TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

Moment Explosion Time in the Rough Heston Model

ausgeführt am

Institut für
Stochastik und Wirtschaftsmathematik
TU Wien

unter der Anleitung von

Associate Prof. Dipl.-Ing. Dr.techn. Stefan Gerhold

durch

Christoph Gerstenecker, B.Sc.
Matrikelnummer: 01125236
Kellergasse 6
2103 Langenzersdorf

Wien, am 28. März 2018

Christoph Gerstenecker

Stefan Gerhold

Kurzfassung

Wir präsentieren einige Resultate aus [Gerhold et al. 2018, arXiv:1801.09458v3]. Hierbei ist das Hauptresultat und Motivation für weitere Anwendungen die Tatsache, dass die Explosionszeit der Momente im sogenannten Rough Heston Modell in der Variante, die von El Euch und Rosenbaum [El Euch, Rosenbaum 2016, arXiv:1609.02108] benutzt wird, genau dann endlich ist, wenn dies auch im gewöhnlichen Heston Modell der Fall ist. Motiviert ist dieses Resultat bzw. der Versuch es zu erlangen durch eine „schöne“ Darstellung der charakteristischen Funktion, die jener im gewöhnlichen Heston Modell ähnelt. Hergeleitet wurde diese Darstellung von El Euch und Rosenbaum in ihrem Paper von 2016. Um das Resultat aus Gerhold et al. zu erreichen, müssen wir die Lösung einer fraktionellen Riccati-Differentialgleichung untersuchen. Diese kann auch als Volterra-Integralgleichung zweiter Art mit schwach singulärem Kern geschrieben werden. Die Autoren machen sich die Mühe rigoros zu zeigen, dass diese Lösung der Volterra-Integralgleichung genau dann explodiert, wenn das Moment bzw. die Momentenerzeugende des Log-Preises dies tut. Nachdem das gesichert ist, wird für einen Spezialfall mittels Potenzreihenansatz ein effizienter Algorithmus zur numerischen Berechnung der Explosionszeit hergeleitet. Weiters wird in dieser Arbeit auch kurz eine empirische Methode zur Bestimmung der Explosionszeit vorgestellt, um die numerischen Tests mit dem Algorithmus vergleichen zu können. Ist nun eine Approximation für die Explosionszeit bestimmt, so kann man die Lösung der Riccati-Gleichung über eine Approximation mit Hilfe von Polylogarithmen darstellen, was auch numerisch überprüft wird. Leserinnen und Leser, die Berechnungen nachvollziehen wollen, werden sich freuen, denn wider den Usus wird eine detaillierte Darstellung des Source Codes in R, der Implementierung und der dabei aufgetretenen Probleme gegeben.

Abstract

We present the results of [Gerhold et al. 2018, arXiv:1801.09458v3], showing that the moment explosion time in the rough Heston model in the version introduced by El Euch and Rosenbaum [El Euch, Rosenbaum 2016, arXiv:1609.02108] is finite if and only if it is finite for the classical Heston model. This is mainly motivated by a representation of the characteristic function of the rough Heston model analogous to the classical Heston model by El Euch and Rosenbaum. Therefore, we need to do some analytics on the (not explicit) solution of a fractional Riccati equation which can be transformed into a weakly singular Volterra integral equation of the second kind. Gerhold et al. rigorously show that this solution explodes if and only if the moment generating function of the rough Heston model explodes. Following Gerhold et al., this fact is used to derive an algorithm to approximate the explosion time through a power series ansatz for a special case. Furthermore, an empirical method to identify the explosion time is presented to be able to compare the results. Having computed an approximation of the explosion time we get an approximation of the solution of the Riccati equation via polylogarithms. The reader's sake for reproduction will be satisfied, since detailed insight is given into the implementation and the used source code in R, as well as comments on issues the author was confronted with during the implementation process.

Acknowledgement

First of all, I would like to thank Stefan Gerhold for supervising this thesis. Thank you for introducing me to this topic, it was really a motivation to work on a "real" research topic which is highly up to date. Thank you that you always had time when I needed help and were not upset about the fact that to graduate students fewer conclusions are obvious than to professors (or even seemed to not be).

I would like to also thank Omar El Euch for valuable comments on the topic and for providing his source code when mine did not work out and I could not find the problem.

I would really like to thank Sandra, the secretary of FAM, who always supported me concerning organizational issues, but also always had an open ear for worries concerning the studies and beyond. I think what you do and the way you are doing it is very important to many students.

A special thanks goes to my parents who gave me the possibility to study at TU Wien. I really appreciate their support during my time of studies and before. I would also really like to thank my girlfriend for her ongoing support and belief in me, especially during the creation time of this thesis. Thank you!

Last but not least, I appreciate financial support under grant P-"apa & Mama" during this work (and the last two decades) ☺

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Wien, am 28. März 2018

Christoph Gerstenecker

Contents

1. Introduction

In modeling financial markets the easiest models, e.g. models with log-normal distribution as Black Scholes and others, provide thin tails which is not very appropriate for modeling real markets. In many models, according to [GGP18], tails of power law type have been proposed. Since these models provide moment explosions thorough analysis has been done on the existence of moments for a various range of classical models; Gerhold et al. emphasize the work of Keller-Ressel [Kel11] on affine stochastic volatility models. Knowing about the critical moments resp. the explosion time—note that we want to model the price as stochastic process S_t, such that $\mathbb{E}[(S_t)^u]$ can explode in t for u fixed—is of great interest, since, according to [GGP18], it allows to approximate the wing behavior of the volatility smile, to assess the convergence rate of some numerical procedures, and to identify models that would assign infinite prices to certain financial products. Gerhold et al. refer to [AP07, Kel11, Con14] for further details and references.

One model, where we have many explicit results, is the well-known Heston model ([Hes93]) which follows the dynamics

$$\begin{aligned} dS_t &= S_t\sqrt{V_t}\,dW_t, \quad S_0 > 0,\\ V_t &= V_0 + \int_0^t \lambda(\overline{v} - V_s)\,ds + \int_0^t \xi\sqrt{V_s}\,dB_s, \end{aligned}$$

where $\lambda, \overline{v}, \xi, V_0 > 0$, and the integrators are two Brownian motions which are correlated with $\rho \in (-1, 1)$, i.e. $\langle dW_t, dB_t\rangle = \rho\,dt$. According to El Euch and Rosenbaum in [ER16], it is shown in [GJR18] that for a very wide range of assets, historical volatility time-series exhibit a behavior which is much rougher than that of a Brownian motion, which is provided by the Heston model. Hence, they suggest, according to [BFG16, GJR18], to use a fractional Brownian motion with small Hurst parameter (see [MV68]) and they give a microstructural foundation for their model since it is a limiting case of modeling the price via Hawkes processes. The "rough" version of the Heston model, which will be called *rough Heston model* in this thesis, is then given via the dynamics

$$\begin{aligned} dS_t &= S_t\sqrt{V_t}dW_t, \quad S_0 > 0,\\ V_t &= V_0 + \frac{1}{\Gamma(\alpha)}\int_0^t (t-s)^{\alpha-1}\lambda\left(\overline{v} - V_s\right)\,ds + \frac{1}{\Gamma(\alpha)}\int_0^t (t-s)^{\alpha-1}\xi\sqrt{V_s}\,dB_s,\\ d\left\langle W, B\right\rangle_t &= \rho\,dt, \end{aligned}$$

where W and B are correlated Brownian motions, $\rho \in (-1, 1)$, $\lambda, \xi, \overline{v}, V_0 > 0$ and smoothness parameter $\alpha \in (1/2, 1)$. This model features a characteristic function that can be evaluated numerically in an efficient way (see [ER16, ER17, GGP18]), by solving a fractional Riccati equation. This is equivalent to solving a weakly singular Volterra integral equation of the second kind (see Chapter 3). Gerhold et al. rigorously show in [GGP18] that the solution of this equation explodes if and only if the moment generating function of the rough Heston model explodes. Following Gerhold et al., this fact is used to derive an algorithm to approximate the explosion time through a power series ansatz for a special case. Furthermore, an empirical method to identify the explosion time is presented to be able to compare the results. Having computed an approximation of the explosion time we get an approximation of the solution of the Riccati equation via polylogarithms. The reader's sake for reproduction will be satisfied, since detailed insight is given into the implementation and the used source code in R, as well as comments on issues the author was confronted with during the implementation process.

The content of this work is structured as follows. Chapters 2 and 3 present some main results of the theory of Volterra integral equations and fractional calculus, where the main reference for them is [Bru04, Bru17, GLS90] and [SKM93, KST06]. This provides the tools for our analysis of the explosion time. In Chapter 4 the analytical treatment of the model can be found. Motivated by the results of El Euch and Rosenbaum [ER16, ER17], we discuss some properties of the solution of a fractional Riccati equation, according to Gerhold et al. [GGP18], to find out more about the explosion behavior. We get analytical bounds for the explosion time of this function and, furthermore, the connection between the explosion time of this solution and the moment generating function of the log-price of our model is established, which was done rigorously in detail by Gerhold et al. Now knowing that explosion of the solution of a Riccati equation leads to explosion of the moment generating function, we establish some algorithms for computing the explosion time in Chapter 5. First, the fractional Adams method, proposed by El Euch and Rosenbaum in [ER16], is used to compute a numerical solution of the Riccati equation. Second, some asymptotics that were established by Gerhold et al. in [GGP18] are tested and then an algorithm to reliably compute the explosion time in a special case is tested. Another approach, to approximate the solution of the Riccati equation via polylogarithms, is established and numerically tested as well. In Chapter 6 there is not an "only source code appendix", but a detailed description of the implementation and further numerical analysis. The reason is that the results in this thesis should be well reproducible for graduate or possibly interested undergraduate students. Furthermore, the detailed description can be seen as a trigger for the readers to test the algorithms by themselves.

2. Volterra Integral Equations

In this chapter I want to give a very short introduction to Volterra integral equations and the objects we use out of this environment. Almost all the definitions and notations of this chapter are from [Bru04, Chapter 2 and 6] and [GLS90, Chapter 12 and 13].

2.1. Linear Volterra integral equations

Let $\mathcal{V} : C(I) \to C(I)$ denote the linear Volterra integral operator defined by

$$(\mathcal{V}\phi)(t) := \int_0^t K(t,s)\phi(s)\,ds, \quad t \in I := [0,T],$$

with $T < \infty$, where the kernel $K = K(t,s)$ is continuous on $D := \{(t,s) : 0 \le s \le t \le T\}$.

Definition 2.1 ([Bru04])**.** The integral equation

$$(\mathcal{V}\psi)(t) = g(t), \quad t \in I, \quad g(0) = 0, \tag{2.1}$$

is called a (linear) *Volterra integral equation of the first kind* for the unknown function $\psi(t)$ and a given continuous function $g(t)$ on I.

Definition 2.2 ([Bru04])**.** The integral equation

$$\psi(t) = g(t) + (\mathcal{V}\psi)(t), \quad t \in I, \tag{2.2}$$

is a (linear) *Volterra integral equation of the second kind* for the unknown function $\psi = \psi(t)$. The function $g = g(t)$ is a given continuous function on I.

As we are going to deal with a linear Volterra integral equation in Chapter 4 we need to know about the concept of the resolvent kernel in order to “eliminate” the solution function itself from the integrand.

Definition 2.3 ([Bru04])**.** Let ψ be the solution of a linear Volterra integral equation of the second kind as in (2.2). Further, let

$$R(t,s) = K(t,s) + \int_s^t K(t,v)R(v,s)\,ds, \quad (t,s) \in D, \tag{2.3}$$

$$R(t,s) = K(t,s) + \int_s^t R(t,v)K(v,s)\,ds, \quad (t,s) \in D. \tag{2.4}$$

Then for $K \in C(D)$, the (unique) *resolvent kernel* $R = R(t,s)$ corresponding to the given kernel K in the linear Volterra integral equation (2.2) is (formally) defined by either of the resolvent equations (2.3) and (2.4).

The existence and uniqueness of solutions to the linear Volterra integral equation (2.2) is established in Theorem 2.4.

Theorem 2.4. *Let $K \in C(D)$, and let R denote the resolvent kernel associated with K. Then for any $g \in C(I)$, the second-kind Volterra integral equation (2.2) has a unique solution $\psi \in C(I)$, and this solution is given by*

$$\psi(t) = g(t) + \int_0^t R(t,s)g(s)\,ds, \quad t \in I. \tag{2.5}$$

Proof. See [Bru04, Theorem 2.1.2]. □

Note that in many situations this is very helpful, since maybe we know some more about the properties of $g(t)$ alone than about the properties of the solution $\psi(t)$. Later on, in Lemma 4.14, we will use this representation to show that the solution $\psi(t)$ is negative resp. positive, with arguing with the sign of $g(t)$ and $R(t,s)$. As next, we have the result, that the regularity of the kernel K transfers to the regularity of the corresponding resolvent kernel R, which we actually do not need in this work, but should be mentioned nevertheless.

Theorem 2.5. *Assume that $K \in C^m(D)$. Then its resolvent R has the same degree of regularity, namely $R \in C^m(D)$. Thus, for any $g \in C^m(I)$ the solution of the Volterra integral equation (2.2) satisfies $\psi \in C^m(I)$.*

Proof. See [Bru04, Theorem 2.1.3]. □

Now let $\mathcal{V}_\alpha : C(I) \to C(I)$ be the linear Volterra integral operator defined by

$$(\mathcal{V}_\alpha \phi)(t) := \int_0^t p_\alpha(t-s)K(t,s)\phi(s)\,ds, \quad t \in I := [0,T], \tag{2.6}$$

with $K \in C(D)$, $K(t,t) \neq 0$ for $t \in I$, and

$$p_\alpha(t-s) := \begin{cases} (t-s)^{-\alpha}, & 0 < \alpha < 1, \\ \log(t-s), & \alpha = 1. \end{cases}$$

The kernel of the corresponding linear Volterra integral equation

$$\psi(t) = g(t) + (\mathcal{V}_\alpha \psi)(t), \quad t \in I, \tag{2.7}$$

is given by $H_\alpha(t,s) := p_\alpha(t-s)K(t,s)$. This integral kernel is called *weakly singular* since the explosion at the singularity is "weaker" than it would be for $(t-s)^{-1}$ resp. the kernel is too weak to let the integral explode. While in Theorem 2.4 we need the kernel to be continuous on D, which is not satisfied for (2.7), we get from [Bru04] a workaround for this problem, if the kernel is weakly singular, i.e. if the kernel is of the form in (2.6), then we get the analogous result to Theorem 2.4.

Theorem 2.6. *Assume that $K \in C(D)$ for K from (2.6), and let $0 < \alpha < 1$. Then for any $g \in C(I)$ the linear, weakly singular Volterra integral equation (2.7) possesses a unique solution $\psi \in C(I)$. This solution is given by*

$$\psi(t) = g(t) + \int_0^t R_\alpha(t,s)g(s)\,ds, \quad t \in I. \tag{2.8}$$

Here, the resolvent kernel R_α corresponding to the kernel $H_\alpha(t,s) := p_\alpha(t-s)K(t,s)$ inherits the weak singularity $(t-s)^{-\alpha}$ and has the form

$$R_\alpha(t,s) = (t-s)^{-\alpha}Q(t,s;\alpha), \quad 0 \le s < t \le T, \tag{2.9}$$

where

$$Q(t,s;\alpha) := \sum_{n=1}^{\infty}(t-s)^{(n-1)(1-\alpha)}\Phi_n(t,s;\alpha).$$

The functions Φ_n are defined recursively by

$$\Phi_n(t,s;\alpha) := \int_0^1 (1-z)^{-\alpha}z^{(n-1)(1-\alpha)-1}K(t,s+(t-s)z)\Phi_{n-1}(s+(t-s)z,s;\alpha)\,dz$$

for $n \ge 2$, with $\Phi_1(t,s;\alpha) := K(t,s)$ and $\Phi_n(\cdot,\cdot;\alpha) \in C(D)$. Moreover, $Q(\cdot,\cdot;\alpha)$ solves the resolvent equations

$$Q(t,s;\alpha) = K(t,s) + (t-s)^\alpha \int_s^t (t-v)^{-\alpha}(v-s)^{-\alpha}K(t,v)Q(v,s;\alpha)\,dv,$$
$$Q(t,s;\alpha) = K(t,s) + (t-s)^\alpha \int_s^t (t-v)^{-\alpha}(v-s)^{-\alpha}Q(t,v;\alpha)K(v,s)\,dv$$

on D.

Proof. See [Bru04, Theorem 6.1.2]. □

Remark 2.7. Note that for the model in Chapter 4 we will use $(t-s)^{\alpha-1}$ as part of the kernel. Brunner mentions that it will have certain advantages in his analysis to choose the weak singularity as $(t-s)^{-\alpha}$. We need to just have this in mind when we apply the theorems of this part later on, since we will have to use the transformation $\widetilde{\alpha} = 1-\alpha$ to exactly apply the theorem to our situation.

2.2. Nonlinear Volterra integral equations

Definition 2.8 ([Bru17, Chapter 2])**.** A standard non-linear VIE of the second kind has the form

$$\psi(t) = f(t) + \int_0^t k(t, s, \psi(s))\, ds, \quad t \in I := [0, T], \tag{2.10}$$

where f and k are given functions. The underlying non-linear Volterra integral operator

$$(\mathcal{U}\psi)(t) := \int_0^t k(t, s, \psi(s))\, ds$$

is usually referred to as the *Volterra-Urysohn* integral operator. If $k = k(t, s, u)$ in (2.10) is of the form $k(t, s, w) = K(t, s)G(s, w)$, the corresponding Volterra integral equation

$$\psi(t) = f(t) + \int_0^t K(t, s)G(s, \psi(s))\, ds, \quad t \in I, \tag{2.11}$$

is called a *Volterra-Hammerstein* integral equation. It corresponds to the *Volterra-Hammerstein* integral operator

$$(\mathcal{H}\psi)(t) := \int_0^t K(t, s)G(s, \psi(s))\, ds.$$

Since our analysis in Chapter 4 is done without explicitly knowing the solution of the Volterra integral equation, we need a result to be able to rely on the existence and uniqueness of continuous solutions. Therefor we have the following.

Theorem 2.9. *Let $\alpha \in (0, 1)$, $g \in C([0, \infty))$, and suppose that $H : \mathbb{R} \to \mathbb{R}$ is locally Lipschitz continuous. Then there is $T^* \in (0, \infty]$ such that the Volterra integral equation*

$$\psi(t) = g(t) + \int_0^t (t - s)^{\alpha - 1} H(\psi(s))\, ds \tag{2.12}$$

has a unique continuous solution ψ on $[0, T^*)$, and there is no continuous solution on any larger right-open interval $[0, T^{**})$.

Proof. See [GGP18, Theorem 2.3]: Uniqueness and existence can be found in [Bru17, Theorem 3.1.4]. The discussion how the solution can be continued to a maximal right-open interval of existence can be found at the beginning of [GLS90, Chapter 12] where it is explained how to prove [GLS90, Theorem 12.1.1]. □

Definition 2.10 ([GLS90])**.** Let $J \subseteq \mathbb{R}$ be an interval. A function $k : J^2 \to \mathbb{C}^{n \times n}$ is called a *kernel of continuous type on J* if k is measurable, if for every $t \in J$ the function $s \mapsto k(t, s)$ is integrable, and if the function $t \mapsto (s \mapsto k(t, s))$ is a continuous function from J into $L^1(J, \mathbb{C}^{n \times n})$.

Lemma 2.11. *Let $a(t,s)$ be the kernel of a Volterra integral equation. If $a(t,s) = k(t-s)$ for $0 \le s \le t$ with $k \in L^1_{\mathrm{loc}}([0,T), \mathbb{R}^{n\times n})$, then a is of continuous type on $[0,T)$.*

Proof. See the remark after [GLS90, Theorem 12.1.1]. □

Now let us introduce a parametric version of (2.11), i.e. for some $\lambda \in \mathbb{R}$, we have

$$\psi(\lambda,t) = f(\lambda,t) + \int_0^t a(\lambda,t,s)\, h(\lambda,s,\psi(\lambda,s))\, ds, \quad t \ge 0. \tag{2.13}$$

If we formally differentiate with $\psi_\lambda(\lambda,t) := \partial_\lambda \psi(\lambda,t)$, we get

$$\begin{aligned}\psi_\lambda(\lambda,t) = f_\lambda(\lambda,t) + \int_0^t \Big\{ & a_\lambda(\lambda,t,s)\, h(\lambda,s,\psi(\lambda,s)) + a(\lambda,t,s)\, \partial_1 h(\lambda,s,\psi(\lambda,s)) \\ & + a(\lambda,t,s)\, \partial_3 h(\lambda,s,\psi(\lambda,s))\, \psi_\lambda(\lambda,s) \Big\}\, ds, \quad t \ge 0.\end{aligned} \tag{2.14}$$

The next theorem shows that this intuition is not too bad and we are really able to differentiate $\psi(\lambda,t)$ and attain the representation (2.14).

Theorem 2.12. *Let $0 < T_\infty \le \infty$, let $f \in C(\mathbb{R}\times[0,T);\mathbb{R}^n)$ be continuously differentiable with respect to its first variable, and let $h \in C(\mathbb{R}\times[0,T)\times\mathbb{R}^n;\mathbb{R}^n)$ be continuously differentiable with respect to its first and third variables. Furthermore, assume that, for each $\lambda \in \mathbb{R}$, the function $a(\lambda,\cdot,\cdot)$ is a Volterra kernel of continuous type on $[0,T)$ that is differentiable with respect to λ in the sense that there exists a Volterra kernel $a_\lambda(\lambda,\cdot,\cdot)$ of continuous type on $[0,T_\infty)$ satisfying, for each $T \in [0,T_\infty)$,*

$$\sup_{t\in[0,T]} \int_0^t |a(\lambda+\epsilon,t,s) - a(\lambda,t,s) - \epsilon\, a_\lambda(\lambda,t,s)|\, ds = o(\epsilon), \quad \epsilon \to 0. \tag{2.15}$$

Then, for each $\lambda \in \mathbb{R}$, there is a unique solution $\psi(\lambda,\cdot)$ of equation (2.13), defined on the maximal interval of existence $[0,T_{\max}(\lambda))$. Moreover, $\psi(\lambda,t)$ is continuously differentiable with respect to λ on the set $\{(\lambda,t)|\lambda \in \mathbb{R}, t \in [0,T_{\max}(\lambda))\}$, and the derivative ψ_λ satisfies (2.14) on $[0,T_{\max}(\lambda))$.

Proof. See [GLS90, Theorem 13.1.2], it is a direct corollary of [GLS90, Theorem 13.3.1]. □

2.3. Asymptotics for the solution of special Volterra integral equations

In Chapter 5 we will develop an algorithm to compute the explosion time in a special case. Therefor we need an asymptotic result for the solution of a Volterra integral equation which can be represented in a special way.

Lemma 2.13. *Let ψ be the unique continuous solution of the nonlinear Volterra integral equation*

$$\psi(t) = \int_{t_0}^{t} k(t-s)G(s,\psi(s))\,ds, \tag{2.16}$$

where

$$G(s,\psi(s)) = r(s)g(u(s)+h(s)).$$

Assume that the following conditions hold:

$$g(w), g'(w), g''(w) > 0, \quad w > 0, \tag{2.17}$$
$$g(w) \sim w^m, \quad m > 1, \quad w \to \infty, \tag{2.18}$$
$$k(t-s) \geq 0,\ k'(t-s) < 0, \quad t_0 \leq s < t, \tag{2.19}$$
$$k(t-s) \sim k_0(t-s) \equiv \frac{1}{\Gamma(\mu)}(t-s)^{\mu-1}, \quad 0 < \mu < 1, \quad s \to t, \tag{2.20}$$
$$r, h \in C^1([t_0,T]),\ r(t), h(t) > 0,\ r'(t), h'(t) \geq 0, \quad t \geq t_0, \tag{2.21}$$
$$\psi(t) \to \infty, \quad t \to \widehat{t} < \infty. \tag{2.22}$$

Then we get for the solution $\psi(t)$ of (2.16) the asymptotics

$$\psi(t) \sim \left(\frac{\Gamma(\frac{m\mu}{m-1})}{r(\widehat{t})\Gamma(\frac{\mu}{m-1})}\right)^{\frac{1}{m-1}} (\widehat{t}-t)^{-\frac{\mu}{m-1}}, \quad t \to \widehat{t}. \tag{2.23}$$

Proof. See [RO96, Section 3: (3.2)]. □

Remark 2.14. Note that according to [GGP18], the proof is not rigorous. Still, there is hope that the statement holds, since there is at least numerical evidence for our special case of the statement, see e.g. Figure 5.2. We use Lemma 2.13 only as a motivation for an algorithm to compute the explosion time which is only verified numerically, so this should not be a great problem.

3. Fractional Calculus

Definition 3.1. The (left-sided) Riemann-Liouville fractional integral I_t^α of order $\alpha \in (0, \infty)$ of a function f is given by

$$I_t^\alpha f(t) := \frac{1}{\Gamma(\alpha)} \int_0^t (t-s)^{\alpha-1} f(s)\, ds \tag{3.1}$$

whenever the integral exists, and the (left-sided) Riemann-Liouville fractional derivative D_t^α of order $\alpha \in [0, 1)$ of f is given by

$$D_t^\alpha f(t) := \frac{d}{dt} I_t^{1-\alpha} f(t) \tag{3.2}$$

whenever this expression exists.

Remark 3.2. Note that I_t^1 is the usual integral, as

$$I_t^1 f(t) = \frac{1}{\Gamma(1)} \int_0^t (t-s)^0 f(s)\, ds = \int_0^t f(s)\, ds.$$

The fractional derivative D_t^α can be defined for $\alpha > 1$ as well, but this is not needed in our context.

Remark 3.3. The motivation for the integral concept of Definition 3.1 is the desire for generalizing the differential reps. the integral concept to orders $\alpha \in (0, \infty)$. Hereby we denote as integral of order $n \in \mathbb{N}$ the n-fold integral

$$\begin{aligned} \int_0^t ds \int_0^t ds \cdots \int_0^t f(s)\, ds &= \frac{1}{(n-1)!} \int_0^t (t-s)^{n-1} f(s)\, ds \\ &= \frac{1}{\Gamma(n)} \int_0^t (t-s)^{n-1} f(s)\, ds, \end{aligned}$$

from which the idea can be well seen; see e.g. [SKM93, Section 1.2.3].

Lemma 3.4. *Let $T > 0$.*

(i) The fractional integral and derivative of power functions can be easily calculated via

$$I_t^\alpha t^\nu = t^{\nu+\alpha} \frac{\Gamma(\nu+1)}{\Gamma(\nu+\alpha+1)}, \qquad \nu > -1,\ \alpha \in (0, \infty), \tag{3.3}$$

$$D_t^\alpha t^\nu = t^{\nu-\alpha} \frac{\Gamma(\nu+1)}{\Gamma(\nu-\alpha+1)}, \qquad \nu > -1+\alpha,\ \alpha \in [0, 1). \tag{3.4}$$

(ii) The fractional integral operators satisfy the semigroup property on $C([0,T])$, i.e.

$$I_t^\alpha I_t^\beta = I_t^{\alpha+\beta} \quad \text{for } \alpha, \beta \in (0,\infty). \tag{3.5}$$

(iii) For $f \in C([0,T])$ and $\alpha \in (0,1)$ the equation

$$D_t^\alpha I_t^\alpha f(t) = f(t) \tag{3.6}$$

holds.

