2.27	17	0.41	13	**0.40**	8	2.12
degen3	1,503	2,604	19	27.30	21	13.26	21	13.38	19	27.52	$\dagger_{-6}$	†	15	**2.00**	12	2.18
dfl001	6,071	12,230	48	4,336.35	50	2,044.54	55	2,205.16	91	3,131.77	-	-	$\dagger_{-5}$	†	22	**7.46**
e226	223	472	21	0.64	20	0.61	21	0.82	20	**0.59**	61	1.17	22	0.60	14	1.97

TABLE 3.2: (cont.) Experiments on NETLIB problems. In each row, red boldface and blue underline denote the fastest and second fastest solvers in CPU time (seconds), respectively.

			Inner-outer iteration solvers						Direct solver		Public-domain solvers				Commercial solver	
			AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		SDPT3		SeDuMi		MOSEK	
Problem	m	n	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
etamacro	400	816	30	**1.23**	31	1.58	31	1.43	30	2.30	-	-	30	2.80	20	2.82
fffff800	524	1,028	32	4.11	30	6.29	33	6.39	32	3.31	44	**0.86**	46	1.60	22	2.55
fit1d	24	1,049	21	0.78	21	0.45	21	0.49	19	**0.38**	36	2.11	18	0.80	13	0.67
fit1p	627	1,677	16	4.01	16	5.31	16	5.14	16	3.56	25	1.78	53	2.00	17	**0.73**
fit2d	25	10,524	20	3.40	21	3.54	21	3.72	20	2.40	41	3.10	15	2.60	18	**0.79**
fit2p	3,000	13,525	19	1,103.13	32	1,755.13	32	1,831.13	19	102.02	27	3.69	40	8.90	17	**0.82**
ganges	1,309	1,706	18	8.21	18	27.73	21	33.06	18	3.80	22	**0.90**	26	1.60	15	0.91
gfrd_pnc	616	1,160	21	1.15	22	1.04	21	0.88	21	0.98	27	**0.85**	20	1.00	29	0.90
grow15	300	645	19	0.43	19	**0.35**	20	0.37	17	0.40	21	0.80	25	1.00	13	0.89
grow22	440	946	20	0.68	20	0.59	22	0.59	18	**0.53**	22	0.93	26	1.40	14	0.95
grow7	140	301	18	**0.12**	18	0.16	18	**0.12**	16	0.16	19	0.66	19	0.70	12	0.69
israel	174	316	24	0.99	27	0.94	27	1.06	25	1.12	34	**0.51**	20	0.60	15	2.14
kb2	43	68	16	0.09	17	**0.08**	17	**0.08**	15	0.11	26	0.71	15	0.50	16	0.75
ken_07	2,426	3,602	17	4.14	18	2.39	17	2.24	16	1.07	33	1.74	18	1.80	15	**0.79**
ken_11	14,694	21,349	22	636.24	23	123.23	23	85.95	22	7.83	$\dagger_{-4}$	†	38	10.60	31	**1.87**
ken_13	28,632	42,659	27	2,633.00	28	365.15	29	348.51	27	23.90	-	-	43	29.50	20	**2.83**
ken_18	105,127	154,699	◇	◇	38	12,893.63	46	21,315.47	38	324.89	-	-	59	765.30	20	**24.98**
lotfi	153	366	16	0.28	16	**0.24**	16	0.32	16	0.39	37	1.14	20	1.20	15	2.47
maros_r7	3,136	9,408	15	57.78	15	29.69	15	31.68	15	11.14	21	5.39	15	4.80	12	**3.29**
modszk1	687	1,620	23	2.70	23	3.60	23	3.48	22	2.54	29	**0.85**	23	1.00	22	0.92
osa_07	1,118	25,067	34	12.35	32	6.26	36	8.51	27	5.85	31	3.90	31	4.90	14	**2.55**
osa_14	2,337	54,797	38	11.41	32	9.11	37	11.81	37	16.07	37	7.65	36	7.30	18	**3.03**
osa_30	4,350	104,374	39	22.69	41	19.08	38	17.16	36	28.98	37	12.49	40	11.50	17	**3.36**
osa_60	10,280	243,246	30	48.25	40	40.12	33	37.26	34	67.90	39	26.73	41	21.70	17	**5.10**
pds_02	2,953	7,716	29	4.43	29	3.43	29	4.16	29	3.16	$\dagger_{-5}$	†	30	6.90	18	**0.82**
pds_06	9,881	29,351	48	49.77	48	44.17	51	45.85	48	44.65	-	-	51	61.50	23	**1.45**

TABLE 3.2: (cont.) Experiments on NETLIB problems. In each row, red boldface and blue underline denote the fastest and second fastest solvers in CPU time (seconds), respectively.

			Inner-outer iteration solvers						Direct solver		Public-domain solvers				Commercial solver	
			AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		SDPT3		SeDuMi		MOSEK	
Problem	m	n	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
pds_10	16,558	49,932	51	91.60	52	87.75	50	79.22	52	130.17	-	-	74	157.20	28	**2.54**
pds_20	33,874	108,175	61	1,365.98	64	1,155.95	62	683.72	62	665.05	-	-	†$_{-7}$	†	34	**11.02**
perold	625	1,506	36	4.71	36	6.71	36	6.97	37	2.82	†$_{-6}$	†	†$_{-7}$	†	24	**0.87**
pilot	1,441	4,860	33	31.54	33	51.15	33	49.36	33	16.18	-	-	81	19.70	39	**1.73**
pilot4	410	1,123	30	2.11	30	2.12	30	2.29	30	2.26	†$_{-7}$	†	-	-	27	**0.78**
pilot87	2,030	6,680	39	55.59	39	105.77	39	102.58	39	33.13	88	11.54	76	12.60	38	**2.45**
pilot_ja	940	2,267	35	13.02	37	19.51	36	14.79	37	4.84	-	-	-	-	29	**0.71**
pilot_we	722	2,928	35	5.67	39	8.58	38	7.62	35	2.42	†$_{-7}$	†	44	4.90	31	**0.71**
pilotnov	975	2,446	24	5.70	25	5.02	27	4.07	22	2.90	-	-	-	-	17	**0.73**
qap12	3,192	8,856	19	758.92	21	144.74	20	99.35	19	50.45	26	21.78	†$_{-7}$	†	17	**6.09**
qap15	6,330	22,275	23	5,530.52	25	789.81	24	581.25	24	335.83	52	330.31	†$_{-7}$	†	17	**21.11**
qap8	912	1,632	11	1.73	12	1.09	11	**0.98**	10	2.75	13	1.25	8	1.10	7	2.16
sc105	105	163	10	0.05	10	0.04	10	0.04	10	**0.02**	20	0.50	10	0.20	8	2.13
sc205	205	317	11	0.17	11	0.09	11	0.07	10	**0.05**	18	0.61	12	0.30	10	2.16
sc50a	50	78	10	0.03	10	**0.00**	6	0.02	10	0.02	12	0.17	8	0.20	8	2.13
sc50b	50	78	7	**0.01**	7	0.02	7	0.02	7	0.03	11	0.26	7	0.20	6	1.94
scagr25	471	671	18	0.93	18	0.69	18	0.71	17	**0.20**	35	0.84	21	0.70	21	2.63
scagr7	129	185	14	0.15	15	0.11	15	0.11	14	**0.07**	33	0.71	17	0.50	19	2.52
scfxm1	330	600	18	1.03	19	1.05	20	1.14	18	**0.70**	52	1.40	20	0.80	15	2.42
scfxm2	660	1,200	21	2.44	22	4.73	23	4.71	21	1.35	58	1.59	24	**1.30**	18	2.56
scfxm3	990	1,800	22	5.94	23	12.64	24	12.10	22	1.64	59	1.79	25	**1.50**	16	2.53
scorpion	388	466	15	0.28	16	0.23	16	0.26	15	**0.20**	17	0.39	11	0.30	11	2.21
scrs8	490	1,275	25	0.91	26	0.78	25	0.77	25	**0.61**	37	1.06	35	1.70	14	2.41
scsd1	77	760	9	0.06	9	0.05	9	**0.03**	9	0.04	12	0.23	8	0.20	8	2.02
scsd6	147	1,350	11	0.17	12	0.12	11	0.13	11	**0.07**	15	0.32	11	0.40	10	2.06
scsd8	397	2,750	12	0.76	12	0.71	12	0.64	11	**0.16**	13	0.32	10	0.60	7	1.93

TABLE 3.2: (cont.) Experiments on NETLIB problems. In each row, red boldface and blue underline denote the fastest and second fastest solvers in CPU time (seconds), respectively.

			Inner-outer iteration solvers						Direct solver		Public-domain solvers				Commercial solver	
			AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		SDPT3		SeDuMi		MOSEK	
Problem	m	n	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
sctap1	300	660	17	0.31	19	0.38	19	0.36	17	**0.12**	20	0.46	20	0.50	11	2.15
sctap2	1,090	2,500	20	1.36	20	1.21	21	1.04	19	1.75	21	**0.48**	12	0.60	9	2.05
sctap3	1,480	3,340	19	1.33	19	1.14	20	1.22	18	2.31	23	0.94	13	**0.40**	9	2.11
share1b	117	253	23	0.50	24	0.51	24	0.48	23	**0.16**	27	0.52	22	0.50	23	2.74
share2b	96	162	12	0.16	14	0.20	16	0.21	12	**0.09**	26	0.60	19	0.30	15	2.47
shell	536	1,777	19	0.61	19	0.57	19	0.58	19	1.68	-	-	31	1.10	22	**0.56**
ship04l	402	2,166	14	**0.26**	14	**0.26**	14	**0.26**	15	1.00	20	0.74	17	0.80	10	1.86
ship04s	402	1,506	15	0.78	15	0.30	15	**0.21**	14	1.14	20	0.67	17	0.70	11	0.48
ship08l	778	4,363	16	0.82	17	1.33	17	1.28	16	2.47	21	**0.51**	18	0.90	11	1.93
ship08s	778	2,467	15	0.44	16	0.46	16	0.60	15	1.82	20	**0.32**	16	0.40	10	1.88
ship12l	1,151	5,533	20	1.48	19	2.21	20	2.01	19	4.66	22	**0.65**	23	1.90	14	2.04
ship12s	1,151	5,533	17	0.90	19	1.00	19	0.94	17	2.66	22	**0.41**	17	0.90	12	1.96
sierra	1,227	2,735	17	1.28	19	1.37	19	1.05	21	1.60	-	-	29	3.50	16	**0.59**
stair	356	614	22	1.43	22	1.63	22	1.87	22	0.96	$\dagger_{-6}$	†	18	0.70	15	**0.52**
standata	359	1,274	18	0.63	17	**0.34**	17	0.38	17	0.86	-	-	19	0.70	9	0.48
standgub	361	1,383	17	0.35	17	**0.30**	17	0.37	17	0.91	-	-	19	0.70	9	0.51
standmps	467	1,274	25	0.81	24	0.68	25	0.82	24	1.71	-	-	15	0.70	17	**0.53**
stocfor1	117	165	19	0.13	21	0.13	20	0.20	19	**0.09**	30	0.71	17	0.50	11	2.21
stocfor2	2,157	3,045	23	37.36	24	18.00	24	17.59	21	13.43	53	**1.95**	$\dagger_{-4}$	†	17	2.54
stocfor3	16,675	23,541	◇	◇	38	4,590.71	37	4,071.37	$\dagger_{-7}$	$\dagger_{32}$	80	11.05	-	-	26	**3.37**
truss	1,000	8,806	19	6.62	21	10.22	22	10.59	19	3.29	21	**1.12**	19	1.90	12	2.27
tuff	333	628	21	1.63	22	1.27	24	2.03	21	1.39	$\dagger_{-7}$	†	21	0.80	18	**0.60**
vtp_base	198	346	24	0.69	24	**0.52**	24	0.61	24	0.77	39	1.26	42	1.30	12	0.69
wood1p	244	2,595	17	1.75	17	1.34	17	**1.19**	-	-	38	2.75	19	2.00	10	2.17
woodw	1,098	8,418	25	5.12	27	6.72	28	7.34	22	3.73	-	-	33	3.20	17	**2.47**

TABLE 3.3: Experiments on QAPLIB problems. In each row, red boldface and blue underline denote the fastest and second fastest solvers in CPU time (seconds), respectively.

			Inner-outer iteration solvers						Direct solver		Public-domain solvers				Commercial solver	
			AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		SDPT3		SeDuMi		MOSEK	
Problem	m	n	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
nug05	201	225	7	0.16	7	**0.09**	7	**0.09**	7	0.27	12	0.36	5	0.20	5	1.81
nug06	372	486	10	0.56	10	0.31	10	0.25	8	0.83	11	0.22	6	**0.10**	6	1.84
nug07	602	931	12	1.83	13	0.96	12	0.72	12	2.02	18	1.48	10	**0.70**	8	2.09
nug08	912	1,632	10	3.27	11	1.03	12	1.06	10	3.26	16	2.08	8	**1.00**	7	1.96
nug12	3,192	8,856	19	1,287.19	20	427.16	19	355.36	20	73.13	$\dagger_{-7}$	†	$\dagger_{-7}$	†	17	**5.57**
nug15	6,330	22,275	23	9,521.25	25	809.23	24	773.55	23	559.88	33	171.64	$\dagger_{-5}$	†	17	**22.13**
nug20	15,240	72,600	25	60,223.29	$\dagger_{-7}$	$\dagger_{28}$	33	16,650.52	$\dagger_{-7}$	$\dagger_{28}$	$\dagger_{-7}$	†	$\dagger_{-5}$	†	19	**243.71**

TABLE 3.4: Experiments on MITTELMANN problems. In each row, red boldface and blue underline denote the fastest and second fastest solvers in CPU time (seconds), respectively.

			Inner-outer iteration solvers						Direct solver		Public-domain solvers				Commercial solver	
			AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		SDPT3		SeDuMi		MOSEK	
Problem	m	n	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
fome11	12,142	24,460	47	6,900.09	48	14,156.31	53	12,270.84	-	-	-	-	$\dagger_{-5}$	†	23	**8.97**
fome12	24,284	48,920	48	12,568.26	48	38,138.98	52	28,159.85	-	-	-	-	$\dagger_{-7}$	†	21	**33.17**
fome13	48,568	97,840	47	25,726.58	50	37,625.03	54	63,301.06	-	-	-	-	$\dagger_{-7}$	†	24	**61.01**
fome20	33,874	108,175	61	1,510.85	64	1,240.23	62	689.71	62	692.71	-	-	$\dagger_{-7}$	†	34	**8.96**
fome21	67,748	216,350	74	12,671.62	74	3,185.03	84	3,822.02	75	1,617.71	-	-	$\dagger_{-6}$	†	39	**18.47**
nug08-3rd	19,728	29,856	12	5,833.97	11	259.01	10	**237.02**	-	-	-	-	-	-	7	257.82
pds-30	49,944	158,489	69	1,964.48	72	1,105.42	70	788.98	69	1,659.21	-	-	103	2,014.70	34	**19.93**
pds-40	66,844	217,531	66	4,878.49	68	1,551.30	77	1,904.76	67	4,012.71	◇	◇	105	4,832.20	34	**31.15**
pds-50	83,060	275,814	73	13,860.17	73	3,274.74	80	3,960.55	73	7,196.51	◇	◇	111	11,433.90	38	**49.74**
pds-60	99,431	336,421	72	25,592.33	75	5,024.43	83	7,535.99	72	11,609.01	◇	◇	$\dagger_{-7}$	†	36	**94.28**
pds-70	114,944	390,005	80	22,564.32	82	4,980.04	85	7,405.50	84	17,575.97	◇	◇	126	44,946.8	46	**136.50**
pds-80	129,181	434,580	80	25,752.26	83	6,279.08	86	9,853.86	85	21,077.53	◇	◇	119	58,286.40	42	**157.64**
rail507	507	63,516	43	1,039.09	51	1,138.80	51	475.47	48	14.98	$\dagger_{-7}$	†	34	7.10	17	**2.69**
rail516	516	47,827	39	496.60	43	700.58	39	536.36	38	11.82	$\dagger_{-7}$	†	19	3.70	11	**2.48**
rail582	582	56,097	44	1,296.56	46	971.35	47	1,422.62	41	17.52	$\dagger_{-7}$	†	40	8.60	16	**2.43**

as $\kappa(\mathcal{A}\mathcal{A}^{\mathsf{T}})$ increased sharply.

