INSTITUTO
UNIVERSITÁRIO
DE LISBOA

UNIVERSIDADE
DE LISBOA

Numerical and Theoretical Analysis of Lognormal and Location-Scale t Mixture Models for Risk-Neutral Density Recovery

Miguel Natal de Brito Boto

Msc in Financial Mathematics

Supervisors:
PhD, João Pedro Nunes, Full Professor,
ISCTE-IUL

PhD, João Pedro Ruas, Integrated Researcher,
BRU-ISCTE Business Research Unit

October, 2025

Department of Finance
Department of Mathematics

Numerical and Theoretical Analysis of Lognormal and Location-Scale t Mixture Models for Risk-Neutral Density Recovery

Miguel Natal de Brito Boto

Msc in Financial Mathematics

Supervisors:
PhD, João Pedro Nunes, Full Professor,
ISCTE-IUL

PhD, João Pedro Ruas, Integrated Researcher,
BRU-ISCTE Business Research Unit

October, 2025

Resumo

Esta tese investiga a estimação de densidades de risco neutro (RNDs) implícitas nos preços de opções de mercado, com foco no índice S&P 500.

O estudo implementa e avalia metodologias de *finite mixture modelling* para estimar a RND, com componentes teóricos, metodológicos e empíricos. Teoricamente, as *finite mixtures* podem ser vistas como *single hidden layer feedforward neural networks*, sustentando a convergência através de *universal approximation theorems*. Introduz-se uma noção de admissibilidade, definida em relação ao comportamento assintótico dos preços das opções, fornecendo uma estrutura teórica para discutir a validade das densidades inferidas.

Empiricamente, duas especificações de mistura—as distribuições lognormais e *location-scale t*—são implementadas recorrendo a rotinas de otimização em MATLAB. O modelo de mistura lognormal demonstra uma clara convergência *in-sample*, com métricas de estimação e precisão de preços a melhorar à medida que aumenta o número de componentes da mistura. Em contrapartida, a mistura do tipo *location-scale t* enfrenta desafios computacionais significativos, devido ao seu espaço paramétrico de elevada dimensão e instabilidade numérica na avaliação de funções especiais.

Em suma, os resultados destacam a eficácia das misturas lognormais para a estimação flexível e precisa de RNDs, ao mesmo tempo que sublinham as dificuldades práticas associadas a especificações mais complexas. A tese contribui ao oferecer tanto um enquadramento metodológico para a estimação de RNDs baseada em misturas como uma perspetiva teórica sobre admissibilidade em precificação assintótica de opções, sugerindo direções para futuras pesquisas em eficiência computacional e validação de modelos.

Abstract

This thesis investigates the estimation of risk-neutral densities (RNDs) implied by option market prices, focusing on the S&P 500 index.

The study implements and evaluates finite mixture model methodologies for estimating the risk-neutral density, with theoretical, methodological, and empirical components. Theoretically, finite mixtures are connected to single hidden layer neural networks, supporting convergence through universal approximation arguments. A notion of admissibility is introduced, defined in relation to the asymptotic behavior of option prices, providing a theoretical framework for discussing the validity of inferred densities.

Empirically, two mixture specifications—the lognormal and the location-scale t distributions—are implemented using MATLAB optimization routines. The lognormal mixture model demonstrates clear in-sample convergence, with estimation metrics and pricing accuracy improving as the number of components increases. In contrast, the location-scale t mixture faces significant computational challenges, arising from its high-dimensional parameter space and numerical instability in evaluating special functions.

Overall, the results highlight the effectiveness of lognormal mixtures for flexible and accurate RND estimation, while underscoring the practical difficulties associated with more complex specifications. The thesis contributes by offering both a methodological framework for mixture-based RND estimation and a theoretical perspective on admissibility in asymptotic option pricing, suggesting directions for future research in computational efficiency and model validation.

Contents

CHAPTER 1

Introduction

Risk neutral densities (RNDs) play a central role in modern financial economics by providing a probabilistic description of future asset price outcomes as implied by market prices. This description can, for example, when applied to stock indices, exchange rates and interest rates, guide an economic political decisor, such as a central bank (Schittenkopf and Dorffner (2000)), extending the directly observed market information he has. In this thesis, the focus is on another role RNDs have in an option pricing model. By recovering these densities from observed option prices, one can infer the market's collective expectations beyond the information contained in implied volatilities alone.

Option pricing models provided by RNDs have the advantage of capturing the expectations of the market participants. Models like Black-Scholes do not have this advantage, but a common approach is to infer the underlying volatility from market data, hence incorporating market sentiment into the model. Our approach extends this technique: instead of extracting only the second moment of the RND (the volatility), we extract all of the moments (the whole RND) from market data.

The key challenge addressed is the extraction of the risk neutral density from observed option market data. We use option data on the S&P 500 index, a widely followed benchmark for the United States equity market. Sophisticated statistical and numerical methods are required to infer the RND from option prices, taking into account market microstructure, data quality, and modeling assumptions.

The goal of this thesis is to implement and assess finite mixture model methodologies for estimating the S&P 500 risk neutral density and to validate the estimates against market data. Theoretically, we relate finite mixture models to single hidden layer neural networks and argue convergence by means of an universal approximation theorem.

This introduction is followed by a brief review of risk neutral densities in the context of dynamic asset pricing theory, and a theoretical framework to guide the rest of the exposition. The methodology section selects the models and describes the source and processing of the market data. We also dedicate a chapter for necessary numerical implementations. Results are presented and conclusions summarize findings and suggest further research. Standard numerical methods are relegated to the Appendix.

CHAPTER 2

Literature Review

In the 20th century, increasing efforts for a systematic theory to price abstract assets under uncertainty lead to the concept of *state prices*. This concept is relevant here because, even before measure theory was applied to finance, it hinted to the existence of a probability measure $\mathbb{Q}$ that, in an unified way, allows for the writing of a single equation that gives the price of any asset S under an arbitrage free market:[1]

$$d(t)S_t = \mathbb{E}_{\mathbb{Q}}[d(T)S_T \mid \mathcal{F}_t] \, . \tag{2.1}$$

We follow Duffie (2001) for the definitions needed to illustrate dynamic pricing under uncertainty with the basic assets (Arrow-Debreu securities). Let there be a system of N assets, with the set $\{1, 2, ..., S\}$ providing the allowable states of the system, at the future date at which we consider the payoff of each asset. Let $D \in \mathbb{R}^{N \times S}$ be such that D_{ij} is the payoff of securiy i in state j under the chosen numeraire. The market prices of the assets are given by $q \in \mathbb{R}^N$. Then a portfolio $\theta \in \mathbb{R}^N$ has market value $q^T\theta$ and payoff $D^T\theta$.

Definition 2.1. *A state-price vector is a vector $\psi \in \mathbb{R}^S_{>0}$ such that $q = D\psi$.*

A state-price vector $\psi = (\psi_1, ..., \psi_S)$ is such that ψ_i is the price of a contract that is certain to pay 1 monetary unit in state i and 0 in all other states. These contracts are called Arrow-Debreu securities and can be thought of as a "basis" for the market, since every asset can be written as a linear combination of these securities.

2.1. From State-Prices to Risk-Neutral Probabilities

In this subsection we illustrate how state-prices can be used to derive equation (1) in this discrete state market setting. We first note that the existence of a state-price vector is equivalent to the non-existence of arbitrage—see, for instance, Duffie (2001).

Let $\psi = (\psi_1, ..., \psi_S)$ be a state-price vector for the payoff and market value pair (D, q). Let $\psi_0 = \psi_1 + \psi_2 + ... + \psi_S$ and define $\hat{\psi}_j = \frac{\psi_j}{\psi_0}$. From the definition of state-price vector we have, for each asset i,

$$q_i = \sum_{j=1}^{S} D_{ij}\, \psi_j = \psi_0 \sum_{j=1}^{S} D_{ij}\, \hat{\psi}_j. \tag{2.2}$$

Notice that we have defined the $\hat{\psi}_j$ such that $\sum_{j=1}^{S} \hat{\psi}_j = 1$, so that $\sum_{j=1}^{S} D_{ij}\, \hat{\psi}_j$ is a weighted mean of the payoffs at each state. $\hat{\psi}_j$ can be interpreted as the weight assigned to state j for the pricing equation (2) to be true.

[1] $\{d(t)\}_t$ represents a discount process and $\{\mathcal{F}_t\}_t$ a filtration

Hence, without using modern measure theoretic ideas, we are inclined to say these weights play a central role in constructing a systematic pricing theory. At this point, the $\hat{\psi}_j$ are only viewed as a solution to equation (2). In order to give them a deeper interpretation, a very general and abstract theory had to be developed. Measure theory allowed for more sophisticated ways to model uncertainty. Arrow-Debreu securities were further developed in mathematical finance, but modern approaches often emphasize martingale methods and risk-neutral measures. For instance, equation (1) can be explained in a very condensed way:

The discounted price process of an asset is a martingale under the risk-neutral measure.

Obviously, this assumes the reader is comfortable with the basics of measure theory and stochastic processes. Being able to prove such a central theorem in one sentence is what mathematicians find very attractive.

CHAPTER 3

Theoretical Framework

The purpose of this chapter is to introduce the theoretical framework in which we will be working with. The key definitions and theorems can be found here. Note that by key definitions we mean the ones that may serve as a basis for the exposition. However, we assume the reader is familiar with the mathematical framework in which the dynamic asset pricing theory is inserted into.

3.1. Notation

Throughtout this exposition we consider an abstract asset and denote by S_t the price of the asset at time t, or simply S if the time is not relevant. Furthermore, we will be using the following notation:

- r will denote the continuosly compounded risk free interest rate.
- c_t will denote the price of a call option contract on the asset S and at time t. For modelling purposes it is useful to think of the price of an option as a function of several inputs, such as the strike X, maturity T and dividend yield δ. In the same way p_t is understood as the time-t put price.
- $\tau := T - t$.
- $\mathbb{Q}$ is the risk neutral measure and q will denote the risk-neutral density of the asset price.

3.2. Key Concepts and Definitions

In the framework of a single asset, we can define the following.

Definition 3.1 (Risk Neutral Measure). *A probability measure $\mathbb{Q}$ is said to be risk-neutral if*

(i) $\mathbb{Q}$ is equivalent to the "physical" probability measure $\mathbb{P}$.

(ii) The discounted asset price is a martingale under this measure $\mathbb{Q}$.

This incredibly abstract definition still carries the famous "risk neutral" description of the market. When evaluating the price in this risk neutral measure, the martingale property ensures the discounted asset price has no tendency to rise, and no tendency to fall. In this way, an agent who's preferences are bound to the risk neutral measure expects his rate of return to equal the risk free rate. On the other hand, condition (i) ensures that changing to the risk neutral measure does not change the paradigm of the original setting. It is much easier to infer on the risk neutral setting, and so the enquivalence can give some confidence that one's inferences are not completely far off. This is, of course,

an informal explanation of this definition, and lacks mathematical context, but should be enough for our purposes.

The asset price random variable S induces a probability measure on $\mathbb{R}$, namely the *push forward measure*. In this way, the expected value of S in our measurable space, which lacks specification, can be reduced to an expectation in the familiar space of real numbers. For this reduction to be applied in practice, one needs to have the probability density function (PDF) of S. Since we are interested in computing expected values in the $\mathbb{Q}$ measure (see equation (2.1)), we need the PDF of S under the risk neutral measure, which is what we call the risk neutral density (RND) of S.