(iv) For $f \in C([0,T])$ such that $D_t^\alpha f \in C([0,T])$ with $\alpha \in (0,1)$ the equation

$$I_t^\alpha D_t^\alpha f(t) = f(t) - \frac{I_t^{1-\alpha} f(0)}{\Gamma(\alpha)} t^{\alpha-1} \tag{3.7}$$

holds.

Proof. Ad (i). By substituting $u = (t-s)/t$ we get straight forward

$$\begin{aligned}
I_t^\alpha t^\nu &= \frac{1}{\Gamma(\alpha)} \int_0^t (t-s)^{\alpha-1} s^\nu \, ds = \frac{1}{\Gamma(\alpha)} \int_0^1 u^{\alpha-1} t^\alpha t^\nu (1-u)^\nu \, du \\
&= \frac{t^{\nu+\alpha}}{\Gamma(\alpha)} \int_0^1 u^{\alpha-1}(1-u)^\nu \, du = \frac{t^{\nu+\alpha}}{\Gamma(\alpha)} B(\alpha, \nu+1) \\
&= \frac{t^{\nu+\alpha}}{\Gamma(\alpha)} \frac{\Gamma(\alpha)\Gamma(\nu+1)}{\Gamma(\alpha+\nu+1)} = t^{\nu+\alpha} \frac{\Gamma(\nu+1)}{\Gamma(\nu+\alpha+1)}.
\end{aligned}$$

By definition of the fractional derivative we get

$$\begin{aligned}
D_t^\alpha t^\nu &= \frac{d}{dt}\left(I_t^{1-\alpha} t^\nu\right) = \frac{d}{dt}\left(t^{\nu+1-\alpha} \frac{\Gamma(\nu+1)}{\Gamma(\nu+1-\alpha+1)}\right) \\
&= t^{\nu-\alpha} \frac{(\nu+1-\alpha)\Gamma(\nu+1)}{\Gamma(\nu+1-\alpha+1)} = t^{\nu-\alpha} \frac{(\nu+1-\alpha)\Gamma(\nu+1)}{(\nu+1-\alpha)\Gamma(\nu+1-\alpha)} \\
&= t^{\nu-\alpha} \frac{\Gamma(\nu+1)}{\Gamma(\nu-\alpha+1)}.
\end{aligned}$$

Ad (ii). Substituting $s = u + \tau(t-u)$, which is suggested by [SKM93], we get

$$\begin{aligned}
\int_u^t (t-s)^{\alpha-1}(s-u)^{\beta-1} \, ds &= \int_0^1 ((t-u)(1-\tau))^{\alpha-1}(\tau(t-u))^{\beta-1}(t-u) \, d\tau \\
&= \int_0^1 (t-u)^{\alpha+\beta-1}(1-\tau)^{\alpha-1}\tau^{\beta-1} \, d\tau \\
&= (t-u)^{\alpha+\beta-1} \int_0^1 (1-\tau)^{\alpha-1}\tau^{\beta-1} \\
&= (t-u)^{\alpha+\beta-1} B(\alpha,\beta).
\end{aligned} \tag{3.8}$$

Since we have $f \in C([0,T])$, we can use Fubini and get

$$\begin{aligned}
I_t^\alpha I_t^\beta f(t) &= I_t^\alpha \left(\frac{1}{\Gamma(\beta)} \int_0^s (s-u)^{\beta-1} f(u)\, du \right) \\
&= \frac{1}{\Gamma(\alpha)} \int_0^t (t-s)^{\alpha-1} \left(\frac{1}{\Gamma(\beta)} \int_0^s (s-u)^{\beta-1} f(u)\, du \right) ds \\
&= \frac{1}{\Gamma(\alpha)\Gamma(\beta)} \int_0^t \int_0^s (t-s)^{\alpha-1}(s-u)^{\beta-1} f(u)\, du\, ds \\
&\overset{\text{Fubini}}{=} \frac{1}{\Gamma(\alpha)\Gamma(\beta)} \int_0^t \int_u^t (t-s)^{\alpha-1}(s-u)^{\beta-1} f(u)\, ds\, du \\
&= \frac{1}{\Gamma(\alpha)\Gamma(\beta)} \int_0^t f(u) \left(\int_u^t (t-s)^{\alpha-1}(s-u)^{\beta-1}\, ds \right) du \\
&\overset{(3.8)}{=} \frac{1}{\Gamma(\alpha)\Gamma(\beta)} \int_0^t f(u)(t-u)^{\alpha+\beta-1} B(\alpha,\beta)\, du \\
&= \frac{B(\alpha,\beta)}{\Gamma(\alpha)\Gamma(\beta)} \int_0^t (t-u)^{\alpha+\beta-1} f(u)\, du \\
&= \frac{1}{\Gamma(\alpha+\beta)} \int_0^t (t-u)^{\alpha+\beta-1} f(u)\, du \\
&= I_t^{\alpha+\beta} f(t).
\end{aligned}$$

Ad (iii). As the requirements of (ii) are fulfilled we get

$$D_t^\alpha I_t^\alpha f(t) = \frac{d}{dt} I_t^{1-\alpha} I_t^\alpha f(t) = \frac{d}{dt} I_t^1 f(t) = f(t).$$

Ad (iv). See [SKM93]. □

Remark 3.5. Note that in Lemma 3.4 (iv) the term for the initial value has to be interpreted as

$$I_t^{1-\alpha} f(0) := \lim_{s \searrow 0} I_t^{1-\alpha} f(s),$$

since writing formally

$$I_t^{1-\alpha} f(0) = \int_0^0 (0-s)^{\alpha-1} f(s)\, ds$$

would not make any sense.

Now we establish a connection between weakly singular Volterra integral equations from Chapter 2 and fractional calculus.

Theorem 3.6. *Let $\alpha \in (0,1)$, $T > 0$ and suppose that $\psi \in C([0,T])$ and $H \in C(\mathbb{R})$. Then ψ satisfies the fractional differential equation*

$$D_t^\alpha \psi(t) = H(\psi(t)), \tag{3.9}$$

$$I_t^{1-\alpha}\psi(0) = 0 \tag{3.10}$$

if and only if it satisfies the Volterra integral equation

$$\psi(t) = \frac{1}{\Gamma(\alpha)} \int_0^t (t-s)^{\alpha-1} H(\psi(s))\, ds. \tag{3.11}$$

Proof. Here we can apply Lemma 3.4 (iv). We get

$$\psi(t) - \frac{I_t^{1-\alpha}\psi(0)}{\Gamma(\alpha)} t^{\alpha-1} = I_t^\alpha D_t^\alpha \psi(t) = I_t^\alpha H(\psi(t))$$

which is equivalent to

$$\psi(t) = \frac{I_t^{1-\alpha}\psi(0)}{\Gamma(\alpha)} t^{\alpha-1} + I_t^\alpha H(\psi(t)),$$

and that is exactly the Volterra integral equation (3.11) due to the initial condition (3.10). For the other direction a little bit more effort is needed. It can be found in [KST06]. □

4. Rough Heston Model

In order to better model typical properties of real markets, the well-known Heston model (see [Hes93]) can be extended in a way to model "rough paths" with the use of a fractional integral. Since for the classical Heston model, there is an explicit representation for the moment generating function, we would be happy to get a similar result for the "rough" model extension. Luckily, El Euch and Rosenbaum show in [ER16] how to attain this result, except their representation is semi-explicit due to the occurrence of a function, that is given as the solution of a fractional Riccati differential equation which cannot be solved analytically. Nevertheless, this helps a lot since numerical applications get easier with this representation. In the following we are going to discuss some analytical properties of the solution of this fractional Riccati equation that can be derived just from the equation resp. from its representation as Volterra integral equation, in order to get boundaries for the explosion time and furthermore we present, at least for a special case, an algorithm to numerically compute the explosion time.

Definition 4.1. (Rough Heston model) The dynamics of the *rough Heston model* are (analogous to the classical Heston model) given by

$$dS_t = S_t\sqrt{V_t}\,dW_t, \quad S_0 > 0, \tag{4.1}$$

$$V_t = V_0 + \frac{1}{\Gamma(\alpha)}\int_0^t (t-s)^{\alpha-1}\lambda\left(\overline{v} - V_s\right)\,ds + \frac{1}{\Gamma(\alpha)}\int_0^t (t-s)^{\alpha-1}\xi\sqrt{V_s}\,dB_s, \tag{4.2}$$

$$d\left\langle W, B\right\rangle_t = \rho\,dt, \tag{4.3}$$

where W and B are correlated Brownian motions, $\rho \in (-1,1)$, $\lambda, \xi, \overline{v}, V_0 > 0$ and smoothness parameter $\alpha \in (1/2, 1)$.

Remark 4.2. Note that the limiting case $\alpha = 1$ resp. $\alpha \nearrow 1$ would correspond to the classical Heston model.

After [ER16] we get a nice representation of the characteristic function of the log-price $X_t := \log(S_t/S_0)$ for at least $\rho \in (-1/\sqrt{2}, 1/\sqrt{2}]$. To be able to use this representation for our purpose we need a more general result. In [ER17] the constraint for the correlation vanishes, so the representation holds for $\rho \in (-1,1)$, and we can plug in arguments with real part different from 0 into the characteristic function, such that we also get this representation for the moment generating function. To ensure the existence of the moment generating function we need the following.

Theorem 4.3 (Generalized characteristic function)**.** *Consider the rough Heston model* (4.1)–(4.3)*. Let $t > 0$ and $z \in \mathbb{C}$ satisfying $z = a + ib$ with $a, b \in \mathbb{R}$, $\lambda - \rho\xi a > 0$ and $a_-(t) < a < a_+(t)$, where*

$$a_-(t) = \frac{\xi^2 - 2\rho\xi X(t) - \sqrt{\Delta(t)}}{2\xi^2(1-\rho^2)}, \quad a_+(t) = \frac{\xi^2 - 2\rho\xi X(t) + \sqrt{\Delta(t)}}{2\xi^2(1-\rho^2)}, \tag{4.4}$$

with

$$X(t) = \lambda + \frac{\alpha t^{-\alpha}}{\Gamma(1-\alpha)}, \quad \Delta(t) = 4\xi^2 X(t)^2 + \xi^4 - 4\rho\xi^3 X(t).$$

Then we have

$$R(z,t) := E[e^{zX_t}] = \exp(\overline{v}\lambda I_t^1 \psi(z,t) + V_0 I_t^{1-\alpha}\psi(z,t)), \tag{4.5}$$

where $\psi(z,\cdot)$ is the unique continuous solution of the fractional Riccati equation

$$D_t^\alpha \psi(z,s) = \frac{1}{2}(z^2 - z) + (z\rho\xi - \lambda)\psi(z,s) + \frac{\xi^2}{2}\psi(z,s)^2, \quad s \le t, \quad I_t^{1-\alpha}\psi(z,0) = 0. \tag{4.6}$$

Proof. See [ER17, Corollary 3.1 and 3.2] with $h(z,t) = \psi(z,t)$ and $\theta^0(s) \equiv \overline{v}$. □

Corollary 4.4. *For each $t > 0$, there is an open interval in $\mathbb{R}$ such that* (4.5) *holds for $z = u$ in that interval.*

Corollary 4.5 (Moment generating function of the log-price)**.** *Let $t > 0$ and $u \in \mathbb{R}$ satisfying the conditions in Theorem 4.3. Then the moment generating function of the log-price $X_t = \log(S_t/S_0)$ is given via*

$$E[e^{uX_t}] = \exp(\overline{v}\lambda I_t^1 \psi(u,t) + V_0 I_t^{1-\alpha}\psi(u,t)), \tag{4.7}$$

where $\psi(u,\cdot)$ satisfies the fractional Riccati equation (4.6)*.*

El Euch and Rosenbaum showed in [ER16] an efficient way to compute the solution of the Riccati equation (4.6) via the fractional Adams method in order to compute the characteristic function if you fix the model parameters (see [ER16, Section 5] for reference). In this section, Corollary 4.5 builds the basis for some analysis of the explosion behavior of the moment generating function of the log-price X_t, respectively the explosion behavior of the solution $\psi(z,\cdot)$ of the Riccati equation (4.6). It mostly consists of results that are only published in preprints so far, and at the end there is done some numerical analysis to verify or at least to show, that for some model choices the proposed asymptotics work out nicely.

At first let us transform our problem (4.6) from a fractional differential equation to a Volterra integral equation, and then we are able to use a large amount of results that are already known for Volterra integral equations. For ease of notation and a better understanding of the computations afterwards let us write the Riccati equation (4.6) as

$$D_t^\alpha \psi(z,t) = R(z,\psi(z,t)), \quad s \le t, \quad I_t^{1-\alpha}\psi(z,0) = 0, \tag{4.8}$$

where

$$R(z,w) := c_1(z) + c_2(z)w + c_3 w^2 \tag{4.9}$$

with

$$c_1(z) := \frac{1}{2}(z^2 - z), \tag{4.10}$$

$$c_2(z) := z\rho\xi - \lambda, \tag{4.11}$$

$$c_3 := \frac{1}{2}\xi^2. \tag{4.12}$$

After Theorem 3.6 we can write (4.8) equivalently for fixed z as the Volterra integral equation

$$\psi(z,t) = \frac{1}{\Gamma(\alpha)}\int_0^t (t-s)^{\alpha-1} R(z,\psi(z,s))\,ds, \tag{4.13}$$

which is the equation for which the numerics at the end of this work is implemented. Note that this equation has a unique continuous solution due to Theorem 2.9, since R is a polynomial with respect to both of its variables and therefore locally Lipschitz (see Lemma A.10). To ease the use of results in [GGP18], the equation, we will mainly refer to with $u \in \mathbb{R}$, is

$$f(u,t) = \frac{1}{\Gamma(\alpha)}\int_0^t (t-s)^{\alpha-1} G(u,f(u,s))\,ds, \tag{4.14}$$

where

$$f(u,t) := c_3\psi(u,t) \tag{4.15}$$

and

$$\begin{aligned} G(u,w) &:= c_3 R\left(u, \frac{w}{c_3}\right) \\ &= (w + e_0(u))^2 - e_1(u), \end{aligned} \tag{4.16}$$

with

$$e_0(u) := \frac{1}{2}c_2(u) = \frac{1}{2}(u\rho\xi - \lambda), \tag{4.17}$$
$$e_1(u) := e_0(u)^2 - c_3c_1(u). \tag{4.18}$$

Note that we use $u \in \mathbb{R}$ as the argument for the generalized characteristic function because we are interested in the moment generating function. Since $c_3 = \xi^2/2 > 0$ in our model, the explosion behavior of $f(u, \cdot)$ is the same as of $\psi(u, \cdot)$.

Remark 4.6. Note that $G(u, w)$ from (4.16) is a polynomial with respect to each of its variables, since $R(u, w)$ is obviously a polynomial in w and $c_1(u), c_2(u), c_3$ are polynomials in u. To have an explicit representation for later on, we can write

$$G(u, w) = \frac{1}{4}\xi^2u^2 + \left(w\rho\xi - \frac{1}{4}\xi^2\right)u + \left(w^2 - w\lambda\right) \tag{4.19}$$

and

$$G(u, w) = w^2 + (\rho\xi u - \lambda)w + \frac{1}{4}\xi^2u(u-1). \tag{4.20}$$

Let us now write for the explosion time in the rough Heston model

$$T^*_\alpha(u) := \sup\{t \geq 0 : \mathbb{E}[S^u_t] < \infty\}, \quad u \in \mathbb{R}. \tag{4.21}$$

In the following we will distinguish between four cases for $u \in \mathbb{R}$.

Definition 4.7. For u real, let $\mathbb{R}$ be disjointed into the four cases:

(A) $c_1(u) > 0$, $e_0(u) \geq 0$,

(B) $c_1(u) > 0$, $e_0(u) < 0$ and $e_1(u) < 0$,

(C) $c_1(u) > 0$, $e_0(u) < 0$ and $e_1(u) \geq 0$,

(D) $c_1(u) \leq 0$.

As we will mainly discuss finite explosion times later on, we will just have a look at the admissible range for $u \in \mathbb{R}$ for the cases where the explosion time is finite. To have a better idea how this segmentation of $\mathbb{R}$ looks like, we have the following Lemma.

Lemma 4.8. *Let $(a, b), (a, b], [a, b)$ be the empty set for $a \geq b$. The real number u satisfies case (A) if and only if it is in the set*

$$D_A := \begin{cases} (-\infty, \frac{\lambda}{\xi\rho}], & \rho < 0, \\ [\frac{\lambda}{\xi\rho} \vee 1, \infty), & \rho > 0, \end{cases} \tag{4.22}$$

where the left endpoint 1 *is not included for* $1 \geq \lambda/(\xi\rho)$. *The real number* u *satisfies case* (B) *if and only if it is in the set*

$$D_{B} := \begin{cases} \left(\frac{\lambda}{\xi\rho}, p_{-}\right) \cup (1 \vee p_{+}, \infty), & \rho < 0, \\ (-\infty, p_{-}) \cup \left(1 \vee p_{+}, \frac{\lambda}{\xi\rho}\right), & \rho > 0. \end{cases} \tag{4.23}$$

Proof. Note that $c_1(u) > 0$ is equivalent to $u \in D_{c_1}^{A,B} := \mathbb{R}\backslash[0,1]$, hence this holds for both cases, (A) and (B). Then for case (A), $e_0(u) \geq 0$ is equivalent to $u\rho\xi - \lambda \geq 0$. This means u must be in $D_{e_0}^{A}$, with

$$D_{e_0}^{A} := \begin{cases} (-\infty, \frac{\lambda}{\xi\rho}] & \rho < 0, \\ [\frac{\lambda}{\xi\rho}, \infty) & \rho > 0. \end{cases}$$

Hence, for $\rho < 0$, there is no further restriction on u and for $\rho > 0$ it has to be greater or equal to $\lambda/(\xi\rho)$ as well as 1.

In case (B) we have that $e_0(u) < 0$ is equivalent to $u\rho\xi - \lambda < 0$ which leads to $u \in D_{e_0}^{B}$, with

$$D_{e_0}^{B} := \begin{cases} (\frac{\lambda}{\xi\rho}, \infty), & \rho < 0, \\ (-\infty, \frac{\lambda}{\xi\rho}), & \rho > 0. \end{cases}$$

The last condition is $e_1(u) < 0$ which can be expressed as polynomial in u such that we get

$$D_{e_1}^{B} := \{u \in \mathbb{R} : -4e_1(u) = q_2u^2 + q_1u + q_0 > 0\}, \tag{4.24}$$

with

$$\begin{aligned} q_0 &:= -\lambda^2 < 0, \\ q_1 &:= \xi(2\lambda\rho - \xi), \\ q_2 &:= \xi^2(1 - \rho^2) > 0. \end{aligned}$$

Note that with $-4e_1(0) = q_0 < 0$, the polynomial in (4.24) can get negative. As $q_2 > 0$, the polynomial will have positive asymptotics for $|u| \to \infty$, so there have to be two roots which bound the range of $\mathbb{R}$, where case (B) is not fulfilled. We set the polynomial to zero and get from

$$u^2 + \frac{q_1}{q_2}u + \frac{q_0}{q_2} = 0$$

the roots as

$$\begin{aligned} p_{-,+} &= -\frac{1}{2}\frac{q_1}{q_2} \mp \sqrt{\frac{1}{4}\left(\frac{q_1}{q_2}\right)^2 - \frac{q_0}{q_2}} \\ &= \frac{-q_1/q_2 \mp \sqrt{(q_1/q_2)^2 - 4q_0/q_2}}{2}. \end{aligned} \tag{4.25}$$

We know that

$$p_- < 0 < p_+,$$

since this is equivalent to

$$-\frac{q_1}{q_2} - \sqrt{\left(\frac{q_1}{q_2}\right)^2 - 4\frac{q_0}{q_2}} < 0 < -\frac{q_1}{q_2} + \sqrt{\left(\frac{q_1}{q_2}\right)^2 - 4\frac{q_0}{q_2}},$$

which we can write as

$$\left(\frac{q_1}{q_2}\right)^2 < \left(\frac{q_1}{q_2}\right)^2 - 4\frac{q_0}{q_2}.$$

This holds due to $q_0 < 0$ and $q_2 > 0$, i.e. the second term on the right-hand side is strictly positive. Now we can write

$$D_{e_1}^{B} = (-\infty, p_-) \cup (p_+, \infty).$$

Considering different cases with the convention $(a,b) = \emptyset$ for $a \geq b$ we get for $\rho < 0$

$$\begin{aligned} u &\in ((-\infty, p_-) \cup (p_+, \infty)) \cap (\frac{\lambda}{\xi\rho}, \infty) \cap (\mathbb{R}\backslash[0,1]) \\ &= \left(\left((-\infty, p_-) \cap (\frac{\lambda}{\xi\rho}, \infty)\right) \cup \left((p_+, \infty) \cap (\frac{\lambda}{\xi\rho}, \infty)\right)\right) \cap (\mathbb{R}\backslash[0,1]) \\ &= \left((\frac{\lambda}{\xi\rho}, p_-) \cup (p_+, \infty)\right) \cap (\mathbb{R}\backslash[0,1]) \\ &= (\frac{\lambda}{\xi\rho}, p_-) \cup (1 \vee p_+, \infty) \end{aligned}$$

and for $\rho > 0$ we get

$$\begin{aligned} u &\in ((-\infty, p_-) \cup (p_+, \infty)) \cap (-\infty, \frac{\lambda}{\xi\rho}) \cap (\mathbb{R}\backslash[0,1]) \\ &= \left(\left((-\infty, p_-) \cap (-\infty, \frac{\lambda}{\xi\rho})\right) \cup \left((p_+, \infty) \cap (-\infty, \frac{\lambda}{\xi\rho})\right)\right) \cap (\mathbb{R}\backslash[0,1]) \\ &= \left((-\infty, p_-) \cup (p_+, \frac{\lambda}{\xi\rho})\right) \cap (\mathbb{R}\backslash[0,1]) \\ &= (-\infty, p_-) \cup (1 \vee p_+, \frac{\lambda}{\xi\rho}). \end{aligned}$$

Finally, we get

$$D_{c_1}^{A,B} \cap D_{e_0}^{B} \cap D_{e_1}^{B} = \begin{cases} \left(\frac{\lambda}{\xi\rho}, p_-\right) \cup (1 \vee p_+, \infty), & \rho < 0, \\ (-\infty, p_-) \cup \left(1 \vee p_+, \frac{\lambda}{\xi\rho}\right), & \rho > 0, \end{cases}$$

with the convention $(a,b) = \emptyset$ for $a \geq b$. This is exactly the representation of D_B. $\square$

Let us now state one of the main results of [GGP18], giving a connection between the conditions on u and the explosions.

Theorem 4.9. *For $u \in \mathbb{R}$, the moment explosion time T^*_α of the rough Heston model is finite if and only if u satisfies case (A) or (B). This is equivalent to $T^*_1(u)$ (explosion time of the classical Heston model) being finite.*

Proof. See [GGP18, Theorem 2.4] resp. [GGP18, Section 5]. The proof consists of two main parts. First, Proposition 4.10 discusses the blowup behavior of the solution of (4.14) in cases (A) and (B), and Lemma 4.11 shows that blowup of f leads to blowup of the right-hand side of (4.7). Second, we show in Theorem 4.18 that the explosion time of $f(u, \cdot)$ (the solution of (4.14)) agrees with $T^*_\alpha(u)$ (the explosion time of the rough Heston model) for all $u \in \mathbb{R}$. □

Theorem 4.9 shows that the rough Heston model is consistent with the classical Heston model, at least considering their explosion behavior. In fact, the fractional Riccati equation (4.6) is, in the case of classical Heston, an ordinary Riccati equation that can be solved analytically. In the following we cite some results of [GGP18] to know more about the qualitative behavior of the solution $f(u, \cdot)$ of (4.14) resp. the results we need to prove Theorem 4.9.

Proposition 4.10. *Let $f(u, \cdot)$ be the solution of (4.14) for $u \in \mathbb{R}$ fixed. Then the following holds depending on the case of u:*

Case (A): $f(u, \cdot)$ starts at 0, is positive thereafter and blows up in finite time.

Case (B): $f(u, \cdot)$ starts at 0, is positive thereafter and blows up in finite time.

Case (C): $f(u, \cdot)$ is non-negative and bounded, and exists globally.

Case (D): $f(u, \cdot)$ is non-positive and bounded, and exists globaly.

Proof. See [GGP18, Propositions 3.2, 3.4, 3.6, 3.7]. □

In the following Lemma we get an idea how the explosion times of f and of the moment generating function are connected.

Lemma 4.11. *If f is a non-negative, continuous function that blows up in finite time with explosion time $\widehat{T}$, then $I^\alpha_t f$ blows up in finite time as well, with the same explosion time $\widehat{T}$. If f is a bounded continuous function, then $I^\alpha_t f$ does not blow up in finite time.*

Proof. See [GGP18, Lemma 3.8]. At first glance it seems natural, but have in mind that we have an integral kernel such that this is not obvious. □

With Lemma 4.11 we are able to transfer explosions of the solution $f(u,\cdot)$ of the Volterra integral equation (4.14) to explosions of the moment generating function of the log-price. So, for most of the (wanted) results it suffices to have an idea how the function $f(u,\cdot)$ behaves. In order to ease numerical computations it would be helpful to have some bounds for the explosion time to get reasonable starting values for iterations.

