Student t-Lévy regression model in yuima

by Hiroki Masuda, Lorenzo Mercuri, and Yuma Uehara

Abstract This paper presents an estimation and simulation method in the R package yuima for a linear regression model driven by a Student-t Lévy process with constant scale and arbitrary degrees of freedom. This process finds applications in several fields, for example finance, physics, biology, etc. The first challenge involves simulating sample paths at high-frequency levels, as only unit-time increments are Student-t distributed. In yuima, we solve this problem by means of the inverse Fourier transform for simulating the increments of a Student-t Lévy defined on an interval with any length. The second challenge is the joint estimation of trend, scale, and degrees of freedom, a problem not previously explored in the literature. In yuima, we develop a two-step estimation procedure that efficiently deals with this issue. Numerical examples are given in order to explain methods and classes used in the yuima package.

1 Introduction

The **yuima** package in R provides several simulation and estimation methods for stochastic processes (YUIMA Project Team, 2024; Brouste et al., 2014; Iacus and Yoshida, 2018). This paper introduces new classes and methods in **yuima** for simulating and estimating a t-Lévy regression model based on high-frequency observations. This model can be seen as a generalization of a t-Lévy process (Heyde and Leonenko, 2005; Cufaro Petroni, 2007) by adding covariates, which can be either deterministic or stochastic processes. Adding covariates to a continuous-time stochastic process is a widely used approach for constructing new processes in various fields. For example, in finance, periodic deterministic covariates can be employed to capture the seasonality observed in commodity markets (Sørensen, 2002). Alternatively, in insurance and medicine, covariates are often incorporated into mortality rate dynamics to account for age and cohort effects (see Haberman and Renshaw, 2009; Castro-Porras et al., 2021, and references therein).

In such fields, data often exhibit heavy-tailed behavior in the margins and thus, the inclusion of a t-Lévy process as a driving noise would be useful. However, despite the simple mathematical definition of a t-Lévy regression model, it poses significant challenges both in deriving its mathematical properties and in implementing numerical methods. A key difficulty arises from the fact that the t-Lévy driving noise is not closed under convolution. Consequently, the distribution of its increments follows a t-distribution only over a unit-time interval.

Motivated by this fact, in **yuima**, we propose three simulation methods for a t-Lévy regression model. The key element is the construction of a random number generator for the noise increments, which is essential for several discretized simulation methods and involves the numerical inversion of the characteristic function. More specifically, our method is based on the approximation of the cumulative distribution function, and our method can relieve numerical instability compared with directly using the density function especially for simulating small time increments. Additionally, **yuima** provides a two-stage estimation procedure and the corresponding estimator has consistency and asymptotic normality as shown in Masuda et al. (2024).

The rest of this paper is organized as follows. We briefly summarize the t-Lévy regression model in Section 2. We introduce the new **yuima** classes and methods in Section 3 and we show some examples with simulated and real data that highlight their usage in Section 4. Finally, Section 5 concludes the paper.

2 Student t-Lévy regression model

In this section, we review the main characteristics of the t-Lévy regression model and the t-Lévy process used as its driving noise. We highlight the main issues that arise in the simulation and estimation procedures for these models. For a complete discussion on all mathematical aspects and the properties of these procedures in both cases, we refer to Masuda et al. (2024).

The proposed t-Lévy regression model is a continuous-time stochastic process of the form:

$$Y_t = X_t \cdot \mu + \sigma J_t, \qquad t \in [0, T_n], \tag{1}$$

where the q-dimensional vector process $X = (X_t)$ with càdlàg paths contains the covariates; the dot denotes the inner product in $\mathbb{R}^q$ and $J = (J_t)$ is a scaled t-Lévy process such that its unit-time distribution $\mathcal{L}(J_1)$ is given by:

$$\mathcal{L}(J_1) = t_\nu := t_\nu(0, 1), \tag{2}$$

where $t_\nu(\mu_1, \sigma_1)$ denotes the scaled Student-t distribution[1] with density: The parameters $\nu > 0$, and $\sigma > 0$ represent the degree of freedom, location, and scale parameters, respectively (see Heyde and Leonenko, 2005; Cufaro Petroni, 2007, for more details on a t-Lévy process). For this model, we consider the situation where we estimate these three unknown parameters based on a discrete-time sample $\{(X_{t_j}, Y_{t_j})\}_{j=0}^{[nT_n]}$ with $t_j = t_j^n := \frac{j}{n}$ and $T_n \to \infty$ as $n \to \infty$.