In practice, one works with *transition density functions*. There should be some amount of information of the state of the system at the initial time t, even if it is just the spot price of the asset, and so conditioning on the known information is the natural exploit. For the rest of the exposition, when we refer to the RND q, it will be implicitly conditioned on the state of the system at time t: $q(S_T) \equiv q(t, S_t; T, S_T)$.

We now present the key theorem of the study.

THEOREM 3.1. *Suppose q is continuous. Then*

$$q(t,\, S_t;\, T,\, S_T) = e^{(r-\delta)\tau} \frac{\partial^2 c_t(S_t,\, X,\, T)}{\partial X^2}\bigg|_{X=S_T},$$

and

$$q(t,\, S_t;\, T,\, S_T) = e^{(r-\delta)\tau} \frac{\partial^2 p_t(S_t,\, X,\, T)}{\partial X^2}\bigg|_{X=S_T}.$$

PROOF. We prove the theorem for the put option.

p_t has the known price equation

$$p_t(S_t,\, X,\, T) = e^{-(r-\delta)\tau} \int_0^X (X - S_T) q(S_T) dS_T \,.$$

Let us extend $e^{(r-\delta)\tau} \frac{\partial p_t(S_t, X, T)}{\partial X}$ according to definition:

$$\begin{aligned}
& \frac{1}{h}\left(\int_0^{X+h} (X + h - S_T)\, q(S_T)\, dS_T - \int_0^X (X - S_T)\, q(S_T)\, dS_T\right) \\
= \;& \frac{1}{h}\left(\int_0^X (X - S)\, q(S)\, dS - \int_0^X (X - S)\, q(S)\, dS + \right. \\
& \left. \int_X^{X+h} (X + h - S)\, q(S)\, dS + \int_0^X h\, q(S)\, dS\right) \\
= \;& \int_0^{X+h} q(S) dS + \frac{1}{h}\int_X^{X+h} (X - S) q(S) dS \,.
\end{aligned}$$

Now prove that

$$\lim_{h\to 0} \int_X^{X+h} \frac{(X - S) q(S)}{h} dS = 0 \,.$$

Using L'Hôpital's rule it becomes

$$\lim_{h\to 0} \frac{\frac{d}{dh}\int_X^{X+h}(X-S)\,q(S)\,dS}{\frac{d}{dh}h}\,.$$

By definition,

$$\begin{aligned}
&\frac{d}{dh}\int_X^{X+h}(X-S)\,q(S)\,dS \\
=\;& \lim_{c\to 0}\frac{1}{c}\left(\int_X^{X+h+c}(X-S)\,q(S)\,dS-\int_X^{X+h}(X-S)\,q(S)\,dS\right) \\
=\;& \frac{d}{dt}\int_X^t(X-S)\,q(S)\,dS\Big|_{t=X+h} \\
=\;& (X-t)\,q(t)\,\Big|_{t=X+h} \\
=\;& (X-X-h)\,q(X+h) \\
=\;& -hq\,(X+h)\,.
\end{aligned}$$

Assuming q is continuous, this tends to 0 as $h \to 0$. Hence

$$e^{(r-\delta)\tau}\frac{\partial p_t(S_t,\,X,\,T)}{\partial X} = \int_0^X q(S)\,dS\,,$$

and the proposition follows by the fundamental theorem of calculus. □

We label this as a key theorem because it provides an entry point to the asymptotic approach we take.

When choosing a template density for q, a good metric is the numerical tractibility of this template. In fact, this metric should be sufficient if one is working with mixtures of this template density. This is because a mixture of densities can define a *single hidden layer feedforward neural network*—see Giacomini et al. (2008). Looking at a mixture of densities in this way one follows a universal approximation theorem for neural networks to declare that a mixture model can indeed approximate a large class of densities. The universal approximation theorem for feedforward networks with a single hidden layer states that, under some conditions of the activation function, the network can act as an "universal approximator", where convergence to a given function can be achieved by means of increasing the number of neurons in the hidden layer, which in mixture models amounts to increasing the number of component densities—Giacomini et al. (2008), Hornik, Stinchcombe, and White (1990).

Although the universal approximation theorem can guarantee convergence, it is an existence result, the theorem is silent on the optimization process. For our estimation of the risk neutral density, we choose a template density $\hat{q}$, and attempt to approximate q by mixing the template. We want to focus on the choice of the template here. For the universal approximation theorem a large class of templates will do, but it says nothing about the training of the mixture. Our approach is to pick a template that satisfies some

conditions that arise from the asymptotics of the option price functions. No matter the preferences of a market agent on the underlying of an option contract, the asymptotic properties should always be the same.

DEFINITION 3.2. *Let $\mathcal{F}_q = \{q(\theta)\}_{\theta\in\Theta}$ be a family of probability density functions. We denote by*

$$\mathcal{F}_q^g := \left\{ \sum_{i=1}^{g} \omega_i q(\theta_i) : (\forall i = 1, ..., g,\ \theta_i \in \Theta,\ 0 < \omega_i) \text{ and } \sum_i \omega_i = 1 \right\} ,$$

and we say $f \in \mathcal{F}_q^g$ is a mixture density.

The constraints on the weights ω_i ensure any mixture density is itself a PDF. Mixtures of probability density functions arise from a natural construction when modeling distributions of heterogenous populations (see, for example, McLachlan and Peel (2000)).

3.3. Asymptotic Approach

In this section we translate the option price asymptotic conditions into the risk neutral density itself. From Theorem 3.1 we can deduce that the option price v_t should satisfy

$$e^{(r-\delta)\tau} v_t(X, T) = D^{-2} q(X) . \tag{3.1}$$

The asymptotic conditions for call options are

$$\lim_{X\to+\infty} c\,(X) = 0 , \tag{3.2}$$

and

$$\lim_{X\to 0} c\,(X) = e^{(r-\delta)\tau} S_t . \tag{3.3}$$

The asymptotic conditions for put options are

$$\lim_{X\to+\infty} p\,(X) = +\infty , \tag{3.4}$$

and

$$\lim_{X\to 0} p\,(X) = 0 . \tag{3.5}$$

Inspired by equation (3.1) we define the following:

DEFINITION 3.3. *(Admissibility) Consider a parameter space Θ. Let $\mathcal{F}_q = \{q(\theta)\}_{\theta\in\Theta}$ be a family of probability density functions. We say $\mathcal{F}_q$ is:*

- ***c-admissible** in Θ if $\forall\theta \in \Theta$, $\exists f_c \in C^2(\mathbb{R})$ such that $f_c'' = q(\theta)$ and $e^{-(r-\delta)} f_c$ satisfies the asymptotic conditions (3.2) and (3.3).*
- ***p-admissible** in Θ if $\forall\theta \in \Theta$, $\exists f_p \in C^2(\mathbb{R})$ such that $f_p'' = q(\theta)$ and $e^{-(r-\delta)} f_p$ satisfies the asymptotic conditions (3.4) and (3.5).*
- ***admissible** in Θ if $\mathcal{F}_q$ is c-admissible and p-admissible in Θ.*
- ***weakly c-admissible** in Θ if $\mathcal{F}_q$ is c-admissible in a non-empty parameter space $\mathcal{S} \subset \Theta$.*
- ***weakly p-admissible** in Θ if $\mathcal{F}_q$ is p-admissible in a non-empty parameter space $\mathcal{S} \subset \Theta$.*

- ***weakly admissible*** *in* Θ *if* $\mathcal{F}_q$ *is weakly c-admissible in* Θ *and weakly p-admissible in* Θ

LEMMA 3.1. *Any admissible family of densities is weakly admissible.*

We shall say $\mathcal{F}_q^g$ is admissible (weakly-admissible) in Θ when $\mathcal{F}_q$ is admissible (weakly-admissible) in Θ. When there is no ambiguity on the parameter space we shall say q is c-admissible (p-admissible) to mean that $\{q(\theta)\}_{\theta\in\Theta}$ is c-admissible (p-admissible).

Our goal now will be to study a density by proving its admissibility, and then using it as a template for a mixing model to estimate daily RND through market data.

CHAPTER 4

Methodology

As hinted by the previous chapter, our approach will be to pick a PDF q to serve as template for the mixing $\mathcal{F}_q^g$. We will then use this mixture as the functional form of the RND and estimate the parameters by fitting the model option price to option market prices.

For this purpose, a software was built in the MATLAB programming language. In this section we describe the methodology used for the implementation.

4.1. Lognormal Density

The first density we study is the one implied by the Black-Scholes-Merton (BSM) model. There are many constructions to the BSM model. The most direct one is to assume the percentage changes of the stock price in a short period of time are normally distributed—see Hull (2009). If the construction of a probabilistic model for the stock price should start with deciding the distribution of the percentage changes, then the natural model is in fact the one that assumes this distribution is normal, as this is the distribution that maximizes the entropy of the random variable, making it the rigorous choice for a model with the least assumptions that can then serve as a benchmark for models with structural suppositions that go beyond the mean and deviation. This in turn implies that S follows a lognormal distribution.

The lognormal distribution has the known PDF

$$f_{\text{lognorm}}(\mu,\, \sigma,\, x) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left(-\frac{(\log x - \mu)^2}{2\sigma^2}\right), \quad x > 0\ ,$$

and $f_{\text{lognorm}}(x) = 0$ for $x \leq 0$.

THEOREM 4.1. *$f_{lognorm}$ is p-admissible.*

PROOF. To show that f_{lognorm} is p-admissible our proof will not rely on the parameter space (the mean and deviation).

For $D^{-1}f_{\text{lognorm}}$ we take $\int_0^x f_{\text{lognorm}}$. This is the known cumulative distribution function (CDF) of the lognormal distribution, since f_{lognorm} is only non-zero for positive arguments. Hence,

$$D^{-1}f_{\text{lognorm}}(x) = \Phi\left(\frac{\log x - \mu}{\sigma}\right),$$

where Φ denotes the CDF of the standard normal distribution.

For $D^{-2}f_{\text{lognorm}}$ we take $\int_0^x \Phi\left(\frac{\log t - \mu}{\sigma}\right) dt$. We do not bother about computing the integral, because we can directly prove the assymptotic relations. Let $f(t) = \Phi\left(\frac{\log t - \mu}{\sigma}\right)$.

Since $f(t)$ is a CDF, $\lim_{t\to+\infty} f(t) = 1$. Therefore, there must be some $a > 0$ such that, for $a' > a$, $f(a') > \frac{1}{2}$, say. By the comparison test $\int_{a'}^{+\infty} f(t)dt = \infty$, which gives the relation (3.4) for f. The relation (3.5) is immediate from the chosen function for $D^{-2}f_{\text{lognorm}}$. □

4.2. Location-Scale t Density.

The second density we implement is the *location-scale t* density, which can be thought as a take on the normal distribution with heavier tails. In fact, it is a generalization of the normal distribution because it converges to it when the number of degrees of freedom grows.

Definition 4.1 (Location-scale t). *The location-scale t distribution $lst_\nu(\mu, \sigma)$ is the distribution of the random variable*

$$X = \mu + \sigma T\,,$$

where $T \sim t_\nu$.