Theorem 4.12. *Let $u \in \mathbb{R}$ fixed, satisfying case (A) or (B). Then, the blow-up time $\widehat{T}_\alpha(u)$ of the solution $f(u,\cdot)$ of (4.14) satisfies*

$$\widehat{T}_-(u) \le \widehat{T}_\alpha(u) \le \widehat{T}_+(u), \tag{4.26}$$

where

$$\widehat{T}_-(u) := \Gamma(1+\alpha)^{1/\alpha} \max_{r>1} \frac{(r^\alpha - 1)^{1/\alpha}}{r(r-1)} \int_{a(u)}^{\infty} \left(\frac{w}{G(u,w)}\right)^{1/\alpha} \frac{dw}{w}, \tag{4.27}$$

$$\widehat{T}_+(u) := 4\Gamma(1+\alpha)^{1/\alpha} \int_0^\infty \left(\frac{w}{\widehat{G}(u,w)}\right)^{1/\alpha} \frac{dw}{w}, \tag{4.28}$$

with

$$a(u) := \begin{cases} 0, & u \text{ in case (A)}, \\ -e_0(u), & u \text{ in case (B)}, \end{cases}$$

and

$$\widehat{G}(u,\cdot) := \begin{cases} G(u,\cdot), & u \text{ in case (A)}, \\ -e_1(u)\mathbb{1}_{[0,-e_0(u))} + G(u,\cdot)\mathbb{1}_{[-e_0(u),\infty)}, & u \text{ in case (B)}. \end{cases}$$

Proof. See [GGP18, Theorem 4.1 and 4.2]. □

Lemma 4.13. *The solution f of the Volterra integral equation (4.14) is differentiable with respect to u, and its derivative satisfies*

$$\partial_1 f(u,t) = \int_0^t \frac{(t-s)^{\alpha-1}}{\Gamma(\alpha)} \left(\partial_1 G(u, f(u,s)) + \partial_2 G(u, f(u,s))\partial_1 f(u,s)\right) ds. \tag{4.29}$$

Proof. We show that we can apply Theorem 2.12 here. Note that our Volterra integral equation (4.14) can be written in the form of (2.13), i.e.

$$\begin{aligned} f(u,t) &= \widetilde{f}(u,t) + \int_0^t a(u,t,s)h(u,s,f(u,s))\,ds \\ &= \int_0^t \frac{(t-s)^{\alpha-1}}{\Gamma(\alpha)} G(u,f(u,s))\,ds, \quad t \ge 0. \end{aligned} \tag{4.30}$$

At first let $T \in [0, T_\infty)$ and $n = 1$. Then $\widetilde{f} \in C(\mathbb{R} \times [0,T); \mathbb{R}^n)$ is continuously differentiable with respect to u since $\widetilde{f} \equiv 0$ in our representation. Second, $h \in C(\mathbb{R} \times [0,T) \times \mathbb{R}^n; \mathbb{R}^n)$ is continuously differentiable with respect to its first and third variables for $h(u,s,w) := G(u,w)$, since G is a polynomial with respect to each variable (see Remark 4.6). Third, our kernel $a(u,t,s) := (t-s)^{\alpha-1}/\Gamma(\alpha)$ is of continuous type on $[0,T)$ by Lemma 2.11, since

$$\int_0^T \frac{s^{\alpha-1}}{\Gamma(\alpha)}\, ds = \frac{T^\alpha}{\Gamma(\alpha+1)}.$$

Additionally it is differentiable with respect to u in the way, stated in Theorem 2.12, since for $a_u(u,t,s) := 0$, which is of continuous type on $[0, T_\infty)$ for the same reason as above, we get for each $T \in [0, T_\infty)$ that

$$\begin{aligned}\sup_{t \in [0,T]} \int_0^t |a(u+\epsilon,t,s) - a(u,t,s) - \epsilon a_u(u,t,s)|\, ds &= \sup_{t\in[0,T]} \int_0^t |\epsilon a_u(u,t,s)|\, ds \\ &= 0 = o(\epsilon), \quad \epsilon \to 0.\end{aligned}$$

Now we can apply Theorem 2.12 and the derivative equals to just interchanging differential and integral operators and we get the representation (4.29). □

In the following lemma we show that $f(\cdot,t)$ is strictly monotonous on certain parts of the real axis.

Lemma 4.14. *Let $f(u,\cdot)$ be the solution of the Volterra integral equation (4.14). Then the following holds:*

(i) In case (A) we have $\partial_1 f(u,t) < 0$ for $u < 0$ and $\partial_1 f(u,t) > 0$ for $u > 0$.

(ii) If u satisfies case (B), then the same holds if $\widehat{T}_\alpha(u) - t$ is sufficiently small.

Proof. Ad (i). Lemma 4.13 allows us to write the partial derivative of f with respect to u as

$$\begin{aligned}\partial_1 f(u,t) &= \int_0^t \frac{(t-s)^{\alpha-1}}{\Gamma(\alpha)} \left(\partial_1 G(u, f(u,s)) + \partial_2 G(u,f(u,s))\partial_1 f(u,s)\right) ds \\ &= \int_0^t \frac{(t-s)^{\alpha-1}}{\Gamma(\alpha)} \partial_1 G(u,f(u,s))\, ds + \int_0^t \frac{(t-s)^{\alpha-1}}{\Gamma(\alpha)} \partial_2 G(u,f(u,s))\partial_1 f(u,s)\, ds \\ &= g(t) + \int_0^t (t-s)^{\alpha-1} K^{(u)}(t,s) \partial_1 f(u,s)\, ds,\end{aligned}$$

with

$$K^{(u)}(t,s) := \frac{\partial_2 G(u,f(u,s))}{\Gamma(\alpha)}, \tag{4.31}$$

$$g(t) := \int_0^t \frac{(t-s)^{\alpha-1}}{\Gamma(\alpha)} \partial_1 G(u,f(u,s))\, ds. \tag{4.32}$$

Now can apply Theorem 2.6 for fixed $u \in \mathbb{R}$, since

$$K^{(u)} : (t, s) \mapsto s \mapsto \frac{\partial_2 G(u, f(u, s))}{\Gamma(\alpha)}$$

is continuous on the product space as G is a polynomial with respect to both of its arguments (see Remark 4.6) and f is continuous with respect to both of its arguments since $f(u, t)$ is differentiable with respect to u after Lemma 4.13 and $f(u, \cdot)$ is continuous after Theorem 2.9 with Lemma A.10. From (2.8) we get the representation for $\partial_1 f(u, t)$ as

$$\partial_1 f(u, t) = g(t) + \int_0^t R_{\widetilde{\alpha}}(t, s) g(s)\, ds, \tag{4.33}$$

with $\widetilde{\alpha} := 1 - \alpha$. First we get

$$\frac{(t-s)^{\alpha-1}}{\Gamma(\alpha)} > 0, \quad s \in (0, t). \tag{4.34}$$

Second we have

$$\partial_1 G(u, f(u, s)) < 0, \quad \rho < 0, \tag{4.35}$$

and

$$\partial_1 G(u, f(u, s)) > 0, \quad \rho > 0, \tag{4.36}$$

for all $s \in (0, t)$. To see this, consider that with (4.19), we have that $\partial_1 G(u, w) < 0$ is equivalent to

$$u < \frac{1}{2} - \frac{2w\rho}{\xi^2},$$

which holds since for $\rho < 0$ in case (A) we have $u \leq \lambda/(\xi\rho)$ (see (4.22)) and therefore we get

$$u \leq \frac{\lambda}{\xi\rho} < 0 \leq \frac{1}{2} + \left(-\frac{2w\rho}{\xi^2}\right),$$

for $w \geq 0$ and $\rho < 0$. Then $w \geq 0$ is included in our case because $f(u, s) \geq 0$ for u satisfying case (A) (see Proposition 4.10). Analogously, but not for a symmetric range of u, we have for $\rho > 0$ that $\partial_1 G(u, w) > 0$ is equivalent to

$$u > \frac{1}{2} - \frac{2w\rho}{\xi^2},$$

which holds, since for $\rho > 0$ we need u to be greater or equal than $\lambda/(\xi\rho) \vee 1$ (see (4.22)), hence we get

$$\frac{1}{2} - \frac{2w\rho}{\xi^2} \leq \frac{1}{2} < 1 \leq u,$$

for $w \geq 0$ and $\rho > 0$. As above, $w \geq 0$ holds, since $f(u,s) \geq 0$ here. Now, with (4.34)–(4.36) we follow that

$$g(t) = \int_0^t \underbrace{\frac{(t-s)^{\alpha-1}}{\Gamma(\alpha)}}_{>0} \underbrace{\partial_1 G(u, f(u,s))}_{<0} \, ds < 0, \quad \rho < 0, \tag{4.37}$$

$$g(t) = \int_0^t \underbrace{\frac{(t-s)^{\alpha-1}}{\Gamma(\alpha)}}_{>0} \underbrace{\partial_1 G(u, f(u,s))}_{>0} \, ds > 0, \quad \rho > 0. \tag{4.38}$$

If we are able to show that $R_{\widetilde{\alpha}} \geq 0$ in (4.33) this will lead to

$$g(s) < 0, \quad \rho < 0, \tag{4.39}$$

$$g(s) > 0, \quad \rho > 0, \tag{4.40}$$

for $s \in (0,t)$. In fact, $R_{\widetilde{\alpha}} \geq 0$ follows directly from the representation (2.9), since

$$\Phi_1(t,s;\alpha) := K^{(u)}(t,s) \geq 0, \tag{4.41}$$

$$\Phi_n(t,s;\alpha) := \int_0^1 I(z) \, dz, \quad n \geq 2, \tag{4.42}$$

with

$$I(z) := (1-z)^{-\widetilde{\alpha}} z^{(n-1)(1-\widetilde{\alpha})} K^{(u)}(t, s+(t-s)z) \Phi_{n-1}(s+(t-s)z, s; \widetilde{\alpha})$$

because of $(1-z), z \geq 0$ and $\phi_{n-1}(s+(t-s)z, s; \widetilde{\alpha}) \geq 0$ inductively. The fact that $K^{(u)}(t,s) \geq 0$ can be seen, as $\partial_2 G(u, f(u,s)) \geq 0$ is equivalent to

$$2w + (\rho\xi u - \lambda) \geq 0$$

for $w \geq 0$, which holds, since $f(u,s) \geq 0$ and $(\rho\xi u - \lambda) \geq 0$ in case (A). Finally we get

$$Q(t,s,\widetilde{\alpha}) := \sum_{n=1}^{\infty} (t-s)^{(n-1)(1-\widetilde{\alpha})} \Phi_n(t,s;\widetilde{\alpha}) \geq 0,$$

which leads to

$$R_{\widetilde{\alpha}}(t,s) = (t-s)^{-\widetilde{\alpha}} Q(t,s;\widetilde{\alpha}) \geq 0.$$

Hence (4.39) and (4.40) hold and we get $\partial_1 f(u,t) < 0$ for $\rho < 0$ and $\partial_1 f(u,t) > 0$ for $\rho > 0$. Note that for $\rho < 0$ we need u to be negative and for $\rho > 0$ we need u to positive, such that statement (i) is proven.

Ad (ii). Let us assume that $u < 0$ because the approach for $u > 0$ is exactly the same. We have to show that

$$\tau(u) := \inf\{0 < t < \widehat{T}_\alpha(u) : \partial_1 f(u,\cdot) < 0 \text{ on } (t, \widehat{T}_\alpha(u))\} \tag{4.43}$$

satisfies $\tau(u) < \widehat{T}_\alpha(u)$. We use the facts that

$$\partial_1 G(u,w) = \frac{1}{2}\xi^2 + w\rho\xi - \frac{1}{4}\xi^2 < 0, \quad w \to \infty, \tag{4.44}$$

$$\partial_2 G(u,w) = 2w + (\rho\xi u - \lambda) > 0, \quad w \to \infty, \tag{4.45}$$

$$f(u,t) \to \infty, \quad t \to \widehat{T}_\alpha(u). \tag{4.46}$$

Note that (4.46) directly follows from Proposition 4.10. Thus, g from (4.32) satisfies

$$\lim_{t \nearrow \widehat{T}_\alpha(u)} g(t) = -\infty, \tag{4.47}$$

and $K^{(u)}$ satisfies $\lim_{t \nearrow \widehat{T}_\alpha}(u) K^{(u)}(t) = +\infty$. We can therefore pick $\varepsilon > 0$ such that

$$g(t) < 0,\ K^{(u)}(t) > 0 \quad \text{for} \quad \widehat{T}_\alpha(u) - \varepsilon \le t < \widehat{T}_\alpha(u).$$

For $z \in [0,1]$ and any $s, t < \widehat{T}_\alpha(u)$ satisfying $\widehat{T}_\alpha(u) - \varepsilon \le s \le t$, we have

$$s + (t-s)z \ge s \ge \widehat{T}_\alpha(u) - \varepsilon.$$

Using this observation in (4.41), we see from induction that

$$\Phi_n(t,s;\widetilde{\alpha}) \ge 0, \quad n \ge 1,\ \widehat{T}_\alpha(u) - \varepsilon \le s \le t < \widehat{T}_\alpha(u).$$

The same then holds for the resolvent kernel,

$$R_{\widetilde{\alpha}}(t,s) \ge 0, \quad \widehat{T}_\alpha(u) - \varepsilon \le s \le t < \widehat{T}_\alpha(u). \tag{4.48}$$

By (4.33), we obtain

$$\partial_1 f(u,t) = g(t) + \int_0^{t-\varepsilon} R_{\widetilde{\alpha}}(t,s) g(s)\, ds + \int_{t-\varepsilon}^t R_{\widetilde{\alpha}}(t,s) g(s)\, ds. \tag{4.49}$$

Now note that

$$\left| \int_0^{t-\varepsilon} R_{\widetilde{\alpha}}(t,s) g(s)\, ds \right| \ll - \int_{t-\varepsilon}^t R_{\widetilde{\alpha}}(t,s) g(s)\, ds, \quad \text{as } t \nearrow \widehat{T}_\alpha(u), \tag{4.50}$$

where the right-hand side is positive. Indeed, (4.50) follows from (4.47) and (4.48), as $g(s)$ on the left-hand side of (4.50) is $\mathcal{O}(1)$, since it is continuous on $(0, t-\varepsilon)$. Thus, letting $t \nearrow \widehat{T}_\alpha(u)$, we find that the negative terms $g(t)$ and $\int_{t-\varepsilon}^t R_{\widehat{\alpha}}(t,s) g(s)\, ds$ on the right-hand side of (4.49) dominate. This completes the proof. $\square$

Lemma 4.15. *Let $u \in \mathbb{R}$ and $0 < t < \widehat{T}_\alpha(u)$. Then $f(\cdot, t)$ is analytic at u.*

Proof. See [GGP18, Lemma 5.4]: According to [Bru17, Section 3.1.1], the solution can be constructed by successive iteration and continuation. In [GGP18] is just shown that the first iteration step leads to an analytic function, because the finitely many further steps needed to arrive at arbitrary $t < \widehat{T}_\alpha(u)$ can be dealt with analogously. Define the iterates $f_0 = 0$ and

$$f_{n+1}(v, s) := \frac{1}{\Gamma(\alpha)} \int_0^s (s-\tau)^{\alpha-1} G(v, f_n(v, \tau))\, d\tau, \quad n \geq 0.$$

On a sufficiently small time interval, $f_n(v, \cdot)$ converges uniformly to $f(v, \cdot)$, and the solution can then be continued by solving an updated integral equation and so on (see [Bru17, Theorem 3.1.2]), until we hit $\widehat{T}_\alpha(v)$. Now fix u and t as in the statement of the lemma. For a sufficiently small open complex neighborhood U of u, it is easy to see that $t < \widehat{T}_\alpha(v)$ holds for $v \in U$. Define

$$\begin{aligned} \gamma &:= 1 \vee \sup_{v \in U} |v|, \\ \eta &:= 1 \vee \frac{t^\alpha}{\Gamma(\alpha+1)}. \end{aligned}$$

There is $c \geq 1$ such that, for arbitrary $v \in U$ and $w \in \mathbb{C}$,

$$\begin{aligned} |G(v, w)| &\leq c\left((|w| \vee 1)^2 \vee \gamma(|w| \vee 1) \vee \gamma^2\right) \\ &\leq c\gamma^2 (|w| \vee 1)^2 \\ &=: \theta(|w| \vee 1)^2. \end{aligned}$$

To see this, note that

$$\begin{aligned} w \leq (|w| \vee 1) &\leq c(|w| \vee 1), \\ (\rho\xi u - \lambda) &\leq c\gamma, \\ \frac{1}{4}\xi^2 u(u-1) &\leq c\gamma, \end{aligned}$$

with $c \geq 1$ appropriate and consider the representation (4.20) of $G(u, w)$. By the definition of f_n, we get that

$$\sup_{\substack{v \in U \\ s \in [0,t]}} |f_n(v, s)| \leq (\theta\eta)^{2^n - 1}, \quad n \geq 0. \tag{4.51}$$

This can be seen inductively, since if (4.51) holds, we get

$$\begin{aligned}
|f_{n+1}(v,s)| &\leq \frac{1}{\Gamma(\alpha)} \int_0^s (s-\tau)^{\alpha-1} |G(v, f_n(v,\tau))| \, d\tau \\
&\leq \frac{1}{\Gamma(\alpha)} \int_0^s (s-\tau)^{\alpha-1} \theta \left(|f_n(v,\tau)| \vee 1 \right)^2 \, d\tau \\
&\leq \frac{1}{\Gamma(\alpha)} \int_0^s (s-\tau)^{\alpha-1} \theta ((\theta\eta)^{2^n-1} \vee 1)^2 \, d\tau \\
&\overset{\theta,\eta\geq 1}{=} \theta (\theta\eta)^{2^{n+1}-2} \frac{1}{\Gamma(\alpha)} \int_0^s (s-\tau)^{\alpha-1} \, d\tau \\
&= \theta^{-1}\eta^{-2}(\theta\eta)^{2^{n+1}} \frac{t^\alpha}{\Gamma(\alpha+1)} \\
&\leq \theta^{-1}\eta^{-2}(\theta\eta)^{2^{n+1}} \left(\frac{t^\alpha}{\Gamma(\alpha+1)} \vee 1 \right) \\
&= (\theta\eta)^{2^{n+1}-1}.
\end{aligned}$$

By a standard result on parameter integrals ([Els05, Theorem IV.5.8]), the bound (4.51) implies that each function $f_n(\cdot,t)$ is analytic in U. From the bounds in [Bru17, Section 3.1.1], it is very easy to see that the convergence $f_n(v,t) \to f(v,t)$ is locally uniform with respect to v for fixed t. It is well-known (see [GK06, Theorem 3.5.1]) that this implies that the limit function $f(\cdot,t)$ is analytic. □

Lemma 4.16. *The function $u \mapsto \widehat{T}_\alpha(u)$ increases for $u \leq 0$ and decreases for $u \geq 1$.*

Proof. See [GGP18, Lemma 5.5]: Recall that $\widehat{T}_\alpha = \infty$ in cases (C) and (D), which include $u \in [0,1]$. For case (A), the assertion directly follows from Lemma 4.14 (i). So let u satisfy case (B), where again we assume without loss of generality that $u < 0$. Suppose that $\widehat{T}_\alpha(\cdot)$ does not increase. Then we can pick $u_0 < 0$ such that any left neighborhood of u_0 contains a point u with $\widehat{T}_\alpha(u) > \widehat{T}_\alpha(u_0)$. From the continuity of $\partial_1 f(u,t)$ (see Lemma 4.15), Lemma 4.14 (ii), and the continuity of τ from (4.43), there are $u_1 < u_0$ satisfying $\widehat{T}_\alpha(u_1) > \widehat{T}_\alpha(u_0)$ and $t_1 < \widehat{T}_\alpha(u_0)$ such that $\partial_1 f(u,t) < 0$ in the rectangle

$$\{(u,t) : u_1 \leq u \leq u_0,\, t_1 \leq t < \widehat{T}_\alpha(u_1)\}. \tag{4.52}$$

Then, $\lim_{t \nearrow \widehat{T}_\alpha(u_0)} f(u_0,t) = \infty$ implies that

$$\lim_{t \nearrow \widehat{T}_\alpha(u_0)} f(u_1,t) = \infty, \tag{4.53}$$

because the inequality $\partial_1 f(u,t) < 0$ shows that $f(u_1,\cdot)$ must explode at least as fast as $f(u_0,\cdot)$. But (4.53) is a contradiction to $\widehat{T}_\alpha(u_1) > \widehat{T}_\alpha(u_0)$. □

Lemma 4.17. *Let $u \in \mathbb{R}$ and $0 < t < \widehat{T}_\alpha(u)$. Then (4.7) holds, where $f = c_3\psi$ and $f(u,\cdot)$ is the solution of (4.14).*

Proof. See [GGP18, Lemma 5.6]: We assume that $u < 0$, as $u \geq 0$ is handled analogously. By Lemma 4.16, $u \mapsto \widehat{T}_\alpha(u)$ increases. In the proof, we write $M(u,t)$ for the right-hand side of (4.7), and $\widetilde{M}(u,t) := \mathbb{E}[e^{uX_t}]$ for the moment generating function. Now fix $u < 0$ and $0 < t < \widehat{T}_\alpha(u)$ such that (u,t) has positive distance from the graph of the increasing function $\widehat{T}_\alpha(\cdot)$. Clearly, it suffices to consider pairs (u,t) with this property. By Corollary 4.4, there are $v^- < v^+$ such that

$$M(v,t) = \widetilde{M}(v,t), \quad v^- < v < v^+. \tag{4.54}$$

We now show that (4.54) extend to $u \leq v \leq v^+$ by analytic continuation. From general results on characteristic functions ([Wid41, Theorems II.5a and II.5b]), $v \mapsto \widetilde{M}(v,t)$ is analytic in a vertical strip $w^- < \mathrm{Re}(v) < w^+$ of the complex plane, and has a singularity at $v = w^-$. If we suppose that $w^- > u$, then Lemma 4.15 leads to a contradiction: The left-hand side of (4.54) would then be analytic at $v = w^-$, and the right-hand side singular. This shows that (4.54) can be extended to the left up to u by analytic continuation. □

Theorem 4.18. *Let $u \in \mathbb{R}$. Then $\widehat{T}_\alpha(u) = T^*_\alpha(u)$, and (4.7) holds for $0 < t < T^*_\alpha(u)$.*

Proof. See [GGP18, Theorem 5.7]: In the light of Lemma 4.17, it only remains to show that $\widehat{T}_\alpha(u) \geq T^*_\alpha(u)$. (Obviously Lemma 4.17 implies that $\widehat{T}_\alpha(u) \leq T^*_\alpha(u)$). Since $t \mapsto X_t$ is continuous, we have with Doob's submartingale inequality and dominated convergence that the map $t \mapsto \widetilde{M}(u,t) = \mathbb{E}[e^{uX_t}]$ is continuous and therefore we get $\widehat{T}_\alpha(u) \geq T^*_\alpha(u)$. □

Theorem 4.18 now ensures, together with Lemma 4.11, that doing our analysis on the solution $f(u,t)$ of (4.14) is sufficient to know about the explosion time of the moment generating function of the log-price.

Despite we have the focus on real u, since we are interested in the moment generating function, it may be helpful to be consistent for complex-valued u, since the characteristic function is necessary for option pricing. So, Gerhold et al. argue in [GGP18, Section 6] that the results above transmit to the complex case.

Theorem 4.19. *Let $u \in \mathbb{C}$. Then $T^*_\alpha(u) = T^*_\alpha(Re(u))$, and (4.7) holds for $0 < t < T^*_\alpha(u)$.*

Lemma 4.20. *Let $u \in \mathbb{C}$. Then $\widehat{T}_\alpha(u) \geq T^*_\alpha(u)$.*

Proof. See [GGP18, Lemma 6.2]: Suppose that $\widehat{T}_\alpha(u) < T^*_\alpha(u)$. The Volterra integral equation (4.14) translates into a two-dimensional Volterra integral equation for $(\mathrm{Re}(f), \mathrm{Im}(f))$. Since Gripenberg et al. did their analysis very general in a multidimensional setting, we get from [GLS90, Theorem 12.1.1] that $(\mathrm{Re}(f), \mathrm{Im}(f))$ explodes for $t \nearrow \widehat{T}_\alpha < \infty$. This contradicts the continuity of $t \mapsto \mathbb{E}[e^{uX_t}]$, which is used in the proof of Theorem 4.18. $\square$

Proof of Theorem 4.19. [GGP18, Theorem 6.2]: The first statement is clear since $|e^{uX_t}| = |e^{\mathrm{Re}(u)X_t}|$. Now let $t > 0$ be arbitrary. As above, we write $\widetilde{M}$ for the moment generating function and M for the right-hand side of (4.7). By Theorem 4.18 we have $M(v,t) = \widetilde{M}(v,t)$ for v in the real interval

$$I := \{v \in \mathbb{R} : T^*_\alpha(v) \geq t\}.$$

The function $\widetilde{M}(\cdot, t)$ is analytic on the strip

$$\{v \in \mathbb{C} : \mathrm{Re}(v) \in I\} = \{v \in \mathbb{C} : T^*_\alpha(v) \geq t\}. \tag{4.55}$$

By the same argument as in Lemma 4.15, the function $M(\cdot, t)$ is analytic on the set $\{v \in \mathbb{C} : \widehat{T}_\alpha(v) \geq t\}$, which contains the strip (4.55) by Lemma 4.20. Therefore, $M(\cdot, t)$ and $\widetilde{M}(\cdot, t)$ agree on (4.55) by analytic continuation. This implies the assertion. $\square$

5. Numerics – The Theory

After the discussion in Chapter 4, especially Theorem 4.18, one of the main results, we now know, is that if we have an idea about the behavior of ψ, the solution of (4.13), we have an idea about the behavior of the moment generating function. Hence, it is of great interest to have an (at least) numerical handling of ψ. In this chapter we will introduce an algorithm to get a numerical solution for the Volterra integral equation (4.13). Furthermore, with this numerical solution we want to use different approaches to compute the explosion time of ψ and some asymptotic properties. More precisely, Gerhold et al. suggest in [GGP18] Algorithm 5.7 for computing the explosion time if u satisfies case (A) from Chapter 4 and Algorithm 5.8 for computing a lower bound for the explosion time if u satisfies case (B).

5.1. Numerical scheme for the Riccati equation

In the following the numerical scheme used in [ER16] is presented. Let us write $g(u,t) = R(u,\psi(u,t))$. Over a regular discrete time-grid $(t_k)_{k\in\{1,\dots,N\}}$ with mesh Δ ($t_k = k\Delta$), we estimate

$$\psi(u,t_{k+1}) = \frac{1}{\Gamma(\alpha)}\int_0^{t_{k+1}} (t_{k+1}-s)^{\alpha-1} g(u,s)\,ds$$

by

$$\widehat{\psi}(u,t_{k+1}) = \frac{1}{\Gamma(\alpha)}\int_0^{t_{k+1}} (t_{k+1}-s)^{\alpha-1} \widehat{g}(u,s)\,ds,$$

where

$$\widehat{g}(u,t) = \frac{t_{j+1}-t}{t_{j+1}-t_j}\widehat{g}(u,t_j) + \frac{t-t_j}{t_{j+1}-t_j}\widehat{g}(u,t_{j+1}), \quad t\in[t_j,t_{j+1}), \quad 0\le j\le k.$$

This corresponds to a trapezoidal discretization of the fractional integral and leads to the following scheme:

$$\widehat{\psi}(u,t_{k+1}) = \sum_{0\le j\le k}\Big(a_{j,k+1}R(u,\widehat{\psi}(u,t_j))\Big) + a_{k+1,k+1}R(u,\widehat{\psi}(u,t_{k+1})), \tag{5.1}$$

with

$$\begin{aligned}
a_{0,k+1} &= \frac{\Delta^\alpha}{\Gamma(\alpha+2)}(k^{\alpha+1} - (k-\alpha)(k+1)^\alpha),\\
a_{j,k+1} &= \frac{\Delta^\alpha}{\Gamma(\alpha+2)}((k-j+2)^{\alpha+1} + (k-j)^{\alpha+1} - 2(k-j+1)^{\alpha+1}), \quad 1 \le j \le k,\\
a_{k+1,k+1} &= \frac{\Delta^\alpha}{\Gamma(\alpha+2)}.
\end{aligned} \tag{5.2}$$

However, $\widehat{\psi}(u, t_{k+1})$ being on both sides of (5.1), this scheme is implicit. Thus, in a first step, we compute a pre-estimation of $\widehat{\psi}(u, t_{k+1})$ based on Riemann sum that we then plug into the trapezoidal quadrature. This pre-estimation, called predictor and that we denote by $\widehat{\psi}^P(u, t_{k+1})$, is defined by

$$\widehat{\psi}^P(u, t_{k+1}) = \frac{1}{\Gamma(\alpha)} \int_0^t (t-s)^{\alpha-1} \widetilde{g}(u, s)\, ds, \tag{5.3}$$

with

$$\widetilde{g}(u, t) = \widehat{g}(u, t_j), \quad t \in [t_j, t_{j+1}), \quad 0 \le j \le k.$$

Therefore,

$$\widehat{\psi}^P(u, t_{k+1}) = \sum_{0 \le j \le k} b_{j,k+1} R(u, \widehat{\psi}(u, t_j)), \tag{5.4}$$

where

$$b_{j,k+1} = \frac{\Delta^\alpha}{\Gamma(\alpha+1)}((k-j+1)^\alpha - (k-j)^\alpha), \quad 0 \le j \le k. \tag{5.5}$$

Thus, the final explicit numerical scheme is given by

$$\widehat{\psi}(u, t_{k+1}) = \sum_{0 \le j \le k} \Big(a_{j,k+1} R(u, \widehat{\psi}(u, t_j)) \Big) + a_{k+1,k+1} R(u, \widehat{\psi}^P(u, t_{k+1})). \tag{5.6}$$

El Euch and Rosenbaum note in [ER16] that theoretical guarantees for the convergence of this scheme are provided in [LT09]. In particular, it is shown that for given $t > 0$ and $u \in \mathbb{R}$,

$$\max_{t_j \in [0,t]} \left| \widehat{\psi}(u, t_j) - \psi(u, t_j) \right| = o(\Delta)$$

and

$$\max_{t_j \in [\varepsilon, t]} \left| \widehat{\psi}(u, t_j) - \psi(u, t_j) \right| = o(\Delta^{2-\alpha}),$$

for any $\varepsilon > 0$.