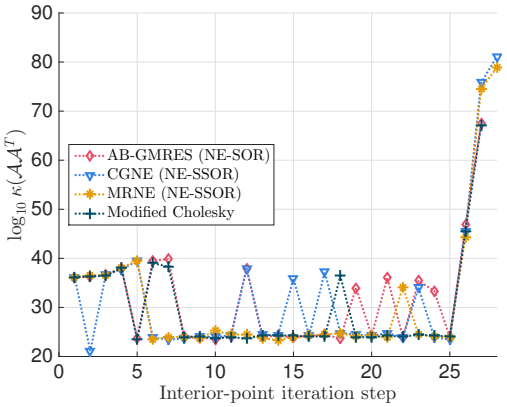
Figs. 3.3c–3.3d show the relative residual norm $\|\boldsymbol{f}_{\mathrm{af}} - \mathcal{A}\mathcal{A}^{\mathsf{T}}\Delta\boldsymbol{y}_{\mathrm{af}}\|_2/\|\boldsymbol{f}_{\mathrm{af}}\|_2$ in the predictor stage and $\|\boldsymbol{f}_{\mathrm{cc}} - \mathcal{A}\mathcal{A}^{\mathsf{T}}\Delta\boldsymbol{y}_{\mathrm{cc}}\|_2/\|\boldsymbol{f}_{\mathrm{cc}}\|_2$ in the corrector stage, respectively. The quantities are in $\log_{10}$ scale. The relative residual norm for modified Cholesky tended to increase with the interior-point iterations and sharply increased in the final phase when it lost accuracy in solving the normal equations for the steps. We observed similar trends for other test problems and, in the worst cases, the inaccuracy in the solutions prevented interior-point convergence. Among the iterative solvers, AB-GMRES (NE-SOR) and MRNE (NE-SSOR) were the most stable in keeping the accuracy of solutions to the normal equations; CGNE (NE-SSOR) performed similarly but lost numerical accuracy at the last few interior-point steps.

Figs. 3.3e–3.3f show the CPU time and number of iterations of the Krylov methods for each interior-point step, respectively. It was observed that the CPU time of the modified Cholesky solver was more evenly distributed in the whole process while that of the iterative solvers tended to be less in the beginning and ending phases. At the final stage, AB-GMRES (NE-SOR) required the fewest number of iterations but cost much more CPU time than the other two iterative solvers. This can be explained as follows: AB-GMRES (NE-SOR) requires increasingly more CPU time and memory with the number of iterations because it has to store the orthonormal vectors in the modified Gram-Schmidt process as well as the Hessenberg matrix. In contrast, CGNE (NE-SSOR) and MRNE (NE-SSOR) based methods require constant memory. CGNE (NE-SSOR) took more iterations and CPU time than MRNE (NE-SSOR). Other than $\mathcal{A}$ and the preconditioner, the memory required for k iterations of AB-GMRES is $\mathcal{O}(k^2 + km + n)$ and that for CGNE and MRNE iterations is $\mathcal{O}(m + n)$[61, 94]. This explains why AB-GMRES (NE-SOR), although requiring less iterations, usually takes longer to obtain the solution at each interior-point step.

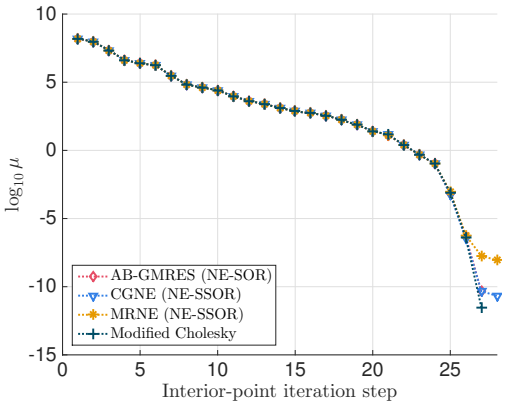
On the other hand, the motivation for using AB-GMRES (NE-SOR) is that GMRES is more robust for ill-conditioned problems than the symmetric solvers CG and MINRES. This is because GMRES uses a modified Gram-Schmidt process to orthogonalize the vectors explicitly; CG and MINRES rely on short recurrences, where orthogonality of vectors may be lost due to rounding error. Moreover, GMRES allows using non-symmetric preconditioning while the symmetric solvers require symmetric preconditioning. For example, using SOR preconditioner is cheaper than SSOR for one iteration because the latter goes forwards and backwards. SOR requires 2MV + 3m operations per inner iteration, while SSOR requires 4MV + 6m. In this sense, the GMRES method has more freedom for choosing preconditioners.

From Fig. 3.3, we may draw a few conclusions. For most problems, the direct solver gave the most efficient result in terms of CPU time. However, for some problems, the direct solver tended to lose accuracy as interior-point iterations proceeded and, in the worst cases, this would inhibit convergence. For problems where the direct method broke down, the proposed inner-iteration preconditioned Krylov subspace methods worked until convergence. It is acceptable to solve iteratively for an

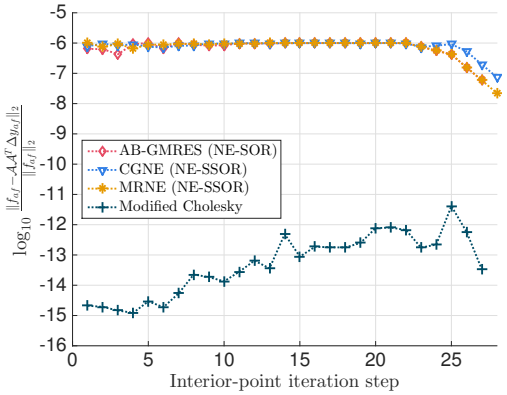
approximate step in the early phase of the interior-point method and then increase the level of accuracy in the late phase.



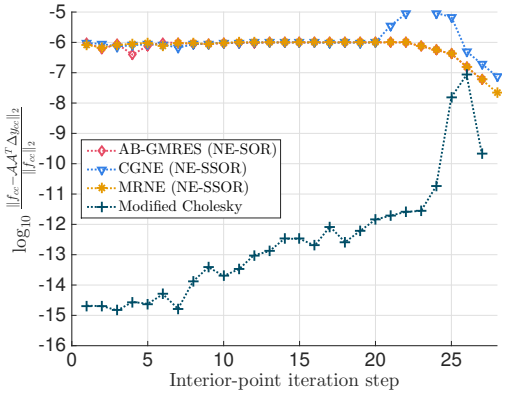
(a) Condition number $\kappa(\mathcal{A}\mathcal{A}^{\mathsf{T}})$.



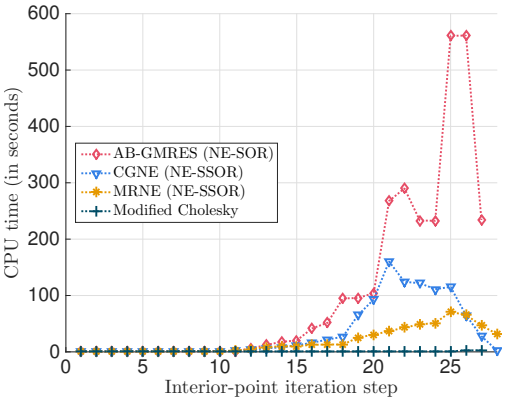
(b) Duality measure μ.



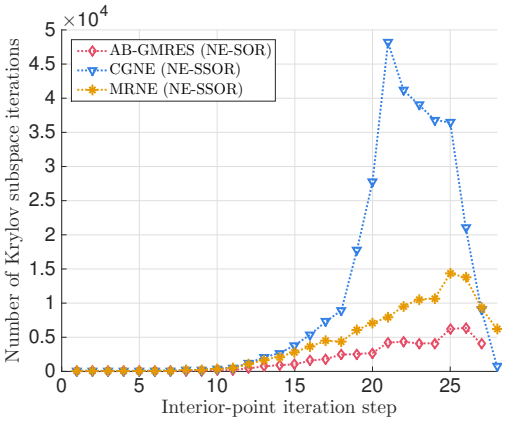
(c) Relative residuals for predictor stage.



(d) Relative residuals for corrector stage.



(e) CPU time for each interior-point step.



(f) Krylov iteration for each interior-point step.

FIG. 3.3: Numerical results for problem ken_13.

3.4.4 Rank-deficient problems

Most of the problems tested in the last section have a sparse and full-rank constraint matrix A. In this section, we enrich the experiment by adding artificial problems with a dense, rank-deficient and ill-conditioned constraint matrix, which challenge some of the solvers.

We first present an experiment to investigate the effect of rank-deficiency on CPU time. Since MOSEK was the most efficient and stable standard solver as presented in the previous section, here we only compare our solvers with MOSEK. We randomly generated a set of constraint matrices A whose rank ranged from 50 to 100 with a step of 5. The elements of $\boldsymbol{x}$ and $\boldsymbol{c}$ were uniformly distributed random numbers, generated by using the MATLAB function `rand`. The location of zero elements of $\boldsymbol{x}$ was also subject to the random uniform distribution. Then, $\boldsymbol{b}$ was generated as $\boldsymbol{b} = A\boldsymbol{x}$. More details are given in Table 3.5.

In Fig. 3.4, we plot the time required for each solver to achieve interior-point convergence versus rank(A). In order to give averaged information, we took an average of the CPU times for 5 different randomly generated problems for each rank, where the CPU time was taken as an average of 10 measurements for each problem. All solvers succeeded in solving the problems. Iterative solvers performed better than modified Cholesky as the rank decreased.

Next, we present an experiment for problems that were both rank-deficient and ill-conditioned. We randomly generated a set of problems, with constraint matrix A as in Table 3.6. The sparsity of A was around 50%.

In Fig. 3.5, we plot the time required for each solver to achieve interior-point convergence versus rank(A). The graphs for modified Cholesky and MOSEK are

TABLE 3.5: Information on artificial problems: completely dense with different rank.

Problem	m	n	Nonzeros	Rank	$\kappa(A)$
Artificial	100	300	30,000	$[50, 100]$	10^2

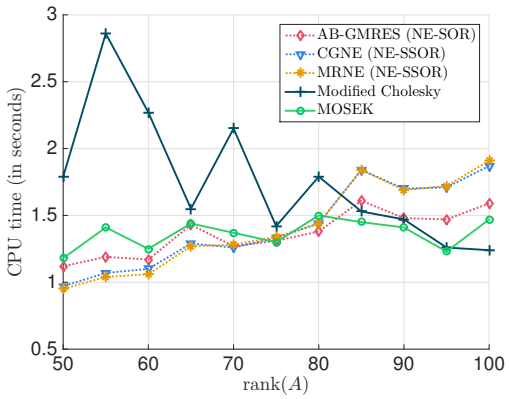


FIG. 3.4: CPU time for artificial problems: completely dense with different rank.

TABLE 3.6: Information on artificial problems: ill-conditioned with different rank.

Problem	m	n	Nonzeros	Rank	$\kappa(A)$
Artificial	100	300	15,000	$[50, 100]$	10^8

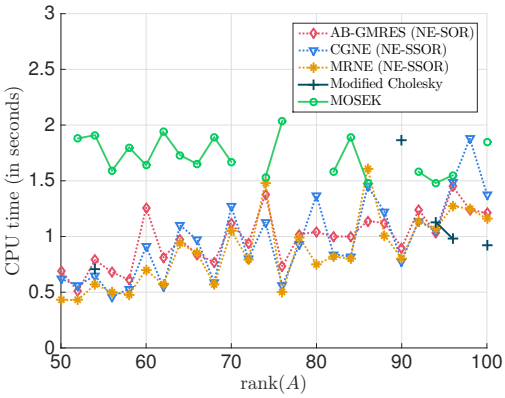


FIG. 3.5: CPU time for artificial problems: ill-conditioned with different rank.

TABLE 3.7: Experiments on artificial problems.

		Inner-outer iteration solvers						Direct solver		Commercial solver	
		AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		MOSEK	
Problem	Rank(A)	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
Rand1	1,000	22	120.04	$\dagger_{-4}$	$\dagger_{18}$	$\dagger_{-6}$	$\dagger_{21}$	-	-	26	6.50
Rand2	999	28	483.85	$\dagger_{-4}$	$\dagger_{18}$	$\dagger_{-6}$	$\dagger_{31}$	-	-	27	11.04
Rand3	998	21	336.19	$\dagger_{-4}$	$\dagger_{21}$	$\dagger_{-6}$	$\dagger_{20}$	-	-	-	-
Rand4	997	24	392.52	$\dagger_{-4}$	$\dagger_{18}$	$\dagger_{-6}$	$\dagger_{20}$	-	-	-	-
Rand5	996	31	441.28	$\dagger_{-4}$	$\dagger_{19}$	$\dagger_{-6}$	$\dagger_{20}$	-	-	-	-
Rand6	995	21	305.69	$\dagger_{-4}$	$\dagger_{21}$	$\dagger_{-6}$	$\dagger_{20}$	-	-	-	-

disconnected because there were failed cases. For example, MOSEK (green line with circles) failed at the point rank(A) = 88 and rank(A) = 90, and hence the points at rank(A) = 86 and rank(A) = 92 were not connected.

This result shows that MOSEK, although fast and stable for the full-rank problems, failed for 7 out of 26 ill-conditioned rank-deficient problems and was almost always slower than the proposed solvers. The modified Cholesky solver broke down due to numerical errors for 21 problems. However, the three iterative solvers overcame this difficulty and solved all the problems.

Note that when the interior-point solver with MOSEK failed to converge, it automatically switched to a simplex method. Although this re-optimisation process can usually give an optimal solution to the LP problem, we consider the interior-point method to have failed.

Similar experiments were carried out on larger problems. We tested the solvers on problems of size $1,000 \times 1,500$ with condition number 10^8 and sparsity around 50%. The result is presented in Table 3.7. The notations † and - have the same meaning as explained in the previous section. The table shows that only AB-GMRES (NE-SOR) succeeded in solving all problems.

3.5 Conclusion

We proposed a new way of preconditioning the normal equations of the second kind arising within interior-point methods for LP problems (3.13). The resulting interior-point solver is composed of three nested iteration schemes. The outer-most layer is the predictor-corrector interior-point method; the middle layer is the Krylov subspace method for least squares problems, where we may use AB-GMRES, CGNE or MRNE; on top of that, we use a row-scaling scheme that does not incur extra CPU time; the inner-most layer, serving as a preconditioner for the middle layer, is the stationary inner iterations. Among the three layers, only the outer-most one runs toward the required accuracy and the other two are terminated prematurely.

The advantage of our method is that it does not break down, even when the matrices become (nearly) singular. The method is competitive for large and sparse problems and may also be well-suited to problems in which matrices are too large and dense for direct approaches to work. Extensive numerical experiments showed that the stability and efficiency of our method outperform the open-source solvers SDPT3 and SeDuMi, and can solve rank-deficient and ill-conditioned problems where the MOSEK interior-point solver fails. It would also be worthwhile to extend our method to problems such as convex quadratic programming and SDP.