Definition 4.1 implies the PDF of lst_ν is

$$f_{\text{lst}}(\nu,\, \mu,\, \sigma) = \frac{1}{k_{\nu,\,\sigma}} \left(1 + \frac{(x-\mu)^2}{\sigma^2\nu}\right)^{-\frac{\nu+1}{2}}\,,$$

where

$$k_{\nu,\,\sigma} := \int_{-\infty}^{+\infty} \left(1 + \frac{(x-\mu)^2}{\sigma^2\nu}\right)^{-\frac{\nu+1}{2}}$$

is the constant that normalizes the function.

Theorem 4.2. *For $\sigma,\, \nu > 0$, the second antiderivative of f_{lst} on the domain $\mathbb{R}\backslash\{\mu\}$ can be written as*

$$D^{-2}f_{lst}(\nu,\, \mu,\, \sigma) =$$

$$c_2 + c_1(x-\mu) + sign\,(x-\mu)\frac{\sigma^2\nu}{(\nu-1)k_{\nu,\,\sigma}}\,{}_2F_1\left(-\frac{1}{2},\, \frac{\nu-1}{2},\, \frac{1}{2};\, -\frac{(x-\mu)^2}{\sigma^2\nu}\right)$$

where ${}_2F_1(a,\, b;\, c;\, z) := \sum_{j=0}^{+\infty} \frac{(a)_j\,(b)_j}{(c)_j}\frac{z^j}{j!}$ is the Gaussian Hypergeometric function and $c_1,\, c_2 \in \mathbb{R}$.

Proof. Write $y = x-\mu$. Let $\lambda = \frac{1}{\sigma^2\nu}$, $b = \frac{\nu+1}{2}$, so that $f_{\text{lst}}(\nu,\, \mu,\, \sigma) = \frac{1}{k_{\nu,\,\sigma}}\,(1+\lambda y^2)^{-b}$. Let

$$F(y) := \int_0^y (1+\lambda t^2)^{-b}dt\,.$$

To evaluate the integral, substitute $(1-u) = (1+\lambda t^2)^{-1}$. This is in order to rewrite the integral as an incomplete beta integral, which can be expressed as a hypergeometric

function (see Abramowitz and Stegun (1964, Equations (6.6.1) and (6.6.8))):

$$\begin{aligned} F(y) =& \frac{1}{2\sqrt{\lambda}} \int_0^{\frac{\lambda y^2}{1+\lambda y^2}} (1-u)^{b-\frac{3}{2}} u^{-\frac{1}{2}}\, du \\ =& \frac{1}{\sqrt{\lambda}} \left(\frac{\lambda y^2}{1+\lambda y^2}\right)^{\frac{1}{2}} {}_2F_1\left(\frac{1}{2},\, \frac{3}{2}-b;\, \frac{3}{2};\, \frac{\lambda y^2}{1+\lambda y^2}\right) \\ =& \frac{|y|}{\sqrt{1+\lambda y^2}}\, {}_2F_1\left(\frac{1}{2},\, \frac{3}{2}-b;\, \frac{3}{2};\, \frac{\lambda y^2}{1+\lambda y^2}\right). \end{aligned}$$

Substituting λ and b, we arrive at

$$F(y) = \frac{|y|\,\sqrt{\sigma^2\nu}}{\sqrt{\sigma^2\nu + y^2}}\, {}_2F_1\left(\frac{1}{2},\, 1-\frac{\nu}{2};\, \frac{3}{2};\, \frac{y^2}{\sigma^2\nu + y^2}\right).$$

Applying a transformation formula from Abramowitz and Stegun (1964, Equation (15.3.4)), then

$${}_2F_1\left(\frac{1}{2},\, 1-\frac{\nu}{2};\, \frac{3}{2};\, \frac{y^2}{\sigma^2\nu + y^2}\right) = \left(\frac{\sigma^2\nu}{y^2+\sigma^2\nu}\right)^{-\frac{1}{2}} {}_2F_1\left(\frac{1}{2},\, \frac{1+\nu}{2};\, \frac{3}{2};\, -\frac{y^2}{\sigma^2\nu}\right),$$

and, hence,

$$D^{-1} f_{\text{lst}}(\nu,\, \mu,\, \sigma) = c_1 + \frac{|y|}{k_{\nu,\,\sigma}}\, {}_2F_1\left(\frac{1}{2},\, \frac{1+\nu}{2};\, \frac{3}{2};\, -\frac{y^2}{\sigma^2\nu}\right),$$

$c_1 \in \mathbb{R}$.

Finally, and using properties of the hypergeometric function from Abramowitz and Stegun (1964, Equation (15.3.4)),

$$D^{-2} f_{\text{lst}}(\nu,\, \mu,\, \sigma) = c_2 + c_1 y + \text{sign}(y)\, \frac{\sigma^2\,\nu}{k_{\nu,\,\sigma}(\nu-1)}\, {}_2F_1\left(-\frac{1}{2},\, \frac{\nu-1}{2};\, \frac{1}{2};\, -\frac{y^2}{\sigma^2\,\nu}\right)$$

for $c_1,\, c_2\, \in \mathbb{R}$ □

Notice that in our derivation we implicitly leave the edge case $x = \mu$ out. The last step does not work when $y = 0$. Since our work is numeric in nature, we are not concerned with that.

The parameter space for the lst model is $\Theta := \mathbb{R}_{>2} \times \mathbb{R} \times \mathbb{R}^+$. We restrict ν to be larger than 2 in order for the standard deviation σ to be defined.

THEOREM 4.3. *f_{lst} is p-admissible.*

PROOF. Imposing $q = 1$ to Olver et al. (2010, Equation (16.11.6)) and taking the zero-order term ($k = 0$) from Olver et al. (2010, Equation (16.11.2)), then

$${}_2F_1(a,\, b;\, c;\, z) \sim \frac{\Gamma(c)}{\Gamma(a)\Gamma(b)} \left[\Gamma(a)\frac{\Gamma(b-a)}{\Gamma(c-a)}(-z)^{-a} + \Gamma(b)\frac{\Gamma(a-b)}{\Gamma(c-b)}(-z)^{-b}\right]$$

$$(|z| \to \infty)$$

Hence,

$$
{}_2F_1\left(-\frac{1}{2},\,\frac{\nu-1}{2};\,\frac{1}{2};\,-\frac{(x-\mu)^2}{\sigma^2\nu}\right)
$$

$$
\sim\ \frac{\sqrt{\pi}\,\Gamma\left(\frac{\nu}{2}\right)}{\sigma\,\sqrt{\nu}\,\Gamma\left(\frac{\nu-1}{2}\right)}\,(x-\mu)+\Gamma\left(\frac{\nu-1}{2}\right)\frac{\Gamma\left(-\frac{\nu}{2}\right)}{\sigma^{1-\nu}\,\nu^{\frac{1-\nu}{2}}\,\Gamma\left(1-\frac{\nu}{2}\right)}(x-\mu)^{1-\nu}
$$

$$
\sim\ \frac{\sqrt{\pi}\,\Gamma\left(\frac{\nu}{2}\right)}{\sigma\,\sqrt{\nu}\,\Gamma\left(\frac{\nu-1}{2}\right)}\,(x-\mu)\quad(x\to\infty)\,.
$$

Applying this to $D^{-2}f_{\text{lst}}$,

$$
(D^{-2}f_{\text{lst}}-c_2+c_1\mu)\sim c_1x+\frac{\sigma\,\sqrt{\nu}\,\Gamma\left(\frac{1}{2}\right)\Gamma\left(\frac{\nu}{2}\right)}{(\nu-1)\,k\,\Gamma\left(\frac{\nu-1}{2}\right)}\,(x-\mu)\qquad(x\to+\infty)\,. \tag{4.1}
$$

Hence $e^{-(r-\delta)}\,D^{-2}f_{\text{lst}}$ satisfies condition (3.4) for any $c_1,\,c_2\in\mathbb{R}$.

For condition (3.5) take $c_1=0$, and

$$
c_2=\ \text{sign}(\mu)\,\frac{\sigma^2\,\nu}{(\nu-1)\,k}\,{}_2F_1\left(-\frac{1}{2},\,\frac{\nu-1}{2};\,\frac{1}{2};\,-\frac{\mu^2}{\sigma^2\,\nu}\right)\,.
$$

□

4.3. Estimation Procedure

In this section we describe the estimation procedure.

4.3.1. Data Description

We use option data from the Chicago Board Options Exchange. The dataset consists of daily option contract data on the S&P 500 index, European-style and cash settled, ranging from 2022-01-03 to 2022-05-31. Naturally, we require essential variables for computing option prices, such as maturity, strike, option price, underlying price, risk free rate and underlying dividend yield. Additionally, we make use of other not so essential variables to enhance our estimation procedure. Those were the option expiration cycle, volume, open interest, underlying futures price and BSM implied volatility.

4.3.2. Option Filtering

We filter out options that have 7 or less days to expiration. A 7 day horizon is usually not of interest for policy makers—Bollinger, Melick, and Thomas (2023). We then pick the earliest available maturity. This way our options are not far from expiring, which should imply more market liquidity, and hence a greater inclination to a free-arbitrage market.

We use options from the seventh expiration cycle. This allows us to graph the evolution of the RND over the dataset period.

We also filter out options that are trading at the minimum tick value, which, for options on the S&P 500, is 0.05 index points. As cleverly pointed out in Bollinger, Melick, and

Thomas (2023), for options trading at the minimum tick value, the call with the smallest strike and the put with the greatest strike still convey information. Therefore, we leave one call and one put trading at minimum tick value.

Furthermore, we filter out all the in-the-money options. Out-of-the-money options have more liquidity and, since our dataset includes calls and puts, we still get data from the whole strike range.

Lastly, if, after every filtering, we find our resulting dataset has five or less rows, we discard that day.

4.3.3. Smoothing the Data

If the resulting dataset has more than 5 rows, we proceed to smooth the data.

We have the strikes and implied volatilities in a two dimensional space, and fit a cubic spline to the scattered data, thus obtaining a smooth curve of strikes to volatilities. We then revert the implied volatility back to the BSM price, to get back to our original setting, with a smoothed out data instead.

When reverting the implied volatility to option price, every contract becomes a put option. We could have just as well retrieved the call price, but we only estimate the RND for put options, which alignes well with our approach since only p-admissibility is proven.

The smooth data obtained from this process is illustrated in Figures 1 and 2 for January 3, 2022.

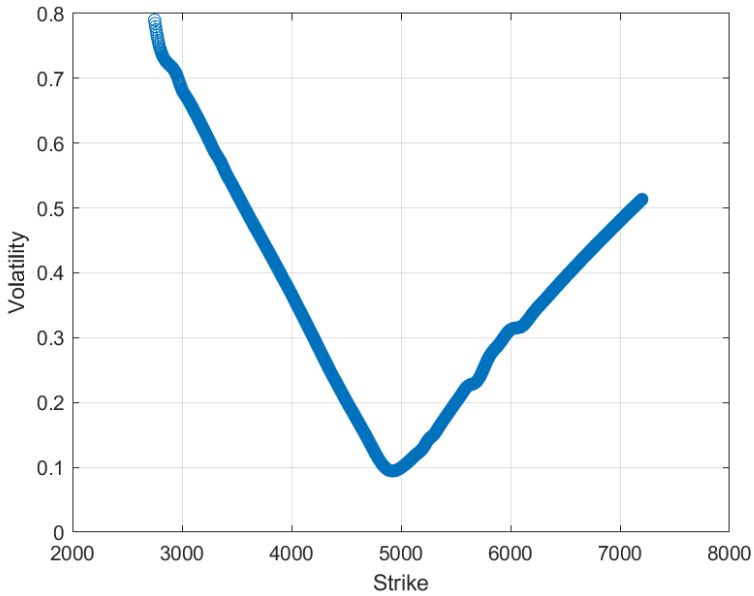

FIGURE 1. Fitted implied volatility curve as a function of the strike price for the option series on January 3, 2022. The curve is obtained using cubic spline interpolation over out-of-the-money options.