From this description we get the following algorithm for a predetermined specific choice for the model parameters in (4.1)–(4.3).

Algorithm 5.1. *Let u be a real number and $t > 0$ a given time point.*

- *Fix $n \in \mathbb{N}$ for the number of intervals for an equidistant grid on $[0, t]$.*
- *For $k \in \{0, \ldots, n-1\}$ compute for all j the $a_{j,k+1}$ in (5.2), the $b_{j,k+1}$ in (5.5), and evaluate $\widehat{\psi}^P(u, t_{k+1})$ in (5.4) and finally $\widehat{\psi}(u, t_{k+1})$ in (5.6).*

Remark 5.2. Note that running Algorithm 5.1 can be quite slow. In Section 6.2 for the implementation, there is a description how vector arithmetics can be used, and we profit from the fact that the $a_{j,k+1}$ and the $b_{j,k+1}$ only depend on the lag $\ell = k - j$ for most of the j such that we do not need to compute these values in every iteration step.

For the implementation of the numerical scheme of this section see Section 6.2 and for the auxiliary functions resp. for understanding how the model is handled in the source code see Section 6.1. As appetizer for the following we have some plots to see how our computations are graphically displayed in Figure 5.1, for which the source code can also be found in Section 6.2. Here we used the rough Heston model of (4.1)–(4.3) with

$$\overline{v} = 0.04,\ \lambda = 0.3,\ \xi = 1,\ v_0 = 1,\ \rho = -0.7,\ \alpha = 0.6, \tag{5.7}$$

which will be used in all the examples.

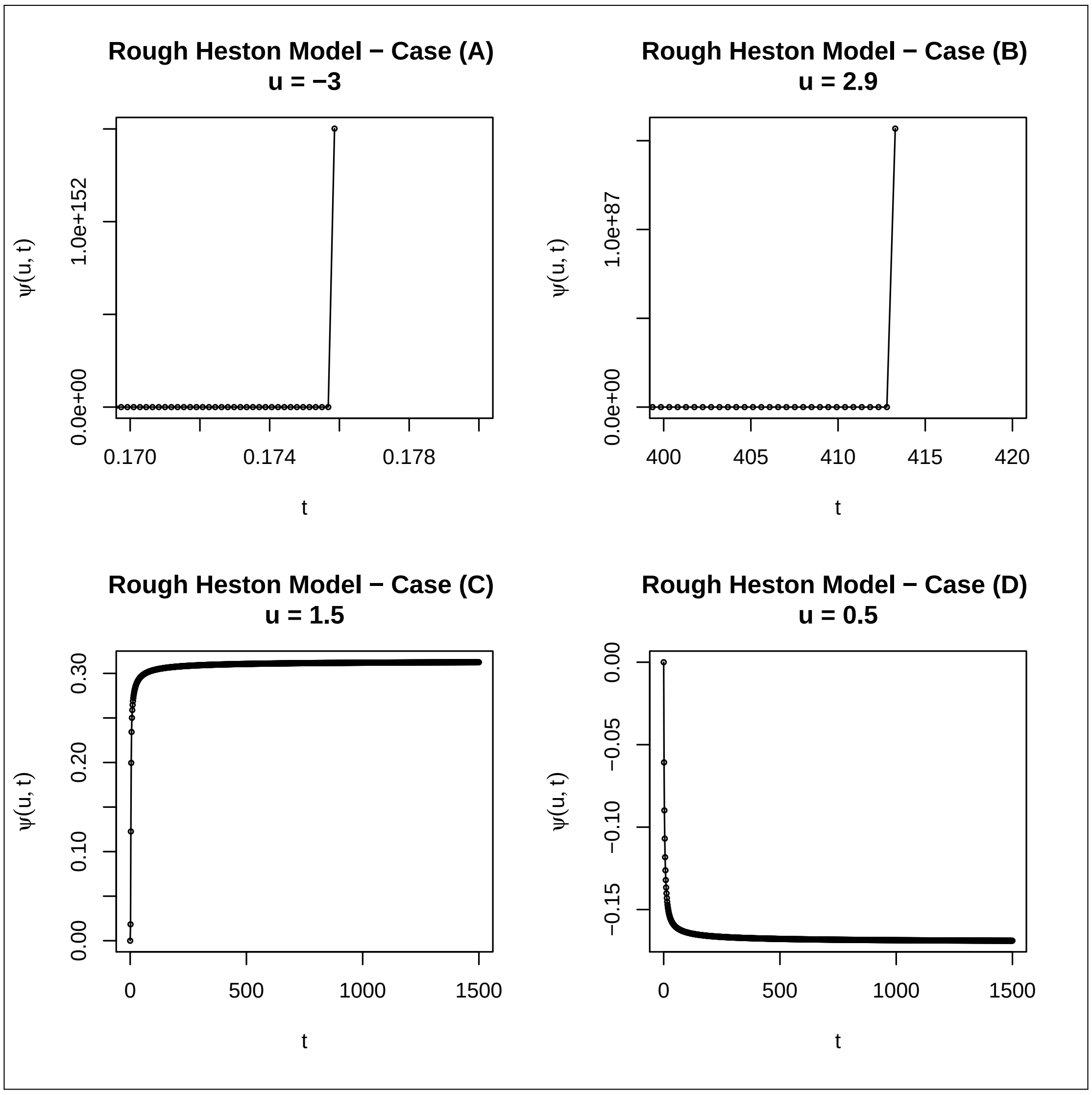

Figure 5.1.: Graphical example for the numerical implementation of $\psi(u, t)$

5.2. Computing the explosion time

In this section I present a way suggested in [GGP18, Section 6] to compute the explosion time at least numerically if u satisfies case (A). For case (B) we get an algorithm for an approximation of a lower bound, which is after [GGP18] sometimes sharper than the explicit bound from Theorem 4.12.

The solution f of (4.14) satisfies the fractional Riccati equation

$$D_t^\alpha f = d_1 + d_2 f + f^2, \tag{5.8}$$

where $d_1(u) = c_1(u)c_3$ and $d_2(u) = c_2(u)$, with initial condition $I^{1-\alpha} f(0) = 0$. We try a fractional power series ansatz

$$f(t) \overset{!}{=} \sum_{n=1}^{\infty} a_n(u) t^{\alpha n} \tag{5.9}$$

with unknown coefficients $a_n = a_n(u)$. By Lemma 3.4 (i), the fractional power series (5.9) (formally) satisfies the initial condition. Inserting (5.9) into (5.8) and using Lemma 3.4 (i) again, we obtain

$$\begin{aligned} \sum_{n=0}^{\infty} a_{n+1} v_{n+1} t^{\alpha n} &= d_1 + \sum_{n=0}^{\infty} d_2 a_n t^{\alpha n} + \sum_{n=2}^{\infty} \left(\sum_{k=1}^{n-1} a_k a_{n-k} \right) t^{\alpha n} \\ &= d_1 + d_2 a_1 t^\alpha + \sum_{n=2}^{\infty} \left(d_2 a_n + \sum_{k=1}^{n-1} a_k a_{n-k} \right) t^{\alpha n}, \end{aligned} \tag{5.10}$$

where

$$v_n = \frac{\Gamma(\alpha n + 1)}{\Gamma(\alpha n - \alpha + 1)}. \tag{5.11}$$

Note that v_n is an increasing sequence; this follows from the fact that $\log \circ \Gamma$ is convex (see Example 11.14 in [Sch05]). By Stirling's formula (Theorem A.6), $v_n \sim (\alpha n)^\alpha$ for $n \to \infty$. From (5.10), we obtain the following convolution recursion for $a_n = a_n(u)$:

$$\begin{aligned} a_1 &= \frac{d_1}{v_1}, \\ a_{n+1} &= \frac{1}{v_{n+1}} \left(d_2 a_n + \sum_{k=1}^{n-1} a_k a_{n-k} \right), \quad n \geq 1. \end{aligned} \tag{5.12}$$

The function f can thus be expressed as $f(u,t) = F(u, t^\alpha)$, where

$$F(u,z) = \sum_{n=1}^{\infty} a_n(u) z^n. \tag{5.13}$$

Lemma 5.3. *Let $u \in \mathbb{R}$, satisfying case (A) from Chapter 4. Then $F(u, \cdot)$ is analytic at zero, with a positive and finite radius of convergence $R(u)$.*

Proof. See [GGP18, Lemma 7.2]: To see that the radius of convergence is positive, we show that there is a number $A = A(u) > 0$ such that

$$|a_n| \leq A^n n^{\alpha-1}, \quad n \geq 1. \tag{5.14}$$

Gerhold et al. note that the factor $n^{\alpha-1}$ is chosen in order to facilitate the proof for this geometric bound. We have

$$\frac{\alpha^{-\alpha}|d_2|n^{-1}}{(n+1)^{\alpha-1}} + \frac{2\alpha^{-\alpha}\Gamma(\alpha)^2 n^{\alpha-1}}{\Gamma(2\alpha)(n+1)^{\alpha-1}} \to \frac{2\alpha^{-\alpha}\Gamma(\alpha)^2}{\Gamma(2\alpha)}, \quad n \to \infty, \tag{5.15}$$

since for $\alpha \in (0,1)$ we have

$$\frac{n^{-1}}{(n+1)^{\alpha-1}} = \frac{(n+1)^{1-\alpha}}{n} = \left(\frac{n+1}{n}\right)^{1-\alpha} \frac{1}{n^\alpha} = \left(1 + \frac{1}{n}\right)^{1-\alpha} \frac{1}{n^\alpha} \to 0, \quad n \to \infty,$$

and

$$\frac{n^{\alpha-1}}{(n+1)^{\alpha-1}} = \left(\frac{n}{n+1}\right)^{\alpha-1} = \left(1 - \frac{1}{n+1}\right)^{\alpha-1} \to 1, \quad n \to \infty.$$

Choose n_0 such that the left-hand side of (5.15) is bounded by $3\alpha^{-\alpha}\Gamma(\alpha)^2/\Gamma(2\alpha)$ for all $n \geq n_0$, and such that $2v_n \geq (\alpha n)^\alpha$ for all $n \geq n_0$. The latter is possible because $v_n \sim (\alpha n)^\alpha$. Fix a number A with $A \geq 3\alpha^{-\alpha}\Gamma(\alpha)^2/\Gamma(2\alpha)$ and such that $A^n n^{\alpha-1} \geq |a_n|$ holds for $1 \leq n \leq n_0$. Let $n \geq n_0$ and assume, inductively, that $|a_k| \leq A^k k^{\alpha-1}$ holds for $1 \leq k \leq n$. From the recurrence (5.12), we then obtain

$$\begin{aligned}
|a_{n+1}| &\leq 2(\alpha n + \alpha)^{-\alpha} \left(|d_2| A^n n^{\alpha-1} + A^n \sum_{k=1}^{n-1} k^{\alpha-1}(n-k)^{\alpha-1}\right) \\
&\leq 2(\alpha n)^{-\alpha} \left(|d_2| A^n n^{\alpha-1} + A^n \sum_{k=1}^{n-1} k^{\alpha-1}(n-k)^{\alpha-1}\right).
\end{aligned}$$

Since $x^{\alpha-1}(n-x)^{\alpha-1}$ is a strictly convex function of x on $(0,n)$ with minimum at $n/2$, it is easy to see that

$$\begin{aligned}
\sum_{k=1}^{n-1} k^{\alpha-1}(n-k)^{\alpha-1} &\leq \int_0^n x^{\alpha-1}(n-x)^{\alpha-1}\,dx \\
&= n^{2\alpha-1} \int_0^1 y^{\alpha-1}(1-y)^{\alpha-1} \\
&= n^{2\alpha-1} B(\alpha,\alpha) \\
&= n^{2\alpha-1} \frac{\Gamma(\alpha)^2}{\Gamma(2\alpha)}.
\end{aligned}$$

We conclude

$$\begin{aligned}|a_{n+1}| &\leq 2\alpha^{-\alpha}|d_2|A^n n^{-1} + \frac{2\alpha^{-\alpha}\Gamma(\alpha)^2}{\Gamma(2\alpha)}A^n n^{\alpha-1} \\ &= A^n(n+1)^{\alpha-1}\left(\frac{2\alpha^{-\alpha}|d_2|n^{-1}}{(n+1)^{\alpha-1}} + \frac{2\alpha^{-\alpha}\Gamma(\alpha)^2 n^{\alpha-1}}{\Gamma(2\alpha)(n+1)^{\alpha-1}}\right) \\ &\leq A^n(n+1)^{\alpha-1}\frac{3\alpha^{-\alpha}\Gamma(\alpha)^2}{\Gamma(2\alpha)} \\ &\leq A^{n+1}(n+1)^{\alpha-1}.\end{aligned}$$

This completes the inductive proof of (5.14).

The finiteness of the radius of convergence will follow from the existence of a number $B = B(u) > 0$ such that

$$a_n \geq B^n, \quad n \geq 1. \tag{5.16}$$

To this end, define

$$r_n := \frac{d_2 + n - 1}{v_{n+1}}, \quad n \geq 1.$$

By Stirling's formula (Theorem A.6), we have $r_n/r_{n-1} = 1 + (1-\alpha)/n + \mathcal{O}(n^{-2})$ as $n \to \infty$, and so r_n eventually increases. Let $n_0 \geq 2$ be such that r_n increases for $n \geq n_0$, and define

$$B := \min\{r_{n_0}, a_1, a_2^{1/2}, \dots, a_{n_0}^{1/n_0}\}.$$

This number satisfies $a_n \geq B^n$ for $n \leq n_0$ by definition. Let us fix some $n \geq n_0$ and assume, inductively, that $a_k \geq B^k$ holds for $1 \leq k \leq n$. By (5.12) we get

$$\begin{aligned}a_{n+1} &\geq \frac{1}{v_{n+1}}(d_2 B^n + (n-1)B^n) \\ &= B^n r_n \geq B^n r_{n_0} \\ &\geq B^{n+1}\end{aligned}$$

Thus, (5.16) is proved by induction. □

From the estimates in the proof of Lemma 5.3, it is clear that termwise fractional derivation of the series (5.9) is allowed, and so the right-hand side of (5.9) really represents the solution f of (5.8) with initial condition $I_t^{1-\alpha}f(0) = 0$, as long as t satisfies $0 \leq t < R(u)^{1/\alpha}$. We proceed to show how the explosion time $T_\alpha^*(u)$ can be computed from the coefficients $a_n(n)$. The essential fact is that there is no gap between $R(u)^{1/\alpha}$ and $T_\alpha^*(u)$. For this, we require the following classical result from complex analysis.

Theorem 5.4 (Pringsheim's theorem)**.** *Suppose that the power series $F(z) = \sum_{n=0}^{\infty} a_n z^n$ has positive finite radius of convergence R, and that all but finitely many of the coefficients are non-negative real numbers. Then F has a singularity at R, i.e. a singularity on the real axis $\mathbb{R}$.*

Proof. See [Rem91, p. 235, Section 8.1.5]. □

Lemma 5.5. *Let $f(u, \cdot)$ be the solution of (5.8). Then $f(u, \cdot)$ is analytic on the whole interval $(0, T_\alpha^*(u))$.*

Proof. Gerhold et al. [GGP18] suggest to use [Lub83, Theorem 1], such that we directly attain the result. □

Theorem 5.6. *Suppose that $u \in \mathbb{R}$ satisfies case (A). Define the sequence $a_n(u)$ by the recurrence (5.12). Then we have*

$$\limsup_{n\to\infty} a_n(u)^{-1/(\alpha n)} = T_\alpha^*(u). \tag{5.17}$$

Proof. See [GGP18, Theorem 7.4]: Recall that $f(u, \cdot)$, the solution of (5.8), also solves the Volterra integral equation (4.14). By Lemma 5.5, we have that $f(u, \cdot)$ is analytic on the whole interval $(0, T_\alpha^*(u))$. As $f(u, t)$ blows up for $t \nearrow T_\alpha^*(u)$ by Proposition 4.10, and $t \mapsto F(u, t^\alpha)$ is analytic on $(0, R(u)^{1/\alpha})$, we must have $R(u)^{1/\alpha} \leq T_\alpha^*(u)$.

Assume for contradiction that $R(u)^{1/\alpha} < T_\alpha^*(u)$. Then f is analytic at $R(u)^{1/\alpha}$. But since $z \mapsto z^{1/\alpha}$ is analytic at $R(u) > 0$, the composition $F(u, z) = f(u, z^{1/\alpha})$ would be analytic at $z = R(u)$ as well, which contradicts Theorem 5.4. Therefore,

$$R(u)^{1/\alpha} = T_\alpha^*(u). \tag{5.18}$$

It is well-known that the radius of convergence is given by the Cauchy-Hadamard formula (see e.g. [Rem91, Section 4.1.3])

$$R(u)^{-1} = \limsup_{n\to\infty} a_n(u)^{1/n}, \tag{5.19}$$

which concludes the proof. □

Note that, in case (B), we can argue similarly as in the preceding proof. However, the coefficients a_n are no longer positive and so Pringsheim's theorem (Theorem 5.4) is not applicable. Then, the inequality $R(u)^{1/\alpha} \leq T_\alpha^*(u)$ need not be an equality. Still, we can compute a lower bound for the explosion time:

$$\limsup_{n\to\infty} |a_n(u)|^{-1/(\alpha n)} \leq T_\alpha^*(u). \tag{5.20}$$

In the following we try to improve the asymptotics (5.17) in order to save computation time. Our approach is to apply Lemma 2.13 to our equation (4.14). The Volterra integral equation (4.14) can be written in the representation of (2.16) with

$$k(t-s) := k_0(t-s) = \frac{(t-s)^{\alpha-1}}{\Gamma(\alpha)} \geq 0, \quad t > s \geq 0,$$

where we have

$$k'(t-s) = -(1-\alpha)\frac{(t-s)^{\alpha-2}}{\Gamma(\alpha)} < 0, \quad t > s \geq 0.$$

Hence the requirements (2.19) and (2.20) for the kernel hold. Then clearly

$$f(t) := f(u,t) \to \infty, \quad t \nearrow \widehat{T}_\alpha(u) = T^*_\alpha(u),$$

hence (2.22) holds. For $r \equiv 1$ and $h \equiv 0$ we then get

$$g(w) := G(u,w) = c_1(u)c_3 + c_2(u)w + w^2 \sim w^2, \quad w \to \infty,$$

such that (2.18) follows. For u satisfying case (A), we have $c_1(u) > 0$ and $c_2(u) \geq 0$, such that we get with $c_3 > 0$ that

$$g(w) > 0, \quad w > 0.$$

The same argument holds for g', such that we get

$$g'(w) = c_2(u) + 2w > 0, \quad w > 0.$$

Then, $g''(w) > 0$ and hence (2.17) follows. Now the requirements of Lemma 2.13 are fulfilled, and we get the asymptotics

$$f(t) \sim \frac{\Gamma(2\alpha)}{\Gamma(\alpha)}(T^*_\alpha(u) - t)^{-\alpha}, \quad t \nearrow T^*_\alpha(u). \tag{5.21}$$

But what is with condition (2.21)? In fact, we need h to be greater 0. So, this only suffices for a heuristic approach to have an idea for the asymptotics to test numerically, since actually h does not quite fulfill the requirements of Lemma 2.13 (Cheat alert ☹). At least, we could choose $h \equiv \varepsilon$ and define

$$g(w) := G(u, w-\varepsilon) = c_1(u)c_3 + c_2(u)(w-\varepsilon) + (w-\varepsilon)^2, \tag{5.22}$$

we would obtain the same results, except, that case (A) is just sufficient for $w > \varepsilon$ and not for $w > 0$ as required in Lemma 2.13. So let us write

$$f(t) \overset{?}{\sim} \frac{\Gamma(2\alpha)}{\Gamma(\alpha)}(T^*_\alpha(u) - t)^{-\alpha}, \quad t \nearrow T^*_\alpha(u), \tag{5.23}$$

instead of (5.21), because we do not quite fulfill the requirements, but also because according to [GGP18], not all steps in [RO96] are rigorous. We proceed, heuristically, to infer refined asymptotics of $a_n(u)$ from (5.23). Define

$$\Phi(z) := \sum_{n=1}^{\infty} a_n(u)R(u)^n z^n,$$

a power series with radius of convergence 1, by the definition of $R(u)$ in Lemma 5.3. Its asymptotics for $z \nearrow 1$ can be derived from (5.23). Recall that the explosion time and the radius of convergence of F are related by $T_\alpha^*(u) = R(u)^{1/\alpha}$.

$$\begin{aligned}\Phi(z) &= f((Rz)^{1/\alpha}) \\ &\overset{?}{\sim} \frac{\Gamma(2\alpha)}{\Gamma(\alpha)}(T_\alpha^* - (Rz)^{1/\alpha})^{-\alpha} \\ &= \frac{\Gamma(2\alpha)}{\Gamma(\alpha)}R^{-1}(1-z^{1/\alpha})^{-\alpha} \\ &\sim \frac{\alpha^\alpha\Gamma(2\alpha)}{\Gamma(\alpha)}R^{-1}(1-z)^{-\alpha}, \quad z \nearrow 1.\end{aligned}$$

The method of singularity analysis (see [FS09, Section VI]) allows us to transfer the asymptotics of Φ to asymptotics of its Taylor coefficients a_nR^n. Sweeping some analytic conditions under the rug, Gehold et al. [GGP18] arrive at

$$a_n(u)R(u)^n \overset{?}{\sim} \frac{\alpha^\alpha\Gamma(2\alpha)}{\Gamma(\alpha)}R^{-1}\frac{n^{\alpha-1}}{\Gamma(\alpha)}, \quad n \to \infty,$$

and thus

$$a_n(u) \overset{?}{\sim} R(u)^{-n-1}n^{\alpha-1}\frac{\alpha^\alpha\Gamma(2\alpha)}{\Gamma(\alpha)^2}, \quad n \to \infty. \tag{5.24}$$

According to [GGP18, Section 7] numerical tests confirm (5.24) (also see e.g. Figure 5.3), and there is little doubt that it is true in case (A). Summing up, T_α^* can be computed by the following algorithm, which converges much faster than the simpler approximation (5.17).

Algorithm 5.7. *Let u be a real number satisfying case* (A).

- *Fix $n_{\max} \in \mathbb{N}$ (e.g. $n_{\max} = 100$),*
- *compute $a_1(u), \dots, a_{n_{\max}}(u)$ by the recursion* (5.12),

- *compute the approximation*

$$\left(a_n(u) n^{1-\alpha} \frac{\Gamma(\alpha)^2}{\alpha^\alpha \Gamma(2\alpha)} \right)^{\frac{-1}{\alpha(n+1)}} \Bigg|_{n=n_{\max}} \approx T_\alpha^*(u) \tag{5.25}$$

for the explosion time.

Algorithm 5.8. *Let u be a real number satisfying case (B).*

- *Fix $n_{\max} \in \mathbb{N}$ (e.g. $n_{\max} = 200$),*
- *compute $a_1(u), \dots, a_{n_{\max}}(u)$ by the recursion (5.12),*
- *compute the approximate lower bound*

$$|a_n(u)|^{-1/(\alpha n)} \Big|_{n=n_{\max}} \lessapprox T_\alpha^*(u) \tag{5.26}$$

of the explosion time.

Remark 5.9. Note that it will be advantageous in the implementation to represent the v_n from (5.11) via the Euler beta function to avoid overflows in the gamma function. This can be done via

$$v_n = \frac{\Gamma(\alpha n + 1)}{\Gamma(\alpha n - \alpha + 1)} = \frac{\Gamma(\alpha n + 1)}{\Gamma(\alpha n - \alpha + 1)\Gamma(\alpha)} \Gamma(\alpha) = \frac{\Gamma(\alpha)}{B(\alpha n - \alpha + 1, \alpha)}.$$

The problem of such an overflow occurred to me during the implementation of the algorithms in R, but this is not an R-specific problem since I found out that e.g. for large arguments Python's `math.gamma`-function also gives an error message instead of evaluating to infinity.

In the following we look at some plots to numerically verify the statements of this section. At first glance, if we consider Figure 5.1, we can think of just trying different times t for the end of the interval, where $\psi(u,t)$ is computed with Algorithm 5.1. This is a motivation for thinking of some "graphical" approach, i.e. some iterations with "human decision" to have an idea or a "guess" about the explosion time to be able to compare with the results from the other tested algorithms. The whole method can be seen in Section 6.3. For the model (5.7) from Section 5.1, we get with this iteration method an explosion time of

$$T_{0.6}^*(-3) \approx 0.1752218 =: T_{\text{graph}}. \tag{5.27}$$

Now we want to compare this to the results that come from Algorithm 5.7. Therefore, we use the computations in Section 6.4. First we try Algorithm 5.7 in order to achieve the explosion time automatically and compare it to (5.27). We get an approximation of

$$T_{0.6}^*(-3) \approx 0.1747451 =: T_{\text{num}}. \tag{5.28}$$

As we can see we have an error of

$$|T_{\text{graph}} - T_{\text{num}}| = 0.0004767, \tag{5.29}$$

which is quite good. Now we want to show, that despite the fact, that the requirements of [RO96] are not quite fulfilled we get, at least numerically, the asymptotic representation (5.23). The implementation therefor can be found in Section 6.4 and results are given in Figure 5.2. As next let us show that the asymptotics (5.24) holds for our example. It is equivalent to

$$a_n (T_\alpha^*)^{\alpha(n+1)} n^{1-\alpha} \frac{\Gamma(\alpha^2)}{\alpha^\alpha \Gamma(2\alpha)} \to 1, \quad n \to \infty, \tag{5.30}$$

which is confirmed by Figure 5.3, whereto the source code can be found in Section 6.4. The last thing in this section which remains to do is to show a reason why we can profit from Algorithm 5.8. A good example is for the model, we used so far, with $u = 2.851852$. The upper bound from Theorem 4.12 is 79842.92 which is significantly worse than the bound of Algorithm 5.7, which is 0.2436943 after the computations in Section 6.4.