To develop the **yuima** simulation and estimation algorithms for the model in (1), we need to address two problems: first, the simulation of the sample path of $J = (J_t)$ on a small time grid with $\Delta t \neq 1$ which is essential for handling the increments:

$$\Delta_j Y = \Delta_j X \cdot \mu + \sigma \Delta_j J, \quad j = 0, 1, \ldots, [nT_n]$$

where $\Delta_j Z$ denotes $Z_{t_j} - Z_{t_{j-1}}$ for any stochastic process $Z = (Z_t)$. Second, the identification of an efficient procedure for estimating the model parameters (μ, σ) in (1) and the degree of freedom ν in (2). We will introduce both the simulation and estimation methods below.

Due to the stationary behavior of the L'evy increments, i.e. $\mathcal{L}(\Delta_i J) = \mathcal{L}(J_h)$ and $h := t_i - t_{i-1} = \frac{1}{n}$ for $i = 1, \ldots, [nT_n]$, $\mathcal{L}(J_h)$ admits the Lebesgue density:

$$\begin{aligned} x \mapsto & \frac{1}{\pi} \int_0^\infty \cos(ux) \{\varphi_{J_1,\nu}(u)\}^h du \\ & = \left(\frac{2^{1-\nu/2}}{\Gamma(\nu/2)} \right)^h \frac{1}{\pi} \int_0^\infty \cos(ux) u^{\nu h/2} \left(K_{\nu/2}(u)\right)^h du, \end{aligned} \tag{3}$$

where $\varphi_{J_1,\nu}(u)$, the characteristic function of $\mathcal{L}(J_1) = t_\nu$ is given by

$$\varphi_{J_1,\nu}(u) := \frac{2^{1-\nu/2}}{\Gamma(\nu/2)} |u|^{\nu/2} K_{\nu/2}(|u|), \qquad u \in \mathbb{R} \tag{4}$$

and $K_\nu(t)$ denotes the modified Bessel function of the second kind ($\nu \in \mathbb{R}$, $t > 0$):

$$K_\nu(t) = \frac{1}{2} \int_0^\infty s^{\nu-1} \exp\left\{ -\frac{t}{2} \left(s + \frac{1}{s} \right) \right\} ds.$$

We here remark that although the explicit expression of (3) cannot be obtained for general h, the tail index of $\mathcal{L}(J_h)$ is the same as that of $\mathcal{L}(J_1)$ (Berg and Vignat, 2008, Theorem 2),

[1] Let J_1 be a scaled Student-t random variable with parameters $\nu > 0$ $\mu_1 = 0$ and $\sigma_1 = 1$, its transformation $Z_1 := \sqrt{\nu} J_1$ is a Student-t r.v. see @johnson1995continuous Ch. 28 with the following density function:

$$f_{Z_1}(z; \nu) := \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\pi\nu}\Gamma(\frac{\nu}{2})} \left\{ 1 + \frac{z^2}{\nu} \right\}^{-(\nu+1)/2}.$$

and hence, modeling tail index via student t-Lévy process makes sense. A classical method for simulating t-Lévy increments based on this Fourier inversion representation is through the rejection method, as discussed in Hubalek (2005) and Devroye (1981). However, this method can become time-consuming and unstable when dealing with very small values of h, primarily due to the oscillatory behavior encountered during the numerical evaluation of the density. Such a problem often arises since to express the high-frequency observed situation from a continuous process, we should take a finer mesh of simulating the underlying process than that of simulating the discrete observations. For this issue, in **yuima**, the Random Number Generator for the increments is developed using the inverse quantile method. The first step involves integrating the density in (3) to obtain the cumulative distribution function (CDF). This approach leads to a more stable behavior in the tails of the CDF compared to the density tails, owing to a mitigation effect observed during the numerical evaluation of the following double integral:

$$F(y) = \int_{-\infty}^{y} \frac{1}{\pi} \int_{0}^{\infty} \cos(ux)\{\varphi_{J_1,\nu}(u)\}^h du dx.$$

Further details on our approach are provided in the next section. Another way to simulate t-Lévy process is using series representation. Numerically, for each time t, we need ad-hoc truncation of the infinite sum. The Gaussian approximation of a small-jump part is also valid (Asmussen and Rosiński, 2001), but the associated error may not be easy to control in a practical manner. For more details, see Massing (2018).