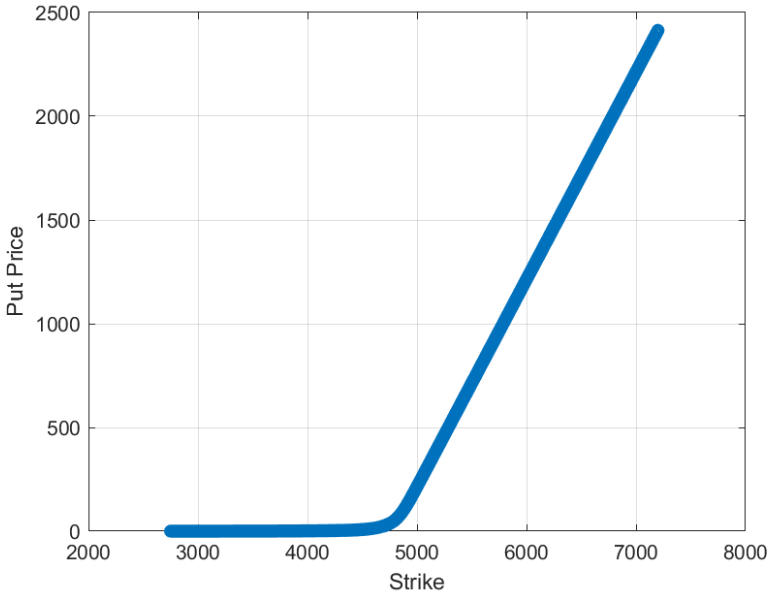

FIGURE 2. Put option prices inferred from market implied volatilities on January 3, 2022.

4.3.4. Parameter Estimation

Given a parameter space Θ and a PDF $f : \mathbb{R} \times \Theta \to \mathbb{R}$, we try to find the $\theta \in \Theta$ that best approximates the function $f(\theta)$ to the RND q of the S&P500.

For a single market day, our procedure is to minimize the sum of squared residuals $\sum(\hat{p} - p)^2$, where $\hat{p}$ is the put price assuming the RND is f, and p is the actual put price. For the optimization, we use the *interior point algorithm*, through MATLAB's `fmincon` function.

4.3.5. fmincon Interior Point Algorithm

We make use of MATLAB's Optimization Toolbox, which provides tools for finding parameters that minimize or maximize objective functions while satisfying given constraints. The toolbox offers two approaches for constrained nonlinear optimization: *Solver-based Optimization* and *Problem-based Optimization*. The latter is a higher-level code, and, hence, is excluded for not being appropriate to the task at hand.

From the Solver-based Optimization approach we make use of the MATLAB function `fmincon`, which finds the minimum of our objective function — sum of squared residuals — in a constrained parameter space. We utilize the default algorithm: *Interior Point*.

Consider the minimization problem

$$\min_x f(x)\,, \tag{4.2}$$

subject to

$$g_E(x) = 0\,,$$

$$g_I(x) \leq 0\,,$$

$f : \mathbb{R}^n \to \mathbb{R}$, $g_E : \mathbb{R}^n \to \mathbb{R}^{m_E}$, $g_I : \mathbb{R}^n \to \mathbb{R}^{m_I}$, and n, m_E, $m_I \in \mathbb{N}$.

Problem (4.2) is an inequality constrained optimization problem. The main idea here is to work with a similar problem and hope for an asymptotic approximation to the solution of the original problem. First note that problem (4.2) is equivalent to

$$\min_x f(x)\,, \tag{4.3}$$

subject to

$$g_E(x) = 0\,,$$

$$g_I(x) + s = 0\,,$$

$$s \geq 0\,.$$

We have replaced the general inequality $g_I(x) \leq 0$ by the simple bound $s \geq 0$. We can simplify the constraints even further using quite an ingenious idea common to interior point methods. We make it so that the objective function is obviously minimized inside of the feasible region, and then get rid of the superfluous constraints. We do this by incorporating a so called *barrier function* into the objective. It is called barrier function because it heavily penalizes values approaching the boundary of the feasible region. A typical choice for problem (4.2) is to take as barrier function

$$\phi(x) = -\sum_{i=1}^{m_I} log(-g_I^i(x))\,.$$

Applying this to problem (4.3), we write

$$\min_x \left[f(x) - \mu \sum_{i=1}^{m_I} log\, s_i\right], \tag{4.4}$$

subject to

$$g_E(x) = 0\,,$$

and

$$g_I(x) + s = 0\,.$$

Notice here the extra parameter μ, which is an essential one, as it will allow us to get back to our original problem. A sequence of problems are solved for decreasing values of μ, allowing the solution to get closer to the optimal constrained solution of problem (4.2). For more details on the algorithm refer to Byrd, Gilbert, and Nocedal (2000).

4.4. Price Function

The parameter estimation procedure relies on one being able to compute the option price dynamically assuming the RND $f(\theta)$. In both of our density templates we use closed form expressions for the option price function. When working with $f_{\text{lognorm}}(\mu,\, \sigma)$, we use the formula

$$p(t) = \; e^{(r-\delta)\tau} \left(X\, \Phi(-d_2) - e^{\mu+\frac{1}{2}\sigma^2} \Phi(-d_1) \right) ,$$

with

$$d_1 = \; \frac{-\log X + \mu + \sigma^2}{\sigma} ,$$

and

$$d_2 = \; d_1 - \sigma \, .$$

The formula can be obtained from Equation (2.1) by computing the integral

$$\begin{aligned} \mathbb{E}_{\mathbb{Q}}[\, p(T) \mid \mathcal{F}_t \,] &= \int (X - S_T)^+ \, d\mathbb{Q} \\ &= \int_{-\infty}^{+\infty} (X - S_T)^+ q(t,\, S_t;\, T,\, S_T)\, dS_T \\ &= \int_{-\infty}^{+\infty} (X - S_T)^+ f_{\text{lognorm}}(S_T)\, dS_T \quad , \end{aligned}$$

and assuming the transition density is f_{lognorm}. Note that in the implementation we substitute q by some $f \in \mathcal{F}^g_{f_{\text{lognorm}}}$, for some $g \in \mathbb{N}$, but since we deal with finite mixtures, the formula reduces to computing the integral for each component.

We also note that the underlying spot price is not explicitly present in the pricing formula. This is because the RND already depends on the spot price, by being a transition density. This does not, however, mean that the model option price is independent of the underlying spot. We estimate the density for a static (but general) t, and each instant t carries its own informative context (by means of the sigma-algebra $\mathcal{F}_t$), including the spot price, mapping to a RND $q(t,\, S_t;\, T,\, S_T)$. Therefore, in practice, the spot influences the optimal parameter θ without being explicitly present in the objective function.

The mixture of $\text{lst}(\nu,\, \mu,\, \sigma)$ model has as price function (again, for a single component) equal to

$$p(t) = e^{-(r-\delta)\tau} \left(c\, (X - \mu) + \right.$$

$$\left. \text{sign}(X-\mu)\, \frac{\sigma^2\, \nu}{k_{\nu,\,\sigma}(\nu-1)}\, {}_2F_1 \left(-\frac{1}{2},\, \frac{\nu-1}{2};\, \frac{1}{2};\, -\frac{(X-\mu)^2}{\sigma^2\, \nu} \right) \right) \; ,$$

with

$$c = \; \text{sign}(\mu)\, \frac{\sigma^2\, \nu}{(\nu-1)\, k}\, {}_2F_1 \left(-\frac{1}{2},\, \frac{\nu-1}{2};\, \frac{1}{2};\, -\frac{\mu^2}{\sigma^2\, \nu} \right) \; .$$

This pricing formula is obtained from equation (2.1) and Theorem 4.2. The constant c is chosen so that the mixture density satisfies the put assymptotic relations (3.4) and (3.5).

The implementation of this pricing function takes considerably more effort than in the lognormal case. The numerical computation of the special function ${}_2F_1$ requires careful handling of different parameter regimes. MATLAB has a built-in function `hypergeom`, but we found it was too slow to input it into `fmincon`, which requires iterative evaluation of the objective function (see Section 4.3.5). It is thus necessary to dedicate a separate section to this implementation.

CHAPTER 5

Numerical Implementation of ${}_2F_1$

A satisfactory definition of the Gaussian Hypergeometric function ${}_2F_1$ asks for the concept of analytic continuation.

Analyic continuation is a technique to extend the domain of a function. It stems from the fact that, if $f\colon U \to \mathbb{C}$ is an analytic function, there is at most one way to extend it to a larger domain $V \supset U$. This is because the identity theorem for analytic functions says that any other analytic function F extending f to a greater domain V is completely determined by its values on U, hence by f itself, since all it takes to define a function is its domain and values.

With this technique in mind we write ${}_2F_1(a,\, b;\, c;\, z) = \sum_{j=0}^{+\infty} \frac{(a)_j\,(b)_j}{(c)_j}\frac{z_j}{j!}$ and casually talk about evaluating the function for some big z even though the series only converges for $|z| < 1$. ${}_2F_1$ is decoupled from the given expression, or any other expression. In fact, interestingly, it can be the case that a function f can be analytically continued to a larger domain V, and proven that no series or integral assigned to f can be made to converge in all of V.

Definition 5.1. *For $c \notin \mathbb{Z}^-$, the Gaussian Hypergeometric function ${}_2F_1$ is given by*

$$ {}_2F_1(a,\, b;\, c;\, z) = \sum_{j=0}^{+\infty} \frac{(a)_j\,(b)_j}{(c)_j}\frac{z_j}{j!} \quad , \tag{5.1}$$

for $z \in \mathbb{C}$, $|z| < 1$, where

$$(a)_n = \begin{cases} 1 & \text{if } n = 0 \\ a\,(a+1)\cdots(a+n-1) & \text{if } n > 0 \end{cases}$$

is the Pochhammer symbol. For other values of $z \in \mathbb{C}$, ${}_2F_1$ is defined by analytic continuation along any path in the complex plane that avoids the points 1 and $+\infty$, which lead to inconsistencies.

The choice of method for numerically computing ${}_2F_1$ depends on the parameters a, b, c, z. Note that, if $|z| < 1$, we have a series representation, and so a natural approach would be to decide on the accuracy we want to achieve and iteratively sum the series terms until that accuracy is reached. Our z inputs are

$$z = -\frac{(X-\mu)^2}{\sigma^2\nu}\,,$$

and

$$z' = -\frac{\mu^2}{\sigma^2\nu}\,.$$

The ν parameter is the degrees of freedom of lst. It is such that $\nu > 2$, which gives $|z'| < \frac{\mu^2}{2\sigma^2}$. The σ parameter represents the standard deviation of the S&P500 index. For our dataset, set the lower bound at 700 index points, so that $|z'| < \frac{\mu^2}{(700\sqrt{2})^2}$. The μ parameter represents the mean of the S&P500 index. Set an upper bound of 4500 points, so that $|z'| < \left(\frac{4500}{700\sqrt{2}}\right)^2$. It is clear now that we shouldn't assume our z values will satisfy $|z| < 1$.