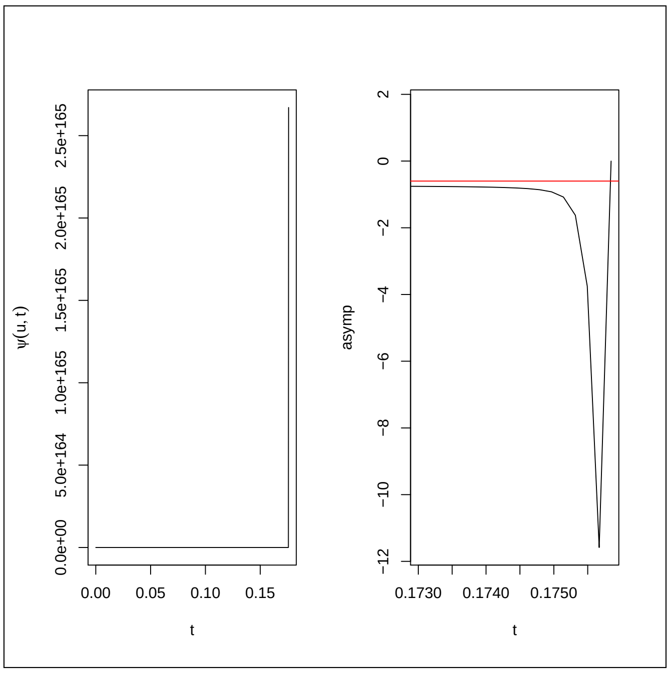

Figure 5.2.: Asymptotics (5.23)

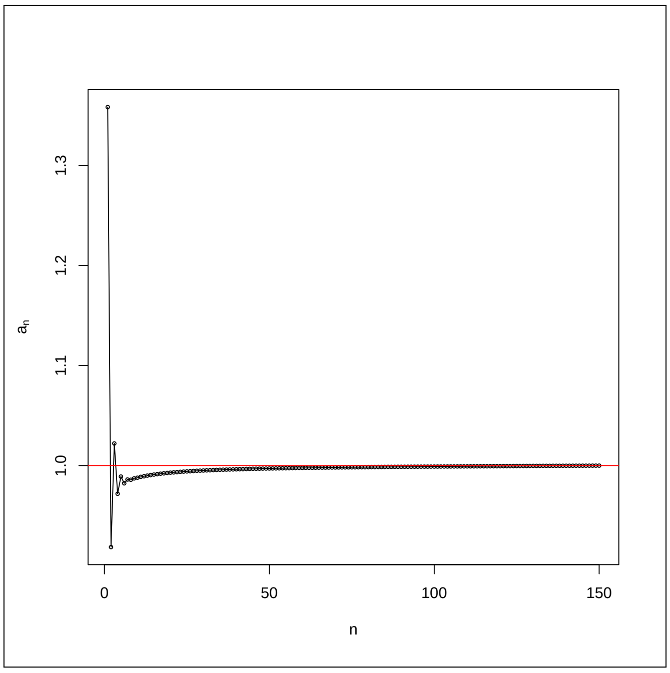

Figure 5.3.: Asymptotics (5.24) for a_n of the power series (5.9)

5.3. Approximation of the Riccati-solution via polylogarithm

The supervisor of this work, Stefan Gerhold, suggested an approximation of $f(u,t)$, the solution of (4.14), via an asymptotic representation with a polylogarithm for $u \in \mathbb{R}$. Write

$$f(u,t) = \sum_{n=1}^{\infty} a_n(u) t^{\alpha n}, \tag{5.31}$$

as in Section 5.2. Define

$$b_n := \frac{\alpha^{\alpha}\Gamma(2\alpha)}{\Gamma(\alpha)^2} n^{\alpha-1} (T_{\alpha}^{*})^{-\alpha(n+1)} \sim a_n, \quad n \to \infty. \tag{5.32}$$

Fix $N \in \mathbb{N}$ and write

$$f(u,t) = \sum_{n=1}^{\infty} b_n t^{\alpha n} + \sum_{n=1}^{N-1} (a_n - b_n) t^{\alpha n} + E_N,$$

where

$$E_N := \sum_{n=N}^{\infty} (a_n - b_n) t^{\alpha n}. \tag{5.33}$$

If E_N is small, we have the approximation

$$\begin{aligned} f(u,t) &\approx \sum_{n=1}^{\infty} b_n t^{\alpha n} + \sum_{n=1}^{N-1} (a_n - b_n) t^{\alpha n} \\ &= \frac{\alpha^\alpha \Gamma(2\alpha)}{\Gamma(\alpha)^2} (T_\alpha^*)^{-\alpha} \operatorname{Li}_{1-\alpha}\left(\left(\frac{t}{T_\alpha^*}\right)^{\alpha}\right) + \sum_{n=1}^{N-1} (a_n - b_n) t^{\alpha n} \end{aligned} \tag{5.34}$$

where the polylogarithm is defined by

$$\operatorname{Li}_\nu(z) := \sum_{n=1}^{\infty} \frac{z^n}{n^\nu}. \tag{5.35}$$

Heuristic error analysis leads to

$$\begin{aligned} |E_N| &= \left| \sum_{n=N}^{\infty} (T_\alpha^*)^{\alpha n} (a_n - b_n) \left(\frac{t}{T_\alpha^*}\right)^{\alpha n} \right| \\ &\leq \sup_{k \geq N} (T_\alpha^*)^{\alpha k} |a_k - b_k| \sum_{n=N}^{\infty} \left(\frac{t}{T_\alpha^*}\right)^{\alpha n} \\ &= \sup_{k \geq N} (T_\alpha^*)^{\alpha k} |a_k - b_k| \frac{(t/T_\alpha^*)^{\alpha N}}{1 - (t/T_\alpha^*)^\alpha} \\ &\lesssim N^\beta (t/T_\alpha^*)^{\alpha N}, \quad N \to \infty, \end{aligned} \tag{5.36}$$

with $\beta < 0$ such that

$$a_n - b_n \sim (\text{const}) n^\beta (T_\alpha^*)^{-\alpha n}, \quad n \to \infty. \tag{5.37}$$

In Section 6.5 we compute the approximation (5.34) and plot it against the real $f(u,t)$ in Figure 5.4. There we can see that it works out quite well if the explosion time T_α^* is already well-approximated. The constant β in (5.37) can be computed via

$$\frac{\log(|a_n - b_n|) + \alpha n \log(T_\alpha^*)}{\log(n)} \to \beta, \quad n \to \infty. \tag{5.38}$$

For our example (5.7) we get in Section 6.5 that $\beta = -0.4176961$; see Figure 5.5.

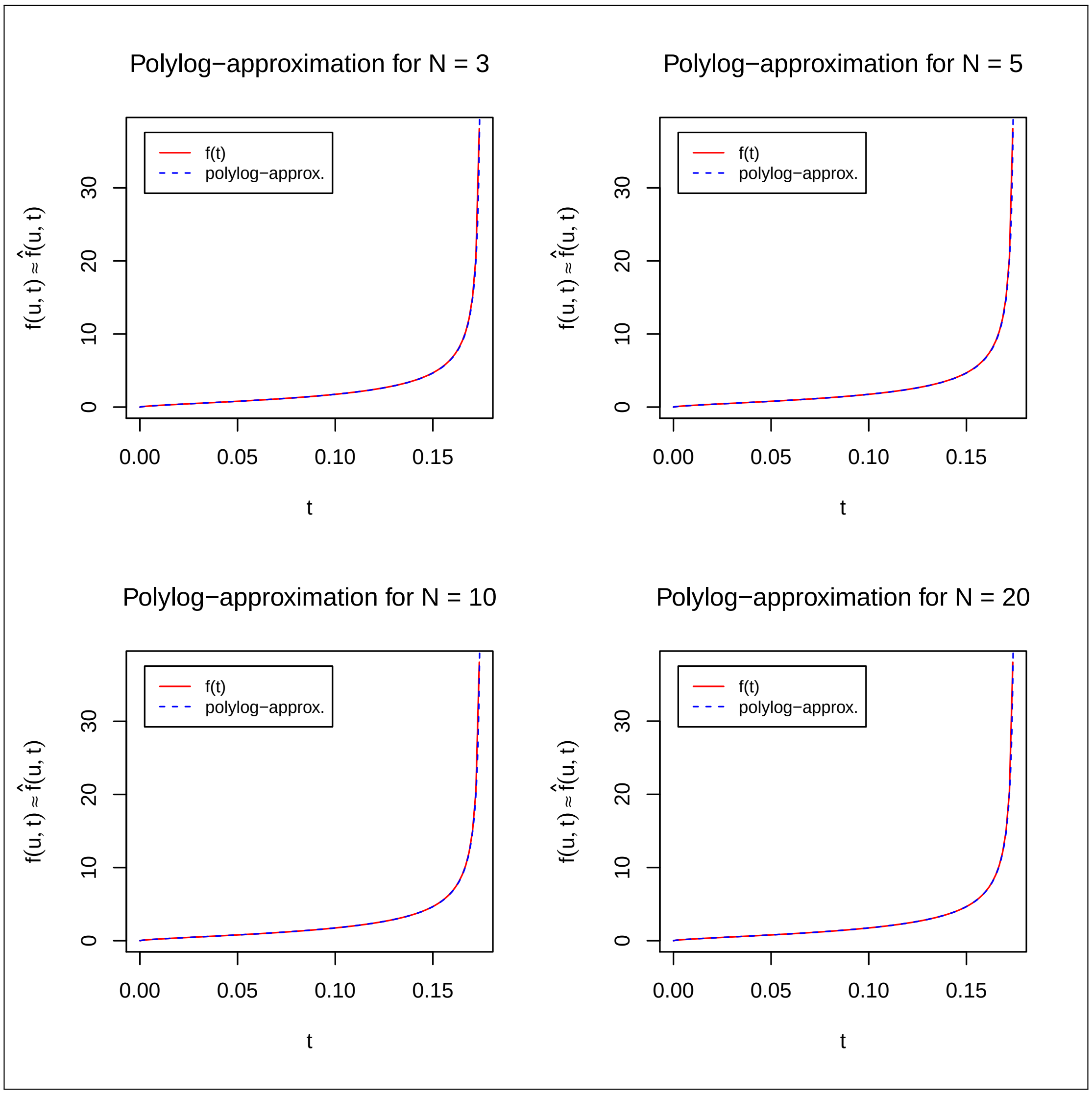

Figure 5.4.: Convergence of the approximation of $f(u, t)$ via (5.34)

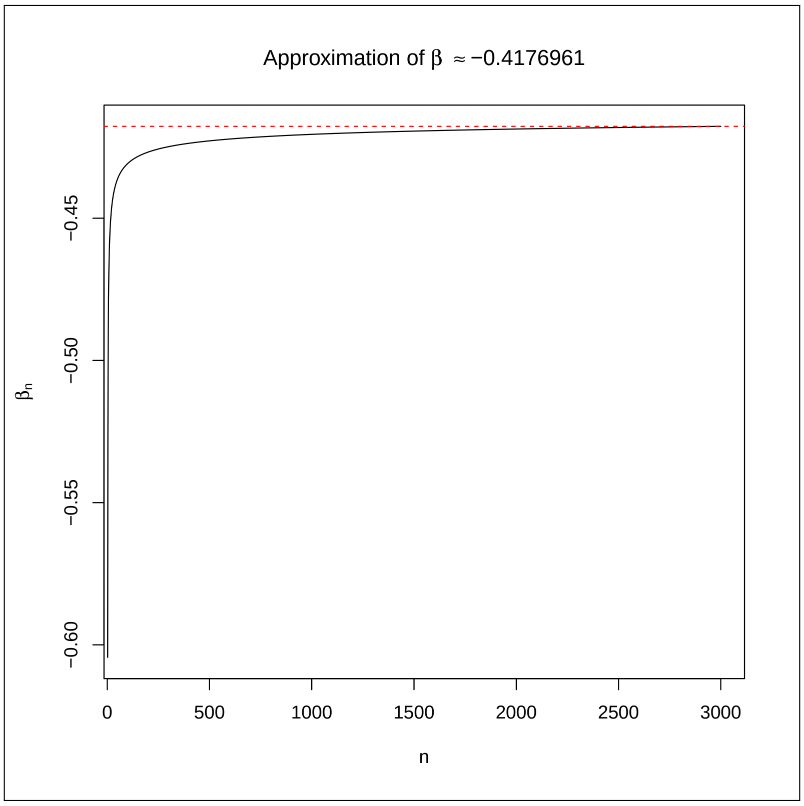

Figure 5.5.: Approximation of β in (5.37)

6. Source Code, Methods & Implementation

In this section I want to show some examples with the algorithms and asymptotics mentioned in Sections 5.1–5.3. Since one aim of this thesis is that a master's degree student can easily follow the topics discussed in here, I am going to give detailed insight into my implementation and the issues I dealt with programming the algorithms in the R-programming language. Therefore, the coding is not shown in the appendix, but between the description of the implementation to trigger the reader to reproduce the procedure. For all the programs R version 3.2.3 has been used.

6.1. Introduction and auxiliary functions

At first let us start with the implementation of the model in R via an R-`data.frame`, since this is quite flexible for our purpose. The model parameters of (4.1)–(4.3) are the contents of the `data.frame` as follows:

$$\begin{aligned} \texttt{v} &:= \overline{v} > 0, \\ \texttt{l} &:= \lambda > 0, \\ \texttt{x} &:= \xi > 0, \\ \texttt{v\_0} &:= V_0 > 0, \\ \texttt{r} &:= \rho \in (-1,1), \\ \texttt{a} &:= \alpha \in (1/2,1). \end{aligned} \tag{6.1}$$

For example we get our model through coding the following:

```r
rh <- data.frame("v" = 0.04, "l" = 0.3, "x" = 1, "v_0" = 1, "r" = -0.7, "a" = 0.6)
```

As later on we will need some basic functions from Chapter 4 to directly take over the notation there into our implementation, I wrote some auxiliary functions. For sure, you could omit them and write their return values directly in the needed place, because most of them are just one-liners.

Auxiliary functions: ans, c_1, c_2, c_3, e_0, e_1, R, G

```
# Description of arguments :
# NA

# Description of return value:
# Returns the last computed value in the current R-environment.

ans <- function(){
  return(.Last.value)
}



# ------------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model
# parameters...data.frame containing the model parameters of a rough Heston model

# Description of return value:
# Returns c_1(u) of R(u,w) = c_1(u) + c_2(u) * w + c_3 * w^2.

# Remark: Actually the variable "parameters" is not needed here but due to
    consistency reasons it is also an argument.

c_1 <- function(u, parameters){
  return(1/2 * u * (u - 1))
}



# ------------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model
# parameters...data.frame containing the model parameters of a rough Heston model

# Description of return value:
# Returns c_2(u) of R(u,w) = c_1(u) + c_2(u) * w + c_3 * w^2.

c_2 <- function(u, parameters){
  return(parameters$r * parameters$x * u - parameters$l) # r = rho, x = xi, l =
      lambda
}



# ------------------------------------------------------------



# Description of arguments :
# parameters...data.frame containing the model parameters of a rough Heston model

# Description of return value:
# Returns c_3 of R(u,w) = c_1(u) + c_2(u) * w + c_3 * w^2.

# Remark: Note that c_3 does not have an argument u since it is not dependend on
    it.

c_3 <- function(parameters){
  return(1/2 * parameters$x * parameters$x) # x = xi
}



# ------------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model
# parameters...data.frame containing the model parameters of a rough Heston model

# Description of return value:
# Returns e_0 of the representation G for the transformed Volterra integral
    equation.

e_0 <- function(u, parameters){
  return(1/2 * c_2(u = u, parameters = parameters))
}



# ------------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model
# parameters...data.frame containing the model parameters of a rough Heston model

# Description of return value:
# Returns e_1 of the representation G for the transformed Volterra integral
    equation.

e_1 <- function(u, parameters){
  return(e_0(u = u, parameters = parameters)^2 - c_3(parameters = parameters) * c_
      1(u = u, parameters = parameters))
}



# ------------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model
# parameters...data.frame containing the model parameters of a rough Heston model
# w...some real value; will be replaced by psi later on

# Description of return value:
# Returns R which is part of the integrand for the Volterra integral equation for
   psi.

R <- function(u, w, parameters){
  return(c_1(u = u, parameters = parameters)
         + c_2(u = u, parameters = parameters) * w
         + c_3(parameters = parameters) * w^2)
}



# ------------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model
# parameters...data.frame containing the model parameters of a rough Heston model
# w...some real value; will be replaced by psi later on

# Description of return value:
# Returns G which is part of the integrand for the Volterra integral equation for
   f.

G <- function(u, w, parameters){
  res <- (w + e_0(u = u, parameters = parameters))^2 - e_1(u = u, parameters =
     parameters)
  return(res)
}
```

Second, it will be quite helpful for performing some tests, if we have functions to check, whether our chosen parameters are admissible in the sense that they fulfill the conditions in (6.1), such that we are not risking getting unreasonable results. Furthermore, we then do not need to extra implement a handling for inappropriate values, since we have checked them before using this values in the functions.

Functions for tests and checks: `cases`, `consistency`, `elro16_condition`, `aux_modelcheck`, `modelcheck`

```
# Description of arguments :
# u...moment of rough Heston model
# parameters...data.frame containing the model parameters of a rough Heston model

# Description of return value:
# Returns a data.frame with objects "Case" and "compl".
#   return$Case...String with the case that u satisfies
#   return$compl...indicator if the moment input was a complex data type

cases <- function(u, parameters){
  if(c_1(u = Re(u), parameters = parameters) > 0){
    if(e_0(u = Re(u), parameters = parameters) >= 0){
      return(data.frame("Case" = "Case (A)", "compl" = is.complex(u),
          stringsAsFactors=FALSE)) # There were some troubles with the collocation
           of data.frames for one dimensional u with the default value of
          stringsAsFactors.
    }else{
      if(e_1(u = Re(u), parameters = parameters) < 0){
        return(data.frame("Case" = "Case (B)", "compl" = is.complex(u),
            stringsAsFactors=FALSE))
      }else{
        return(data.frame("Case" = "Case (C)", "compl" = is.complex(u),
            stringsAsFactors=FALSE))
      }
    }
  }else{
    return(data.frame("Case" = "Case (D)", "compl" = is.complex(u),
        stringsAsFactors=FALSE))
  }
}



# ------------------------------------------------------------



# Description of arguments :
# parameters...data.frame containing the model parameters of a rough Heston model

# Description of return value:
# Returns a boolean value to check if the chosen parameters really correspond to
    the restrictions of our model.

consistency <- function(parameters){
  if(parameters$v <= 0){
    return(c(FALSE, "v"))
  }
  if(parameters$l <= 0){
    return(c(FALSE, "l"))
  }
  if(parameters$x <= 0){
    return(c(FALSE, "x"))
  }
  if(parameters$v_0 <= 0){
    return(c(FALSE, "v_0"))
  }
  if(abs(parameters$r) >= 1){
    return(c(FALSE, "r"))
  }
  if(parameters$a <= 1/2){
    return(c(FALSE, "a"))
  }
  if(parameters$a >= 1){
    return(c(FALSE, "a"))
  }
  return(TRUE)
}


```

```

# ------------------------------------------------------------



# Description of arguments :
# parameters...data.frame containing the model parameters of a rough Heston model

# Description of return value:
# Returns a boolean value if the restriction of (-1/sqrt(2), 1/sqrt(2)] of [ElRo16
   ] is fulfilled; else the range is given back.

elro16_condition <- function(parameters){
  if(parameters$r <= - 1 / sqrt(2)){                 # r = rho
    return(c(FALSE, "Lower than - 1 / sqrt(2)"))
  }
  if(parameters$r > 1 / sqrt(2)){
    return(c(FALSE, "Higher than 1 / sqrt(2)"))
  }
  return(TRUE)
}



# ------------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model
# parameters...data.frame containing the model parameters of a rough Heston model

# Description of return value:
# Returns a data.frame with with "Consistency", "ElRo16_Condition", "Cases" using
   the functions above.

# Remark: The "auxiliary"-prefix, because here for one-dimensional argument u.


aux_modelcheck <- function(u, parameters){
  ret_1 <- consistency(parameters = parameters)
  ret_2 <- elro16_condition(parameters = parameters)
  ret_3 <- cases(u = u, parameters = parameters)

  return(data.frame("Consistency" = ret_1, "ElRo16_Condition" = ret_2, "Cases" =
      ret_3))
}



# ------------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model
# parameters...data.frame containing the model parameters of a rough Heston model

# Description of return value:
# Returns a collocation of the checkfunctions above for multi-dimensional u.

modelcheck <- function(u, parameters){
  n_u <- length(u)
  if(n_u < 2){
    aux_modelcheck(u = u, parameters = parameters)
  }else{
    res <- data.frame("u" = u,  "Consistency" = FALSE, "ElRo16_Condition" = FALSE,
         "Cases" = data.frame("Case" = "Case (X)", "compl" = FALSE,
        stringsAsFactors=FALSE)) # values don"t matter, this is only for adjusting
         the correct data.frame
    m_res <- ncol(res)
    for(i in (1:n_u)){
      res[i,(2:m_res)] <- aux_modelcheck(u = u[i], parameters = parameters) # For
          successful collocation we need "stringAsFactors = TRUE" in the data.
          frames for dim(u) = 1!
    }
    return(res)
  }
}
```

As we will need some reasonable starting values for iterations in Section 6.3 later on, we will profit from having the boundaries from Theorem 4.12 for the explosion time available as function.

Boundaries for the explosion time: `aux_max_boundaries`, `max_boundaries`, `lower_bound`, `upper_bound`, `boundaries`

```
# Description of arguments :
# alpha...some real number between 0 and 1.

# Description of return value:
# Returns the maximum of the function, that needs to be maximized for the
    boundaries of the explosion time.

# Remark: alpha in (0, 1) is not checked, since this function is mainly an
    auxiliary function such that only admissible input will be used.

aux_max_boundaries <- function(alpha){
  to_max <- function(r){
    return((r^alpha - 1)^(1 / alpha) / (r * (r - 1)))
  }
  # "optimize" seemed to work nice for our purpose
  # Maybe one should implement this a little bit more propper.
  res <- optimize(f = to_max, interval = c(1 + 1 / Inf, 10), maximum = TRUE)
  return(res$objective)
}



# ------------------------------------------------------------


```

```r

# Description of arguments :
# alpha...some real number between 0 and 1.

# Description of return value:
# Returns a data.frame containing "alpha" and "f(r_max)" to realize "aux_max_
    boundaries" for vector-valued alpha.
#   return$alpha...input to see the corresponding output correctli
#   return$f(r_max)...maximum of the function, that needs to be maximized for the
    boundaries of the explosion time

# Remark: alpha in (0, 1) is not checked, since this function is mainly an
    auxiliary function such that only admissible input will be used.

max_boundaries <- function(alpha){
  n_alpha <- length(alpha)
  if(n_alpha < 2){
    aux_max_boundaries(alpha = alpha)
  }else{
    res <- vector("list", length = n_alpha)
    for(j in (1:n_alpha)){
      res[[j]] <- aux_max_boundaries(alpha = alpha[j])
    }
    res <- as.data.frame(cbind(alpha, res))
    colnames(res) <- c("alpha", "f(r_max)")
        return(res)
  }
}



# ------------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model
# parameters...data.frame containing the model parameters of a rough Heston model

# Description of return value:
# Returns the lower bound for the explosion time.

lower_bound <- function(u, parameters){
  u <- Re(u)
  case <- cases(u = u, parameters = parameters)$Case
  if(case == "Case (A)"){
    a_u <- 0
  }else if(case == "Case (B)"){
    a_u <- -e_0(u = u, parameters = parameters)
  }else{
    return (0) # stop("ATTENTION: We are in Case (C) or (D)!")
  }

  integrand <- function(w){
    (w / G(u = u, w = w, parameters = parameters))^(1 / parameters$a) / w
  }
  fac_3 <- integrate(f = integrand, lower = a_u, upper = Inf)

  fac_1 <- gamma(1 + parameters$a)^(1 / parameters$a)
  fac_2 <- max_boundaries(alpha = parameters$a)
  fac_3 <- fac_3$value # we need to assign this to just get the value

  res <- fac_1 * fac_2 * fac_3
  return(res)
}



# ------------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model
# parameters...data.frame containing the model parameters of a rough Heston model

# Description of return value:
# Returns the upper bound for the explosion time.

upper_bound <- function(u, parameters){
  u <- Re(u)
  case <- cases(u = u, parameters = parameters)$Case
  if(case == "Case (A)"){
    a_u <- 0
    fac_3 <- 0
  }else if(case == "Case (B)"){
    a_u <- -e_0(u = u, parameters = parameters)
    fac_3 <- parameters$a * (e_0(u = u, parameters = parameters) / e_1(u = u,
        parameters))^(1 / parameters$a) # Analytical solution of int_0^{-e_0}
  }else{
    return(Inf) # stop("ATTENTION: We are in Case (C) or (D)!")
  }

  integrand <- function(w){
    (w / G(u = u, w = w, parameters = parameters))^(1 / parameters$a) / w
  }
  fac_3 <- fac_3 + integrate(f = integrand, lower = a_u, upper = Inf)$value

  fac_1 <- 4 * gamma(1 + parameters$a)^(1 / parameters$a)

  res <- fac_1 * fac_3
  return(res)
}



# ------------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model
# parameters...data.frame containing the model parameters of a rough Heston model

# Description of return value:
# Returns the boundaries of the explosion time.
```

```r

boundaries <- function(u, parameters){
  n_u <- length(u)
  u_re <- Re(u)

  res <- data.frame("u" = u, "T_low" = 0, "T_up" = 0, "Complex" = is.complex(u))

  for(j in (1:n_u)){
    res$T_low[j] <- lower_bound(u = u_re[j], parameters = parameters)
    res$T_up[j] <- upper_bound(u = u_re[j], parameters = parameters)
  }

  return(res)
}
```