Chapter 4

Stability of Calibration Procedures: Fractals in the Black-Scholes-Merton Model

Usually, in the Black-Scholes-Merton pricing theory the volatility is a positive real parameter. Here we explore what happens if it is allowed to be a complex number. The function for pricing a European option with a complex volatility has essential singularities at zero and infinity. The singularity at zero reflects the put-call parity. In this framework, the function yielding the implied volatility has not only a real root, but also infinitely many complex roots in a neighbourhood of the origin. The Newton-Raphson calculation of the complex implied volatility has a chaotic nature described by fractal objects.

4.1 Introduction

Every day, the implied volatility consistent with the given price of a European option is computed millions of times in trading and risk management systems throughout the financial industry. This is typically done with the Newton-Raphson method [68, 112], which can exhibit chaotic phenomena when hunting a successively better approximation to the root. Neglecting the practical meaning, these phenomena are best described in the complex plane by means of the associated fractal Julia sets. In a one-page paper published in 1879, Cayley [15] first suggested the difficulty in extending the Newton-Raphson method (which he called Newton-Fourier) to the following cases:

1. complex polynomials with degree higher than 2;
2. a complex initial value leading the following iterations;
3. the allowance of complex roots.

Furthermore, in the study of this problem he proposed a concept which later was called an attraction basin:

> *"To determine the region of the plane, such that P (the initial point) being taken at pleasure anywhere within one region we arrive ultimately at the point A (a root); anywhere within another region at the point B; and so for the several points representing the roots of the equation."*

In this work, we extend the Black-Scholes-Merton valuation [12, 84] to a complex implied volatility parameter, allowing the initial value of the Newton-Raphson method to be complex; then we explore the fractal objects that describe the chaotic nature of the Newton-Raphson calculation of the implied volatility.

4.1.1 Implied volatility

The Black-Scholes-Merton model does not adequately take into account important characteristics of the market dynamics such as skewness, fat tails and the correlation between the asset's value and its volatility. Other models have been devised to better approximate the fair price of derivatives, as discussed in a large body of research. However, dealers still prefer to describe the price of an option V, obtained either by these refined models or from a market quote, in terms of the volatility σ such that the Black-Scholes-Merton formula replicates the given price. This parameter σ is called the *implied volatility* and is often described, following Rebonato [101, pg. 169], as:

> *"the wrong number to put in the wrong formula to get the right price of plain-vanilla (European) options".*

From the perspective of a trader, implied volatility results from a rescaling process that allows to compare the relative worth of options with different maturities or involving different assets or currencies, where a crude comparison in terms of premium would be inapplicable. For similar reasons it is also used in the interpolation of prices of options with different maturities and strikes.

4.1.2 Numerical scheme to calculate the implied volatility

To calibrate the implied volatility $\sigma \geq 0$, the Newton-Raphson method is used to solve the equation which matches the market price V and the Black-Scholes-Merton valuation for a European option:

$$V = e^{-rT}\theta\left\{F\Phi\left[\theta\left(\frac{\log\frac{F}{K}}{\sigma\sqrt{T}} + \frac{1}{2}\sigma\sqrt{T}\right)\right] - K\Phi\left[\theta\left(\frac{\log\frac{F}{K}}{\sigma\sqrt{T}} - \frac{1}{2}\sigma\sqrt{T}\right)\right]\right\}, \tag{4.1}$$

where $\theta = 1$ for call options and $\theta = -1$ for put options, $F := S_0 e^{(r-d)T}$ is the fair forward price to maturity T, r is the risk-free interest rate, d is the dividend rate (or foreign interest rate for a foreign exchange rate contract), S_0 is the spot price of the underlying, K is the strike price, and $\Phi(\cdot)$ is the standard normal cumulative

distribution function

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{u^2}{2}\right) \mathrm{d}u, \tag{4.2}$$

which can be expressed through the error function as

$$\Phi(x) = \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \int_{0}^{x} \exp\left(-\frac{u^2}{2}\right) \mathrm{d}u \tag{4.3a}$$

$$= \frac{1}{2} + \frac{1}{2} \operatorname{erf} \frac{x}{\sqrt{2}}. \tag{4.3b}$$

If we denote by $f(\sigma)$ the right-hand side of Eq. (4.1), then the Newton-Raphson iteration to solve $V = f(\sigma)$ is given by

$$\sigma_{n+1} = \sigma_n - \frac{f(\sigma_n) - V}{f'(\sigma_n)}, \tag{4.4}$$

starting from an arbitrary initial guess $\sigma_0 \in \mathbb{R}_+$. However, as explained by Jäckel [68], $f(\sigma)$ is convex for low volatilities and concave for higher volatilities, causing instabilities in the algorithm. The article demonstrates how taking the logarithm of both the market price and the Black-Scholes-Merton price overcomes this convergence problem.

Inspired by the chaotic phenomenon arising from the Newton-Raphson search for implied volatility around the origin, where $f(\sigma)$ is too flat according to Jäckel, we perform a further experiment on the calibration of implied volatility where the search domain is extended to the complex plane and the Black-Scholes-Merton price is extended as an analytic function on $\mathbb{C}_* = \mathbb{C} \setminus \{0\}$ with essential singularities at zero and infinity. We observe infinitely many complex roots for Eq. (4.1) other than the real one, as will be illustrated by means of fractal attraction basins.

4.2 Analytic extension of the pricing function

In this section, we show the analytic extension of the major component of the Black-Scholes-Merton pricing function, i.e., the normal cumulative distribution function, and therefore the extension of the pricing function itself. We discuss the singularity of this function and hence the complex roots of Eq. (4.1). Illustrations are given at the end.

4.2.1 Analytic extension of the cumulative normal distribution

The standard normal cumulative distribution function $\Phi(z)$ with a complex argument $z \in \mathbb{C}$ is related to several special functions that arise often in applied mathematics and engineering; see for example Fettis, Caslin and Cramer [43] for an analysis of the zeros of $\operatorname{erf}(z)$.

Theorem 4.1. *The cumulative density function of the standard normal distribution*

can be extended as a complex entire function on $\mathbb{C}$.

Proof. Given that $\exp(-z^2/2)$ is an entire function, by general results of complex analysis [2] the function

$$\Phi(z) = \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \int_0^z \exp\left(-\frac{u^2}{2}\right) \mathrm{d}u \tag{4.5}$$

is well defined and entire. Trivially, its restriction to the real line is the function $\Phi(x)$ defined in Eq. (4.3a). □

An analytic function is said to have an isolated singularity at a point if the function is analytic in a neighbourhood of the point with the point excluded. Isolated singularities of analytic functions in one variable are classified as [2]:

Removable if the function can be assigned a value at that point such that the resulting extended function is analytic. A typical example of a removable singularity is $z = 0$ for $f(z) = (\sin z)/z$.

Pole if the norm of the function tends to ∞ as that point is approached. A typical example of a pole singularity is $z = 0$ for $f(z) = 1/z$.

Essential in all other cases. A typical example of an essential singularity is $z = 0$ for $f(z) = \exp(1/z)$.

Remark 4.1. The complex cumulative normal distribution function $\Phi(z)$ has an essential singularity at $z = \infty$. This is because along the real axis, when $z \to +\infty$, $\Phi(z) \to 1$ and when $z \to -\infty$, $\Phi(z) \to 0$. On the other hand, along the imaginary axis, $\Phi(z)$ is unbounded for $z \to \pm i\infty$: for real y, substituting $u = iv$,

$$\Phi(iy) = \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \int_0^{iy} \exp\left(-\frac{u^2}{2}\right) \mathrm{d}u \tag{4.6a}$$

$$= \frac{1}{2} + i\frac{1}{\sqrt{2\pi}} \int_0^{y} \exp\frac{v^2}{2} \mathrm{d}v, \tag{4.6b}$$

which grows to $\pm i\infty$ as $y \to \pm\infty$.

4.2.2 Analytic extension of the Black-Scholes-Merton pricing formula

Theorem 4.2. *The Black-Scholes-Merton price as a function of the volatility* $f(\sigma)$ *can be extended as an analytic function on* $\mathbb{C}_* = \mathbb{C} \setminus \{0\}$. *The singularities at zero and infinity are essential.*

Proof. The necessary and sufficient condition for a point z_0 to be an essential singularity of $f(z)$ is that $\lim_{z \to z_0} f(z)$ does not exist. Let $d_\pm := \log(F/K)/(\sigma\sqrt{T}) \pm \sigma\sqrt{T}/2$, then it is easily seen that when σ approaches zero along the imaginary axis, $d_\pm$ approach $+i\infty$ or $-i\infty$, depending on the ratio F/K. As shown in Remark 4.1,

$\lim_{y\to\infty} \Phi(iy)$ is indefinite. Thus $\lim_{\sigma\to 0} f(\sigma)$ equals neither a finite complex number nor ∞, i.e., the limit does not exist. Similarly one can verify that the singularity at $\sigma = \infty$ is essential.

From another perspective, the singularity is not removable, as otherwise it would have a zero Taylor expansion, and it is not a pole either as otherwise the function would tend to infinity. □

Remark 4.2. If we denote by $f(\sigma)$ the RHS of Eq. (4.1), then we have that $f(-\sigma)$ is the opposite of the price of the put option with the same maturity and strike. This has as a consequence that along the real axis

$$\lim_{\sigma\to 0^+} f(\sigma) = \begin{cases} e^{-rT}(F-K) & \text{if } F \geq K \\ 0 & \text{if } F \leq K \end{cases} \tag{4.7}$$

$$\lim_{\sigma\to 0^-} f(\sigma) = \begin{cases} 0 & \text{if } F \geq K \\ e^{-rT}(F-K) & \text{if } F \leq K, \end{cases} \tag{4.8}$$

which is reflected in Fig. 4.1 and again implies that the singularity at $\sigma = 0$ is essential. By the put-call parity $f(\sigma) - f(-\sigma)$ is independent of σ.

A striking consequence of the previous result is that the implied volatility equation has now infinitely many solutions near zero.

Corollary 4.1. *In each neighbourhood of 0, there are infinitely many complex values of the volatility σ such that Eq. (4.1) holds.*

Proof. This is a consequence of the previous theorem and the Weierstrass-Casorati theorem [2, Ch. 4, Thm. 9]. □

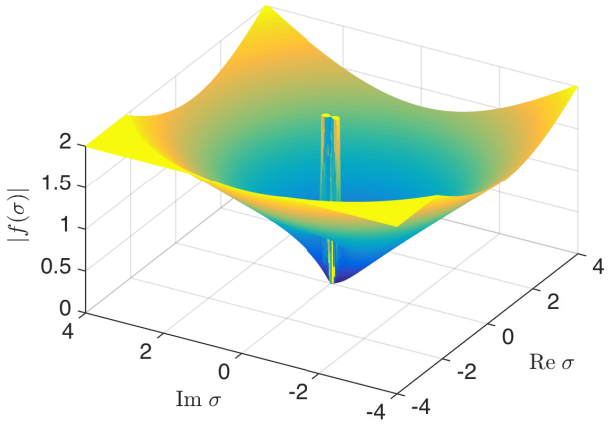


FIG. 4.1: The modulus $|f(\sigma)|$ of the complex Black-Scholes-Merton price as a function of complex volatility σ. Note the gap along the real axis between the value at 0^- and 0^+ as explained in Remark 4.2. The function has a similar behaviour in a neighbourhood of $\sigma = \infty$. The plot is truncated from above at 2 because in a neighbourhood of zero the surface goes to infinity infinitely many times.

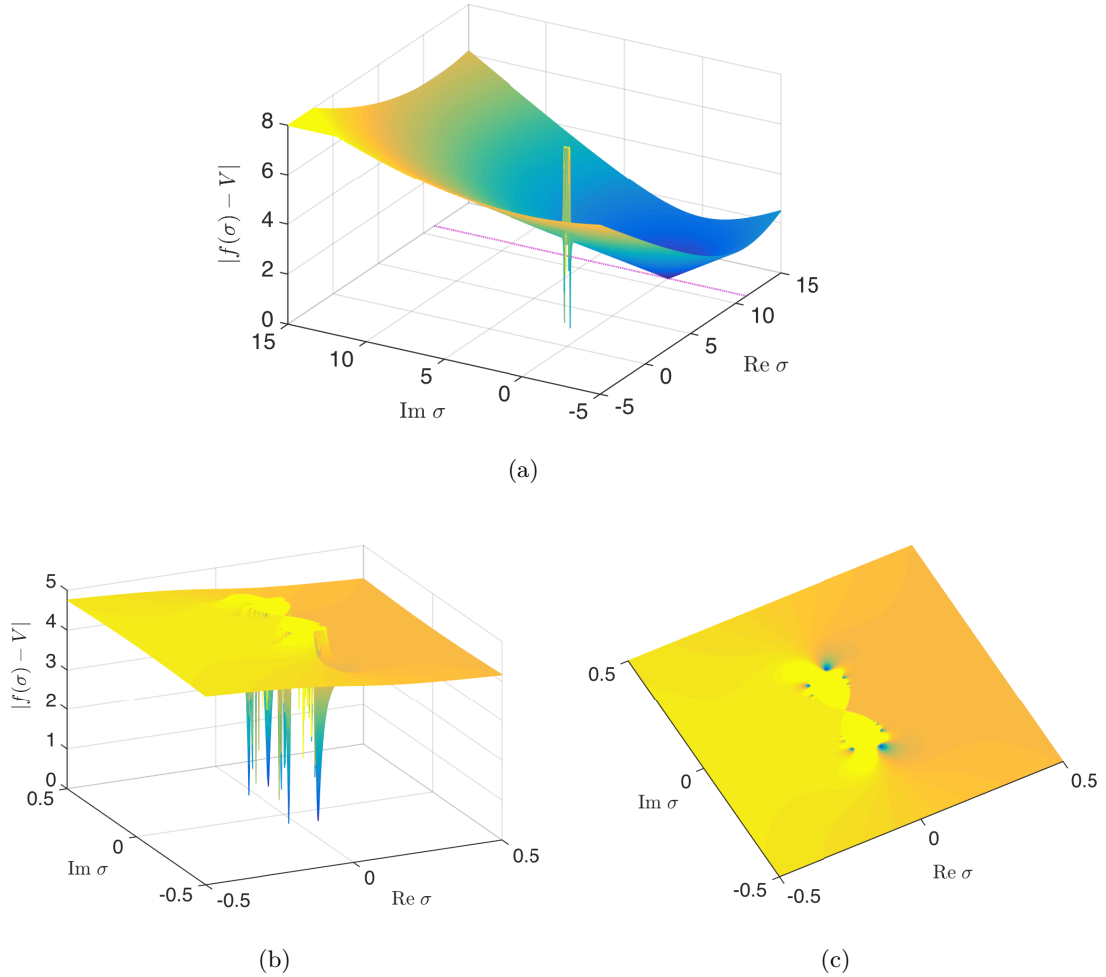


FIG. 4.2: The modulus of calibration error $|f(\sigma) - V|$ as a function of complex volatility σ. (a): The minima of the surface correspond to the scaled real implied volatility at 11.2 vol and to the infinitely many complex roots close to the origin; the plot is truncated from above at 8. (b): A blow-up of $|f(\sigma) - V|$ around the multiple complex roots close to the origin; the plot is truncated from above at 5. (c): Contour map of $|f(\sigma) - V|$.

Fig. 4.2 displays the function $|f(\sigma) - V|$ for a fixed V which according to Corollary 4.1 exhibits infinitely many complex zeros other than the real implied volatility.