Pearson (2009, chap. 4.6) provides transformation formulae for normalizing a $z \in \mathbb{R}$ argument of ${}_2F_1$. Particularly useful for our case are

$$\begin{aligned} {}_2F_1(a,\, b;\; c;\, z) = (1-z)^{-a}\, \frac{\Gamma(c)\Gamma(b-a)}{\Gamma(b)\Gamma(c-a)}\, {}_2F_1\left(a,\, c-b;\, a-b+1;\, \frac{1}{1-z}\right) \\ + (1-z)^{-b}\, \frac{\Gamma(c)\Gamma(a-b)}{\Gamma(a)\Gamma(c-b)}\, {}_2F_1\left(b,\, c-a;\, b-a+1;\, \frac{1}{1-z}\right), \end{aligned} \tag{5.2}$$

$$-\infty < z < -1\,,$$

and

$${}_2F_1(a,\, b;\; c;\, z) = (1-z)^{-a}\, {}_2F_1\left(a,\, c-b;\, c;\, \frac{z}{z-1}\right), \tag{5.3}$$

$$-1 \le z < 0\,.$$

If $-1 \le z < 0$, we apply transformation (5.3) and reduce the hypergeometric input to $z' = \frac{z}{z-1}$, satisfying $|z'| \le \frac{1}{2}$. If $-\infty < z \le -1$, we apply transformation (5.2), reducing the hypergeometric input to $z' = \frac{1}{1-z}$, also satisfying $|z'| \le \frac{1}{2}$. We note that ${}_2F_1$ supports transformation formulae for getting z into the series's whole convergence interval $|z| < 1$—see Pearson (2009). We choose a more restricted interval to ease the convergence of our methods.

After normalizing z to $|z| < \frac{1}{2}$, we pass ${}_2F_1$ through three layers of numerical methods. See Algorithm 1.

Algorithm 1 Compute $\lambda\, {}_2F_1(a,\, b;\, c;\, z)$

```
if |z| > 1/2 then
    (λ, a, b, c, z) ← transformation(λ, a, b, c, z)
end if
output ← taylorMethod(λ, a, b, c, z)
if not converged(output) then
    output ← singleFractionMethod(λ, a, b, c, z)
    if not converged(output) then
        output ← gaussJacobiQuadrature(λ, a, b, c, z)
    end if
end if
```

The λ variable is added because the transformation formulas add a multiplication factor to the output, and the procedure needs to keep track of those factors.

5.1. Accuracy Tests

We built accuracy tests using MATLAB's `matlab.unittest` framework.

The data was first randomly initiated in a class based setup:

```
% Class based setup
methods (TestClassSetup)

    % Prepare the data.
    function prepareData(testCase)

        % Data is randomly initiated.
        % 1000 sample points.
        testCase.aData = -0.5 * ones(1000, 1);
        testCase.bData = rand(1000, 1);
        testCase.cData = 0.5 * ones(1000, 1);
        testCase.zData = rand(1000, 1);

    end
end
```

The a and c inputs are constant as in the location-scale t price function of Section 4.4. We wrote two accuracy tests, one for z inputs satisfying $-40 \leq z \leq -1$ (`betweenLargeAndMinusOne`) and one for z inputs satisfying $-1 \leq z \leq 0$ (`betweenMinusOneAndZero`). The b input is tested for values of ν such that $5 \leq \nu \leq 101$.

```
% Test z inputs between a large negative (here -40) and -1.
function betweenLargeAndMinusOne(testCase)
    a = testCase.aData;
    b = 2 + (50 - 2) * testCase.bData;
    c = testCase.cData;
    z = -40 + (-1 - (-40)) * testCase.zData;

    for iIn = 1 : numel(a)
        hypergeometric = Hypergeometric(...
            1,...
            a(iIn),...
            b(iIn),...
            c(iIn),...
            z(iIn)...
        );
```

```
        % Calculate.
        approximation = hypergeometric.computeApproximation();

        % Built-in functionality.
        truth = hypergeom([a(iIn), b(iIn)], c(iIn), z(iIn));

        % Verify.
        testCase.verifyEqual(approximation, truth, 'RelTol', 1e-12);
    end
end

% Test z inputs between -1 and 0.
function betweenMinusOneAndZero(testCase)
    a = testCase.aData;
    b = 2 + (50 - 2) * testCase.bData;
    c = testCase.cData;
    z = -1 + (0 - (-1)) * testCase.zData;

    for iIn = 1 : numel(a)
        hypergeometric = Hypergeometric(...
            1,...
            a(iIn),...
            b(iIn),...
            c(iIn),...
            z(iIn)...
        );

        % Calculate.
        approximation = hypergeometric.computeApproximation();

        % Built-in functionality.
        truth = hypergeom([a(iIn), b(iIn)], c(iIn), z(iIn));

        % Verify.
        testCase.verifyEqual(approximation, truth, 'RelTol', 1e-10);
    end
end
```

A test passes if all 1000 sampled inputs give

```
(approximation - truth) / truth < RelTol
```

where the `RelTol` variable is the relative tolerance of the test, provided as an input in the `verifyEqual` method of the `matlab.unittest.TestCase` object. We use a relative tolerance instead of an absolute tolerance because the outputs may range from $1e-60$ to $1e60$. The `betweenLargeAndMinusOne` test passed for a relative tolerance of $1e-12$, failing for tolerances of lower orders and the `betweenMinusOneAndZero` test passed for a relative tolerance of $1e-10$, failing for tolerances of lower orders.

5.2. Performance Tests

MATLAB's built-in calculator for ${}_2F_1$ already has the desired accuracy. The improvement we were looking for was on runtime performance. In this section we show that we have significantly improved the performance, whilst maintaining an acceptable accuracy (see Section 5.1 for the accuracy results). Note that our procedure is designed to perform on the parameter regimes we require, in fact, it fails most of the accuracy and performance tests for other parameter regimes.

Our performance tests are class based. We briefly describe how MATLAB's `matlab.perftest` framework conducts a performance comparison between two procedures.

We create a class called `CalculatorPerformanceTests` and make it inherit from `matlab.perftest.TestCase`. We store the input data common to all the tests as class properties:

```
properties
    lambdaData
    aData
    bData
    cData
    zData
end
```

The properties are then filled in a method based setup:

```
% Method based setup
methods (TestMethodSetup)

    % Prepare the data.
    function prepareData(testCase)

        % Use a different seed at each iteration.
        rng('shuffle');

        % Data is randomly initiated.
        testCase.lambdaData = 1 * ones(1000, 1);
        testCase.aData      = -0.5 * ones(1000, 1);
```

```
        testCase.bData      = rand(1000, 1);
        testCase.cData      = 0.5 * ones(1000, 1);
        testCase.zData      = rand(1000, 1);
    end
end
```

A method with the `TestMethodSetup` attribute is run before every individual test method. When running the tests as performance tests, the `prepareData` method is run before every individual measurement iteration. Our approach is to use different and randomly generated input data for each measurement. Since we suspect performance to vary greatly depending on the generated inputs, our setup shuffles the seed of the pseudo-random number generator.

As in the accuracy tests, we build performance tests for the two regimes $-40 \leq z \leq -1$ and $-1 \leq z \leq 0$. We have one test method for our calculator and one test method for the built-in calculator for each parameter regime. We showcase the regime $-40 \leq z \leq -1$, as the other is analogous.

```
% Performance tests.
methods (Test)
    function calculatorBetweenLargeAndMinusOne(testCase)
        lambda = testCase.lambdaData;
        a      = testCase.aData;
        b      = 2 + (50 - 2) * testCase.bData;
        c      = testCase.cData;
        z      = -40 + (-1 - (-40)) * testCase.zData;

        for iIn = 1 : numel(a)
            % Create the Hypergeometric object.
            hypergeometric = Hypergeometric(...
                lambda(iIn),...
                a(iIn),...
                b(iIn),...
                c(iIn),...
                z(iIn)...
            );

            % Compute the approximation.
            approximation =...
                hypergeometric.computeApproximation();
        end
    end
    function builtinBetweenLargeAndMinusOne(testCase)
```

```
        lambda = testCase.lambdaData;
        a      = testCase.aData;
        b      = 2 + (50 - 2) * testCase.bData;
        c      = testCase.cData;
        z      = -40 + (-1 - (-40)) * testCase.zData;
        for iIn = 1 : numel(a)
            truth = lambda(iIn) * hypergeom(...
                [a(iIn), b(iIn)],...
                c(iIn),...
                z(iIn)...
            );
        end
    end
end
```

Table 1 describes MATLAB's performance testing workflow for each test.

Step	Task
1	Calls method setup
2	Runs and times test method
3	Repeats steps 1 and 2 multiple times
4	Builds timing statistics

TABLE 1. MATLAB's Performance Testing Workflow

Step 1 is running the setup, which generates new random input data. Step 2 is running the test. Step 3 is necessary because MATLAB's first run of a function is not representative of its steady-state speed, this is because of MATLAB's *just-in-time* (JIT) compilation: MATLAB doesn't interpret every line of .m code each time it is executed. For example, when a process calls a procedure for the first time, MATLAB compiles the code into optimized machine-instructions. These instructions are then cached and able to be reused for subsequent calls, hence the first call of a function can be noticeably slower than subsequent calls. This is also the reason we didn't use MATLAB's stopwatch timer in the performance tests. Simply timing a process with `tic` may be enough for a large process, but since we are testing isolated procedures it is best to work around the JIT compilation.

Table 2: Test Results

Test	Sample Size	Mean
calculatorBetweenLargeAndMinusOne	8	0.088
calculatorBetweenMinusOneAndZero	13	0.033
builtinBetweenLargeAndMinusOne	4	17.182
builtinBetweenMinusOneAndZero	41	18.485

Table 2 (continued)

Standard Deviation	Min	Median	Max
0.0065	0.081	0.087	0.095
0.0032	0.031	0.032	0.043
0.1189	17.061	17.167	17.334
3.4996	13.509	18.901	25.010

Table 2 shows the sample summary output. The results show our procedure to be, on average, 195.25 times faster in the regime $-40 \leq z \leq -1$ and 560.15 times faster in the regime $-1 \leq z \leq 0$.

CHAPTER 6

Results

We present the results by density model and number of mixture components g. Our discussion in Section 3 reasons convergence can be achieved by means of increasing g. We aim to empirically show the convergence. The benchmark for comparison is the strike to option price obtained in Figure 2 after the smoothing of market data.

In this section we present the results for the first admissible day of the dataset: January 3, 2022.

6.1. Lognormal Mixture Model

6.1.1. $g = 2$

Tables 1 and 2 show error statistics of model to benchmark price of option contracts. Figure 1 is a plot of model to benchmark prices.