6.2. Implementation of numerical scheme of Section 5.1

Note that the relevant equations for our algorithm, (5.2) and (5.4)–(5.6), are generally quite easy to implement at first glance, but with an intelligent code we can save a lot of computation time. In our case we can omit some inner loops and replace them by smart vector arithmetics to use the power of R. In order to achieve this we need to simplify the given algorithm Algorithm 5.1. At this point I want to thank Omar El Euch who kindly provided his Python code while my first implementations were slow and did not work out well. I translated it into R and in the following I will shortly describe the implementation by discussing the used variables, such that it can be reproduced, modified and understood easily (which is actually one aim of this section):

avect Note that the inner part of (5.2), i.e. the $a_{j,k+1}$ for $j \in \{1, \ldots, k\}$, is only dependent of the lag $\ell = k - j$. This can be seen as with

$$a(\ell) := \frac{\Delta\alpha}{\Gamma(\alpha+2)}\left((\ell+2)^{\alpha+1} + \ell^{\alpha+1} - 2(\ell+1)^{\alpha+1}\right), \quad \ell \in \mathbb{N} \cup \{0\}, \tag{6.2}$$

we get

$$a_{j,k+1} = a(k-j), \quad 1 \leq j \leq k, \quad k \in \mathbb{N}. \tag{6.3}$$

Now, if $n \in \mathbb{N}$ denotes the number of steps in our algorithm, we can once compute a vector containing $a(\cdot)$ for all lags till $\ell = n - 1$ and then for $k \leq n$ we get

$$\begin{pmatrix} a_{1,k+1} \\ a_{2,k+1} \\ \vdots \\ a_{k-1,k+1} \\ a_{k,k+1} \end{pmatrix} = \begin{pmatrix} a(k-1) \\ a(k-2) \\ \vdots \\ a(1) \\ a(0) \end{pmatrix}. \tag{6.4}$$

Since we do not need to compute the $a_{j,k+1}$ from new on, but just have an index access we save a lot of time. The indexing is easily realized, since for the vector with the $a_{j,k+1}$ we need the last k elements of the vector

$$\texttt{avect} := \begin{pmatrix} a(n-1) \\ a(n-2) \\ \vdots \\ a(1) \\ a(0) \end{pmatrix}, \tag{6.5}$$

hence we get

$$\texttt{avect[(n - k + 1):n]} = \begin{pmatrix} a_{1,k+1} \\ a_{2,k+1} \\ \vdots \\ a_{k-1,k+1} \\ a_{k,k+1} \end{pmatrix}. \tag{6.6}$$

Note that with using fancy indexing conventions in R we can write `avect[-(1:(n - k))]` instead of `avect[(n - k + 1):n]`, but here it seems to me that the code is better readable with the standard indexing.

bvect In the same way as above the $b_{j,k+1}$ of (5.5) are only dependent of the lag too for $j \in \{0, \dots, k\}$. Denote the lag function by

$$b(\ell) := \frac{\Delta^\alpha}{\Gamma(\alpha+1)}((\ell+1)^\alpha - \ell^\alpha), \quad \ell \in \mathbb{N} \cup \{0\}, \tag{6.7}$$

then we have

$$b_{j,k+1} = b(k-j), \quad 0 \le j \le k, \quad k \in \mathbb{N}. \tag{6.8}$$

As above we produce the vector

$$\texttt{bvect} := \begin{pmatrix} b(n-1) \\ b(n-2) \\ \vdots \\ b(1) \\ b(0) \end{pmatrix}, \tag{6.9}$$

where we access the $b_{j,k+1}$ via

$$\texttt{bvect[(n - k):n]} = \begin{pmatrix} b_{0,k+1} \\ b_{1,k+1} \\ \vdots \\ b_{k-1,k+1} \\ b_{k,k+1} \end{pmatrix}. \tag{6.10}$$

Note that different to `avect` in `bvect` we need the last $k+1$ elements, so we get one entry more.

res The object `res` is a matrix of dimension $(n+1) \times \dim(u)$, where n is the number of steps for the iteration. So we have for $\dim(u) = 1$

$$\texttt{res} = \begin{pmatrix} \widehat{\psi}(u, t_0) \\ \widehat{\psi}(u, t_1) \\ \vdots \\ \widehat{\psi}(u, t_n) \end{pmatrix}. \tag{6.11}$$

We used the vector- resp. matrix-arithmetics of R to implement everything pointwise for a given vector u.

res_aux This is an auxiliary matrix of dimension $(n+1) \times \dim(u)$ with n is the number of steps for the iteration. It contains the factors $R(u, \widehat{\psi}(u, t_k))$ which are needed for the pre-estimates in (5.4). For $\dim(u) = 1$ we have

$$\texttt{res\_aux} := \begin{pmatrix} R(u, \widehat{\psi}(u, t_0)) \\ R(u, \widehat{\psi}(u, t_1)) \\ \vdots \\ R(u, \widehat{\psi}(u, t_n)) \end{pmatrix}, \tag{6.12}$$

where actually the last value can be omitted, since it would only be needed for $\widehat{\psi}(u, t_{n+1})$. Due to simplifications it is computed, since it is the last value of a loop, and we would have to write an extra code for the last index of `res`. For a given vector u the implementation is pointwise.

resp This is an auxiliary vector of length $\dim(u)$ which is renewed in every iteration step. For $\dim(u) = 1$ we have for `k = 0` that

$$\texttt{resp} = \frac{\Delta^\alpha}{\Gamma(\alpha+1)} R(u, \widehat{\psi}(u, t_0)), \tag{6.13}$$

and for `k in (1:(n - 1))` we use

$$\begin{aligned}
\texttt{resp} &= \texttt{res\_aux[1:(k + 1)]} \cdot \begin{pmatrix} \texttt{bvect[n - k]} \\ \texttt{bvect[n - k + 1]} \\ \vdots \\ \texttt{bvect[n]} \end{pmatrix} \\
&\overset{(6.9) \text{ and } (6.12)}{=} (R(u, \widehat{\psi}(u, t_0)), \dots, R(u, \widehat{\psi}(u, t_k)) \cdot \begin{pmatrix} b(k) \\ b(k-1) \\ \vdots \\ b(0) \end{pmatrix} \qquad (6.14) \\
&= \sum_{j=0}^{k} b(k-j) R(u, \widehat{\psi}(u, t_j)) \\
&\overset{(6.8)}{=} \sum_{j=0}^{k} b_{j,k+1} R(u, \widehat{\psi}(u, t_k)) \\
&= \widehat{\psi}^P(u, t_{k+1}).
\end{aligned}$$

For $\dim(u) > 1$, the implementation above is pointwise in the vector u.

`res_aux_p` This is an auxiliary vector of length $\dim(u)$. For $\dim(u) = 1$ we have

$$\texttt{res\_aux\_p} := R(u, w = \texttt{resp}) \overset{(6.14)}{=} R(u, \widehat{\psi}^P(u, t_{k+1})). \qquad (6.15)$$

This is a factor of the last summand in (5.6). For the last summand we then have

$$a_{k+1,k+1} R(u, \widehat{\psi}^P(u, t_{k+1})) = \frac{\Delta^\alpha}{\Gamma(\alpha + 2)} \cdot \texttt{res\_aux}. \qquad (6.16)$$

Hence, `resp` and `res_aux_p` are just the auxiliary vectors for computing the last summand in the recursion (5.6).

Since all the relevant objects of the function are discussed, let us view the code. Here, the function `aux_psi` realizes the recursion (5.6) for u with $\dim(u) \geq 1$. Additionally, the function `psi` also handles a maturity vector T with $\dim(T) \geq 1$. Since a vectorized implementation for higher dimensional T is not going to work out for this approach, we just have a loop which runs `aux_psi` for every entry of T. Additionally, we have `aux_psi_mpfr` and `psi_mpfr` which do the same as `aux_psi` and `psi`, except the fact, that with the objects of type `mpfr` we can handle large numbers which would cause overflow errors. Despite the fact, that this numbers actually cannot be plotted (they need to be converted to `Inf`-values, e.g. via `base::as.numeric` or `Rmpfr::asNumeric`), it will sometimes be useful to have the possibility to deal with "large numbers", since some computations need a division by large numbers, and overflows in the denominator result in explosion time zero.

Numerics – fractional addams method: aux_psi, psi, aux_psi_mpfr, psi_mpfr

```r
# Description of arguments :
# u...moment of rough Heston model (dim(u) > 1)
# parameters...data.frame containing the model parameters of a rough Heston model
# T...psi(u, t) is computed from 0 to T
# n...grid size with n * dt = T

# Description of return value:
# Returns the data.frame containing "grid" and "res", resp. "res.1", "res.2", ...
    "res.n" for dim(u) = n.
#   return$grid...the grid which is determined by T an n
#   return$res...psi(u, t) for r in {0, dt, 2dt, ..., ndt = T}

# Remark: The grid is given back too, since plot manipulation gets a little bit
    easier and more transparent if you store the values and corresponding grid
    together.

aux_psi <- function(u , parameters, T, n){
  dt <- T / n
  n_u <- length(u)
  gamma_1 <- gamma(parameters$a + 1)
  gamma_2 <- gamma(parameters$a + 2)
  res <- matrix(data = 0, nrow = (n + 1), ncol = n_u)*1i # Can also be applied for
      u complex, res = psi(u, t) for u fix
  res_aux <- matrix(data = 0, nrow = (n + 1), ncol = n_u)*1i
  resp <- rep(x = 0, times = n_u)*1i
  res_aux_p <- rep(x = 0, times = n_u)*1i

  res_aux[1,] <- c_1(u = u, parameters = parameters)
  resp <- dt^parameters$a * res_aux[1,] / gamma_1
  res_aux_p <- R(u = u, w = resp, parameters = parameters)
  res[2,] <- dt^parameters$a * parameters$a * res_aux[1,] / gamma_2 + dt^
      parameters$a * res_aux_p / gamma_2
  res_aux[2,] <- R(u = u, w = res[2,], parameters = parameters)

  nn <- seq(from = n - 1, to = 0, by = -1)
  bvect <- dt^parameters$a * ((nn + 1)^parameters$a - nn^parameters$a) / gamma_1
  avect <- dt^parameters$a * ((nn + 2)^(parameters$a + 1) + nn^(parameters$a + 1)
      - 2 * (nn + 1)^(parameters$a + 1)) / gamma_2

  k <- 1 # k = 1 appart, since something didn't work out with the matrix-
      multiplication %*%.
  resp <- t(t(res_aux[1:(k + 1),]) %*% bvect[(n - k):n]) # Note that "resp" and "
      res_aux_p" are renewed in every step. "res_aux" will be the psi^(u, t_k) for
       k in {0, ..., n} at the end.
  res_aux_p <- R(u = u, w = resp, parameters = parameters)
  res[(k + 2),] <- res_aux[1,] * (k^(parameters$a + 1) - (k - parameters$a) * (k +
       1)^parameters$a) * dt^parameters$a / gamma_2
  res[(k + 2),] <- res[(k + 2),] + res_aux_p * dt^parameters$a / gamma_2
  res[(k + 2),] <- res[(k + 2),] + t(as.matrix(res_aux[2:(k + 1),]) %*% avect[((n
      - k + 1):n)])
  res_aux[(k + 2),] <- R(u = u, w = res[(k + 2),], parameters = parameters)

  for(k in (2:(n - 1))){
    resp <- t(t(res_aux[1:(k + 1),]) %*% bvect[(n - k):n]) # Note that "resp" and
        "res_aux_p" are renewed in every step. "res_aux" will be the psi^(u, t_k)
        for k in {0, ..., n} at the end.
    res_aux_p <- R(u = u, w = resp, parameters = parameters)
    res[(k + 2),] <- res_aux[1,] * (k^(parameters$a + 1) - (k - parameters$a) * (k
        + 1)^parameters$a) * dt^parameters$a / gamma_2
    res[(k + 2),] <- res[(k + 2),] + res_aux_p * dt^parameters$a / gamma_2
    res[(k + 2),] <- res[(k + 2),] + t(t(res_aux[2:(k + 1),]) %*% avect[((n - k +
        1):n)])
    res_aux[(k + 2),] <- R(u = u, w = res[(k + 2),], parameters = parameters)
  }

  grid <- seq(from = 0, to = T, by = dt)
  res[!is.finite(Re(res))] <- Inf # Probably quite improper, but to have an
      appropriate representation for further analysis it will be good to avoid
      NaNs, so we hope (and in fact it has to be in Case(A) and (B)) that the NaNs
       are errors where Infs would be the result.)
  res[!is.finite(Im(res))] <- 1i * Inf

  # Output "res" equals to a sequence psi(u,t_0), psi(u,t_1), ..., psi(u, t_n)
      with t_0 = 0, t_n = T
  return(data.frame("grid" = grid, "res" = res))
}



# ------------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model (dim(u) > 1)
# parameters...data.frame containing the model parameters of a rough Heston model
# T...psi(u, t) is computed from 0 to T (dim(T) > 1)
# n...grid size with n * dt = T

# Description of return value:
# Returns a list containing for each T[j] the result of aux_psi.

psi <- function(u, parameters, T, n){
  n_T <- length(T)
  if(n_T < 2){
    aux_psi(u = u, parameters = parameters, T = T, n = n)
  }else{
    res <- list()
    for(i in (1:n_T)){
      tmp <- aux_psi(u = u, parameters = parameters, T = T[i], n = n)
      res[[i]] <- data.frame("T" = T[i], res = tmp$res, grid = tmp$grid)
    }

    return(res)
  }
}



# ------------------------------------------------------------



# Description of arguments :
```

```r
# u...moment of rough Heston model (dim(u) = 1)
# parameters...data.frame containing the model parameters of a rough Heston model
# T...psi(u, t) is computed from 0 to T (dim(T) = 1)
# n...grid size with n * dt = T

# Description of return value:
# Returns the data.frame containing "grid" and "res", resp. "res.1", "res.2", ...
    "res.n" for dim(u) = n.
#   return$grid...the grid which is determined by T an n
#   return$res...psi(u, t) for r in {0, dt, 2dt, ..., ndt = T}

# Remark: Since we are dealing with explosions, it will sometimes be interesting
    to compute numbers that would normally cause an overflow.

library(Rmpfr)

aux_psi_mpfr <- function(u , parameters, T, n){
  dt <- T / n
  gamma_1 <- gamma(parameters$a + 1)
  gamma_2 <- gamma(parameters$a + 2)
  res <- mpfr(rep(x = 0, times = n + 1),53) # 53 = double precision for mpfr
  res_aux <- mpfr(rep(x = 0, times = n + 1),53)
  resp <- mpfr(rep(x = 0, times = 1),53)
  res_aux_p <- mpfr(rep(x = 0, times = 1),53)

  res_aux[1] <- c_1(u = u, parameters = parameters)
  resp <- dt^parameters$a * res_aux[1] / gamma_1
  res_aux_p <- R(u = u, w = resp, parameters = parameters)
  res[2] <- dt^parameters$a * parameters$a * res_aux[1] / gamma_2 + dt^parameters$
      a * res_aux_p / gamma_2
  res_aux[2] <- R(u = u, w = res[2], parameters = parameters)

  nn <- seq(from = n - 1, to = 0, by = -1)
  bvect <- dt^parameters$a * ((nn + 1)^parameters$a - nn^parameters$a) / gamma_1
  avect <- dt^parameters$a * ((nn + 2)^(parameters$a + 1) + nn^(parameters$a + 1)
      - 2 * (nn + 1)^(parameters$a + 1)) / gamma_2

  for(k in (1:(n - 1))){ # The problem for k = 1 in "aux_psi" does not occur in
      the case dim(u) = 1.
    resp <- t(res_aux[1:(k + 1)]) %*% bvect[(n - k):length(bvect)]
    res_aux_p <- R(u = u, w = resp, parameters = parameters)
    res[(k + 2)] <- res_aux[1] * (k^(parameters$a + 1) - (k - parameters$a) * (k +
        1)^parameters$a) * dt^parameters$a / gamma_2
    res[(k + 2)] <- res[(k + 2)] + res_aux_p * dt^parameters$a / gamma_2
    res[(k + 2)] <- res[(k + 2)] + t(res_aux[2:(k + 1)]) %*% avect[(n - k + 1):
        length(avect)]
    res_aux[(k + 2)] <- R(u = u, w = res[(k + 2)], parameters = parameters)
  }

  grid <- seq(from = 0, to = T, length.out = n + 1)
  return(data.frame("grid" = grid, "res" = asNumeric(res)))
}



# ------------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model (dim(u) = 1)
# parameters...data.frame containing the model parameters of a rough Heston model
# T...psi(u, t) is computed from 0 to T (dim(T) > 1)
# n...grid size with n * dt = T

# Description of return value:
# Returns a list containing for each T[j] the result of aux_psi_mpfr.

psi_mpfr <- function(u, parameters, T, n){
  n_T <- length(T)
  if(n_T < 2){
    aux_psi_mpfr(u = u, parameters = parameters, T = T, n = n)
  }else{
    res <- list()
    for(i in (1:n_T)){
      tmp <- aux_psi_mpfr(u = u, parameters = parameters, T = T[i], n = n)
      res[[i]] <- data.frame("T" = T[i], res = tmp$res, grid = tmp$grid)
    }

    return(res)
  }
}
```

As an example of the numerics we plotted a picture for a vector u satisfying each case of (A)–(D); see Figure 5.1. We choose

$$u = (-3, 2.9, 1.5, 0.5), \tag{6.17}$$

and take the following model:

```
rh <- data.frame("v" = 0.04, "l" = 0.3, "x" = 1, "v_0" = 1, "r" = -0.7, "a" = 0.6)
```

Note that one should have loaded all the R-functions which were introduced above at this point. We made sure that the components of u are satisfying the proposed cases:

```
# Cases (A), (B), (C), (D)
moment <- data.frame(A = -3, B = 2.9, C = 1.5, D = 0.5)
modelcheck(u = as.numeric(moment), parameters = rh)
```

```
Output:
> # Cases (A), (B), (C), (D)
> moment <- data.frame(A = -3, B = 2.9, C = 1.5, D = 0.5)
> modelcheck(u = as.numeric(moment), parameters = rh)
```

```
      u Consistency ElRo16_Condition Cases.Case Cases.compl
 1 -3.0        TRUE             TRUE   Case (A)       FALSE
 2  2.9        TRUE             TRUE   Case (B)       FALSE
 3  1.5        TRUE             TRUE   Case (C)       FALSE
 4  0.5        TRUE             TRUE   Case (D)       FALSE
```

Then we run the code, where the output can be seen in Figure 5.1.

```r
steps <- 1000
par(mfrow = c(1,1))
par(mfrow = c(2,2))

# Case(A)
maturity <- 0.18
estimate <- psi(u = moment$A, parameters = rh, T = maturity, n = steps)
x <- estimate$grid
y <- Re(estimate$res)
plot(x = x, y = y, type = "o", xlim = c(0.17, 0.18), cex = 0.5, xlab = expression(
    t), ylab = expression(psi(u,t)),  main = paste("Rough Heston Model - Case (A)
    \nu = ", toString(moment$A), sep = ""))

# Case (B)
maturity <- 480
estimate <- psi(u = moment$B, parameters = rh, T = maturity, n = steps)
x <- estimate$grid
y <- Re(estimate$res)
plot(x = x, y = y, type = "o", xlim = c(400, 420), cex = 0.5, xlab = expression(t)
    , ylab = expression(psi(u, t)), main = paste("Rough Heston Model - Case (B) \
    nu = ", toString(moment$B), sep = ""))

# Case (C)
maturity <- 1500
estimate <- psi(u = moment$C, parameters = rh, T = maturity, n = steps)
x <- estimate$grid
y <- Re(estimate$res)
plot(x = x, y = y, type = "o", cex = 0.5, xlab = expression(t), ylab = expression(
    psi(u, t)), main = paste("Rough Heston Model - Case (C) \nu = ", toString(
    moment$C), sep = ""))

# Case (D)
maturity <- 1500
estimate <- psi(u = moment$D, parameters = rh, T = maturity, n = steps)
x <- estimate$grid
y <- Re(estimate$res)
plot(x = x, y = y, type = "o", cex = 0.5, xlab = expression(t), ylab = expression(
    psi(u, t)), main = paste("Rough Heston Model - Case (D) \nu = ", toString(
    moment$D), sep = ""))

par(mfrow = c(1,1))
```

6.3. Graphical iteration method

Having loaded all the functions of Sections 6.1 and 6.2 we now develop a graphical scheme with "human decision" to have a guess about the explosion time. Here we use the function `boundaries` from Section 6.1 to determine the starting values for the iteration. The term `iter` in the code is written as a macro (function `defmacro` of `gtools`) that does the iteration for the maturity and plots the results for the graphical decision. There is little hope that this approach leads to a very stable algorithm that could be implemented without "human decision", since some experiments have shown that first, the success sometimes depends strongly on the initial situation and second, how fast the function $\psi(u,t)$ explodes is strongly dependent on the model choice. Nevertheless, if you are aware of some dangers, like numerical errors due to underflows or overflows, or the fact that it strongly depends on the explosion time (which we do not know at the beginning of this) if the number of steps is appropriate, this approach works out quite nicely and leads to reasonable results. After the source code we have the output in the console, which is mainly generated by the macro `iter` to show how big our maximal error is such that we know when we have done a sufficient number of iterations. The vectors `bounds` are assigned manually dependent on the plots in Figures 6.2–6.7. Note that in the subtitle of these plots, the index number of the vector `maturity` is printed. Figure 6.1 shows a plot of $\psi(u,t)$ with the two starting values for the maximal value T of t, which we get from the boundaries of Theorem 4.12.

Script for graphical iteration

```
rh <- data.frame("v" = 0.04, "l" = 0.3, "x" = 1, "v_0" = 1, "r" = -0.7, "a" = 0.6)
moment <- -3
modelcheck(u = moment, parameters = rh)
T_bounds <- boundaries(u = moment, parameters = rh)
T_bounds
maturity <- c(T_bounds$T_low, T_bounds$T_up) # Starting values = boundaries
steps <- 1000

par(mfrow = c(1,1))
par(mfrow = c(1,2))
estimate <- psi(u = moment, parameters = rh, T = maturity, n = steps)
for(j in (1:2)){
  plot(x = estimate[[j]]$grid, y = Re(estimate[[j]]$res), type = "o", cex = 0.5,
      main = bquote(paste(psi(u, t), " for T = ", .(maturity[j]))), ylab =
      expression(psi(u, t)), xlab = expression(t))
}
par(mfrow = c(1,1))
# The plots verify the explosion in between.

# Now we start the iteration
library(gtools) # for defining macro

# ITERATION START
iter <- defmacro(.GlobalEnv, expr={
  n <- 6 # since the plot visualization is done manually, the par(mfrow = [...])
      and n need to match!
  left <- bounds[1]
  right <- bounds[2]
  maturity <- seq(from = maturity[bounds[1]], to = maturity[bounds[2]], length.out
       = n)
  max_error <- maturity[n] - maturity[1]
  res <- psi(u = moment, parameters = rh, T = maturity, n = steps)

  par(mfrow = c(1, 1))
  par(mfrow = c(2, 3))
  for(j in (1:n)){
    df1 <- data.frame(x = res[[j]]$grid, y = Re(res[[j]]$res))
    plot(df1, type = "o", cex = 0.5, xlab = "t", ylab = expression(psi(u, t)),
         main = bquote(atop(paste(psi(u,t), " for T = ", .(round(maturity[j],4)),
             sep = ""), paste("Max.num = " , .(max(na.omit(df1$y))), sep = "" )
                )),
         sub = paste("Nr. ", toString(j), sep = ""))
  }
  par(mfrow = c(1, 1))
  max_error
})
# ITERATION END

maturity
bounds <- c(1,2)
iter()

bounds <- c(2,3)
iter()

bounds <- c(3,4)
iter()

bounds <- c(1,3)
iter()

bounds <- c(3,4)
iter()

bounds <- c(2,5)
iter()

T_exp_graph <- mean(maturity[1:3])
T_exp_graph
```

```
Output:
> rh <- data.frame('v' = 0.04, 'l' = 0.3, 'x' = 1, 'v_0' = 1, 'r' = -0.7, 'a' = 0.6)
> moment <- -3
> modelcheck(u = moment, parameters = rh)
  Consistency ElRo16_Condition Cases.Case Cases.compl
```

```
1        TRUE             TRUE   Case (A)       FALSE
> T_bounds <- boundaries(u = moment, parameters = rh)
> T_bounds
   u     T_low      T_up Complex
1 -3 0.0229439 0.5527463   FALSE
--------------------------------------------------------------------
####################################################################
--------------------------------------------------------------------
> maturity
[1] 0.0229439 0.5527463
> bounds <- c(1,2)
> iter()
[1] 0.5298024
>
> bounds <- c(2,3)
> iter()
[1] 0.1059605
>
> bounds <- c(3,4)
> iter()
[1] 0.02119209
>
> bounds <- c(1,3)
> iter()
[1] 0.008476838
>
> bounds <- c(3,4)
> iter()
[1] 0.001695368
>
> bounds <- c(2,5)
> iter()
[1] 0.001017221
>
> T_exp_graph <- mean(maturity[1:3])
> T_exp_graph
[1] 0.1752218
```