Following Masuda et al. (2024), **yuima** provides a two-stage estimation algorithm for $(\mu, \sigma, \nu) \in \Theta_\mu \times \Theta_\sigma \times \Theta_\nu$. Denote by (μ_0, σ_0, ν_0) the true parameter values of the model in (1); the two-step algorithm can be summarized as follows:

1. An estimator $\hat{a} := (\hat{\mu}_n, \hat{\sigma}_n)$ of $a_0 = (\mu_0, \sigma_0)$ is obtained by maximizing the *Cauchy quasi-likelihood*:
$$\hat{a}_n := (\hat{\mu}_n, \hat{\sigma}_n) \in \underset{a \in \overline{\Theta_\mu \times \Theta_\sigma}}{\operatorname{argmax}} \mathbb{H}_{1,n}(a). \tag{5}$$
The Cauchy quasi-(log-)likelihood $\mathbb{H}_{1,n}(a)$ conditional on X has the following form:
$$\begin{aligned}\mathbb{H}_{1,n}(a) &:= \sum_{j=1}^{N_n} \log\left\{\frac{1}{h\sigma}\phi_1\left(\frac{\Delta_j Y - \mu \cdot \Delta_j X}{h\sigma}\right)\right\} \\ &= C_n - \sum_{j=1}^{N_n}\left\{\log\sigma + \log\left(1 + \epsilon_j(a)^2\right)\right\},\end{aligned}$$
where ϕ_1 denotes the density function of the standard Cauchy distribution and the term C_n does not depend on $a = (\mu, \sigma)$. $N_n := [nB_n]$ is the number of observations in the part $[0, B_n]$ of the entire period $[0, T_n]$ where (B_n) is a positive sequence satisfying $B_n \le T_n$ and $n^{\epsilon''} \lesssim B_n \lesssim n^{1-\epsilon'}$ for some $\epsilon', \epsilon'' \in (0,1)$.

2. Then, we construct an estimator $\hat{\nu}_n$ of ν_0 using the *Student-t quasi-likelihood* on the "unit-time'' residual sequence $\hat{\epsilon}_i$:
$$\hat{\epsilon}_i := \hat{\sigma}_n^{-1}\left(Y_i - Y_{i-1} - \hat{\mu}_n \cdot (X_i - X_{i-1})\right),$$
for $i = 1, \ldots, [T_n]$, which is expected to be approximately i.i.d. t_ν-distributed. Therefore $\hat{\nu}_n$ solves:
$$\hat{\nu}_n \in \underset{\nu \in \overline{\Theta}_\nu}{\operatorname{argmax}} \mathbb{H}_{2,n}(\nu),$$
where
$$\mathbb{H}_{2,n}(\nu) := \sum_{i=1}^{[T_n]}\left(-\frac{1}{2}\log\pi + \log\Gamma\left(\frac{\nu+1}{2}\right) - \log\Gamma\left(\frac{\nu}{2}\right) - \frac{\nu+1}{2}\log\left(1 + \hat{\epsilon}_i^2\right)\right).$$

We note that the first estimation scheme is based on the locally Cauchy property of J: $h^{-1}J_h \xrightarrow{\mathcal{L}} t_1$ (standard Cauchy) as $h \to 0$. Let $\hat{u}_{a,n} := \sqrt{N_n}(\hat{a}_n - a_0)$ and $\hat{u}_{\nu,n} := \sqrt{T_n}(\hat{\nu}_n - \nu_0)$. Under some regularity conditions on the covariate process $X = (X_t)$ (see Masuda et al., 2024, Assumption 2.1 in for all requirements on X), the estimators have the following joint asymptotic normality:

$$(\hat{\Gamma}_{a,n}^{1/2}\hat{u}_{a,n}, \hat{\Gamma}_{\nu,n}^{1/2}\hat{u}_{\nu,n}) \xrightarrow{\mathcal{L}} N_{q+2}(0, I_{q+2}),$$

where ψ_1 denotes the trigamma function and

$$\hat{\Gamma}_{a,n} = \mathrm{diag}\left(\frac{1}{2\hat{\sigma}_n^2 N_n}\sum_{j=1}^{N_n}\left(\frac{1}{h}\Delta_j X\right)^{\otimes 2}, \frac{1}{2\hat{\sigma}_n^2}\right),$$

$$\hat{\Gamma}_{\nu,n} = \frac{1}{4}\left(\psi_1\left(\frac{\hat{\nu}_n}{2}\right) - \psi_1\left(\frac{\hat{\nu}_n + 1}{2}\right)\right),$$