Table 1: Signed error statistics for the Lognormal mixture model with $g = 2$

Day	Mean(%)	Median(%)	Standard Deviation(%)
2022-01-03	-42.27	-0.031	48.71

Table 1 (continued)

Min(%)	Max(%)	q25(%)	q75(%)	Skewness	Kurtosis
-100	9.66	-100	-0.021	-0.29	1.12

Table 2: Absolute error statistics for the Lognormal mixture model with $g = 2$

Day	Mean(%)	Median(%)	Standard Deviation(%)
2022-01-03	42.85	1.32	48.19

Table 2 (continued)

Min(%)	Max(%)	q25(%)	q75(%)	Skewness	Kurtosis
1.29e-05	100	0.026	100	0.29	1.12

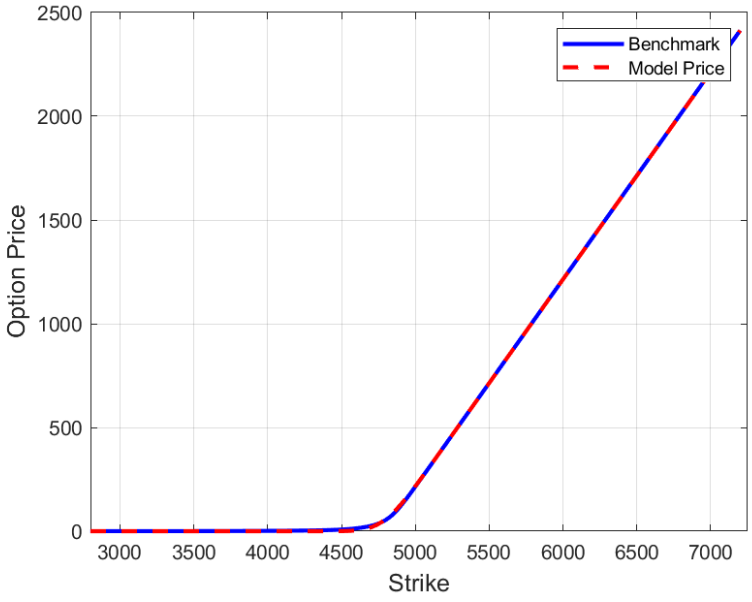

FIGURE 1. Model to benchmark price as a function of strike for a lognormal mixture with $g = 2$ components.

The model price has a negative bias, with a mean signed error of -42.27%. The estimation also seems to be failing completely for some contracts, where the error reaches -100%, and Figure 1 is not capturing this failure at naked eye. We were able to flag the failures for out-of-the-money (OTM) contracts, more precisely, the model price starts diverging from the market price at strikes around 4750 points, a little below the at-the-money (ATM) strike range (see Figure 2).

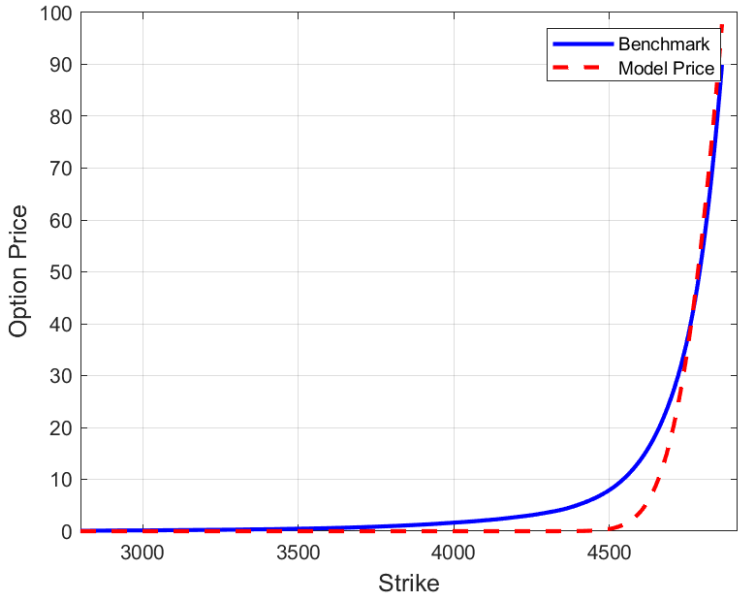

FIGURE 2. Model to benchmark price as a function of strike, for a lognormal mixture with $g = 2$ components and strikes under 4500 index points.

Figure 2 provides visual evidence for the failure of the estimation for OTM puts. We note that the plot is MATLAB's fitting of the data points, hence the errors cannot be observed directly from the graph.

6.1.2. $g = 4$

Tables 3 and 4 show error statistics for the lognormal mixture model with $g = 4$ components.

Table 3: Signed error statistics for the Lognormal mixture model with $g = 4$

Day	Mean(%)	Median(%)	Standard Deviation(%)
2022-01-03	-9.26	-0.0042	23.72

Table 3 (continued)

Min(%)	Max(%)	q25(%)	q75(%)	Skewness	Kurtosis
-92.88	8.33	-0.84	0.016	-2.23	6.60

Table 4: Absolute error statistics for the Lognormal mixture model with $g = 4$

Day	Mean(%)	Median(%)	Standard Deviation(%)
2022-01-03	10.78	0.036	23.06

Table 4 (continued)

Min(%)	Max(%)	q25(%)	q75(%)	Skewness	Kurtosis
3.22e-05	92.88	0.0072	6.31	2.26	6.73

Increasing the number of mixture components has addressed our problem for out-of-the-money contracts. The signed error distribution improves substantially: the mean goes from -42.27% to -9.26% and the percentil 25 goes from -100% to -0.84%. The absolute error distribution also improves.

Figure 3 plots model to benchmark price in the same strike range as Figure 2.

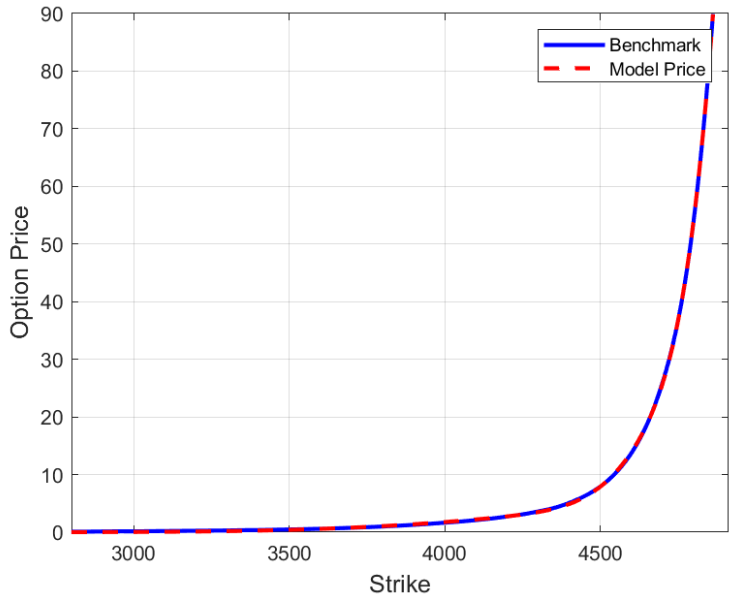

FIGURE 3. Model to benchmark price as a function of strike for a lognormal mixture with $g = 4$ components and strikes under 4500 index points.

There is a clear convergence in out-of-the-money contracts with the doubling of the number of mixture components.

Again, to visualise the points that have not converged, we go deeper out-of-the-money in Figure 4.

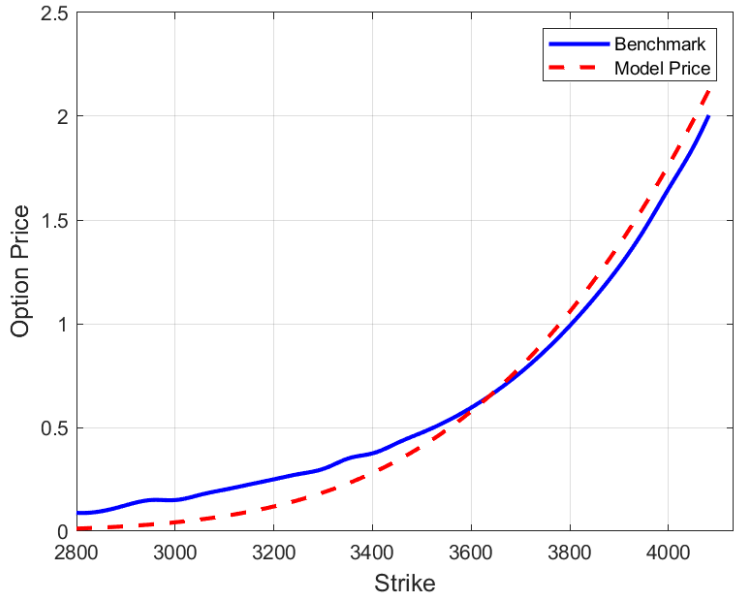

FIGURE 4. Model to benchmark price as a function of strike for a lognormal mixture with $g = 4$ components and strikes under 4000 index points.

6.1.3. $g = 16$

Tables 5 and 6 show error statistics for the lognormal mixture model with $g = 16$ components.

Table 5: Signed error statistics for the Lognormal mixture model with $g = 16$

Day	Mean(%)	Median(%)	Standard Deviation(%)
2022-01-03	1.54	0.00072	5.00

Table 5 (continued)

Min(%)	Max(%)	q25(%)	q75(%)	Skewness	Kurtosis
-20.22	33.31	-0.0035	1.26	3.68	21.04

Table 6: Absolute error statistics for the Lognormal mixture model with $g = 16$

Day	Mean(%)	Median(%)	Standard Deviation(%)
2022-01-03	2.10	0.049	4.79

Table 6 (continued)

Min(%)	Max(%)	q25(%)	q75(%)	Skewness	Kurtosis
1.43e-06	33.31	0.0020	2.18	4.12	22.59

Figure 5 plots model to benchmark price in the same strike range as Figure 4. Figure 6 plots model to benchmark price for strikes under 3200 index points.

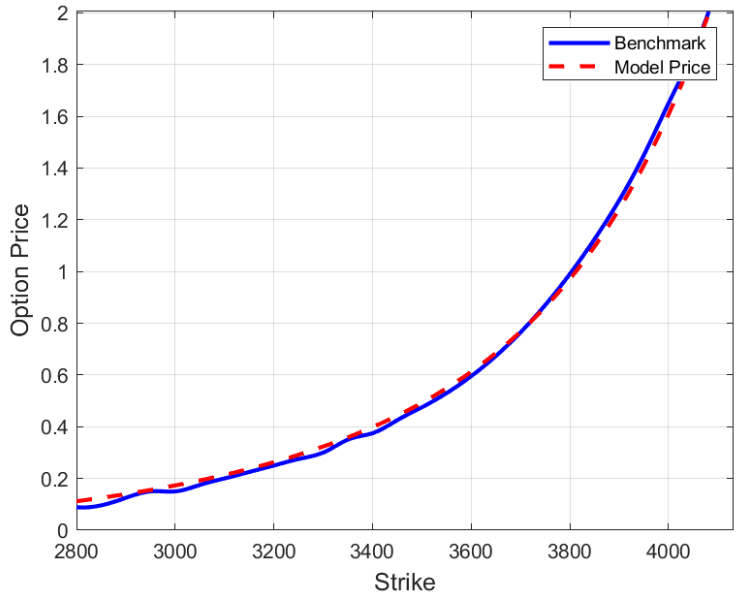

FIGURE 5. Model to market price as a function of strike for a lognormal mixture with $g = 16$ components and strikes under 4000 index points.