In Figs. 4.1–4.3 we use market data of 28 November 2013 for a 1Y at-the-money (ATM) USDJPY call option: the spot is $S = 102.10$, the strike is $K = 102.76$, and the scaled ATM volatility is $\sigma = 11.2$ vol; the interest rate for JPY, is $r = 2.68\%$ and that for USD is $q = 2.71$. The unit "vol" is a measure of volatility used habitually by practitioners, with 1 vol = 1%. We scale our complex plane in vols, so that the label 30 on the x or y axis means 30% or $i30\%$.

4.3 Implied volatility fractals

4.3.1 Newton-Raphson fractals

Fractals arising from the Newton-Raphson algorithm have been studied in several articles [28, 99]. In fact the first fractals arose as an attempt to respond a question by Cayley [15] on the loci of complex numbers converging to the several roots of a polynomial by the Newton-Raphson algorithm.

Definition 4.1. Given a function g, for each positive integer n we denote by g^n the n-fold composition of g:

$$g^n(x) = \underbrace{g(g(\ldots g(x)\ldots))}_{n \text{ times}}. \tag{4.9}$$

Then we can associate to each fixed point x_* of g its basin of attraction

$$B_g(x_*) = \left\{ x \in \mathbb{C} | \lim_{n\to\infty} g^n(x) = x_* \right\}. \tag{4.10}$$

If to solve a non-linear equation $y(x) = 0$ we apply the Newton-Raphson iteration

$$x_{n+1} = x_n - \frac{y(x_n)}{y'(x_n)} \tag{4.11}$$

and define

$$g(x) = x - \frac{y(x)}{y'(x)}, \tag{4.12}$$

then the initial values x_0 where the Newton-Raphson method converges to a given root x_* are in the attraction basin $B_g(x_*)$.

The Julia set J_g is the boundary of the attraction basin $B_g(x_*)$ of a fixed point x_* [86]. It has been proved that when g is a rational function the Julia set thus defined is independent of the fixed point and coincides with the closure of the repelling fixed points. Several illustrations of attraction basins coloured according to the convergence speed and also of Julia sets can be found in the book by Peitgen and Richter [98].

4.3.2 Fractals associated to implied volatility

In this subsection, we present an empirical analysis of attraction basins and Julia sets for the Newton-Raphson iteration associated to Eq. (4.1). For simplification we use an equivalent equation given by Jäckel [68],

$$b = h(\hat{\sigma}) \tag{4.13}$$

where $b := Ve^{rT}/\sqrt{FK}$,

$$h(\hat{\sigma}) := \theta e^{a/2}\Phi\left[\theta\left(\frac{a}{\hat{\sigma}} + \frac{\hat{\sigma}}{2}\right)\right] - \theta e^{-a/2}\Phi\left[\theta\left(\frac{a}{\hat{\sigma}} - \frac{\hat{\sigma}}{2}\right)\right], \tag{4.14}$$

$a := \log(F/K)$, and $\hat{\sigma} := \sigma\sqrt{T}$. The Newton-Raphson iteration to find the implied volatility from equation (4.13) is

$$\hat{\sigma}_{n+1} = \hat{\sigma}_n - \frac{h(\hat{\sigma}_n) - b}{h'(\hat{\sigma}_n)} \tag{4.15a}$$

$$= \hat{\sigma}_n - \frac{\sqrt{2\pi}\,(h(\hat{\sigma}_n) - b)}{\exp\left(-\frac{a^2}{2\hat{\sigma}_n^2} - \frac{\hat{\sigma}_n^2}{8}\right)}. \tag{4.15b}$$

The termination criterion is

$$\frac{\|\hat{\sigma}_{n+1} - \hat{\sigma}_n\|}{\|\hat{\sigma}_n\|} \leq \epsilon \tag{4.16}$$

or $n = L$. Given an initial point $\hat{\sigma}_0$, a tolerance level ϵ and a maximum iteration number L, the Newton-Raphson iteration (4.15b) terminates in three cases:

1. it converges to the real root;
2. it converges to one of the many complex roots;
3. it does not converge until the maximum iteration number is reached, or encounters other numerical problems.

We produced fractals by plotting each initial point $\hat{\sigma}_0$ in a different colour according to how the corresponding iterations terminate. Specifically, for the three termination cases above, we used the following colour scheme:

1. a shade of blue, linearly scaling from dark to light by the number of steps it takes to converge: the points in dark blue take fewer steps to converge than those in light blue;
2. a shade of red, linearly scaling from dark to light by the number of steps it takes to converge: the points in dark red take fewer steps to converge than those in light red;
3. black.

In this colouring scheme, the attraction basins are the regions that are not in black. In all figures, we used 1001×1001 initial points. Fig. 4.3 shows an implied volatility fractal under different magnifications. Note that the enlargement of panel 4.3a around the origin shown in panel 4.3b is not observable in panel 4.3a because of the limited resolution of the latter.

Fig. 4.4 shows the fractals for options with different values of $\Delta = \partial f/\partial S_0$, i.e. the rate of change of the option price with respect to the change in the spot price of the underlying. In the Black-Scholes-Merton model,

$$\Delta = \theta e^{-dt}\Phi\left[\theta\left(\log(F/K)/(\sigma\sqrt{T}) + \sigma\sqrt{T}/2\right)\right]. \tag{4.17}$$

We use call and put options with $\Delta = 25\%$ and $\Delta = 10\%$ because they are often used by practitioners to depict the volatility smile (the ATM call option with $\Delta = 50\%$

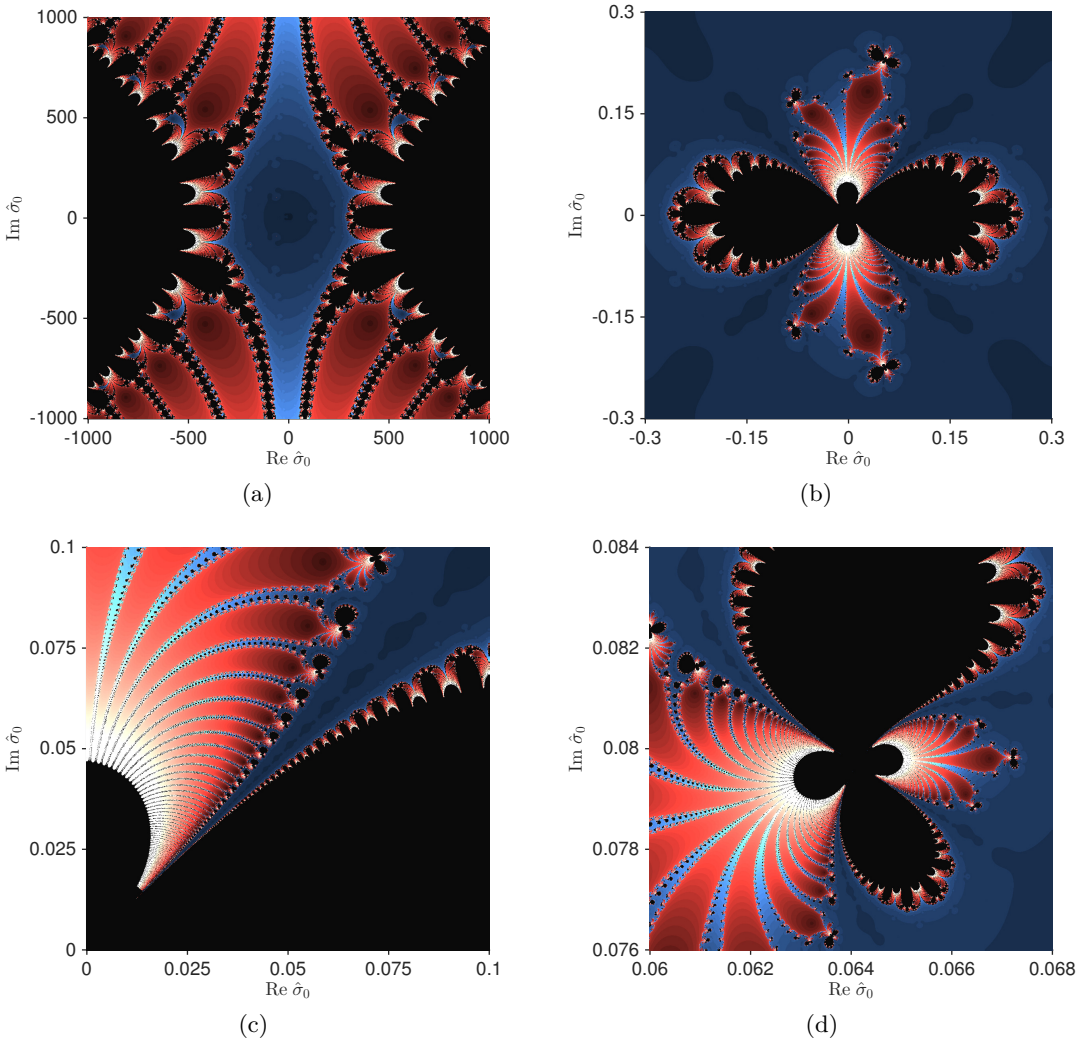
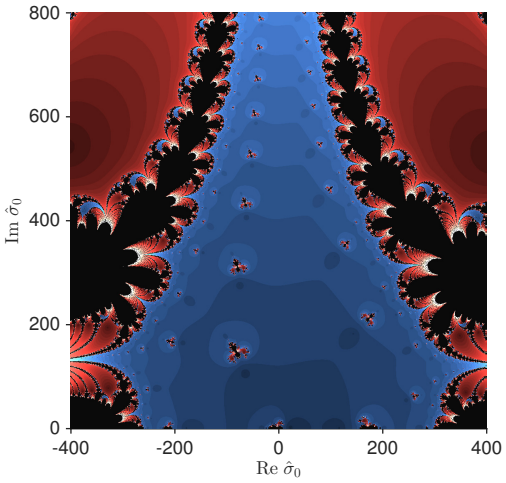


FIG. 4.3: (a) Implied volatility fractal for an ATM option with $\epsilon = 10^{-8}$, $L = 100$. The axis scale is in vols. (b) A zoom-in of (a) around the origin. (c) A zoom-in of (b) in the first quadrant. (d) A zoom-in of one petal in (c).

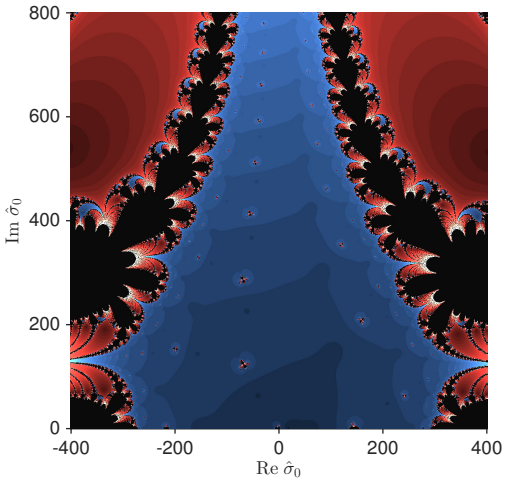
is shown in Fig. 4.3). The volatility smile is an important measurement indicating that the implied volatility changes with the strike. We show the fractals in the upper half of the complex plane because the fractals are symmetric with respect to the real axis, as can be seen in Fig. 4.3. Notice the difference in the number of attraction basins for different values of Δ and the similarity between call and put options with the same Δ.

Fig. 4.5 shows a fractal for an ATM option with the same parameters as Fig. 4.3 and Jäckel's aforementioned modification of the equation to be solved [68]. This modification subtracts from both sides of Eq. (4.13) the intrinsic value $\tau := 2\theta H(\theta a)\sinh(a/2)$, where $H(\cdot)$ is the Heaviside function, and solves the equivalent form on a logarithmic scale

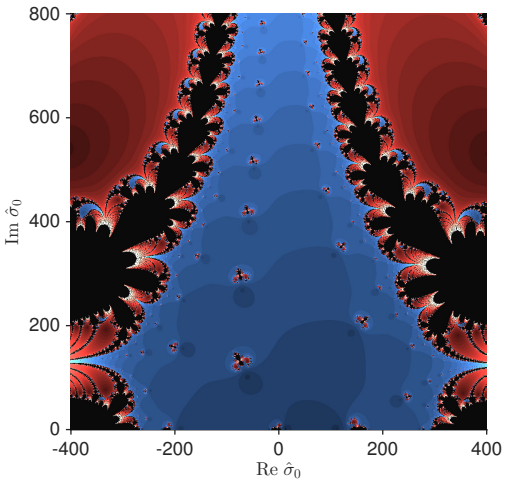
$$\log \frac{h(\hat{\sigma}) - \tau}{b - \tau} = 0. \tag{4.18}$$



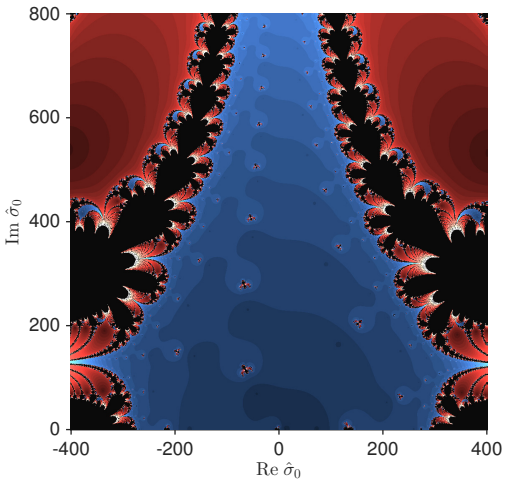
(a) Put, $\Delta = -10\%$, $K = 76.74$, $\sigma = 25$ vol.



(b) Put, $\Delta = -25\%$, $K = 91.40$, $\sigma = 20$ vol.

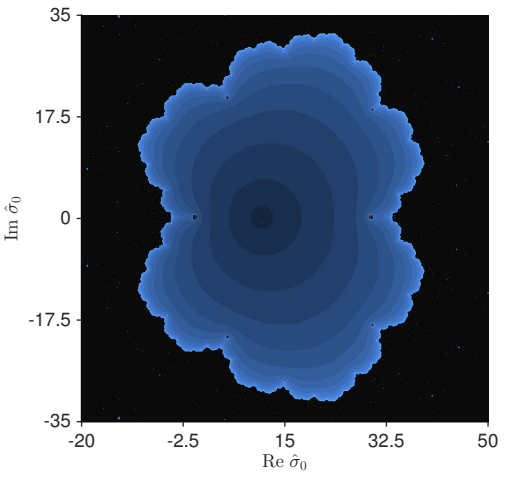


(c) Call, $\Delta = 10\%$, $K = 126.50$, $\sigma = 16$ vol.

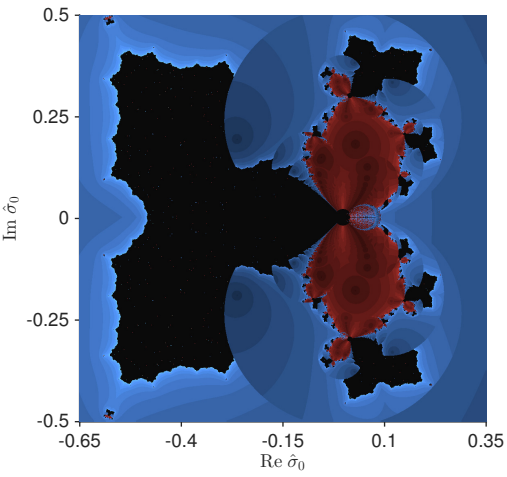


(d) Call, $\Delta = 25\%$, $K = 113.85$, $\sigma = 15$ vol.