Since $\hat{\Gamma}_{a,n}^{1/2}$ and $\hat{\Gamma}_{\nu,n}$ can be constructed only by the observations, we can easily obtain the confidence intervals of each parameter.
We conclude this section by noting that the function $\mathbb{H}_{2,n}(\nu)$ is concave with respect to ν, as demonstrated in Masuda et al. (2024). This result follows directly from the fact that the driving noise in the t-Lévy regression model is a scaled t-Lévy process with a unit-time distribution as defined in (2).

3 Classes and methods for t-lévy regression models

This section provides an overview of the new classes and methods introduced in **yuima** for the mathematical definition, trajectory simulation, and estimation of a Student Lévy Regression model. To handle this model, the first step involves constructing an object of the `yuima.LevyRM-class`. As an extension of the `yuima-class` refer to Brouste et al. (2014) for more details, `yuima.LevyRM-class` inherits slots such as `@data`, `@model`, `@sampling`, and `@functional` from its parent class. The remaining slots store specific information related to the Student-t-Lévy regression model.

Notably, the slot `@unit_Levy` contains an object of the `yuima.th` class, which represents the mathematical description of the Student-t Lévy process J_t (see the subsequent section for detailed explanations). The labels of the regressors are saved in the slot `@regressors`, while slots `@LevyRM` and `@paramRM` respectively cache the names of the output process Y_t and a `string` vector reporting the regressors' coefficients, the scale parameter, and the degree of freedom. The `yuima.th-class` is obtained by the new `setLaw_th` constructor. This function requires the arguments used for the numerical inversion of the characteristic function, and its usage is discussed in the next section.

Once the `yuima.th-object` is created, we define the system of stochastic differential equations (SDEs) that describes the behavior of the regressors, with their mathematical definitions stored in an object of the `yuima.model-class`. Both `yuima.th` and `yuima.model` objects are used as inputs for the `setLRM` constructor, which returns an object of the `yuima.LevyRM-class`. The following chunk code reports the input for this new function.

```
setLRM(unit_Levy, yuima_regressors, LevyRM = "Y", coeff = c("mu", "sigma0"), data = NULL,
  sampling = NULL, characteristic = NULL, functional = NULL)
```

As is customary for any class extending the `yuima-class`, the `simulate` method enables the generation of sample paths for the Student Lévy Regression model. To simulate trajectories, an object of the `yuima.sampling-class` is constructed to represent an equally spaced grid-time used in trajectory simulation. The regressors' paths are obtained using the Euler scheme, while the increments of the Student-t Lévy process are simulated using the random number generator available in the slot `@rng` of the `yuima.th-object`.