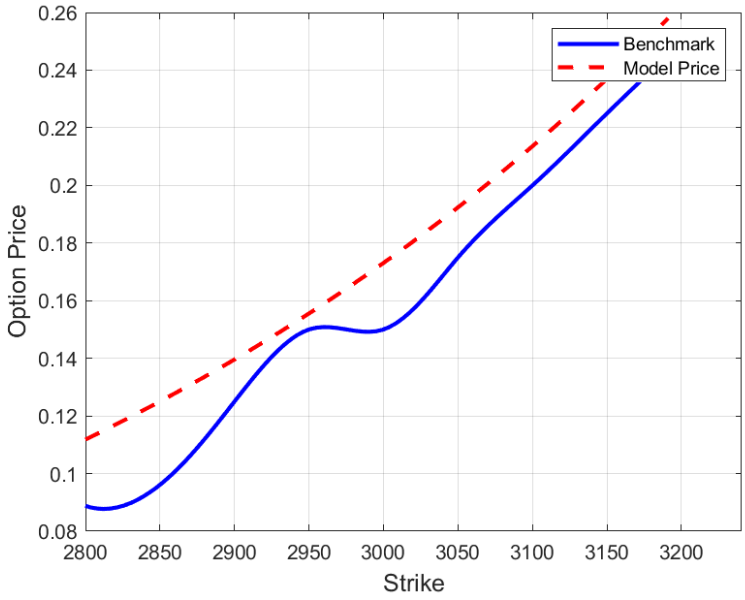

FIGURE 6. Model to market price as a function of strike for a lognormal mixture with $g = 16$ components and strikes under 3200 index points.

At $g = 16$ we again see greater convergence in lower strikes: please compare Figure 4 to Figure 5. It is interesting to note that the option price bias is no longer negative, given that the signed errors mean is positive. Figure 6 shows that the errors are now coming from overestimating the option price.

6.1.4. $g = 20$

Tables 7 and 8 show error statistics for the lognormal mixture model with $g = 20$ components.

Table 7: Signed error statistics for the Lognormal mixture model with $g = 20$

Day	Mean(%)	Median(%)	Standard Deviation(%)
2022-01-03	-0.82	-0.00073	3.53

Table 7 (continued)

Min(%)	Max(%)	q25(%)	q75(%)	Skewness	Kurtosis
-30.032	15.42	-0.21	0.0054	-1.39	15.71

Table 8: Absolute error statistics for the lognormal mixture model with $g = 20$

Day	Mean(%)	Median(%)	Standard Deviation(%)
2022-01-03	1.51	0.017	3.30

Table 8 (continued)

Min(%)	Max(%)	q25(%)	q75(%)	Skewness	Kurtosis
1.94e-06	30.032	0.0010	1.30	3.14	15.76

Overall, we observe in-sample convergence across all of the moneyness regimes with the increase in the number of mixture components in the lognormal mixture model.

6.2. Location-Scale t Mixture Model

6.2.1. $g = 2$

Tables 9 and 10 show error statistics for the location-scale t mixture model with $g = 2$ mixture components.

Table 9: Signed error statistics for the location-scaled t mixture model with $g = 2$

Day	Mean(%)	Median(%)	Standard Deviation(%)
2022-01-03	54.73	-0.59	142.23

Table 9 (continued)

Min(%)	Max(%)	q25(%)	q75(%)	Skewness	Kurtosis
-90.40	496.28	-7.95	10.58	2.087	6.040

Table 10: Absolute error statistics for the location-scale t mixture model with $g = 2$

Day	Mean(%)	Median(%)	Standard Deviation(%)
2022-01-03	70.00	8.13	135.36

Table 10 (continued)

Min(%)	Max(%)	q25(%)	q75(%)	Skewness	Kurtosis
0.011	496.29	3.27	52.68	2.16	6.25

Figure 7 plots the model to benchmark price.

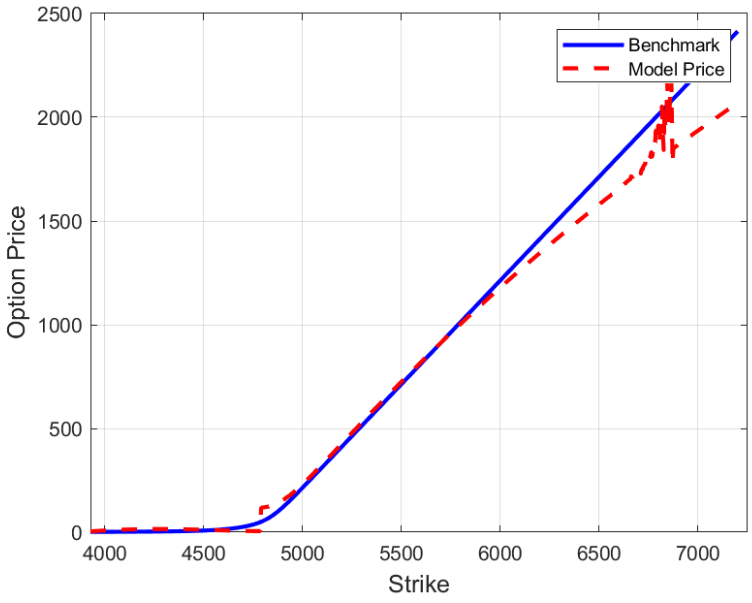

FIGURE 7. Model to benchmark price as a function of strike for the location-scale t mixture with $g = 2$ components.

The signed error statistics show the model has a pronounced positive bias overall. The model to benchmark plot, however, reveals an underestimation in in-the-money option contracts.

Figure 8 compares model to benchmark price for at-the-money option contracts.

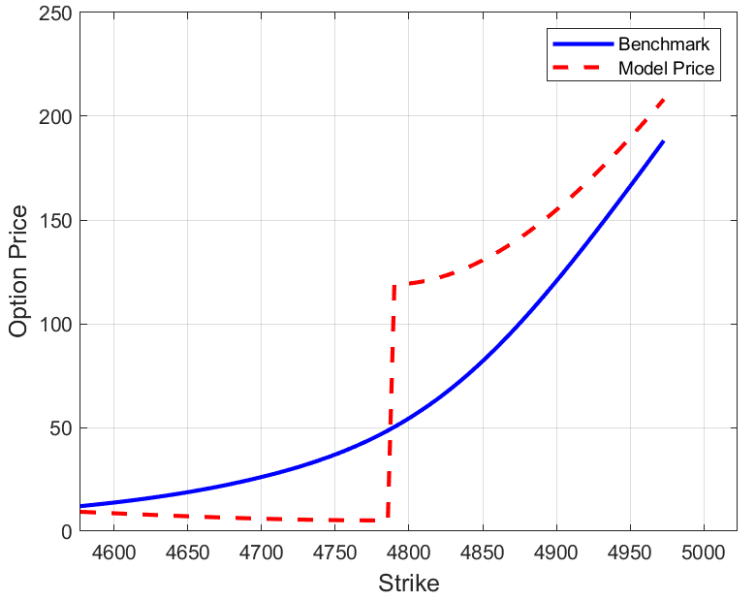

FIGURE 8. Model to benchmark price as a function of strike for the location-scale t with $g = 2$ components and strikes between 4600 and 5000 index points.

Notice that the model has a big spike at exactly the S&P 500 spot price. This is likely due to us leaving the edge case of $X = \mu$ out in our derivation of the price function for the location-scale t density in Theorem 4.2.

CHAPTER 7

Discussion

The empirical component of this thesis was focused on

(1) The relevance of the assymptotic admissibility according to Definition 3.3.
(2) Convergence in mixture models by means of increasing the number of mixture components.

The rationale for topic 1. comes from the fact that, given equation (3.1), the true RND q must be admissible, which implies that, if it is given by a finite mixture model, every component must itself be admissible. It is therefore intuitive to think that admissible densities have better convergence properties when used in finite mixture models. In order to actually validate this argument one would need to pick densities belonging to different classes of Definition 3.3 and compare the convergence qualities. This was not done in this exposition and serves as a suggestion for future work for the interested researcher.

Topic 2. contrasts with topic 1. because it argues admissibility is not a prerequisite for convergence. Giacomini et al. (2008) notes that the functional form of mixture models itself is enough to approximate arbitrary densities. Topic 2. was empirically validated for the lognormal mixture model, where the estimation metrics improved significantly at each increase in the number of mixture components.

The location-scale t mixture model's parameter space has $3g$ dimensions, more than the $2g$ dimensions of the lognormal mixture model. It also involves special care in initializing the parameters and numerically computing the price function. Multiple factors contributed to the lack of results for this choice of density. Namely:

- Computational complexity: MATLAB's `fmincon` uses iterative algorithms that scale poorly with the number of parameters. High-dimensional problems require more function evaluations and gradient computations, leading to increased computational cost and time.
- Initialization challenges: Finding a good initial guess becomes increasingly difficult in high dimensions. Without a robust initialization technique `fmincon` may converge slowly, converge to a suboptimal solution, such as a local minimum, or fail to find a feasible solution.
- Gradient dependence: `fmincon` relies on gradient-based methods, which can struggle in high-dimensional spaces with poorly conditioned functions, such as ${}_2F_1$.
- Numerical precision issues: The computation of ${}_2F_1$ requires numerical algorithms that were only tested in a restricted parameter space. Furthermore, in high

dimensions, small numerical errors in gradient or Hessian approximations can accumulate, leading to convergence issues or inaccurate results.

There are several techniques that can be used to mitigate many of the problems. For example, in order to mitigate the gradient dependence factor, one can supply exact derivatives to `fmincon`. This way the algorithm can skip the finite difference approximations, which can be inaccurate and computationally expensive. The increase in computational complexity can be mitigated by the usage of MATLAB's parallel computing capabilities. To work with the challenges in initializing parameters and local minima traps one can combine `fmincon` with global optimization methods, as these are designed to avoid getting trapped in local minima and to broadly sample the solution space.

CHAPTER 8

Conclusion

This thesis undertook an empirical investigation of the risk neutral density (RND) approach to modeling S&P 500 options, focusing on admissibility conditions and convergence behavior as the number of mixture components increases.

The concept of admissibility was introduced as a theoretical consideration regarding the structure of finite mixture models used for RND estimation. While this notion provided a framework for discussion, no claims regarding its novelty or definitive practical relevance are made.

Both lognormal and location-scale t finite mixture models were analyzed. The lognormal mixture demonstrated clear in-sample convergence, with increasing components yielding improved estimation metrics and put option pricing accuracy. This supports the effectiveness of lognormal mixtures for flexible and accurate risk neutral density approximation.

In contrast, the location-scale t mixture model faced significant computational and optimization challenges. These difficulties prevented clear demonstration of convergence and underscore practical limitations in applying more complex mixture specifications, due to their higher-dimensional parameter spaces and numerical complexities.

It is important to note that all validation was performed in-sample; out-of-sample or predictive testing was beyond the scope of this work and represents a relevant direction for future study.

Overall, this thesis provides an empirical assessment of mixture model RND estimation validated through put option pricing, highlighting the
strengths of lognormal mixtures and the practical challenges associated with more complex specifications. Recommendations for future work include extending validation frameworks, improving computational methods, and exploring alternative mixture structures, particularly through the use of Definition 3.3.

APPENDIX A

Taylor Series Method

This is the Taylor series method (a) in Pearson, S. Olver, and Porter (2017, chap. 4.2). It amounts to assigning ${}_2F_1 = S_N$, where $\{S_n\}_n$ is the sequence of partial sums of the series (5.1) and N is big enough to satisfy the convergence criteria.