FIG. 4.4: Implied volatility fractals for options with $\epsilon = 10^{-8}$, $L = 100$ and various values of Δ.



(a)



(b)

FIG. 4.5: Implied volatility fractal for an ATM option with Jäckel's modification.

The corresponding Newton-Raphson iteration is

$$\hat{\sigma}_{n+1} = \hat{\sigma}_n - \frac{\log \frac{h(\hat{\sigma}_n)-\tau}{b-\tau}}{\frac{1}{h(\hat{\sigma}_n)-\tau} h'(\hat{\sigma}_n)} \tag{4.19a}$$

$$= \hat{\sigma}_n - \frac{\sqrt{2\pi}(h(\hat{\sigma}_n)-\tau)\log \frac{h(\hat{\sigma}_n)-\tau}{b-\tau}}{\exp\left(-\frac{a^2}{2\hat{\sigma}_n^2} - \frac{\hat{\sigma}_n^2}{8}\right)}. \tag{4.19b}$$

Notice that this modification largely reduced the red area (the initial points that lead to a convergence to the complex roots) and the blue area (the initial points that lead to a convergence to the real root). The enlargement panel 4.5b shows the exquisite fractal near the origin.

4.4 Conclusion

We extended the Black-Scholes-Merton price as a function of the volatility as an analytic function on $\mathbb{C}_* = \mathbb{C} \setminus \{0\}$ and showed that the singularities at zero and infinity are essential. As a result, the objective function for finding the implied volatility has infinitely many complex solutions near zero. Following the practice, we adopt the Newton-Raphson method to resolve for the implied volatility. The chaotic nature of the calibration of the implied volatility is described in the complex plane by means of the associated fractal Julia sets. Fractals associated with the searching process are shown for this interesting problem prevalent in the industry. Among other things, these fractals visualise dramatically the effect of a modification suggested by Jäckel to improve the stability and convergence of the search for the implied volatility.

Chapter 5

Model Consistency Under Spot Inversion in the Foreign Exchange Market

5.1 Introduction

The symmetries of the foreign exchange (FX) market distinguish it from the others. It is common knowledge that the USDJPY exchange rate is just the reciprocal of the JPYUSD exchange rate.

An ideal model for an exchange rate S_t should guarantee that its inverse $1/S_t$ also follows a process within the same class. Furthermore, let $S^{(1)}$ denotes the USDJPY rate, $S^{(2)}$ denotes the JPYGBP rate, then their product $S^{(3)} = S^{(1)}S^{(2)}$ is the USDGBP rate and should obey a model in the same class as the other two. However in reality these properties are barely satisfied by the existing models.

The Black-Scholes-Merton model [12, 84] can guarantee that the inversion of spot rate follows another Black-Scholes-Merton model. But many stochastic volatility (SV) models, which are more popular in the financial industry, do not enjoy this property. In this chapter, we investigate this property on the SV model and use the results to verify an arbitrage example in the FX market.

For the following discussion, we avoid using the word *domestic* and *foreign* as they cause confusions. Instead, we use the FX terminology that the first currency is called the *base* currency and the second is called the *term* currency. f_x denotes the first order derivative of f with respect to x, f_{xy} denotes the second order derivative of f with respect to x and y. The exception is the subscript t for S and v which denotes a process as time passes, e.g., S_t represents the process of S.

5.2 Definition and theorem of model consistency

We give the definitions of SV model and the model consistency under spot inversions.

Definition 5.1. A *stochastic volatility (SV) model* for an exchange rate S_t is a

stochastic process of the form

$$\mathrm{d}S_t = (r_{\text{term}} - r_{\text{base}})S_t\mathrm{d}t + \sqrt{v_t}S_t\mathrm{d}W_t^1, \tag{5.1a}$$
$$\mathrm{d}v_t = a(v_t, t)\mathrm{d}t + b(v_t, t)\mathrm{d}W_t^2, \tag{5.1b}$$

with

$$\mathrm{d}W_t^1\mathrm{d}W_t^2 = \rho\mathrm{d}t, \tag{5.1c}$$

where r_{term} and r_{base} denote the risk-free interest rate of the term currency and base currency, respectively.

Definition 5.2. A family of stochastic volatility processes is called *consistent under spot inversions* if whenever an exchange rate S_t follows one such process under its term currency risk neutral measure, the inverse rate S_t^{-1} in the measure associated to its term currency (which is the base currency of S_t) is also the solution of an SDE in the family.

Next, we give the theorem on SV model consistency.

Theorem 5.1. *If an exchange rate S_t follows the process Eqs. (5.1) in the risk-neutral measure associated to the term currency, then the inverse exchange rate $S_t^* := S_t^{-1}$ follows*

$$\mathrm{d}S_t^* = (r_{\text{base}} - r_{\text{term}})S_t^*\mathrm{d}t + \sqrt{v_t^*}S_t^*\mathrm{d}\bar{W}_t^1, \tag{5.2a}$$
$$\mathrm{d}v_t^* = \left[a(v_t^*, t) + \rho\sqrt{v_t^*}b(v_t^*, t)\right]\mathrm{d}t + b(v_t^*, t)\mathrm{d}\bar{W}_t^2, \tag{5.2b}$$

with

$$\mathrm{d}\bar{W}_t^1\mathrm{d}\bar{W}_t^2 = -\rho\mathrm{d}t. \tag{5.2c}$$

Proof. Let S_t be the spot rate which follows the general SV proecess Eqs. (5.1), and $f(S_t, v_t, t)$ be a derivative contract with S_t as the underlying. The price for $f(S_t, v_t, t)$ in term currency satisfies the PDE [21]

$$f_t + \frac{1}{2}vS^2f_{SS} + \frac{1}{2}b^2f_{vv} + \rho\sqrt{V}Sbf_{Sv} + (r_{\text{term}} - r_{\text{base}})Sf_S + af_v - r_{\text{term}}f = 0. \tag{5.3}$$

Let $f^* := f/S$ be the price of this derivative in base currency. Then we deduce the PDE that is satisfied by f^* as a function of S_t^*.

First note that

$$S^* = S^{-1}, \tag{5.4a}$$
$$f_S = -f_{S^*}S^{*2}, \tag{5.4b}$$
$$f_{SS} = f_{S^*S^*}(S^*)^4 + 2f_{S^*}(S^*)^3, \tag{5.4c}$$
$$f_{Sv} = \frac{\partial f_v}{\partial S^*}\frac{\partial S^*}{\partial S} = -f_{S^*v}S^{*2}. \tag{5.4d}$$

Then the pricing PDE Eq. (5.3) in terms of S_t^* is

$$\begin{aligned} f_t + \frac{1}{2}vS^*\left(f_{S^*S^*}S^* + 2f_{S^*}\right) + \frac{1}{2}b^2 f_{vv} - \rho\sqrt{v}S^* b f_{S^*v} \\ -(r_{\text{term}} - r_{\text{base}})S^* f_S + a f_v - r_{\text{term}} f = 0. \end{aligned} \tag{5.5}$$

Next, we change to $f^* = fS^*$ noting that

$$f_{S^*} = \frac{1}{S^*}f^*_{S^*} - \frac{1}{S^{*2}}f^*, \tag{5.6a}$$

$$f_{S^*S^*} = \frac{1}{S^*}f^*_{S^*S^*} - \frac{2}{S^{*2}}f^*_{S^*} + \frac{2}{(S^*)^3}f^*, \tag{5.6b}$$

$$f_{S^*v} = \frac{1}{S^*}f^*_{S^*v} - \frac{1}{S^{*2}}f^*_v, \tag{5.6c}$$

$$f_{vv} = \frac{f^*_{vv}}{S^*}, \tag{5.6d}$$

$$f_v = \frac{f^*_v}{S^*}, \tag{5.6e}$$

$$f_t = \frac{f^*_t}{S^*}. \tag{5.6f}$$

The PDE Eq. (5.5) becomes

$$\begin{aligned} \frac{f^*_t}{S^*} + \frac{1}{2}v(S^*)^2 f_{S^*S^*} + vS^* f_{S^*} + \frac{1}{2}b^2 f_{vv} - \rho\sqrt{v}S^* b f_{vS^*} \\ -(r_{\text{term}} - r_{\text{base}})S^* f_{S^*} + a f_v - r_{\text{term}}\frac{f^*}{S^*} = 0. \end{aligned} \tag{5.7}$$

After arranging terms, we reach

$$\begin{aligned} f^*_t + \frac{1}{2}f^*_{S^*S^*} - (r_{\text{term}} - r_{\text{base}})S^* f^*_{S^*} + (a + \rho\sqrt{v}b)f^*_v + \frac{1}{2}b^2 f^*_{vv} \\ -\rho\sqrt{v}bS^* f^*_{vS^*} - r_{\text{base}} f^* = 0, \end{aligned} \tag{5.8}$$

which is the pricing PDE for the stochastic process Eqs. (5.2). □

5.3 Theorem for specific stochastic volatility models

We first list the variance processes (5.1b) and the function forms of $a(v_t, t)$ and $b(v_t, t)$ for the popular SV models.

1. The Heston model [63].
   The variance v_t follows

$$\mathrm{d}v_t = \kappa(\bar{v} - v_t)\mathrm{d}t + \sigma\sqrt{v_t}\mathrm{d}W_t^2, \tag{5.9}$$

   and the drift and variance terms are

$$a(v_t, t) = \kappa(\bar{v} - v_t), \tag{5.10a}$$

$$b(v_t, t) = \sigma\sqrt{v_t}. \tag{5.10b}$$

2. The Stein & Stein model [107].
   The volatility $u_t := \sqrt{v_t}$ follows an Ornstein-Uhlenbeck process

$$\mathrm{d}u_t = \kappa(\bar{u} - u_t)\mathrm{d}t + \alpha \mathrm{d}W_t^2, \tag{5.11}$$

   and the variance v_t follows

$$\begin{aligned} \mathrm{d}v_t &= 2u_t \mathrm{d}u_t + \alpha^2 \mathrm{d}t, && \text{(5.12a)} \\ &= (2\kappa\bar{u}\sqrt{v_t} - 2\kappa v_t + \alpha^2)\mathrm{d}t + 2\alpha\sqrt{v_t}\mathrm{d}W_t^2, && \text{(5.12b)} \end{aligned}$$

   and the drift and variance terms are

$$\begin{aligned} a(v_t, t) &= 2\kappa\bar{u}\sqrt{v_t} - 2\kappa v_t + \alpha^2, && \text{(5.13a)} \\ b(v_t, t) &= 2\alpha\sqrt{v_t}. && \text{(5.13b)} \end{aligned}$$

3. The Scott model [106].
   The logarithm of square root of variance $y_t := \log\sqrt{v_t}$ follows an Ornstein-Uhlenbeck process

$$\mathrm{d}y_t = \kappa(\bar{y} - y_t)\mathrm{d}t + \beta \mathrm{d}W_t^2, \tag{5.14}$$

   and the variance v_t follows

$$\begin{aligned} \mathrm{d}v_t &= 2v_t\left[\kappa(\bar{y} - y_t)\mathrm{d}t + \beta\mathrm{d}W_t^2\right] + 2v_t\beta^2\mathrm{d}t, && \text{(5.15a)} \\ &= 2v_t\left[\kappa(\bar{y} - \log\sqrt{v_t}) + \beta^2\right]\mathrm{d}t + 2\beta v_t \mathrm{d}W_t^2, && \text{(5.15b)} \end{aligned}$$

   and the drift and variance terms are

$$\begin{aligned} a(v_t, t) &= 2v_t\left[\kappa(\bar{y} - \log\sqrt{v_t}) + \beta^2\right], && \text{(5.16a)} \\ b(v_t, t) &= 2\beta v_t. && \text{(5.16b)} \end{aligned}$$

4. The Hull & White model [64].
   The variance v_t follows a log-normal process

$$\mathrm{d}v_t = \mu_v v_t \mathrm{d}t + \sigma v_t \mathrm{d}W_t^2, \tag{5.17}$$

   and the drift and variance terms are

$$\begin{aligned} a(v_t, t) &= \mu_v v_t, && \text{(5.18a)} \\ b(v_t, t) &= \sigma v_t. && \text{(5.18b)} \end{aligned}$$

Then, we give theorems for each SV model introduced.

Corollary 5.1. *The Heston model is consistent with spot inversions. Namely, if a spot rate is modelled by a Heston process with parameters* $(\kappa, \bar{v}, \sigma, \rho, v_0)$*, then the*

inverse spot rate in the appropriate numéraire is modelled by a Heston model with parameters

$$(\kappa^*, \bar{v}^*, \sigma^*, \rho^*, v_0^*) = \left(\kappa - \rho\sigma, \frac{\kappa}{\kappa - \rho\sigma}\bar{v}, \sigma, -\rho, v_0\right). \tag{5.19}$$

Proof. This is a direct result of Theorem 5.1 by inserting $a(v_t, t) = \kappa(\bar{v} - v_t)$ and $b(v_t, t) = \sigma\sqrt{v_t}$ for Heston model into the Eq. (5.2b)

$$\begin{aligned} \mathrm{d}v_t^* &= \left[\kappa(\bar{v} - v_t^*) + \rho\sqrt{v_t^*}\sigma\sqrt{v_t^*}\right]\mathrm{d}t + \sigma\sqrt{v_t}\mathrm{d}\bar{W}_t^2, && (5.20a)\\ &= (\kappa - \rho\sigma)\left[\frac{\kappa}{\kappa - \rho\sigma}\bar{v} - v_t^*\right]\mathrm{d}t + \sigma\sqrt{v_t^*}\mathrm{d}\bar{W}_t^2, && (5.20b) \end{aligned}$$

which is the variance process of Heston model with parameters as in Eq. (5.19). □

Corollary 5.2. *The Stein & Stein model, Scott model, and Hull & White model as specified in Section 5.2 are not self-consistent under the inversion of spot rate.*

Proof.

1. For the Stein & Stein model, the function $a^*(v_t^*, t)$ is

$$\begin{aligned} a^*(v_t^*, t) &= a(v_t^*, t) + \rho\sqrt{v_t^*}b(v_t^*, t), && (5.21a)\\ &= 2\kappa\bar{u}^*\sqrt{v_t^*} - 2\kappa v_t^* + \alpha^2 + \rho\sqrt{v_t^*}2\alpha\sqrt{v_t^*}, && (5.21b)\\ &= 2\kappa\bar{u}^*\sqrt{v_t^*} + 2(\alpha\rho - \kappa)v_t^* + \alpha^2, && (5.21c) \end{aligned}$$

   which is of the form in Eq. (5.13a) only if $\alpha\rho = 0$.

2. For the Scott model, the function $a^*(v_t^*, t)$ is

$$\begin{aligned} a^*(v_t^*, t) &= a(v_t^*, t) + \rho\sqrt{v_t^*}b(v_t^*, t), && (5.22a)\\ &= \left[\kappa(\bar{y}^* - \log v_t^*) + \frac{1}{2}\beta^2\right]v_t^* + \rho\beta(v_t^*)^{\frac{3}{2}}, && (5.22b) \end{aligned}$$

   which contains the term $(v_t^*)^{3/2}$ and thus does not have the form in Eq. (5.16a).

3. For the Hull & White model, the function $a^*(v_t^*, t)$ is

$$\begin{aligned} a^*(v_t^*, t) &= a(v_t^*, t) + \rho\sqrt{v_t^*}b(v_t^*, t), && (5.23a)\\ &= \mu_v v_t^* + \rho\sigma(v_t^*)^{\frac{3}{2}}, && (5.23b) \end{aligned}$$

   which contains the term $(v_t^*)^{\frac{3}{2}}$ and thus does not have the form in Eq. (5.18a).