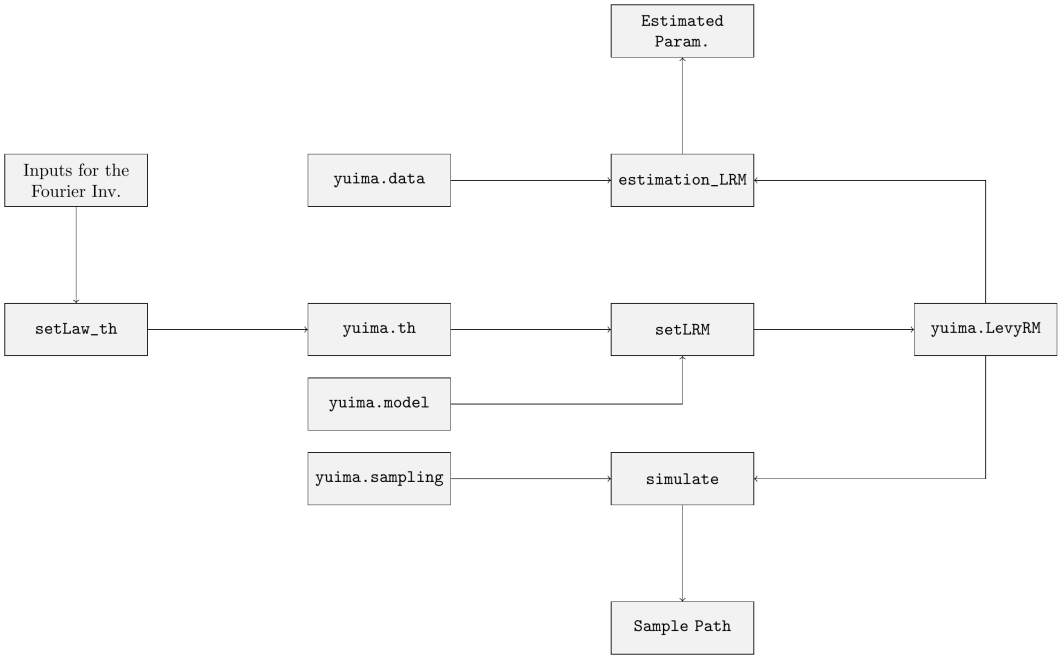

Figure 1: Scheme of classes and methods in yuima for the t-Lévy Regression model.

The last method available in **yuima** is `estimation_LRM`. This method allows the users to estimate the model using either real or simulated data. The estimation follows a two-step procedure, introduced in Masuda et al. (2024).

```
estimation_LRM(start, model, data, upper, lower)
```

For this function, the minimal inputs are `@start`, `@model`, `@data`, `@upper` and `@lower`. The arguments `@start` are the initial points for the optimization routine while `@upper` and `@lower` correspond to the box constraints. The `@yuima.LevyRM-object` is passed to the function through the input `@model` while the input `@data` is used to pass the dataset to the internal optimization routine. Figure 1 describes these new classes and methods, along with their respective usage.

3.1 yuima.th: A new class for mathematical description of a Student-t Lévy process

In this section, the steps for the construction of an `yuima.th-object` are presented. As remarked in Section 3, this object contains all information on the scaled Student-t Lévy process. Moreover, detailed information is provided regarding the numerical algorithms utilized for evaluating the density function (3). The construction of this object is accomplished using the `setLaw_th` constructor, and the subsequent code snippet displays its corresponding arguments.

```
setLaw_th(h = 1, method = "LAG", up = 7, low = -7, N = 180, N_grid = 1000,
  regular_par = NULL)
```

The input `h` is the length of the step-size of each time interval for the Student-t Lévy increment $\Delta J_h = J_h - J_0$. Its default value, `h=1`, indicates that the `yuima.th-object` describes completely the process J_t at time 1. The argument `method` refers to the type of quadrature used for the computation of the integral in (3) while the remaining arguments govern the precision of the integration routine.

The `yuima.th-class` inherits the slots `@rng`, `@density`, `@cdf` and `@quantile` from its parent class `yuima-law` Masuda et al. (2022). These slots store respectively the random number generator, the density function, the cumulative distribution function and the quantile function of J_h.

As mentioned in Section 1, the density function with $h \neq 1$ does not have a closed-form formula and, therefore, the inversion of the characteristic function is necessary. **yuima** provides three methods for this purpose: the Laguerre quadrature, the COS method and the Fast Fourier Transform. The Gauss-Laguerre quadrature is a numerical integration method employed for evaluating integrals in the following form:

$$\mathcal{I} = \int_0^{+\infty} f(x)\, e^{-x} \mathrm{d}x.$$

This procedure has been recently used for the computation of the density of the variance gamma and the transition density of a CARMA model driven by a time-changed Brownian motion (Loregian et al., 2012; Mercuri et al., 2021), this motivates its application in this paper.

Let $f(x)$ be a continuous function defined on $[0, +\infty)$ such that:

$$\mathcal{I} = \int_0^{+\infty} f(x)\, e^{-x} \mathrm{d}x < +\infty;$$

the integral $\mathcal{I}$ can be approximated as follows:

$$\mathcal{I} \approx \sum_{j=1}^{N} \omega(k_j)\, f(k_j), \tag{6}$$

where k_j is the jth-root of the N-order Laguerre polynomial[2] $L_N(x)$ and the weights $\omega(k_j), j = 1, \ldots, N$ are defined as:

$$\omega(k_j) = \frac{k_j}{(N+1)^2 L_{N+1}^2(k_j)}. \tag{7}$$

To apply the approximation in (6), we rewrite the inversion formula in (3) as follows:

$$\begin{aligned} f(x) &= \frac{1}{\pi}\left(\frac{2^{1-\frac{\nu}{2}}}{\Gamma(\nu/2)}\right)^h \int_0^{+\infty} \cos(ux)\, u^{\nu h/2} \left(K_{\nu/2}(u)\right)^h \mathrm{d}u \\ &= \frac{1}{\pi}\left(\frac{2^{1-\frac{\nu}{2}}}{\Gamma(\nu/2)}\right)^h \int_0^{+\infty} \cos(ux)\, u^{\nu h/2} \left(K_{\nu/2}(u)\right)^h e^u e^{-u} \mathrm{d}u. \end{aligned} \tag{8}$$