A.1. Convergence Criteria

We set an upper bound of 500 terms for N. We assume convergence when we reach an N such that the next term of the series C_{N+1} is at least twelve orders smaller than S_N:

$$\frac{|C_{N+1}|}{|S_N|} < 10^{-12} \tag{A.1}$$

APPENDIX B

Single Fraction Method

This is method (c) in Pearson, S. Olver, and Porter (2017, chap. 4.2). We use it in our calculator when the Taylor series method shows no convergence (see Algorithm 1). While the Taylor series method gives a polynomial function approximation, the single fraction method gives a rational function approximation. It amounts to approximating the sequence of partial sums $\{S_n\}_n$ by a single fraction. The fraction is given by

$$\alpha_j = (\alpha_{j-1} + \beta_{j-1})\, j(c + j - 1)\,, \tag{B.1}$$

$$\beta_j = \beta_{j-1}\, (a + j - 1)\, (b + j - 1)z\,, \tag{B.2}$$

$$\gamma_j = \gamma_{j-1}\, j(c + j - 1)\,, \tag{B.3}$$

$$S_j = \frac{\alpha_j + \beta_j}{\gamma_j}\,, \quad j = 1,\, 2,\, \ldots \quad , \tag{B.4}$$

with $\alpha_0 = 0$, $\beta_0 = 1$, $\gamma_0 = 1$ and $S_0 = 1$.

The single fraction method may show convergence when the Taylor series method does not. A rational function can capture behaviour that a polynomial function cannot, eg., the denominator can model poles and singularities, particularly useful for when c is close to a non-positive integer. It is also less prone to accumulated rounding errors, as only one division is involved in the output.

B.1. Convergence Criteria

As in the Taylor series method, we set an upper bound of 500 iterations. Convergence is assumed when we have, for some N,

$$\frac{S_N - S_{N-1}}{S_{N-1}} < 10^{-12} \quad ,$$

and

$$\frac{S_{N-1} - S_{N-2}}{S_{N-2}} < 10^{-12} \quad .$$

Note that this criterion differs from criterion (A.1). In Taylor series approximation, convergence is usually assessed by the size of the last computed term, and this works because partial sums are built by adding terms one at a time, hence criterion (A.1) can let us know if the addition of the next term will have any impact on the output. For the single fraction method, the approximants are not built by adding terms but by solving the recurrence relations (B.1).

APPENDIX C

Gauss Jacobi Quadrature

The Gauss Jacobi quadrature is a quadrature rule, i.e, a method for numerical integration. It looks at the integral representation of ${}_2F_1$, instead of at the series representation. ${}_2F_1$ has the following integral representation (Pearson, S. Olver, and Porter (2017)):

$$ {}_2F_1(a,\, b;\, c;\, z) = \frac{\Gamma(c)}{\Gamma(b)\Gamma(c-b)} \int_0^1 (1-zx)^{-a}\,(1-x)^{c-b-1}\,x^{b-1}\,dx\,. \tag{C.1} $$

The Gauss-Legendre quadrature would attempt to compute the integral in equation (C.1) by finding weigths $w_1,\, w_2,\, \ldots w_n$ and nodes $x_1,\, x_2,\, \ldots x_n$ such that

$$ \int_0^1 f(x)\,dx = \sum_{i=1}^{n} w_i\, f(x_i)\,, $$

where

$$ f(x) = (1-zx)^{-a}\,(1-x)^{c-b-1}\,x^{b-1} $$

is the integrand of the right-hand-side of equation (C.1). However, the Gauss-Legendre quadrature would fail for this integral because it has an endpoint singularity at $x = 1$. The quadrature is designed to be exact for polynomials, therefore has no need to sample the interval's endpoints.

The Gauss-Jacobi rule is used to adress this problem. The method approximates integrals of the form

$$ \int_{-1}^{1} f(x)\,(1-x)^{\alpha}\,(1+x)^{\beta}\,dx $$

where f is a smooth function on $[-1,\, 1]$ and $\alpha,\, \beta > -1$.

Pearson, S. Olver, and Porter (2017, chap. 4.3) write equation (C.1) as

$$ {}_2F_1(a,\, b;\, c;\, z) $$

$$ = \frac{\Gamma(c)}{2^{c-1}\,\Gamma(b)\Gamma(c-b)} \int_{-1}^{1} \left(1 - \frac{1}{2}z(1+x)\right)^{-a} (1-x)^{c-b-1}\,(1+x)^{b-1}\,dx\,, $$

enabling the application of this method if $c > b$ and $b > 0$. If $c \leq b$ then one can try again with a by symmetry: ${}_2F_1(a,\, b;\, c;\, z) = {}_2F_1(b,\, a;\, c;\, z)$.

The Gauss-Jacobi quadrature is used as a last resort in Algorithm 1 because it is computationally more expensive.

Bibliography

Abadir, Karim M. (1999). "An introduction to hypergeometric functions for economists". *Econometric Reviews* 18, pp. 287–330.

Abadir, Karim M. and Michael Rockinger (2003). "Density functionals, with an option-pricing application". *Econometric Theory* 19, pp. 778–811.

Abramowitz, Milton and Irene A. Stegun (1964). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Vol. 55. National Bureau of Standards Applied Mathematics Series. Reprinted by Dover Publications, New York, 1972. Washington, D.C.: U.S. Department of Commerce, National Bureau of Standards.

Analytic Continuation (2025). Wikipedia. URL: `https://en.wikipedia.org/wiki/Analytic_continuation` (visited on 09/17/2025).

Arrow, K. J. (1964). "The Role of Securities in the Optimal Allocation of Risk-bearing". *The Review of Economic Studies* 31.2, pp. 91–96.

Bahra, Bhupinder (1997). "Implied risk-neutral probability density functions from option prices: theory and application". *Bank of England Working Paper* 66.

Bollinger, Thomas R., William R. Melick, and Charles P. Thomas (2023). "Principled pasting: attaching tails to risk-neutral probability density functions recovered from option prices". *Quantitative Finance* 23, pp. 1751–1768.

Brigo, Damiano, Gianvittorio Mauri, and Fabio Mercurio (2001). *Lognormal-Mixture Dynamics and Calibration to Volatility Smiles and Skews*. Working paper, Social Science Research Network. DOI: `10.2139/ssrn.276204`. URL: `https://ssrn.com/abstract=276204` (visited on 09/09/2025).

Byrd, Richard, Jean Charles Gilbert, and Jorge Nocedal (2000). "A trust region method based on interior point techniques for nonlinear programming". *Math. Program.* 89, pp. 149–185.

Control Random Number Generation (rng) (2025). MathWorks. URL: `https://www.mathworks.com/help/matlab/ref/rng.html` (visited on 09/24/2025).

Duffie, Darrel (2001). *Dynamic Asset Pricing Theory*. Princeton University Press.

Figlewski, Stephen (2017). "Risk Neutral Densities: A Review". *Annual Review of Financial Economics* 10, pp. 329–359.

Gaussian quadrature (2025). Wikipedia. URL: https://en.wikipedia.org/wiki/Gaussian_quadrature (visited on 09/21/2025).

Giacomini, Raffaella et al. (2008). "Mixtures of t-distributions for finance and forecasting". *Journal of Econometrics* 144, pp. 175–192.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1990). "Universal Approximation of an Unknown Mapping and Its Derivatives Using Multilayer Feedforward Networks". *Neural Networks* 3, pp. 551–560.

Hull, John C. (2009). *Options, Futures, and Other Derivatives*. 7th ed. Pearson Education, Inc. and Dorling Kindersley Publishing, Inc.

Log-normal distribution (2025). Wikipedia. URL: https://en.wikipedia.org/wiki/Log-normal_distribution (visited on 09/08/2025).

MATLAB Execution Engine Improvements (2025). MathWorks. URL: https://www.mathworks.com/help/matlab/matlab_prog/matlab-execution-engine.html (visited on 09/24/2025).

McLachlan, Geoffrey and David Peel (2000). *Finite Mixture Models*. Wiley Series in Probability and Statistics. New York: John Wiley & Sons.

Measure Code Performance (runperf) (2025). MathWorks. URL: https://www.mathworks.com/help/matlab/ref/runperf.html (visited on 09/24/2025).

Measure Performance Using timeit (2025). MathWorks. URL: https://www.mathworks.com/help/matlab/ref/timeit.html (visited on 09/24/2025).

Monteiro, Ana, Reha Tütüncü, and Luis Vicente (2008). "Recovering risk-neutral probability density functions from options prices using cubic splines and ensuring nonnegativity". *European Journal of Operational Research* 187, pp. 525–542.

Nikiforov, Arnold F. and Vasilii B. Uvarov (1988). *Special Functions of Mathematical Physics*. Germany: Birkhäuser.

Olver et al. (2010). "NIST Handbook of Mathematical Functions".

Optimization Toolbox Documentation (2025). MathWorks. URL: https://www.mathworks.com/help/optim/index.html?s_tid=CRUX_lftnav (visited on 02/28/2025).

Pearson, John (2009). "Computation of Hypergeometric Functions". MSc Thesis. University of Oxford.

Pearson, John, Sheehan Olver, and Mason A. Porter (2017). "Numerical methods for the computation of the confluent and Gauss hypergeometric functions". *Numer Algor* 74, pp. 821–866.

Problem-Based or Solver-Based Approach (2025). MathWorks. URL: https://www.mathworks.com/help/optim/ug/first-choose-problem-based-or-solver-based-approach.html (visited on 02/28/2025).

rand (2025). MathWorks. URL: https://www.mathworks.com/help/matlab/ref/rand.html (visited on 09/24/2025).

randn (2025). MathWorks. URL: https://www.mathworks.com/help/matlab/ref/randn.html (visited on 09/24/2025).

Schittenkopf, Christian and Georg Dorffner (2000). "Risk-neutral density extraction from option prices: Improved pricing with mixture density networks". *IEE Transactions on Neural Networks* 12, pp. 716–725.

ScienceDirect (2025). *Gaussian Quadrature*. URL: https://www.sciencedirect.com/topics/mathematics/gaussian-quadrature (visited on 09/21/2025).

Shreve, Steven E. (2004). *Stochastic Calculus for Finance II: Continuous-Time Models*. Springer Finance. New York: Springer.

Student's t-distribution (2025). Wikipedia. URL: https://en.wikipedia.org/wiki/Student%27s_t-distribution (visited on 09/09/2025).

Trefethen, Lloyd N. (2013). *Approximation Theory and Approximation Practice*. Philadelphia: SIAM.

Universal Approximation Theorem (2025). Wikipedia. URL: https://en.wikipedia.org/w/index.php?title=Universal_approximation_theorem&oldid=XXXXXXX (visited on 09/06/2025).

What is the MATLAB JIT Accelerator? (2025). MathWorks. URL: https://www.mathworks.com/matlabcentral/answers/96347-what-is-the-matlab-jit-accelerator (visited on 09/24/2025).

White, Halbert (1996). "Parametric statistical estimation with artificial neural networks". *Mathematical Perspectives on Neural Networks*. Taylor & Francis, pp. 714–776.

Write Performance Tests (matlab.perftest) (2025). MathWorks. URL: https://www.mathworks.com/help/matlab/matlab_prog/write-performance-tests.html (visited on 09/24/2025).