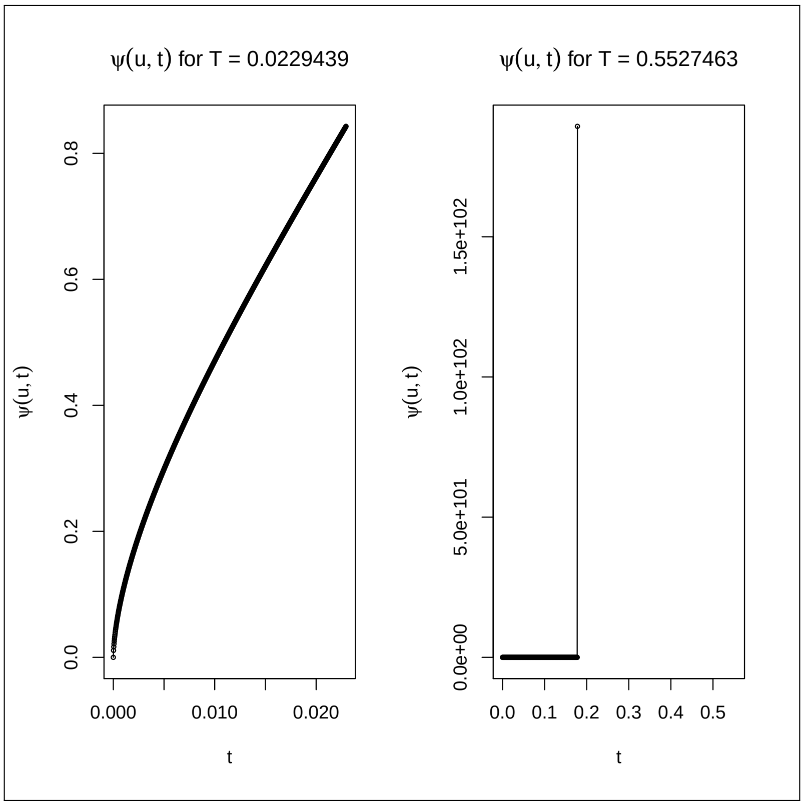

Figure 6.1.: Starting values for t in graphical iteration

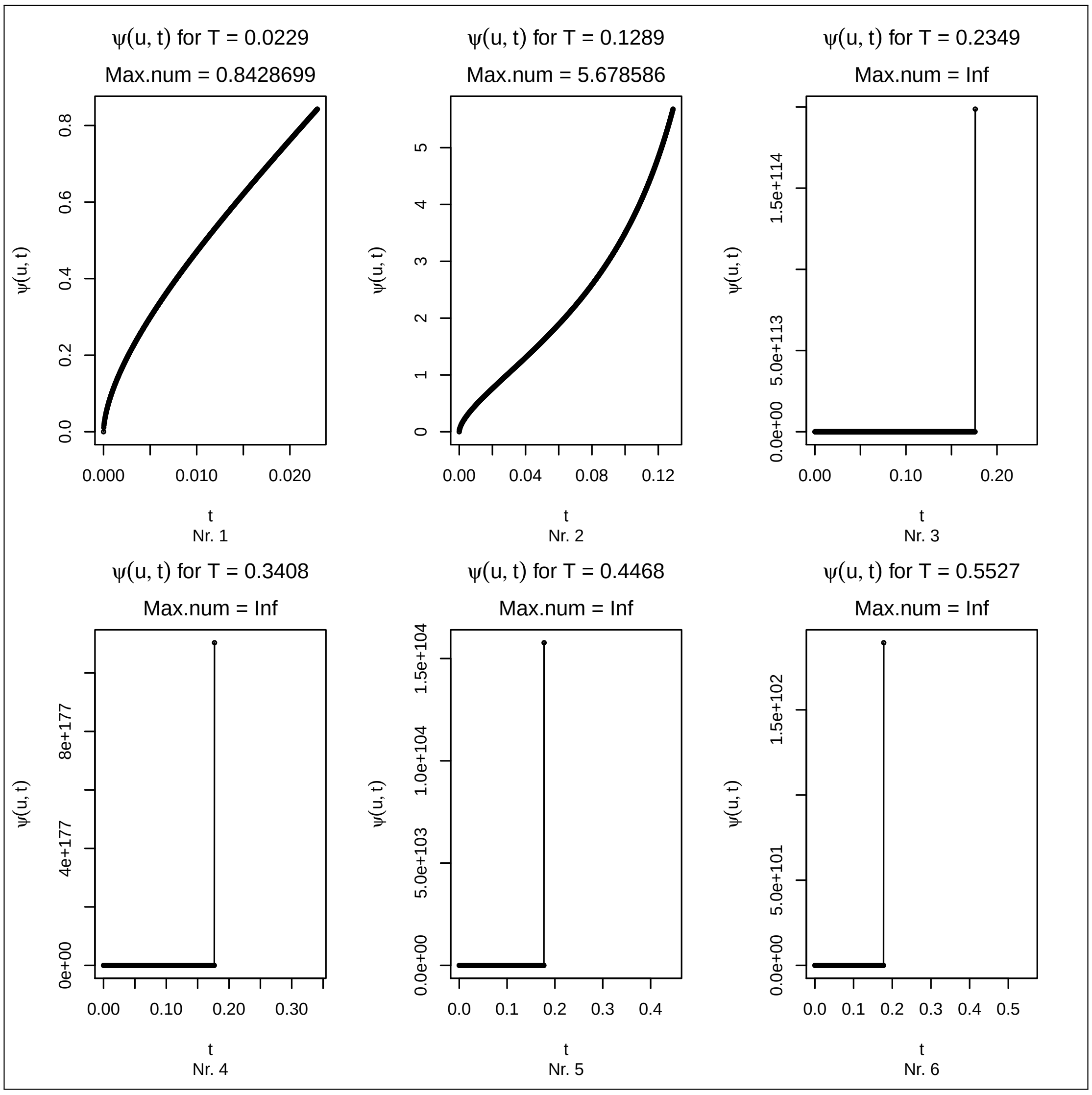

Figure 6.2.: Iteration #1, maximal error = 0.5298024

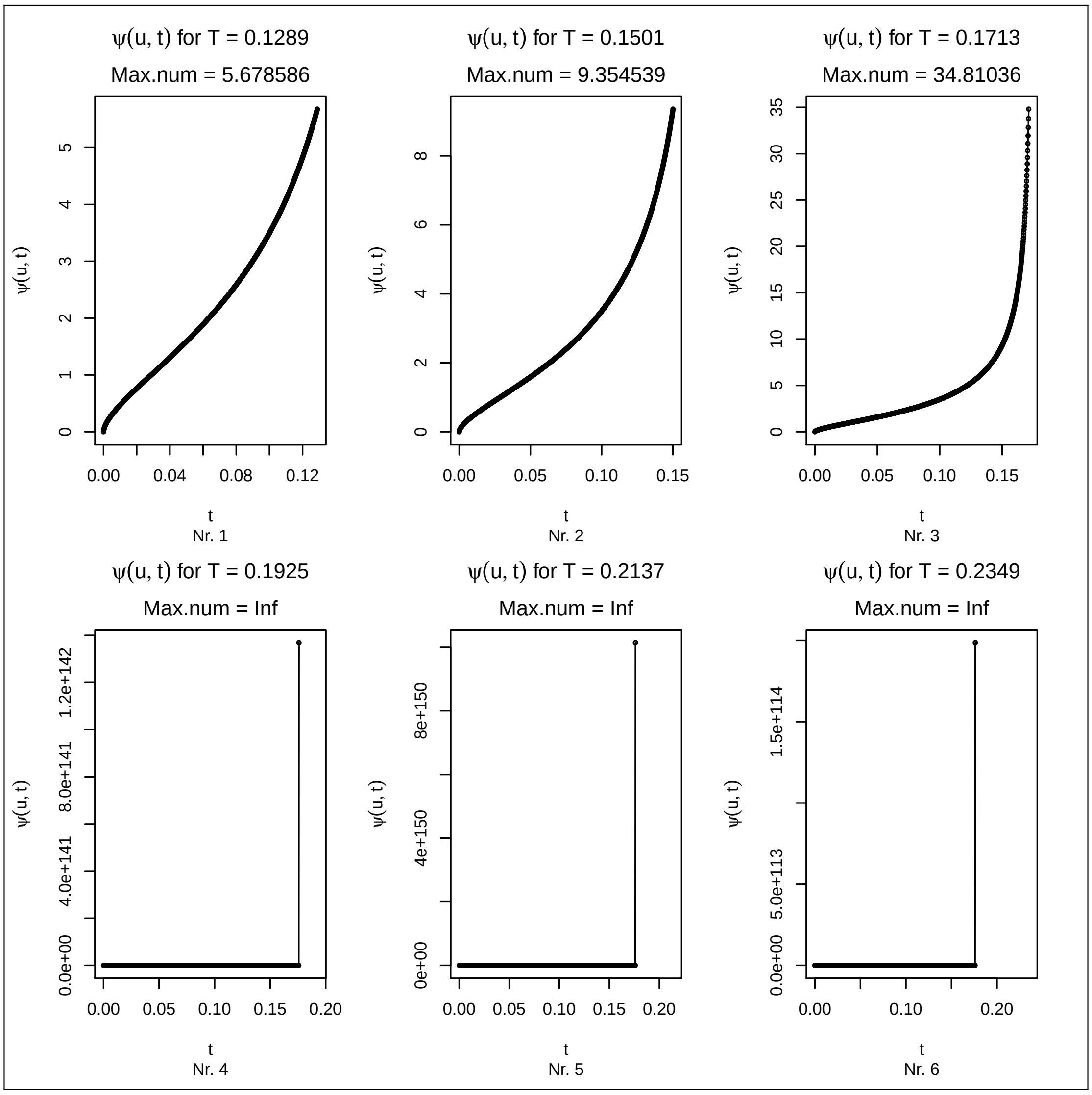

Figure 6.3.: Iteration #2, maximal error = 0.1059605

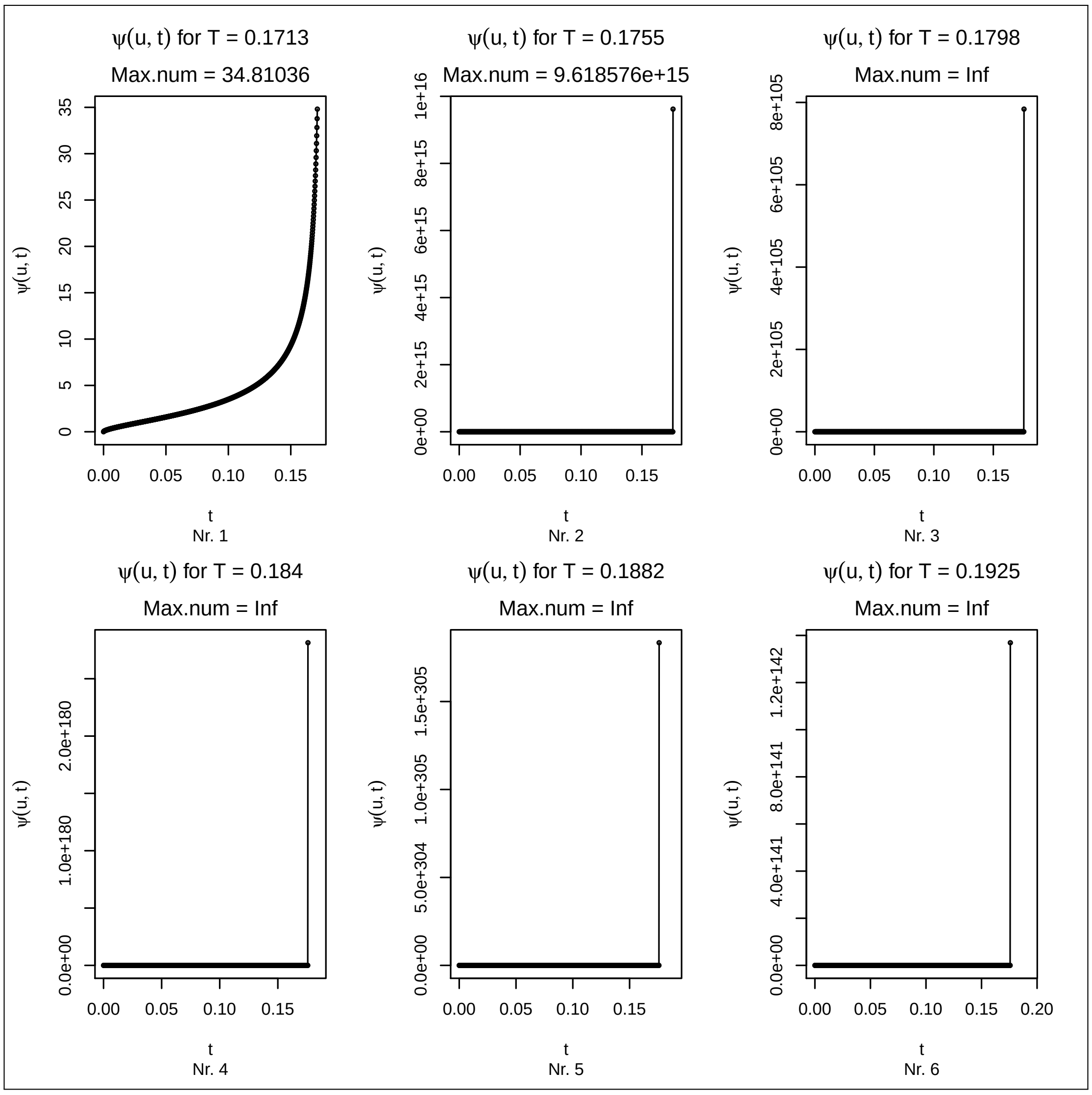

Figure 6.4.: Iteration #3, maximal error = 0.02119209

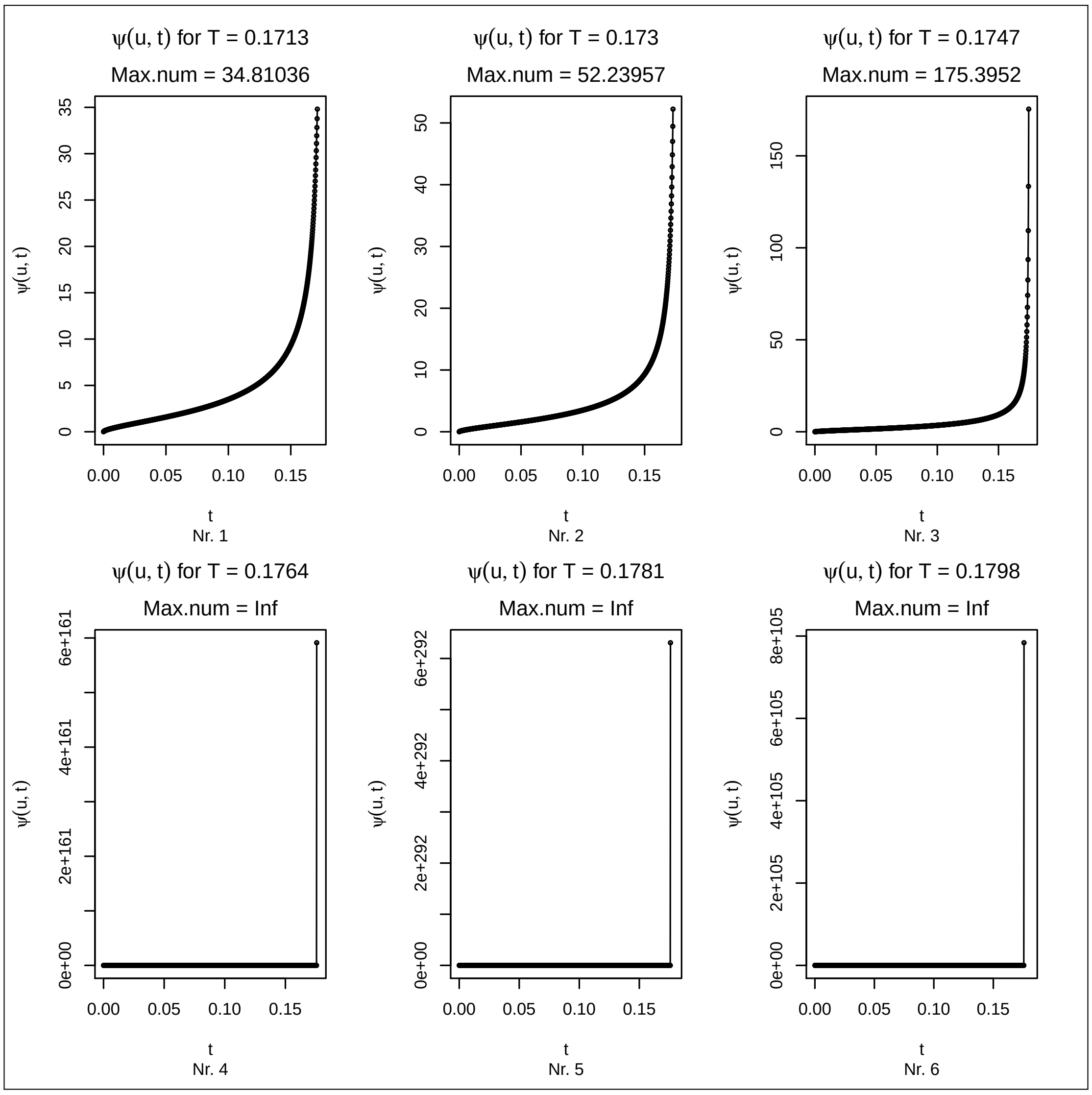

Figure 6.5.: Iteration #4, maximal error = 0.008476838

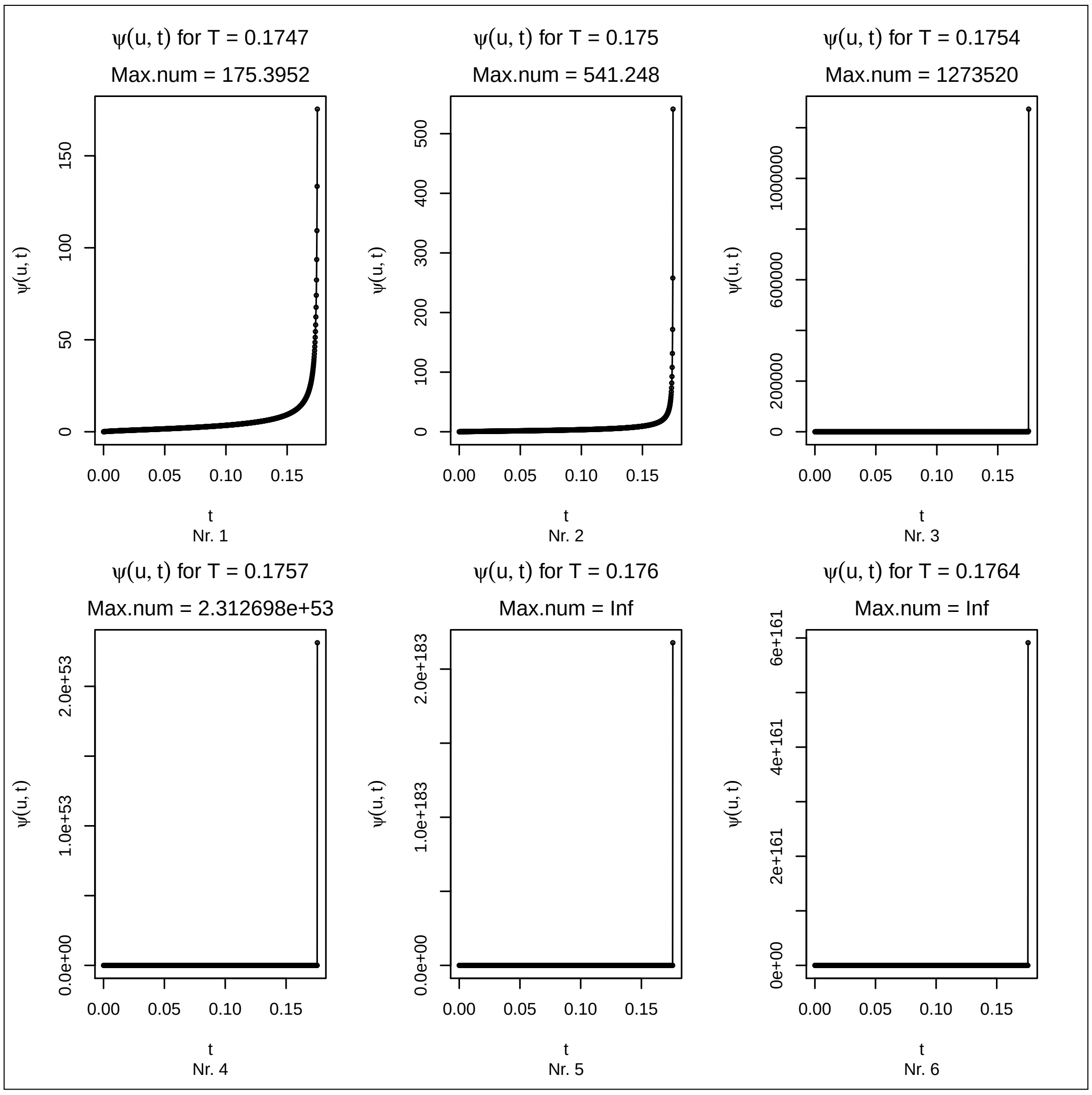

Figure 6.6.: Iteration #5, maximal error = 0.001695368

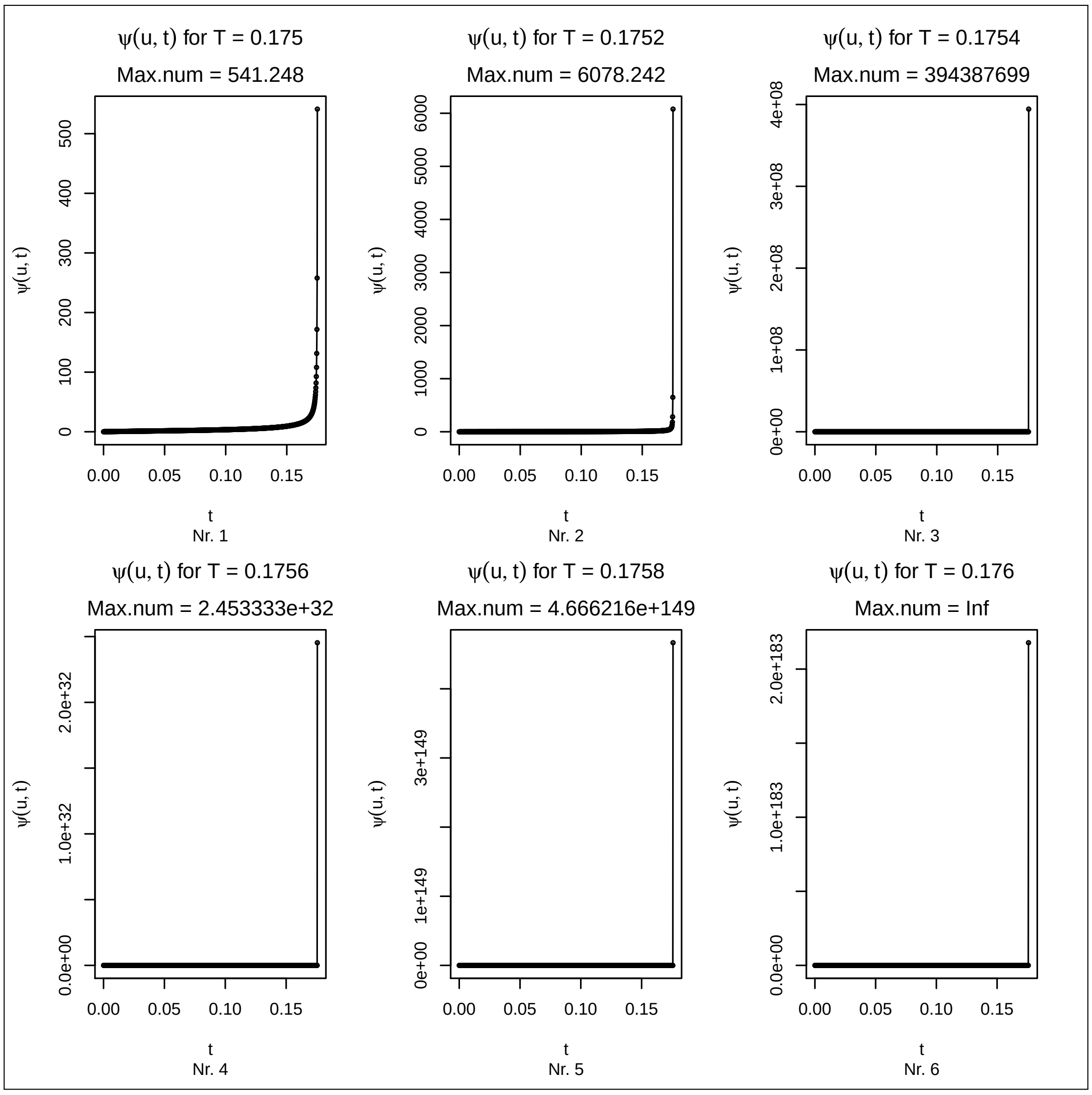

Figure 6.7.: Iteration #6, maximal error = 0.001017221

6.4. Computing explosion time via Algorithm 5.7 and lower bound via Algorithm 5.8

Explosion time via Algorithm 5.7 and bound via Algorithm 5.8: `explosion_time`, `explosion_time_mpfr`

```r
# Description of arguments :
# u...moment of rough Heston model
# parameters...data.frame containing the model parameters of a rough Heston model
# n...maximal coefficient
# verbose...

# Description of return value:
# For verbose == FALSE it returns the explosion time computed via the power series
    .
# For verbose == TRUE it returns a list containing the explosion time and the
    vector with the coefficients a_n.

explosion_time <- function(u, parameters , n,verbose = FALSE){
  case <- cases(u = u, parameters = parameters)$Case
  if(case == "Case (A)" | case == "Case (B)"){
    ints <- (1:n)
    v <- gamma(parameters$a) / beta(parameters$a * (ints - 1) + 1, parameters$a) #
         representation through beta advantageous to avoid overflow
    a <- rep(x = 0.00, times = n)
    a[1] <- c_1(u = u, parameters = parameters) * c_3(parameters = parameters) / v
        [1]
    a[2] <- c_2(u = u, parameters = parameters) * a[1] / v[2]
    for(k in (3:n)){
      a[k] <- 1 / v[k] * (c_2(u = u, parameters = parameters) * a[(k-1)]  +  a
          [(1:(k-2))] %*% rev(a[(1:(k-2))]))
    }

    if(case == "Case (A)"){
      res <- 1 / (a[n] * n^(1 - parameters$a) * gamma(parameters$a)^2 / (
          parameters$a^parameters$a * gamma(2 * parameters$a)) )^(1/(parameters$a
          * (n + 1)))
      if(verbose == TRUE){
        res <- list(ExplosionTime = res, a = a) # Both too look if the results are
            reliable
      }
      return(res)

    }else if(case == "Case (B)"){
      res <- abs(a[n])^(-1/(parameters$a * n) )
      if(verbose){
      res <- list(LowerBound = res, a = a)
      }
      return(res)
    }
  }else{
    stop("ATTENTION: We are in Case (B), (C) or (D)! This Algorithm is just for
        Case (A)")
  }
}

```

```


# ------------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model
# parameters...data.frame containing the model parameters of a rough Heston model
# n...maximal coefficient

# Description of return value:
# Returns the explosion time computed via the power series.

# Remark: This is the "mpfr"-version of "explosion_time". Here it is possible to
    choose n arbitrary large. This is not senseless since our explosion time can
    be very small but the a_n are very large such that they cause overflows while
    computing the explosion time.

explosion_time_mpfr <- function(u, parameters, n){
  case <- cases(u = u, parameters = parameters)$Case

  ints <- mpfr((1:n), 32) # 32 bits precision for integer
  v <- gamma(parameters$a) / beta(parameters$a * (ints - 1) + 1, parameters$a)
  a <- rep(mpfr(0,53), times = n)
  a[1] <- c_1(u = u, parameters = parameters) * c_3(parameters = parameters) / v
      [1]
  a[2] <- c_2(u = u, parameters = parameters) * a[1] / v[2]
  for(k in (3:n)){
    a[k] <- (c_2(u = u, parameters = parameters) * a[(k - 1)] + a[1:(k-2)] %*% rev
        (a[1:(k-2)]))/v[k]
  }

  if(case == "Case (A)"){
    res <- (a[n] * n^(1-parameters$a) * gamma(parameters$a)^2 / (parameters$a^(
        parameters$a) * gamma(2 * parameters$a)))^(-1/(parameters$a*(n + 1)))
    return(asNumeric(res)) # convert to numeric-type. Standard-overflows will
        become "Inf"s
  }else if(case == "Case (B)"){
    res <- abs(a[n])^(-1/(parameters$a * n) )
    return(asNumeric(res))
  }else{
    stop("Not appropriate case. Only implemented for Case (A) and (B)!")
  }
}


# ------------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model
# parameters...data.frame containing the model parameters of a rough Heston model
# n...maximal coefficient

# Description of return value:
# Returns vector a_n which is of type "mpfr".
```

```

a_n_mpfr <- function(u, parameters, n){
  ints <- mpfr((1:n), 32) # 32 bits precision for integer
  v <- gamma(parameters$a) / beta(parameters$a * (ints - 1) + 1, parameters$a)
  a <- rep(mpfr(0,53), times = n)
  a[1] <- c_1(u = u, parameters = parameters) * c_3(parameters = parameters) / v
      [1]
  a[2] <- c_2(u = u, parameters = parameters) * a[1] / v[2]
  for(k in (3:n)){
    a[k] <- (c_2(u = u, parameters = parameters) * a[(k - 1)] + a[1:(k-2)] %*% rev
        (a[1:(k-2)]))/v[k]
  }
  return(a)
}
```

Let us now make a first computation for the model (5.7) with $u = -3$.

```
rh <- data.frame("v" = 0.04, "l" = 0.3, "x" = 1, "v_0" = 1, "r" = -0.7, "a" = 0.6)
modelcheck(u = -3, parameters = rh) # We know already that u = -3 is ok, but we
    should always check.
T_exp <- explosion_time(u = -3, parameters = rh, n = 670)
T_exp
```

```
Output:
> rh <- data.frame('v' = 0.04, 'l' = 0.3, 'x' = 1, 'v_0' = 1, 'r' = -0.7, 'a' = 0.6)
> modelcheck(u = -3, parameters = rh) # We know already that u = -3 is ok, but we
should always check.
  Consistency ElRo16_Condition Cases.Case Cases.compl
1        TRUE             TRUE   Case (A)       FALSE
> T_exp <- explosion_time(u = -3, parameters = rh, n = 670)
> T_exp
[1] 0.1747451
```

Since the approximation of the explosion time is not exact, we have to try a little bit. Then we get the plot in Figure 5.2.