Applying the result in (6), the density function of J_h can be approximated with the formula reported below:

$$\hat{f}_N(x_j) = \frac{1}{\pi}\left(\frac{2^{1-\frac{\nu}{2}}}{\Gamma(\nu/2)}\right)^h \sum_{j=1}^{N} \cos(k_j x)\, k_j^{\nu h/2} \left(K_{\nu/2}(k_j)\right)^h e^{k_j} \omega(k_j). \tag{9}$$

Notably, the approximation formula's precision in equation (9) can be enhanced through the argument `N` in the `setLaw_th` constructor. The roots k_j and the weights $\omega(k_j)$ are internally computed using the `gauss.quad` function from the R package **statmod** (Smyth et al., 2023; Giner and Smyth, 2016), with a maximum allowed order of 180 for the Laguerre polynomial.

The COS method is based on the Fourier Cosine expansion employed for an even function with the compact domain $[-\pi, \pi]$. This method has been widely applied in the finance literature for the computation of the exercise probability of an option and its no-arbitrage price, we refer to Fang and Oosterlee (2009), Fang and Oosterlee (2011) and references therein for details. Let $g(\theta) : [-\pi, \pi] \to \mathbb{R}$ be an even function, its Fourier Cosine expansion reads:

$$g(\theta) = \frac{1}{2}a_0 + \sum_{k=1}^{\infty} a_k \cos(k\theta) \tag{10}$$

[2]The Laguerre polynomial can be defined recursively as follows: $L_0(x), L_1(x) = 1 - x, \ldots, L_N(x) = \frac{(2N-x)L_{N-1}(x) - (N-1)L_{N-2}(x)}{N}$

with

$$a_k = \frac{2}{\pi}\int_0^{\pi} g(\theta)\cos(k\theta)\,\mathrm{d}\theta.$$

Denoting with $f(x)$ the density function of J_h, the new function $\bar{g}(\theta)$ is defined as:

$$\bar{g}(\theta) := f\left(\frac{L}{\pi}\theta\right)\frac{L}{\pi}\mathbb{1}_{\{-\pi\leq\theta\leq\pi\}}, \tag{11}$$

can be applied. The coefficient a_k is determined as follows:

$$\begin{aligned} a_k &= \frac{2}{\pi}\int_0^{\pi}\bar{g}(\theta)\cos(k\theta)\,\mathrm{d}\theta \\ &= \frac{2}{\pi}\int_0^{\pi} f\left(\frac{L}{\pi}\theta\right)\cos(k\theta)\,\frac{L}{\pi}\mathrm{d}\theta. \end{aligned}$$

Setting $\theta = \frac{\pi}{L}x$, we have:

$$a_k = \frac{2}{\pi}\int_0^{L} f(x)\cos\left(k\frac{\pi}{L}x\right)\mathrm{d}x. \tag{12}$$

The coefficient a_k can be rewritten using the characteristic function of J_h at $k\frac{\pi}{L}$, i.e.:

$$a_k = \frac{2}{\pi}\left[\varphi\left(k\frac{\pi}{L}\right) - \int_L^{+\infty} f(x)\cos\left(k\frac{\pi}{L}x\right)\mathrm{d}x\right]. \tag{13}$$

For a sufficiently large L, the coefficient a_k can be approximated as follows:

$$a_k \approx \frac{2}{\pi}\varphi\left(k\frac{\pi}{L}\right). \tag{14}$$

The following series expansion is achieved:

$$f(x) = \frac{1}{2}\bar{a}_0 + \sum_{k=1}^{+\infty}\bar{a}_k\cos\left(k\frac{\pi}{L}x\right) \tag{15}$$

with

$$\bar{a}_k = a_k\frac{\pi}{L} \approx \frac{2}{L}\varphi\left(k\frac{\pi}{L}\right).$$