□

5.4 Consequence on variance swap pricing

A *variance swap* is a contract that pays out a linear function of the realized historical variance of the returns of an asset in a specified set of dates. An example of a fairly standard deal is a one year USDJPY variance swap paying 100,000 USD per volatility

point where the USDJPY rate is taken every business day at 4 p.m. London time from a given Reuters fixing page. A deal like this is quoted by giving a fair variance level in a similar fashion to how forwards and futures are dealt with.

Obviously the volatility exposure of a USDJPY variance swap paying a rebate in USD is different from that paying an equivalent amount in JPY. The reason for this is that the USDJPY spot is negatively correlated with its volatility. A variance swap paying in JPY is more valuable since a high realized variance scenario is likely to occur on JPY gaining value against USD. Inexperienced dealers can be arbitraged in this way. This observation can be verified by the previous conclusion on the Heston model as in the following analysis.

In mathematical modelling the variance is replaced by the continuously sampled variance which is also called quadratic variation, in the Heston model this is the stochastic variable

$$\mathrm{Var} = \frac{1}{T}\int_0^{\mathsf{T}} V_t \mathrm{d}t, \tag{5.24}$$

where T is the expiration date of the contract expressed in years. The fair level for a variance swap paying in term currency is the expectation of the annualized accrued variance which in the Heston model is

$$\mathbb{E}_{\mathrm{term}}\left(\frac{1}{T}\int_0^{\mathsf{T}} V_t \mathrm{d}t\right) = \bar{v} + (v_0 - \bar{v})\frac{1 - e^{-\kappa T}}{\kappa T}, \tag{5.25}$$

which is a number between the starting level of the variance v_0 and its mean reversion level $\bar{v}$.

For a variance swap paying in base currency we just need to invert the Heston process and apply this formula with Corollary 5.1, the level is therefore

$$\mathbb{E}_{\mathrm{base}}\left(\frac{1}{T}\int_0^{\mathsf{T}} V_t \mathrm{d}t\right) = \bar{v}^* + (v_0^* - \bar{v}^*)\frac{1 - e^{-\kappa^* T}}{\kappa^* T}, \tag{5.26a}$$

$$= \frac{\kappa}{\kappa - \rho\sigma}\bar{v} + \left(v_0^* - \frac{\kappa}{\kappa - \rho\sigma}\bar{v}\right)\frac{1 - e^{-(\kappa - \rho\sigma)T}}{(\kappa - \rho\sigma)T}, \tag{5.26b}$$

which is a number between the starting level v_0 and the modified mean reversion $\bar{v}\kappa/(\kappa - \rho\sigma)$. These calculations are in agreement with the observations above regarding USDJPY since in this case the correlation ρ is negative which depresses the value of the variance swap value when pricing in base currency.

5.5 Extension to the affine diffusion model

The *affine jump-diffusion* (AJD) model [39] assumes state vector $\boldsymbol{X}$ follows the process

$$\mathrm{d}\boldsymbol{X}_t = \boldsymbol{\mu}(\boldsymbol{X}_t)\mathrm{d}t + \boldsymbol{\sigma}(\boldsymbol{X}_t)\mathrm{d}\boldsymbol{Z}_t + \mathrm{d}\boldsymbol{J}_t, \tag{5.27}$$

where the drift vector $\boldsymbol{\mu}(\boldsymbol{X}_t)$, the instantaneous covariance matrix $\boldsymbol{\sigma}(\boldsymbol{X}_t)$, and the jump intensities all have affine dependence on $\boldsymbol{X}$. The Heston model is a particular

affine stochastic volatility model that belongs to the AJD class.

For our analysis, we make two assumptions on the AJD model. Firstly we assume there is no jump. Secondly we assume the first element in the vector $\boldsymbol{X}_t$ is the underlying price S_t under the FX settings. We work on the special form of AJD.

$$\mathrm{d}\begin{pmatrix} S_t \\ v_t \end{pmatrix} = \begin{pmatrix} (r_{\text{term}} - r_{\text{base}})S_t \\ \mu(v_t, t) \end{pmatrix} \mathrm{d}t + \begin{pmatrix} \sigma_{11}S_t & 0 \\ \sigma_{21} & \sigma_{22} \end{pmatrix} \begin{pmatrix} \mathrm{d}Z_t^1 \\ \mathrm{d}Z_t^2 \end{pmatrix}, \tag{5.28}$$

with $\mathrm{d}Z_t^1 \mathrm{d}Z_t^2 = 0$.

Theorem 5.2. *If an exchange rate S_t follows the process Eq. (5.28) in the risk-neutral measure associated to the term currency, then the inverse exchange rate $S_t^* := S_t^{-1}$ follows*

$$\mathrm{d}\begin{pmatrix} S_t^* \\ v_t^* \end{pmatrix} = \begin{pmatrix} (r_{\text{base}} - r_{\text{term}})S_t^* \\ \mu(v_t^*, t) + \sigma_{11}\sigma_{21} \end{pmatrix} \mathrm{d}t + \begin{pmatrix} \sigma_{11}S^* & 0 \\ -\sigma_{21} & \sigma_{22} \end{pmatrix} \begin{pmatrix} \mathrm{d}\bar{Z}_t^1 \\ \mathrm{d}\bar{Z}_t^2 \end{pmatrix}, \tag{5.29a}$$

with

$$\mathrm{d}\bar{Z}_t^1 \mathrm{d}\bar{Z}_t^2 = 0. \tag{5.29b}$$

Proof. Let S_t be the exchange rate, v_t be the variance of S_t. The vector (S_t, v_t) follows the AJD proecess Eq. (5.28), and $f(S_t, v_t, t)$ be a derivative contract with S_t as the underlying. Then the pricing PDE for $f(S_t, v_t, t)$ is

$$\begin{aligned} f_t + (r_{\text{term}} - r_{\text{base}})rSf_S + (\mu(v_t, t) + \alpha - \lambda\beta)\, f_v + \frac{1}{2}\sigma_{11}^2 S^2 f_{SS} \\ + \sigma_{11}\sigma_{21} S f_{Sv} + \frac{1}{2}(\sigma_{21}^2 + \sigma_{22}^2) f_{vv} = r_{\text{term}} f. \end{aligned} \tag{5.30}$$

See Appendix A for the deduction of the PDE (5.30).

Let $f^* := f/S$ be the price of this derivative in base currency. Then we deduce the PDE that is satisfied by f^* as a function of S_t^*. Apply the inverse in Eqs. (5.4), then Eq. (5.30) in terms of S_t^* is

$$\begin{aligned} f_t - (r_{\text{term}} - r_{\text{base}}) f_{S^*} S^* + [\mu(v_t, t) + (\alpha - \lambda\beta)]\, f_v + \frac{1}{2}\sigma_{11}^2 (f_{S^*S^*} S^{*2} + 2 f_{S^*} S^*) \\ - \sigma_{11}\sigma_{21} f_{S^*v} S^* + \frac{1}{2}(\sigma_{21}^2 + \sigma_{22}^2) f_{vv} = r_{\text{term}} f. \end{aligned} \tag{5.31}$$

Next, we change to $f^* = fS^*$ by applying Eqs. (5.6)

$$\begin{aligned} \frac{f_t^*}{S^*} - (r_{\text{term}} - r_{\text{base}}) S^* \left(\frac{1}{S^*} f_{S^*}^* - \frac{1}{S^{*2}} f^*\right) + [\mu(v_t, t) + (\alpha - \lambda\beta)] \frac{f_v^*}{S^*} \\ + \frac{1}{2}\sigma_{11}^2 \left[S^{*2} \left(\frac{1}{S^*} f_{S^*S^*}^* - \frac{2}{S^{*2}} f_{S^*}^* + \frac{2}{(S^*)^3} f^*\right) + 2S^* \left(\frac{1}{S^*} f_{S^*}^* - \frac{1}{S^{*2}} f^*\right) \right] \\ - \sigma_{11}\sigma_{21} S^* \left[\frac{1}{S^*} f_{S^*v}^* - \frac{1}{S^{*2}} f_v^* \right] + \frac{1}{2}(\sigma_{21}^2 + \sigma_{22}^2) \frac{f_{vv}^*}{S^*} = r_{\text{term}} \frac{f^*}{S^*}. \end{aligned} \tag{5.32}$$

After arranging terms, we reach

$$f_t^* - (r_{\text{term}} - r_{\text{base}})S^* f_{S^*}^* + [\mu(V_t, t) + (\alpha - \lambda\beta + \sigma_{11}\sigma_{21})]\, f_v^* + \frac{1}{2}\sigma_{11}^2 S^{*2} f_{S^*S^*}^* - \sigma_{11}\sigma_{21} S^* f_{S^*V}^* + \frac{1}{2}(\sigma_{21}^2 + \sigma_{22}^2) f_{vv}^* = r_{\text{base}} f^*, \quad (5.33)$$

which is the pricing PDE for the stochastic process Eqs. (5.29). □

5.6 Conclusion

We define and investigate the consistency of a class of stochastic volatility models under spot inversions, and hence their applicability in FX market. We give a general result for the parameters of SV model which is followed by the inverse spot rate. The Heston model, among the other members in the SV family, is the only one that we found to be consistent under the spot inversion. The result is further extended to the affine SV model. The conclusion on the Heston model verifies the arbitrage opportunity in variance swap trading.

Chapter 6

General Conclusion

The thesis contains four projects with the topic ranging from numerical optimisation to derivative pricing, which are dealt with in Project 2 and 4, respectively. The intersection of the two areas is presented in Project 1 and 3, reflecting the rising status of numerical optimisation technique in the financial industry. Each project has its own conclusion given at the end of the corresponding chapter. The conclusion in this chapter is brief and high-level.

The first work addresses an important practical issue by proposing a novel method to calibrate the Heston stochastic volatility (SV) model. The Heston model has been prevalently used in the industry for equity and foreign exchange (FX) markets. The calibration problem is as important as the model itself. Solving this issue is also important for local SV (LSV) models whose SV component is the Heston model. The work received a lot of interests from practitioners in the financial industry.

The linear programming (LP) problem is of significance in a wide range of applications. Most interior-point solvers for optimisation have computed search directions using direct solvers or iterative solvers with direct-type preconditioners. When the problem is getting larger and larger and the structure becoming dense, the iterative solver is the only choice to go. Project 2 implements an innovative interior-point method (IPM) entirely based on Krylov subspace methods for the LP problems, and succeeds for the first time in the history of IPM to solve an extensive set of benchmark problems from Netlib, Qaplib, and Mittelmann collections. The number of variables of the largest problem is 434,580. With the standard stopping criteria, the computational results show that this implementation outperforms the popular public-domain solvers SeDuMi and SDPT3, and is able to solve the rank-deficient problems which failed the interior-point solver of MOSEK, one of the most popular commercial solver.

Project 3 exhibits chaotic phenomena of the Newton-Raphson calculation of the implied volatility (IV) consistent with the given price of a European option. The computation is performed millions of times in trading and risk management systems throughout the financial industry every day. This work allows IV to be complex, and shows that the function that yields the parameter has singularity at zero, which

reflects the put-call parity, and that at infinity. The Newton-Raphson hunting of a successively better approximation to the IV is described via the fractal tool.

SV models have been extensively used by FX practitioners. Project 4 investigates the feasibility of such application from the perspective of model consistency under spot inversion. A conclusion on the general form of SV model is given, so that further research may be conducted on the consistency of any SV model and on the relation between consistency and the functional form of drift and variance terms of SV process. The analysis shows that the Heston model is the only one among the studied SV models that is consistent when applied to model FX exchange rate.

Financial mathematics and numerical methods are meaningful as they are closely connected with practice. The thesis makes an effort to deliver research that is of practical meaning and hopefully it does.

Bibliography

[1] I. Adler, M. Resende, G. Veiga, and N. Karmarkar, *An implementation of Karmarkar's algorithm for linear programming*, Mathematical Programming, 44 (1989), pp. 297–335, http://dx.doi.org/10.1007/BF01587095.

[2] L. Ahlfors, *Complex Analysis: an Introduction to the Theory of Analytic Functions of One Complex Variable*, McGraw-Hill, New York, 3rd ed., 1979, http://dx.doi.org/10.1137/1022075.

[3] Y. Aït-Sahalia and R. Kimmel, *Maximum likelihood estimation of stochastic volatility models*, Journal of Financial Economics, 83 (2007), pp. 413–452, http://dx.doi.org/10.1016/j.jfineco.2005.10.006.

[4] H. Albrecher, P. Mayer, W. Schoutens, and J. Tistaert, *The little Heston trap*, Wilmott Magazine, January (2007), pp. 83–92, http://www.wilmott.com/pdfs/121218_heston.pdf.

[5] E. Andersen and K. Andersen, *Presolving in linear programming*, Mathematical Programming, 71 (1995), pp. 221–245, http://dx.doi.org/10.1007/BF01586000.

[6] E. Andersen, J. Gondzio, C. Mészáros, and X. Xu, *Implementation of interior-point methods for large scale linear programs*, in Interior Point Methods of Mathematical Programming, P. M. Pardalos and D. Hearn, eds., vol. 5 of Applied Optimization, Kluwer Academic Publishers, Dordrecht, 1996, http://dx.doi.org/10.1007/978-1-4613-3449-1_6.

[7] L. Andersen, *Simple and efficient simulation of the Heston stochastic volatility model*, Journal of Computational Finance, 11 (2008), pp. 1–42, http://www.risk.net/journal-of-computational-finance/technical-paper/2160370/simple-efficient-simulation-heston-stochastic-volatility-model.

[8] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 3 ed., 1999, http://dx.doi.org/10.1137/1.9780898719604.

[9] R. Bauer, *Fast calibration in the Heston model*, 2012, http://www.fam.tuwien.ac.at/~sgerhold/pub_files/theses/bauer.pdf. Master's thesis, Vienna University of Technology.

[10] L. Bergamaschi, J. Gondzio, and G. Zilli, *Preconditioning indefinite systems in interior point methods for optimization*, Computational Optimization and Applications, 28 (2004), pp. 149–171, http://dx.doi.org/10.1023/B:COAP.0000026882.34332.1b.

[11] Å. Björck and T. Elfving, *Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations*, BIT Numerical Mathematics, 19 (1979), pp. 145–163, http://dx.doi.org/10.1007/BF01930845.

[12] F. Black and M. Scholes, *The pricing of options and corporate liabilities*, Journal of Political Economy, 81 (1973), pp. 631–659, http://dx.doi.org/10.1086/260062.

[13] J. Bunch, L. Kaufman, and B. Parlett, *Decomposition of a symmetric matrix*, Numerische Mathematik, 27 (1976), pp. 95–109, http://dx.doi.org/10.1007/BF01399088.

[14] T. Carpenter and D. Shanno, *An interior point method for quadratic programs based on conjugate projected gradients*, Computational Optimization and Applications, 2 (1993), pp. 5–28, http://dx.doi.org/10.1007/BF01299140.

[15] A. Cayley, *The Newton-Fourier imaginary problem*, American Journal of Mathematics, 2 (1879), pp. 97–97, http://dx.doi.org/10.2307/2369201.

[16] B. Chen, *Calibration of the Heston model with application in derivative pricing and hedging*, 2007, http://resolver.tudelft.nl/uuid:25dc8109-4b39-44b8-8722-3fd680c7c4ad. Master's thesis, Technical University of Delft.

[17] S. Chen, D. Donoho, and M. Saunders, *Atomic decomposition by basis pursuit*, SIAM Journal on Scientific Computing, 20 (1998), pp. 33–61, http://dx.doi.org/10.1137/S003614450037906X.