Numerical evidence for asymptotics (5.23)

```
T_exp <- explosion_time(u = -3, parameters = rh, n = 670)

library(gtools)
iter_2 <- defmacro(.GlobalEnv, expr = {
  T_exp <- T_exp + 0.0001
  par(mfrow = c(1,1))
  par(mfrow = c(1,2))

  p <- psi(u = -3, parameters = rh, T = T_exp, n = 1000)
  h <- Re(p$res)
  df1 <- data.frame(x = p$grid, y = h)
  plot(df1, type = "l", xlab = expression(t), ylab = expression(psi(u, t)))

  c <- gamma(2 * rh$a) / gamma(rh$a)
  asymp <- (log(h) - log(c)) / log(T_exp - p$grid)
  df2 <- data.frame(x = p$grid, y = asymp)
  plot(df2, type = "l", xlab = expression(t), ylab = "asymp", xlim = c(0.173,T_exp
      ))
  abline(h = -rh$a, col = "red")
  T_exp

  par(mfrow = c(1,1))
})

iter_2(); iter_2(); iter_2(); iter_2(); iter_2()
iter_2(); iter_2(); iter_2(); iter_2(); iter_2()
iter_2() # iterations necessary since explosion time is not exact
```

Numerical evidence for asymptotics (5.24)

```
N <- 150
avec <- a_n_mpfr(u = -3, parameters = rh, n = N)
T_exp <- explosion_time_mpfr(u = -3, parameters = rh, n = N)
ints <- mpfr((1:N), 32)
asymp <- avec * T_exp^(rh$a * (ints + 1)) * ints^(1-rh$a) * gamma(rh$a)^2 / (rh$a^
    rh$a * gamma(2 * rh$a))
asymp <- as.numeric(asymp)
df2 <- data.frame(x = (1:length(asymp)), y = asymp)
plot(df2, type = "o", cex = 0.5, xlab = expression(n), ylab = expression(a[n]))
abline(h = 1, col = "red")
```

Now a good example, where the upper bound of Algorithm 5.8 is nice to have is for $u = 2.851852$.

Upper bound of Algorithm 5.8 better than Theorem 4.12

```
modelcheck(u = 2.851852, parameters = rh)
T_exp <- explosion_time_mpfr(u = 2.851852, parameters = rh, n = 1000)
T_exp
bounds <- boundaries(u = 2.851852, parameters = rh)
bounds
bounds$T_up - T_exp
```

```
Output:
> modelcheck(u = 2.851852, parameters = rh)
  Consistency ElRo16_Condition Cases.Case Cases.compl
1        TRUE             TRUE   Case (B)       FALSE
> T_exp <- explosion_time_mpfr(u = 2.851852, parameters = rh, n = 1000)
> T_exp
[1] 0.2436943
> bounds <- boundaries(u = 2.851852, parameters = rh)
> bounds
         u    T_low     T_up Complex
1 2.851852 191.2771 79842.92   FALSE
> bounds$T_up - T_exp
[1] 79842.67
```

As we can see, the upper bound of 0.2436943 is very valuable, since from Theorem 4.12, we just get an upper bound of 79842.92.

6.5. Implementation of polylogarithmic approximation

First, let us consider some functions we need for the computation of the approximation (5.34).

Functions for polylog-approximation: `Li`, `rhs`, `rhs_vect`, `rhs_abvect`, `rhs_mpfr`, `rhs_mpfr_abvec`, `Error`

```r
# Description of arguments :
# z...real value |.| <= 1
# nu...real value
# n...number of maximal summands

# Description of return value:
# Returns the Li- or polylogarithmic function.

Li <- function(z, nu, n = 100000){
  if(abs(z) <= 1){
    res <- sum(z^(1:n) / (1:n)^nu)
  }else{
    res <- Inf
  }
  return(res)
}



# ------------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model
# t...time t
# N...N of the approximation of f via polylogarithm
# T_exp...Explosion time we get from an estimator
# parameters...data.frame containing the model parameters of a rough Heston model

# Description of return value:
# Returns the right-hand side of the approximation of f with a polylogarithm.

rhs <- function(u, t, N, T_exp, parameters){
  fac_1 <- parameters$a^(parameters$a) * gamma(2 * parameters$a) / gamma(
      parameters$a)^2 * T_exp^(-parameters$a)
  fac_2 <- Li(z = (t/T_exp)^(parameters$a), nu = (1 - parameters$a))
  sum_1 <- fac_1 * fac_2

  # Create vector a
  ints <- (1:N)
  v <- gamma(parameters$a) / beta(parameters$a * (ints - 1) + 1, parameters$a)
  a <- rep(x = 0, times = N)
  a[1] <- c_1(u = u, parameters = parameters) * c_3(parameters = parameters) / v
      [1]
  a[2] <- c_2(u = u, parameters = parameters) * a[1] / v[2]
  for(k in (3:N)){
    a[k] <- 1 / v[k] * (c_2(u = u, parameters = parameters) * a[(k-1)]   +   a
        [(1:(k-2))] %*% rev(a[(1:(k-2))]))
  }

  # Create vector b
  b <- fac_1 * ints^(parameters$a-1) * T_exp^(-parameters$a * ints)
  sum_2 <-  sum((a - b)[(1:(N-1))] * t^(parameters$a * ints[(1:(N-1))]))
  res <- sum_1 + sum_2
  return(res)
}



# ------------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model
# t...time t (dim(t) >= 1)
# N...N of the approximation of f via polylogarithm
# T_exp...Explosion time we get from an estimator
# parameters...data.frame containing the model parameters of a rough Heston model

# Description of return value:
# Returns a vector with that ealuates "rhs" for each t.

rhs_vect <- function(u, t, N, T_exp, parameters){
  n_t <- length(t)
  res <- rep(0, n_t)
  for(j in (1:n_t)){
    res[j] <- rhs(u, t[j] , N , T_exp, parameters)
  }
  return(res)
}



# -----------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model
# t...time t (dim(t) >= 1)
# N...N of the approximation of f via polylogarithm
# T_exp...Explosion time we get from an estimator
# parameters...data.frame containing the model parameters of a rough Heston model

# Description of return value:
# Returns the vectors a and b which are used for the approximation of f via
    polylogarithms.

rhs_abvec <- function(u, t, N, T_exp, parameters){
  fac_1 <- parameters$a^(parameters$a) * gamma(2 * parameters$a) / gamma(
      parameters$a)^2 * T_exp^(-parameters$a)
  fac_2 <- Li(z = (t/T_exp)^(parameters$a), nu = (1 - parameters$a))
  sum_1 <- fac_1 * fac_2

  # Create vector a
  ints <- (1:N)
  v <- gamma(parameters$a) / beta(parameters$a * (ints - 1) + 1, parameters$a)
  a <- rep(x = 0, times = N)
  a[1] <- c_1(u = u, parameters = parameters) * c_3(parameters = parameters) / v
      [1]
  a[2] <- c_2(u = u, parameters = parameters) * a[1] / v[2]
  for(k in (3:N)){
    a[k] <- 1 / v[k] * (c_2(u = u, parameters = parameters) * a[(k-1)]   +   a
        [(1:(k-2))] %*% rev(a[(1:(k-2))]))
  }

  # Create vector b
  b <- fac_1 * ints^(parameters$a-1) * T_exp^(-parameters$a * ints)
  sum_2 <-  sum((a - b)[(1:(N-1))] * t^(parameters$a * ints[(1:(N-1))]))
  res <- sum_1 + sum_2
  return(list(a = a, b = b))
}



# -----------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model
# t...time t
# N...N of the approximation of f via polylogarithm
# T_exp...Explosion time we get from an estimator
# parameters...data.frame containing the model parameters of a rough Heston model

# Description of return value:
# mpfr-method of rhs.

rhs_mpfr <- function(u, t, N, T_exp, parameters){
  fac_1 <- parameters$a^(parameters$a) * gamma(2 * parameters$a) / gamma(
      parameters$a)^2 * T_exp^(-parameters$a)
  fac_2 <- Li(z = (t/T_exp)^(parameters$a), nu = (1 - parameters$a))
  sum_1 <- fac_1 * fac_2

  # Create vector a
  ints <- mpfr((1:N),32)
  v <- gamma(parameters$a) / beta(parameters$a * (ints - 1) + 1, parameters$a)
  a <- mpfr(rep(x = 0, times = N),53)
  a[1] <- c_1(u = u, parameters = parameters) * c_3(parameters = parameters) / v
      [1]
  a[2] <- c_2(u = u, parameters = parameters) * a[1] / v[2]
  for(k in (3:N)){
    a[k] <- 1 / v[k] * (c_2(u = u, parameters = parameters) * a[(k-1)]   +   a
        [(1:(k-2))] %*% rev(a[(1:(k-2))]))
  }

  # Create vector b
  b <- fac_1 * ints^(parameters$a-1) * T_exp^(-parameters$a * ints)
  sum_2 <-  sum((a - b)[(1:(N-1))] * t^(parameters$a * ints[(1:(N-1))]))
  res <- sum_1 + sum_2
  return(res)
}



# ------------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model
# t...time t
# N...N of the approximation of f via polylogarithm
# T_exp...Explosion time we get from an estimator
# parameters...data.frame containing the model parameters of a rough Heston model

# Description of return value:
# mpfr-method of rhs_abvec.

rhs_mpfr_abvec <- function(u, t, N, T_exp, parameters){
  fac_1 <- parameters$a^(parameters$a) * gamma(2 * parameters$a) / gamma(
      parameters$a)^2 * T_exp^(-parameters$a)
  fac_2 <- Li(z = (t/T_exp)^(parameters$a), nu = (1 - parameters$a))
  sum_1 <- fac_1 * fac_2

  # Create vector a
  ints <- mpfr((1:N),32)
  v <- gamma(parameters$a) / beta(parameters$a * (ints - 1) + 1, parameters$a)
  a <- mpfr(rep(x = 0, times = N),53)
  a[1] <- c_1(u = u, parameters = parameters) * c_3(parameters = parameters) / v
      [1]
  a[2] <- c_2(u = u, parameters = parameters) * a[1] / v[2]
  for(k in (3:N)){
```

```r
    a[k] <- 1 / v[k] * (c_2(u = u, parameters = parameters) * a[(k-1)]   +   a
        [(1:(k-2))] %*% rev(a[(1:(k-2))]))
  }

  # Create vector b
  b <- fac_1 * ints^(parameters$a-1) * T_exp^(-parameters$a * ints)
  sum_2 <-  sum((a - b)[(1:(N-1))] * t^(parameters$a * ints[(1:(N-1))]))
  res <- sum_1 + sum_2
  return(list(a = a, b = b))
}



# ------------------------------------------------------------



# Description of arguments :
# u...moment of rough Heston model
# t...time t
# N...N of the approximation of f via polylogarithm
# T_exp...Explosion time we get from an estimator
# parameters...data.frame containing the model parameters of a rough Heston model
# N_max...defines the maximal sumand for the "infinite" sum

# Description of return value:
# Returns the error from putting the asymptotic b_n into the power series for the
    a_n.

Error <- function(u, t, N, T_exp, parameters, N_max = 3000){
  res <- rhs_mpfr_abvec(u = u, t = t, N = N_max, T_exp = T_exp, parameters =
      parameters)
  res <- sum((res$a[N:N_max] - res$b[N:N_max]) * t^(parameters$a * (N:N_max)))
  return(asNumeric(res))
}
```

Script for plotting the approximation

```r
rh <- data.frame("v" = 0.04, "l" = 0.3, "x" = 1, "v_0" = 1, "r" = -0.7, "a" = 0.6)
T_exp <- explosion_time(u = -3, parameters = rh, n = 670)
p <- psi(u = -3, parameters = rh, T = T_exp, n = 1000)

f <- c_3(parameters = rh)*Re(p$res)

library(gtools)
approx <- defmacro(.GlobalEnv, expr = {
  T_exp <- explosion_time(u = -3, parameters = rh, n = 670)
  x <- p$grid
  y <- function(t){
    res <- rhs_vect(u = -3, t = t, N = N, T_exp = T_exp, parameters = rh)
    return(res)
  }
  plot(y, type = "l", col = "red", xlim = c(0, T_exp-0.001),
       xlab = expression(t), ylab = bquote(paste(f(u, t), phantom() %~~% phantom()
           , widehat(f)(u, t) ,  sep = "")),
      main = bquote(paste("Polylog-approximation for N = ", .(N), sep = ""))) #
          ploting a function automatically adjusts the ylim to the xlim !
  lines(x = x, y = f, col = "blue",lty = "dashed")
  legend("topleft", inset = 0.05,  legend = c("f(t)", "polylog-approx."), col = c(
      "red", "blue"), cex = 0.8, lty=1:2)
})

ma_standard <- c(5,4,4,2) + 0.1
ma <- c(5,4.4,4,2) + 0.1
par(mar = ma)

par(mfrow = c(1,1))
par(mfrow = c(2,2))

N <- 3
approx()

N <- 5
approx()

N <- 10
approx()

N <- 20
approx()

par(mfrow = c(1,1))
```

The outcome, i.e. the plots, can be seen in Figure 5.4. Then let us determine the constant β from (5.37).

Power β for error bound

```
T_exp <- explosion_time(u = -3, parameters = rh, n = 670)
t <- T_exp - 0.01
N <- 3000
ab_vec <- rhs_mpfr_abvec(u = 0, t = t, N = N, T_exp = T_exp, parameters = rh)
a_vec <- ab_vec$a
b_vec <- ab_vec$b
ints <- mpfr((1:N),32)
beta <- (log(abs(a_vec - b_vec)) + rh$a * ints * log(T_exp))/log(ints)
plot(asNumeric(beta),  type = "l", xlim = c(100,3000), xlab = expression(n), ylab
    = expression(beta[n]),
     main = bquote(paste("Approximation of ", beta, " ", phantom() %~~% phantom(),
          .(asNumeric(beta[length(beta)])), sep = "")))
abline(h = asNumeric(beta[length(beta)]), col = "red", lty = "dashed")
asNumeric(beta[length(beta)])
```

```
Output:
---------------------------------------------------------------------
#####################################################################
---------------------------------------------------------------------
> asNumeric(beta[length(beta)])
[1] -0.4176961
```

The plot for the approximation of β can be seen in Figure 5.5.

A. Appendix

A.1. Asymptotics

Definition A.1 ([FS09])**.** Let $\mathbb{S}$ be a set and $s_0 \in \mathbb{S}$. We assume a notion of neighborhood to exist on $\mathbb{S}$, such that $s_0 \in \overline{\mathbb{S}}$ is possible, e.g. $\mathbb{S} = \mathbb{R}$ and $s_0 = +\infty$. Two functions $f, g : \mathbb{S} \backslash \{s_0\} \to \mathbb{R}$ ($\mathbb{C}$) are given.

(i) Write

$$f(s) = o(g(s)), \quad s \to s_0, \tag{A.1}$$

if

$$\lim_{s \to s_0} \left| \frac{f(s)}{g(s)} \right| = 0. \tag{A.2}$$

In other words, for any (arbitrary small) $\varepsilon > 0$, there exists a neighborhood $\mathcal{V}_\varepsilon$ of s_0 (depending on ε), such that

$$|f(s)| \leq \varepsilon |g(s)|, \quad s \in \mathcal{V}_\varepsilon, \quad s \neq s_0.$$

We say *"f is of order smaller than g"*, or *"f is little-oh of g"*, or *"f is asymptotically dominated by g"* (as s tends to s_0).

(ii) Write

$$f(s) = \mathcal{O}(g(s)), \quad s \to s_0, \tag{A.3}$$

if

$$\limsup_{s \to s_0} \left| \frac{f(s)}{g(s)} \right| < \infty. \tag{A.4}$$

In other words, there exists a neighborhood $\mathcal{V}$ of s_0 and a constant $C > 0$ such that

$$|f(s)| \leq C |g(s)|, \quad s \in \mathcal{V}, \quad s \neq s_0.$$

One also says that *"f is of order at most g"*, or *"f is big-oh of g"*, or *"f is bounded from above by g (up to a constant factor) asymptotically"* (as s tends to s_0).

(iii) Write

$$f(s) \sim g(s), \quad s \to s_0, \tag{A.5}$$

if

$$\lim_{s \to s_0} \frac{f(s)}{g(s)} = 1. \tag{A.6}$$

One also says that *"f and g are asymptotically equivalent"* (as s tends to s_0).

Remark A.2. Note that very often the term $s \to s_0$ is omitted, since it can mostly be identified by the context.

Remark A.3. Note that $f(s) = \mathcal{O}(1)$ resp. $f \in \mathcal{O}(1)$ (for $s \to s_0$) means that f is bounded for $s \to s_0$, since this is equivalent to

$$\limsup_{s \to s_0} \frac{|f(s)|}{1} < \infty.$$

Lemma A.4. *Some properties of the Landau-Notation are the following:*

(i) ***Product****: For $f_1 = \mathcal{O}(g_1)$ and $f_2 = \mathcal{O}(g_2)$ we have $f_1 f_2 = \mathcal{O}(g_1 g_2)$. Especially we have $f\mathcal{O}(g) = \mathcal{O}(fg)$.*

(ii) ***Sum****: For $f_1 = \mathcal{O}(g_1)$ and $f_2 = \mathcal{O}(g_2)$ we have $f_1 + f_2 = \mathcal{O}(|g_1| + |g_2|)$. This implies that for $f_1, f_2 \in \mathcal{O}(g)$ we get $f_1 + f_2 \in \mathcal{O}(g)$ which means that $\mathcal{O}(g)$ is a convex cone. If f and g are positive functions, we get $\mathcal{O}(f) + \mathcal{O}(g) = \mathcal{O}(f + g)$.*

(iii) ***Constant multiplication****: Let k be a constant. Then we have $\mathcal{O}(kg) = \mathcal{O}(g)$ if k is nonzero. From $f = \mathcal{O}(g)$ we get $kf = \mathcal{O}(g)$.*

Proof. The properties above can be computed straight forward using the definition of the big-$\mathcal{O}$-notation.

Ad (i). Here we get with suppression of the argument for ease of notation

$$\limsup_{s \to s_0} \left| \frac{f_1 f_2}{g_1 g_2} \right| \leq \limsup_{s \to s_0} \left(\left| \frac{f_1}{g_1} \right| \left| \frac{f_1}{g_1} \right| \right) \leq \left(\limsup_{s \to s_0} \left| \frac{f_1}{g_1} \right| \right) \left(\limsup_{s \to s_0} \left| \frac{f_2}{g_2} \right| \right) < \infty.$$

Ad (ii). We have for $|f_1/g_1|, |f_2/g_2| < C$ that

$$\frac{|f_1 + f_2|}{|g_1| + |g_2|} \leq \frac{|f_1|}{|g_1| + |g_2|} + \frac{|f_2|}{|g_1| + |g_2|} \leq \frac{|f_1|}{|g_1|} + \frac{|f_2|}{|g_2|} < 2C$$

which leads to $f_1 + f_2 = \mathcal{O}(|g_1| + |g_2|)$.

Ad (iii). For $|f/g| < C$ we get

$$\left| \frac{kf}{g} \right| < |k| C.$$

□

Remark A.5. Note that the notation can be combined with other arithmetic operators, e.g. $g(s) = h(s) + \mathcal{O}(f(s))$ expresses the same as $g(s) - h(s) = \mathcal{O}(f(s))$.

Theorem A.6 (Stirling's formula). *For every $x > 0$ exists $\theta(x) \in (0, 1)$ such that*

$$\Gamma(x) = \sqrt{2\pi} x^{x-1/2} e^{-x} e^{\theta(x)/12}.$$

Proof. See [AE06, Theorem VI.9.10.]. □

Remark A.7. Note that in [FS09] we have

$$n! = \sqrt{2\pi} n^{n-1/2} e^{-n}(1 + \epsilon_n), \quad 0 < \epsilon_n < \frac{1}{12n},$$

so we have

$$n! \sim \sqrt{2\pi} n^{n-1/2} e^{-n}, \quad n \to \infty, \tag{A.7}$$

and therefore

$$\Gamma(x) \sim \sqrt{2\pi} x^{x-1/2} e^{-x}, \quad x \to \infty. \tag{A.8}$$

A.2. Lipschitz Continuity

Definition A.8 ([Bru17]). Let $k = k(t, s, u)$ be defined for $0 \leq s \leq t < \infty$ and u in $\mathbb{R}$.

(i) If, for any given positive constants A and B, there exists a constant $L \geq 0$ such that

$$|k(t, s, u_1) - k(t, s, u_2)| \leq L|u_1 - u_2|, \qquad 0 \leq s \leq t \leq T, \quad A \leq u_1, u_2 \leq B, \tag{A.9}$$

then k is said to be *locally Lipschitz continuous* in u.

(ii) If the Lipschitz constant L does not depend on B, then k is called *globally Lipschitz continuous* in u.

Remark A.9. Note that the reason for $k(t, s, u)$ in Definition A.8 to be dependent on 3 variables is that we are dealing with 3- resp. 2-variate functions which are said to be locally Lipschitz in Chapter 4. Brunner seemingly just wants a most general definition to be appropriate for his representation (2.10) of nonlinear Volterra integral equation in [Bru17].

Lemma A.10. *Polynomials are locally Lipschitz continuous.*

Proof. The idea for this short proof is from [Cla13]. Let $p : \mathbb{R} \to \mathbb{R}$ be a polynomial. Then, for $\widetilde{p} := p_{|[A,B]} : [A, B] \to \mathbb{R}$, we get that the continuous function $\partial_x \widetilde{p}(x)$ on $[A, B]$ is bounded, i.e.

$$|\partial_x \widetilde{p}(x)| \leq L, \quad x \in [A, B],$$

and with the mean value theorem we get for some $\xi \in (A, B)$

$$|\widetilde{p}(y) - \widetilde{p}(x)| = |\partial_x \widetilde{p}(\xi)||y - x| \leq L|y - y|, \quad x, y \in (A, B),$$

which concludes the proof. $\square$

Bibliography

[AE06] Herbert Amann and Joachim Escher. *Analysis 1, dritte Auflage.* Birkhäuser, 2006.

[AP07] Leif B. G. Andersen and Vladimir V. Piterbarg. Moment explosions in stochastic volatility models. *Finance Stoch.*, 11(1):29–50, 2007.

[BFG16] Christian Bayer, Peter Friz, and Jim Gatheral. Pricing under rough volatility. *Quant. Finance*, 16(6):887–904, 2016.

[Bru04] Hermann Brunner. *Collocation methods for Volterra integral and related functional differential equations*, volume 15 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2004.

[Bru17] Hermann Brunner. *Volterra integral equations*, volume 30 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2017.

[Cla13] Pete L. Clark. Show polynomial is a Lipschitz function. https://math.stackexchange.com/q/362858, 2013.

[Con14] Rama Cont. *Encyclopedia of Quantitative Finance.* John Wiley and Sons, 2014.

[Els05] Jürgen Elstrodt. *Maß- und Integrationstheorie.* Springer-Lehrbuch. [Springer Textbook]. Springer-Verlag, Berlin, fourth edition, 2005.

[ER16] Omar El Euch and Mathieu Rosenbaum. The characteristic function of rough Heston models. Preprint, arXiv:1609.02108, 2016.

[ER17] Omar El Euch and Mathieu Rosenbaum. Perfect hedging in rough Heston models. Preprint, arXiv:1703.05049, 2017.

[FS09] Philippe Flajolet and Robert Sedgewick. *Analytic combinatorics.* Cambridge University Press, Cambridge, 2009.

[GGP18] Stefan Gerhold, Christoph Gerstenecker, and Arpad Pinter. Moment explosions in the rough Heston model. Preprint, arXiv:1801.09458v3, 2018.

[GJR18] Jim Gatheral, Thibault Jaisson, and Mathieu Rosenbaum. Volatility is rough. *Quant. Finance*, 2018. Available online.

[GK06] Robert E. Greene and Steven G. Krantz. *Function theory of one complex variable*, volume 40 of *Graduate Studies in Mathematics.* American Mathematical Society, Providence, RI, third edition, 2006.

[GLS90] Gustaf Gripenberg, Stig-Olof Londen, and Olof Staffans. *Volterra integral and functional equations*, volume 34 of *Encyclopedia of Mathematics and its Applications.* Cambridge University Press, Cambridge, 1990.

[Hes93] Steven L. Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Rev. Financial Stud.*, 6(2):327–343, 1993.

[Kel11] Martin Keller-Ressel. Moment explosions and long-term behavior of affine stochastic volatility models. *Math. Finance*, 21(1):73–98, 2011.

[KST06] Anatoly A. Kilbas, Hari M. Srivastava, and Juan J. Trujillo. *Theory and applications of fractional differential equations*, volume 204 of *North-Holland Mathematics Studies.* Elsevier Science B.V., Amsterdam, 2006.

[LT09] Changpin Li and Chunxing Tao. On the fractional Adams method. *Comput. Math. Appl.*, 58(8):1573–1588, 2009.

[Lub83] Christian Lubich. Runge-Kutta theory for Volterra and Abel integral equations of the second kind. *Math. Comp.*, 41(163):87–102, 1983.

[MV68] Benoit B. Mandelbrot and John W. Van Ness. Fractional Brownian motions, fractional noises and applications. *SIAM Rev.*, 10(4):422–437, 1968.

[Rem91] Reinhold Remmert. *Theory of complex functions*, volume 122 of *Graduate Texts in Mathematics.* Springer-Verlag, New York, 1991.

[RO96] Catherine A. Roberts and W. Edward Olmstead. Growth rates for blow-up solutions of nonlinear Volterra equations. *Quart. Appl. Math.*, 54(1):153–159, 1996.

[Sch05] René L. Schilling. *Measures, integrals and martingales.* Cambridge University Press, New York, 2005.

[SKM93] Stefan G. Samko, Anatoly A. Kilbas, and Oleg I. Marichev. *Fractional integrals and derivatives.* Gordon and Breach Science Publishers, Yverdon, 1993.

[Wid41] David V. Widder. *The Laplace Transform.* Princeton Mathematical Series, v. 6. Princeton University Press, Princeton, N. J., 1941.