Finally, (15) can be approximated by truncating the series as follows:

$$\hat{f}_N(x) = \frac{1}{2}\hat{a}_{0,L} + \sum_{k=1}^{N}\hat{a}_{k,L}\cos\left(k\frac{\pi}{L}x\right) \tag{16}$$

with

$$\hat{a}_{k,L} = \frac{2}{L}\varphi\left(k\frac{\pi}{L}\right).$$

The precision of $\hat{f}_{N,L}(x)$ in (16) depends on N and L. Users can select these two quantities using `N` and the couple `(up, low)` respectively. L is computed internally in the `setLaw_th` constructor as `L = max(|low|,up)`. The last method available in **yuima** for the computation of the density of t-Lévy increments over a time interval with length h is based on the inversion of the characteristic function by means of the Fast Fourier Transform `FFT` Singleton (1969), Cooley and Tukey (1965) [3]. Denoting the density function as $f(x)$ and the characteristic function as $\varphi(u) := \mathbb{E}\left[e^{iuJ_h}\right]$ for the scaled t-Lévy J_h, the density $f(x)$ can be obtained as follows:

[3] Since **yuima** manages SDEs driven by a Lévy process. For a user-defined noise, the random number generator in the simulation algorithm and the density function for the estimation procedure are constructed using the **yuima** function `FromCF2yuima_law` see the documentation YUIMA Project Team (2024) for more details. This function internally implements the same `FFT` algorthim described in this paper.

$$\begin{aligned} f(x) &= \frac{1}{2\pi}\int_{-\infty}^{+\infty} e^{-iux}\mathbb{E}\left[e^{iuJ_h}\right]\mathrm{d}u \\ &= \frac{1}{2\pi}\int_{-\infty}^{+\infty} e^{-iux}\varphi(u)\,\mathrm{d}u. \end{aligned} \tag{17}$$

As first step, we apply to the integral in (17) the change variable $u = 2\pi\omega$ and we have:

$$f(x) = \int_{-\infty}^{+\infty} e^{-i2\pi\omega x}\varphi(2\pi\omega)\,\mathrm{d}\omega. \tag{18}$$

We consider discrete support for x and ω. The x grid has the following structure:

$$x_0 = a, \ldots, x_j = a + j\Delta x, \ldots, x_N = b \tag{19}$$

with $\Delta x = \frac{b-a}{N}$ where $N+1$ is the number of the points in the grid in (19). Similarly, we define a ω grid:

$$\omega_0 = -\frac{N}{2}\Delta\omega, \ldots, \omega_n = -\frac{N}{2} + n\Delta\omega \ldots, \omega_N = \frac{N}{2}\Delta\omega \tag{20}$$

where $\Delta\omega = \frac{1}{N\Delta x} = \frac{1}{b-a}$. Both grids have the same dimension $N+1$, Δx shrinks as $N \to +\infty$ while $\Delta\omega$ reduces for large values of $b-a$. For any x_j in the grid we can approximate the integral in (17) using the left Riemann summation, therefore, we have the approximation $\hat{f}_N(x_j)$:

$$\begin{aligned} \hat{f}_N(x_j) &= \Delta\omega \sum_{n=1}^{N} e^{2i\pi\left(-\frac{N}{2}\Delta\omega + (n-1)\Delta u\right)x_j}\varphi\left(-\pi N\Delta u + 2\pi(n-1)\right) \\ &= \Delta\omega e^{i\pi N\Delta\omega}\sum_{n=1}^{N} e^{-2i\pi(n-1)\Delta u(a+(j-1)\Delta x)}\varphi\left(-\pi N\Delta\omega + 2\pi(n-1)\right) \\ &= \Delta\omega e^{i\pi N\Delta\omega}\sum_{n=1}^{N} e^{\frac{-2i\pi(n-1)(j-1)}{N}}\varphi\left(-\pi N\Delta\omega + 2\pi(n-1)\right)e^{\frac{-2i\pi(n-1)a}{N}}. \end{aligned} \tag{21}$$

The last equality in (21) is due to the identity $\Delta\omega\Delta x = \frac{1}{N}$. To evaluate the summation in (21), we use the `FFT` algorithm and

$$\hat{f}_N(x_j) = \Delta\omega e^{i\pi N\Delta\omega}\texttt{FFT}\left[\varphi\left(-\pi N\Delta\omega + 2\pi(n-1)\right)e^{\frac{-2i\pi(n-1)a}{N}}\right]. \tag{22}$$

In this case, we have two sources of approximation errors that we can control using the arguments `up`, `low` and `N` in the `setLaw_th` constructor. `N` denotes the number of intervals in the grid used for the Left-Riemann summation while `up`, `low` are used to compute the step size of this grid.