[18] P. Chin and A. Vannelli, *PCG techniques for interior point algorithms*, in Proceedings of the 36th midwest symposium on circuits and systems, IEEE, 1994, pp. 200–203, http://dx.doi.org/10.1109/MWSCAS.1993.343095.

[19] P. Christoffersen, S. Heston, and K. Jacobs, *The shape and term structure of the index option smirk: Why multifactor stochastic volatility models work so well*, Management Science, 55 (2009), pp. 1914–1932, http://www.jstor.org/stable/40539255.

[20] P. Christoffersen and K. Jacobs, *The importance of the loss function in option valuation*, Journal of Financial Economics, 72 (2004), pp. 291–318, http://dx.doi.org/10.1016/j.jfineco.2003.02.001.

[21] I. J. Clark, *Foreign Exchange Option Pricing: A Practitioner's Guide*, Wiley, Chichester, 2011, http://dx.doi.org/10.1002/9781119208679.ch10.

[22] R. Cont and P. Tankov, *Financial Modelling with Jump Processes*, vol. 2 of Financial Mathematics Series, Chapman and Hall/CRC, London, 2003, http://dx.doi.org/10.1201/9780203485217.

[23] E. Craig, *The N-step iteration procedures*, Journal of Mathematical Physics, 34 (1995), pp. 64–73, http://dx.doi.org/10.1002/sapm195534164.

[24] X. Cui, *Approximate Generalized Inverse Preconditioning Methods for Least Squares Problems*, PhD thesis, Department of Informatics, School of Multidisciplinary Sciences, The Graduate University for Advanced Studies (SOKENDAI), 2009, http://id.nii.ac.jp/1013/00001492/.

[25] X. Cui, K. Hayami, and J.-F. Yin, *Greville's method for preconditioning least squares problems*, Advances in Computational Mathematics, 35 (2011), pp. 243 – 269, http://dx.doi.org/10.1007/s10444-011-9171-x.

[26] Y. Cui, S. del Baño Rollin, and G. Germano, *Full and fast calibration of the Heston stochastic volatility model*, 2015. Preprint available at https://arxiv.org/abs/1511.08718v2.

[27] Y. Cui, K. Morikuni, T. Tsuchiya, and K. Hayami, *Implementation of interior-point methods for LP based on Krylov subspace iterative solvers with inner-iteration preconditioning*, 2016. Preprint available at http://www.nii.ac.jp/TechReports/public_html/16-003E.html, http://www.optimization-online.org/DB_HTML/2016/04/5421.html and https://arxiv.org/abs/1604.07491v2.

[28] J. H. Curry, L. Garnett, and D. Sullivan, *On the iteration of a rational function: computer experiments with Newton's method*, Communications in Mathematical Physics, 91 (1983), pp. 267–277, http://dx.doi.org/10.1007/BF01211162.

[29] J. Czyzyk, S. Mehrotra, M. Wagner, and S. Wright, *PCx: An interior-point code for linear programming*, Optimization Methods and Software, 11 (1999), pp. 397–430, http://dx.doi.org/10.1080/10556789908805757.

[30] G. Dahl and K. R. Dahl, *Linear optimization and mathematical finance*, 2012. Preprint available at http://www.uio.no/studier/emner/matnat/math/MAT2700/h12/dualitetmat2700.pdf.

[31] M. D'Apuzzo, V. De Simone, and D. Di Serafino, *On mutual impact of numerical linear algebra and large-scale optimization with focus on interior point methods*, Computational Optimization and Applications, 45 (2010), pp. 283–310, http://dx.doi.org/10.1007/s10589-008-9226-1.

[32] T. Davis, *CSparse: A concise sparse matrix package*, 2014, http://www.suitesparse.com. Version 3.1.4.

[33] T. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Transactions on Mathematical Software, 38 (2011), pp. 1:1–1:25, http://www.cise.ufl.edu/research/sparse/matrices/.

[34] A. Dax, *The convergence of linear stationary iterative processes for solving singular unstructured systems of linear equations*, SIAM Review, 32 (1990), pp. 611–635, http://dx.doi.org/10.1137/1032122.

[35] S. del Baño Rollin, A. Ferreiro-Castilla, and F. Utzet, *On the density of log-spot in the Heston volatility model*, Stochastic Processes and their Applications, 120 (2010), pp. 2037–2063, http://dx.doi.org/10.1016/j.spa.2010.06.003.

[36] F. Devernay, *C/C++ MINPACK*, 2007, http://devernay.free.fr/hacks/cminpack.

[37] E. Dolan and J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming, 91 (2002), pp. 201–213, http://dx.doi.org/10.1007/s101070100263.

[38] I. Duff, *MA57 - a new code for the solution of sparse symmetric definite systems*, ACM Transactions on Mathematical Software, 30 (2004), pp. 118–144, http://dx.doi.org/10.1145/992200.992202.

[39] D. Duffie and J. Pan, *Transform analysis and asset pricing for affine jump-diffusions*, Econometrica, 68 (2000), pp. 1343–1376, http://dx.doi.org/10.1111/1468-0262.00164.

[40] L. Fatone, F. Mariani, M. C. Recchioni, and F. Zirilli, *The calibration of some stochastic volatility models used in mathematical finance*, Open Journal of Applied Sciences, 4 (2014), pp. 23–33, http://dx.doi.org/10.4236/ojapps.2014.42004.

[41] J. Fernández, A. Ferreiro, J. García-Rodríguez, A. Leitao, J. López-Salas, and C. Vázquez, *Static and dynamic SABR stochastic volatility models: Calibration and option pricing using GPUs*, Mathematics and Computers in Simulation, 94 (2013), pp. 55–75, http://dx.doi.org/10.1016/j.matcom.2013.05.007.

[42] M. Ferris and T. Munson, *Interior-point methods for massive support vector machines*, SIAM Journal on Optimization, 13 (2002), pp. 783–804, http://dx.doi.org/10.1137/S1052623400374379.

[43] H. Fettis, J. Caslin, and K. Cramer, *Complex zeros of the error function and of the complementary error function*, Mathematics of Computation, 27 (1973), pp. 401–407.

[44] R. Fourer and S. Mehrotra, *Solving symmetric indefinite systems in an interior-point method for linear programming*, Mathematical Programming, 62 (1993), pp. 15–39, http://dx.doi.org/10.1007/BF01585158.

[45] R. Freund and F. Jarre, *A QMR-based interior-point algorithm for solving linear programs*, Mathematical Programming, 76 (1997), pp. 183–210, http://dx.doi.org/10.1007/BF02614383.

[46] R. Freund, F. Jarre, and S. Mizuno, *Convergence of a class of inexact interior-point algorithms for linear programs*, Mathematics of Operations Research, 24 (1999), pp. 50–71, http://dx.doi.org/10.1287/moor.24.1.50.

[47] G. Fusai and A. Roncoroni, *Implementing Models in Quantitative Finance: Methods and Cases*, Springer-Verlag Berlin, Philadelphia, PA, 2008, http://dx.doi.org/10.1007/978-3-540-49959-6.

[48] J. Gatheral, *The Volatility Surface: A Practitioner's Guide*, vol. 357 of Wiley Finance, Wiley, New Jersey, 2006, http://dx.doi.org/10.1002/9781119202073.ch3.

[49] F. Gerlich, A. Giese, J. Maruhn, and E. Sachs, *Parameter identification in financial market models with a feasible point SQP algorithm*, Computational Optimization and Applications, 51 (2012), pp. 1137–1161, http://dx.doi.org/10.1007/s10589-010-9369-8.

[50] P. Gill, W. Murray, M. Saunders, J. Tomlin, and M. Wright, *On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method*, Mathematical Programming, 36 (1986), pp. 183–209, http://dx.doi.org/10.1007/BF02592025.

[51] M. Gilli and E. Schumann, *Calibrating option pricing models with heuristics*, in Natural Computing in Computational Finance, A. Brabazon, M. O'Neill, and D. Maringer, eds., vol. 380 of Studies in Computational Intelligence, Springer, Berlin, 2012, pp. 9–37, http://dx.doi.org/10.1007/978-3-642-23336-4_2.

[52] M. Gilli and E. Schumann, *Heuristic optimisation in financial modelling*, Annals of Operations Research, 193 (2012), pp. 129–158, http://dx.doi.org/10.1007/s10479-011-0862-y.

[53] P. Glasserman and K. Kim, *Gamma expansion of the Heston stochastic volatility model*, Finance and Stochastics, 15 (2011), pp. 267–296, http://dx.doi.org/10.1007/s00780-009-0115-y.

[54] J. Gondzio, *Multiple centrality corrections in a primal-dual method for linear programming*, Computational Optimization and Applications, 6 (1996), pp. 137–156, http://dx.doi.org/10.1007/BF00249643.

[55] J. Gondzio, *Presolve analysis of linear programs prior to applying an interior point method*, INFORMS Journal on Computing, 9 (1997), pp. 73–91, http://dx.doi.org/10.1287/ijoc.9.1.73.

[56] J. Gondzio, *Interior point methods 25 years later*, European Journal of Operational Research, 218 (2012), pp. 587–601, http://dx.doi.org/10.1016/j.ejor.2011.09.017.

[57] J. Gondzio, *Matrix-free interior point method*, Computational Optimization and Applications, 51 (2012), pp. 457–480, http://dx.doi.org/10.1007/s10589-010-9361-3.

[58] J. Gondzio and T. Terlaky, *A computational view of interior point methods*, in Advances in Linear and Integer Programming, J. E. Beasley, ed., Oxford University Press: Oxford, 1996, pp. 103–144, http://dl.acm.org/citation.cfm?id=247975.247978.

[59] M. Grant and S. Boyd, *Graph implementations for nonsmooth convex programs*, in Recent Advances in Learning and Control, V. Blondel, S. Boyd, and H. Kimura, eds., Lecture Notes in Control and Information Sciences, Springer-Verlag Limited, 2008, pp. 95–110, http://dx.doi.org/10.1007/978-1-84800-155-8_7.

[60] M. Grant and S. Boyd, *CVX: MATLAB software for disciplined convex programming*, March 2014, http://cvxr.com/cvx. version 2.1.

[61] K. Hayami, J.-F. Yin, and T. Ito, *GMRES methods for least squares problems*, SIAM Journal on Matrix Analysis and Applications, 31 (2010), pp. 2400–2430, http://dx.doi.org/10.1137/070696313.

[62] M. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, Journal of Research of the National Bureau of Standards, 49 (1952), pp. 409–436, http://dx.doi.org/10.6028/jres.049.044.

[63] S. L. Heston, *A closed-form solution for options with stochastic volatility and applications to bond and currency options*, Review of Financial Studies, 6 (1993), pp. 327–343, http://dx.doi.org/10.1093/rfs/6.2.327.

[64] J. Hull and A. White, *The pricing of options on assets with stochastic volatilities*, The Journal of Finance, 42 (1987), pp. 281–300, http://dx.doi.org/10.1111/j.1540-6261.1987.tb02568.x.

[65] J. C. Hull, *Options, Futures, and Other Derivatives*, Pearson, Boston, 9 ed., 2014.

[66] A. Hurn, K. Lindsay, and A. McClelland, *Estimating the parameters of stochastic volatility models using option price data*, Journal of Business and Economic Statistics, 33 (2015), pp. 579–594, http://dx.doi.org/10.1080/07350015.2014.981634.

[67] A. Jacquier and C. Martini, *Heston 2010*, 2011, http://dx.doi.org/10.2139/ssrn.1769744. Preprint.

[68] P. Jäeckel, *By implication*, Wilmott Magazine, November (2006), pp. 60–66, http://jaeckel.16mb.com/ByImplication.pdf.

[69] A. Janek, T. Kluge, R. Weron, and U. Wystup, *FX smile in the Heston model*, in Statistical Tools for Finance and Insurance, P. Cizek, W. K. Härdle, and R. Weron, eds., Berlin, 2011, Springer, pp. 133–162, http://dx.doi.org/10.1007/978-3-642-18062-0_4.

[70] J. Júdice, J. Patricio, L. Portugal, M. Resende, and G. Veiga, *A study of preconditioners for network interior point methods*, Computational Optimization and Applications, 24 (2003), pp. 5–35, http://dx.doi.org/10.1023/A:1021882330897.

[71] K. Morikuni and K. Hayami, *AB-GMRES preconditioned by NE-SOR inner iterations*, 2014, http://researchmap.jp/KeiichiMorikuni/Implementations/.

[72] C. Kahl and P. Jäckel, *Not-so-complex logarithms in the Heston model*, Wilmott Magazine, September (2005), pp. 94–103, http://www.wilmott.com/pdfs/110208_heston.pdf.

[73] N. Karmarkar and K. Ramakrishnan, *Computational results of an interior point algorithm for large scale linear programming*, Mathematical Programming, 52 (1991), pp. 555–586, http://dx.doi.org/10.1007/BF01582905.

[74] M. Kojima, S. Mizuno, and A. Yoshise, *A polynomial-time algorithm for a class of linear complementarity problems*, Mathematical Programming, 4 (1989), pp. 1–26, http://dx.doi.org/10.1007/BF01587074.

[75] J. Korzak, *Convergence analysis of inexact infeasible-interior-point algorithms for solving linear programming problems*, SIAM Journal on Optimization, 11 (2000), pp. 133–148, http://dx.doi.org/10.1137/S1052623497329993.

[76] M. I. A. Lourakis, *levmar: Levenberg-Marquardt nonlinear least squares algorithms in C/C++*, July 2004, http://www.ics.forth.gr/~lourakis/levmar.

[77] M. I. A. Lourakis, *Sparse non-linear least squares optimization for geometric vision*, in Computer Vision–ECCV 2010, K. Daniilidis, P. Maragos, and N. Paragios, eds., vol. 6311 of Image Processing, Computer Vision, Pattern Recognition, and Graphics, Berlin, 2010, Springer, pp. 43–56, http://dx.doi.org/10.1007/978-3-642-15552-9_4.

[78] I. Lustig, R. Marsten, and D. Shanno, *On implementing Mehrotra's predictor-corrector interior-point method for linear programming*, SIAM Journal on Optimization, 2 (1992), pp. 435–449, http://dx.doi.org/10.1137/0802022.

[79] I. Lustig, R. Marsten, and D. Shanno, *Interior point methods for linear programming: Computational state of the art*, ORSA Journal on Computing, 6 (1994), pp. 1–14, http://dx.doi.org/10.1287/ijoc.6.1.1.

[80] S. Mehrotra, *Implementations of affine scaling methods: approximate solutions of systems of linear equations using preconditioned conjugate gradient methods*, ORSA Journal on Computing, 4 (1992), pp. 103–118, http://dx.doi.org/10.1287/ijoc.4.2.103.

[81] S. Mehrotra, *On the implementation of a primal-dual interior point method*, SIAM Journal on Optimization, 2 (1992), pp. 575–601, http://dx.doi.org/10.1137/0802028.

[82] S. Mehrotra and Z. Li, *Convergence conditions and Krylov subspace-based corrections for primal-dual interior-point method*, SIAM Journal on Optimization, 15 (2005), pp. 635–653, http://dx.doi.org/10.1137/S1052623403431494.

[83] S. Mehrotra and J. Wang, *Conjugate gradient based implementation of interior point methods for network flow problems*, in Linear and nonlinear conjugate gradient-related methods, L. Adams and J. Nazareth, eds., SIAM, Philadelphia, PA, 1996, pp. 124–142, https://www.researchgate.net/publication/236626908_Conjugate_gradient_based_implementation_of_interior_point_methods_for_network_flow_problems.

[84] R. C. Merton, *Theory of rational option pricing*, The Bell Journal of Economics and Management Science, 4 (1973), pp. 141–183, http://dx.doi.org/10.2307/3003143.

[85] S. Mikhailov and U. Nögel, *Heston's stochastic volatility model: Implementation, calibration and some extensions*, Wilmott Magazine, July (2003), pp. 74–79, http://www.wilmott.com/pdfs/051111_mikh.pdf.

[86] J. W. Milnor, *Dynamics in one complex variable*, vol. 160, Springer, Wiesbaden, 2006, http://dx.doi.org/10.1007/978-3-663-08092-3.

[87] R. Monteiro and I. Adler, *Interior path following primal-dual algorithms. Part I: Linear programming*, Mathematical Programming, 44 (1989), pp. 27–41, http://dx.doi.org/10.1007/BF01587075.

[88] R. Monteiro and J. O'Neal, *Convergence analysis of a long-step primal-dual infeasible interior-point LP algorithm based on iterative linear solvers*, tech. report, Georgia Institute of Technology, 2003, http://www.optimization-online.org/DB_FILE/2003/10/768.pdf.

[89] R. Monteiro, J. O'Neal, and T. Tsuchiya, *Uniform boundedness of a preconditioned normal matrix used in interior-point methods*, SIAM Journal on Optimization, 15 (2004), pp. 96–100, http://dx.doi.org/10.1137/S1052623403426398.

[90] N. Moodley, *The Heston model: A practical approach with MATLAB code*, 2009, http://math.nyu.edu/~atm262/fall06/compmethods/a1/nimalinmoodley.pdf. Bachelor's thesis, University of the Witwatersrand.

[91] J. J. Moré, *The Levenberg-Marquardt algorithm: implementation and theory*, in Numerical Analysis, G. A. Watson, ed., vol. 630 of Lecture Notes in Mathematics, Berlin, 1978, Springer, pp. 105–116, http://dx.doi.org/10.1007/BFb0067700.

[92] K. Morikuni, *Symmetric inner-iteration preconditioning for rank-deficient least squares problems*, 2015, arXiv:1504.00889 [math.NA].

[93] K. Morikuni and K. Hayami, *Inner-iteration Krylov subspace methods for least squares problems*, SIAM Journal on Matrix Analysis and Applications, 34 (2013), pp. 1–22, http://dx.doi.org/10.1137/110828472.

[94] K. Morikuni and K. Hayami, *Convergence of inner-iteration GMRES methods for rank-deficient least squares problems*, SIAM Journal on Matrix Analysis and Applications, 36(1) (2015), pp. 225–250, http://dx.doi.org/10.1137/130946009.

[95] MOSEK ApS, *The MOSEK optimization toolbox for MATLAB manual*, 2015, http://docs.mosek.com/7.0/toolbox/. Version 7.1 (Revision 35).

[96] A. Oliveira and D. Sorensen, *A new class of preconditioners for large-scale linear systems from interior point methods for linear programming*, Linear Algebra and its Applications, 394 (2005), pp. 1–24, http://dx.doi.org/10.1016/j.laa.2004.08.019.

[97] C. Paige and M. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM Journal on Numerical Analysis, (1975), pp. 617–629, http://dx.doi.org/10.1137/0712047.

[98] H.-O. Peitgen and P. H. Richter, *The Beauty of Fractals: Image of Complex Dynamical Systems*, Springer-Verlag, Berlin, 1986, http://dx.doi.org/10.1007/978-3-642-61717-1.

[99] H.-O. Peitgen, D. Saupe, and F. v. Haeseler, *Cayley's problem and Julia sets*, Mathematical Intelligencer, 6 (1984), pp. 11–20, http://dx.doi.org/10.1007/BF03024150.

[100] L. Portugal, M. Resende, G. Veiga, and J. Júdice, *A truncated primal-infeasible dual-feasible network interior point method*, Networks, 35 (2000), pp. 91–108, http://dx.doi.org/10.1002/(SICI)1097-0037(200003)35:2<91::AID-NET1>3.0.CO;2-T.

[101] R. Rebonato, *Volatility and Correlation: The Perfect Hedger and the Fox*, John Wiley, Chichester, 2 ed., 2004, http://dx.doi.org/10.1002/9781118673539.

[102] M. Resende and G. Veiga, *An implementation of the dual affine scaling algorithm for minimum-cost flow on bipartite uncapacitated networks*, SIAM Journal on Optimization, 3 (1993), pp. 516–537, http://dx.doi.org/10.1137/0803025.

[103] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, 2nd ed., 2003, http://dx.doi.org/10.1137/1.9780898718003.

[104] Y. Saad and M. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 7 (1986), pp. 856–869, http://dx.doi.org/10.1137/0907058.

[105] W. Schoutens, E. Simons, and J. Tistaert, *A perfect calibration! Now what?*, Wilmott Magazine, March (2004), pp. 66–78, http://www.wilmott.com/pdfs/070319_schoutens.pdf.

[106] L. O. Scott, *Option pricing when the variance changes randomly: Theory, estimation, and an application*, Journal of Financial and Quantitative Analysis, 22 (1987), pp. 419–438, http://dx.doi.org/10.2307/2330793.

[107] E. M. Stein and J. C. Stein, *Stock price distributions with stochastic volatility: an analytic approach*, Review of Financial Studies, 4 (1991), pp. 727–752, http://dx.doi.org/10.1093/rfs/4.4.727.

[108] J. Sturm, *Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones*, Optimization Methods and Software, 11-12 (1999 Special issue on Interior Point Methods), pp. 625–633, http://dx.doi.org/10.1080/10556789908805766.

[109] K. Tanabe, *Centered Newton method for mathematical programming*, in System Modeling and Optimization, vol. 113 of Lecture Notes in Control and Information Sciences, Springer-Verlag, 1988, pp. 197–206, http://dx.doi.org/10.1007/BFb0042787.

[110] K. Toh, M. Todd, and R. Tütüncü, *SDPT3 — a MATLAB software package for semidefinite programming*, Optimization Methods and Software, 11 (1999), pp. 545–581, http://dx.doi.org/10.1080/10556789908805762.

[111] R. Tütüncü, K. Toh, and M. Todd, *Solving semidefinite-quadratic-linear programs using SDPT3*, Mathematical Programming Series B, 95 (2003), pp. 189–217, http://dx.doi.org/10.1007/s10107-002-0347-5.

[112] P. Wilmott, *Paul Wilmott Introduces Quantitative Finance*, John Wiley, Chichester, 2 ed., 2007.

[113] S. Wright, *Primal-Dual Interior-Point Methods*, SIAM, 1997, http://dx.doi.org/10.1137/1.9781611971453.

[114] S. Wright, *Modified Cholesky factorizations in interior-point algorithms for linear programming*, SIAM Journal on Optimization, 9 (1999), pp. 1159–1191, http://dx.doi.org/10.1137/S1052623496304712.

[115] Y. Zhang, *On the convergence of a class of infeasible interior-point methods for the horizontal linear complementarity problem*, SIAM Journal on Optimization, 4 (1994), pp. 208–227, http://dx.doi.org/10.1137/0804012.

[116] Y. Zhang, *Solving large-scale linear programs by interior-point methods under the MATLAB environment*, Optimization Methods and Software, 10 (1998), pp. 1–31, http://dx.doi.org/10.1080/10556789808805699.

Appendix A

Derivation of the pricing PDE for affine diffusion model under foreign exchange settings

Let S_t be the exchange rate, V_t be the variance of S_t. The vector (S_t, V_t) follows the AJD proecess (5.28) with zero-jump, and $f(S_t, V_t, t)$ be a derivative contract with S_t as the underlying. The dynamic of $f(S_t, V_t, t)$ is

$$\begin{aligned} \mathrm{d}f = [f_t + (r_{\text{term}} - r_{\text{base}})Sf_S + \mu(V_t, t)f_V + \\ \frac{1}{2}\left(\sigma_{11}^2 S^2 f_{SS} + 2\sigma_{11}\sigma_{21} S f_{SV} + (\sigma_{21}^2 + \sigma_{22}^2) f_{VV}\right)\Big] \mathrm{d}t + \\ (\sigma_{11} S f_S + \sigma_{21} f_V)\mathrm{d}Z_t^1 + \sigma_{22} f_V \mathrm{d}Z_t^2. \end{aligned} \tag{A.1}$$

For simplicity, let

$$\begin{aligned} \mathcal{M} := f_t + (r_{\text{term}} - r_{\text{base}})Sf_S + \mu(V_t, t)f_V + \\ \frac{1}{2}\left(\sigma_{11}^2 S^2 f_{SS} + 2\sigma_{11}\sigma_{21} S f_{SV} + (\sigma_{21}^2 + \sigma_{22}^2) f_{VV}\right) \end{aligned} \tag{A.2}$$

be the factor multiplied by $\mathrm{d}t$, and hence

$$\mathrm{d}f = \mathcal{M}\mathrm{d}t + (\sigma_{11} S f_S + \sigma_{21} f_V)\mathrm{d}W_t^1 + \sigma_{22} f_V \mathrm{d}W_t^2. \tag{A.3}$$

When constructing the replicating portfolio of $f(S_t, V_t, t)$, we need to note that the exchange rate S_t is not in itself a trade-able quantity. The closest trade-able quantity is the term currency value of a base currency bond: $B_t^{(b)} S_t$. Consider portfolio Π constituted with 1 unit of the derivative $f(S_t, V_t, t)$, Δ unit of $B_t^{(b)} S_t$, $\tilde{\Delta}$ unit of another derivative $\tilde{f}(S_t, V_t, t)$ with the same underlying.

$$\Pi = f - \Delta B_t^{(b)} S_t - \tilde{\Delta}\tilde{f}. \tag{A.4}$$

The dynamics of Π is

$$\begin{aligned}
\mathrm{d}\Pi &= \mathrm{d}f - \Delta \mathrm{d}(B^{(b)}S) - \tilde{\Delta}\mathrm{d}\tilde{f}, \\
&= \mathrm{d}f - \Delta B^{(b)} S_t \left(r_{\text{term}}\mathrm{d}t + \sigma_{11}\mathrm{d}W_t^1\right) - \tilde{\Delta}\mathrm{d}\tilde{f}, \\
&= \mathcal{M}\mathrm{d}t + (\sigma_{11}Sf_S + \sigma_{21}f_V)\mathrm{d}W_t^1 + \sigma_{22}f_V\mathrm{d}W_t^2 - \Delta B^{(b)}S_t(r_{\text{term}}\mathrm{d}t + \sigma_{11}\mathrm{d}W_t^1) \\
&\quad - \tilde{\Delta}\left[\tilde{M}\mathrm{d}t + (\sigma_{11}S\tilde{f}_S + \sigma_{21}\tilde{f}_V)\mathrm{d}W_t^1 + \sigma_{22}\tilde{f}_V\mathrm{d}W_t^2\right], \\
&= \left(\mathcal{M} - \Delta B^{(b)}S_t r_{\text{term}} - \tilde{\Delta}\tilde{\mathcal{M}}\right)\mathrm{d}t + \sigma_{22}(f_V - \tilde{\Delta}\tilde{f}_V)\mathrm{d}W_t^2 \\
&\quad + \left[\sigma_{11}Sf_S + \sigma_{21}f_V - \tilde{\Delta}(\sigma_{11}S\tilde{f}_S + \sigma_{21}\tilde{f}_V) - \Delta B^{(b)}S_t\sigma_{11}\right]\mathrm{d}W_t^1. \qquad \text{(A.5)}
\end{aligned}$$

By no arbitrage, the portfolio Π is (term) risk-free and the coefficient of the Brownian motions should be zero. Mathematically it holds that

$$\sigma_{11}Sf_S + \sigma_{21}f_V - \tilde{\Delta}(\sigma_{11}S\tilde{f}_S + \sigma_{21}\tilde{f}_V) - \Delta B^{(b)}S\sigma_{11} = 0, \qquad \text{(A.6a)}$$

$$\sigma_{22}(f_V - \tilde{\Delta}\tilde{f}_V) = 0, \qquad \text{(A.6b)}$$

$$\begin{aligned}
\mathrm{d}\Pi &= r_{\text{term}}\Pi\mathrm{d}t \\
&= r_{\text{term}}(f - \Delta B^{(b)}S - \tilde{\Delta}\tilde{f})\mathrm{d}t.
\end{aligned} \qquad \text{(A.6c)}$$

Then from (A.6b), we get $\tilde{\Delta}$

$$\tilde{\Delta} = \frac{f_V}{\tilde{f}_V}. \qquad \text{(A.7)}$$

Δ can be obtained from (A.6a)

$$\begin{aligned}
\sigma_{11}Sf_S - \frac{f_V}{\tilde{f}_V}S\sigma_{11}\tilde{f}_S - \Delta B^{(b)}S\sigma_{11} &= 0, \\
f_S - \frac{f_V\tilde{f}_S}{\tilde{f}_V} - \Delta B^{(b)} &= 0, \\
\Delta &= \frac{f_S - \tilde{f}_S\tilde{\Delta}}{B^{(b)}}. \qquad \text{(A.8)}
\end{aligned}$$

Substituting the expression of Δ and $\tilde{\Delta}$ into (A.5)

$$\begin{aligned}
d\Pi &= \left[\mathcal{M} - \Delta B^{(t)}Sr_{\text{term}} - \tilde{\Delta}\tilde{\mathcal{M}}\right]\mathrm{d}t, \\
&= \left[\mathcal{M} - (f_S - \tilde{f}_S\tilde{\Delta})Sr_{\text{term}} - \tilde{\Delta}\tilde{\mathcal{M}}\right]\mathrm{d}t. \qquad \text{(A.9)}
\end{aligned}$$

Equate and (A.9) and (A.6c),

$$\begin{aligned}
\mathcal{M} - (f_S - \tilde{f}_S\tilde{\Delta})Sr_{\text{term}} - \tilde{\Delta}\tilde{\mathcal{M}} &= r_{\text{term}}(f - \Delta B^{(t)}S - \tilde{\Delta}\tilde{f}), \\
\mathcal{M} - (f_S - \tilde{f}_S\frac{f_V}{\tilde{f}_V})Sr_{\text{term}} - \frac{f_V}{\tilde{f}_V}\tilde{M} &= r_{\text{term}}(f - (f_S - \tilde{f}_S\frac{f_V}{\tilde{f}_V})S - \frac{f_V}{\tilde{f}_V}\tilde{f}), \\
\frac{\mathcal{M} - r_{\text{term}}f}{f_V} &= \frac{\tilde{\mathcal{M}} - r_{\text{term}}\tilde{f}}{\tilde{f}_V}, \qquad \text{(A.10)}
\end{aligned}$$

which holds independent of the expression of derivative. Assume that

$$\frac{\mathcal{M} - r_{\text{term}} f}{f_V} = g(S_t, V_t, t), \tag{A.11}$$

where $g(S_t, V_t, t) = -(\alpha - \lambda\beta)$, $\lambda := \frac{g(S_t, V_t, t) + \alpha}{\beta}$ is called market price of volatility risk and is often assumed to be proportional to V, that is $\lambda = \theta v$.Substitute the definition of $\mathcal{M}$ and $g(S_t, V_t, t)$ into (A.11), we have the pricing PDE

$$f_t + (r_{\text{term}} - r_{\text{base}}) S f_S + [\mu(V_t, t) + (\alpha - \lambda\beta)] f_V + \frac{1}{2}\sigma_{11}^2 S^2 f_{SS} + \sigma_{11}\sigma_{21} S f_{SV} + \frac{1}{2}(\sigma_{21}^2 + \sigma_{22}^2) f_{VV} = r_{\text{term}} f. \tag{A.12}$$