Once the density function has been obtained using one of the three methods described above, it is possible to approximate the cumulative distribution function using the Left-Riemann summation computed on the grid in (19), therefore the cumulative distribution function $F(\cdot)$ for each x_j on the grid is determined as follows:

$$\hat{F}(x_j) = \sum_{x_k < x_j}\hat{f}_N(x_k)\,\Delta x. \tag{23}$$

In this way, we can construct a table that we can use internally in the `cdf` and `quantile` functions. Moreover, to evaluate the cumulative distribution function at any $x \in (x_{j-1}, x_j)$, we interpolate linearly its value using the couples $(x_{j-1}, \hat{F}(x_{j-1}))$ and $(x_j, \hat{F}(x_j))$. The random numbers can be obtained using the inversion sampling method.

We conclude this section with a numerical comparison among the three methods im-

plemented in **yuima**. To assess the precision of our code we use as a target the cumulative distribution function of a Student-t with $\nu = 3$ computed through the R function `pt` (R Core Team, 2024). To conduct this comparison, we construct three `yuima.th-objects` as displayed in the code snippet reported below[4].

```
# To instal YUIMA from Github repository
R> library(pak)
R> pak::pkg_install("yuimaproject/yuima")

##########
# Inputs #
##########
R> library(yuima)
R> nu <- 3
R> h <- 1 # step size for the interval time
R> up <- 10
R> low <- -10

# Support definition for variable x
R> x <- seq(low,up,length.out=100001)

# Definition of yuima.th-object
R> law_LAG <- setLaw_th(h = h, method = "LAG", up = up, low = low, N = 180) # Laguerre
R> law_COS <- setLaw_th(h = h, method = "COS", up = up, low = low, N = 180) # COS
R> law_FFT <- setLaw_th(h = h, method = "FFT", up = up, low = low, N = 180) # FFT

# Cumulative Distribution Function: we apply the cdf method to the yuima.th-object
R> Lag_time <- system.time(ycdf_LAG <- cdf(law_LAG, x, list(nu=nu))) # Laguerre
R> COS_time <- system.time(ycdf_COS<-cdf(law_COS, x, list(nu=nu))) # COS
R> FFT_time <- system.time(ycdf_FFT<-cdf(law_FFT, x, list(nu=nu))) # FFT
```

User can specify the numerical methods of the inversion of the characteristic function using the input `method`. This argument assumes three values: `"LAG"` for the Gauss-Laguerre quadrature, `"COS"` for the Cosine Series Expansion and `"FFT"` for the Fast Fourier Transform. Once the `yuima.th-object` has been constructed, its cumulative distribution function is computed by applying the **yuima** method[5] `cdf` that returns the cumulative distribution function for the `numeric` vector `x`. To enable a direct comparison with the outputs of the R functions `dt`, `pt`, `rt`, and `qt`, the **yuima** methods `dens`, `cdf`, `quantile`, and `rand` refer to the increments of a t-Lévy process Z_t, rather than the corresponding scaled t-Lévy process J_t. Specifically, when $t = 1$, the functions `pt` and `cdf` compute the same cumulative distribution function, as illustrated in Figure 2.

For all numerical approaches available for `cdf`, we have a good level of precision that is also confirmed by Table 1. As expected the fastest method is the `FFT` which seems to be also the most precise.

To further investigate this fact, we study the behaviour of the cumulative distribution function when h varies. For $h = 0.01$, we observe an oscillatory behaviour on tails for the `FFT` and `COS` while the Laguerre method seems to be more stable as shown in Figure 3. To have a fair comparison, we set `N = 180`, however we notice that the precision of `FFT` can be drastically improved by tuning the inputs `N`, `up` and `low`.

[4]We recommend to download the latest **yuima** version using the function `pk_install` available in **pak** Csárdi and Hester (2024).

[5]An object of `yuima.th-class` inherits the `dens` method for the density computation, `cdf` for the evaluation of the cumulative distribution function, `quantile` for the quantile function and `rand` for the generation of the random sample. We refer to Masuda et al. (2022) for the usage of these methods.

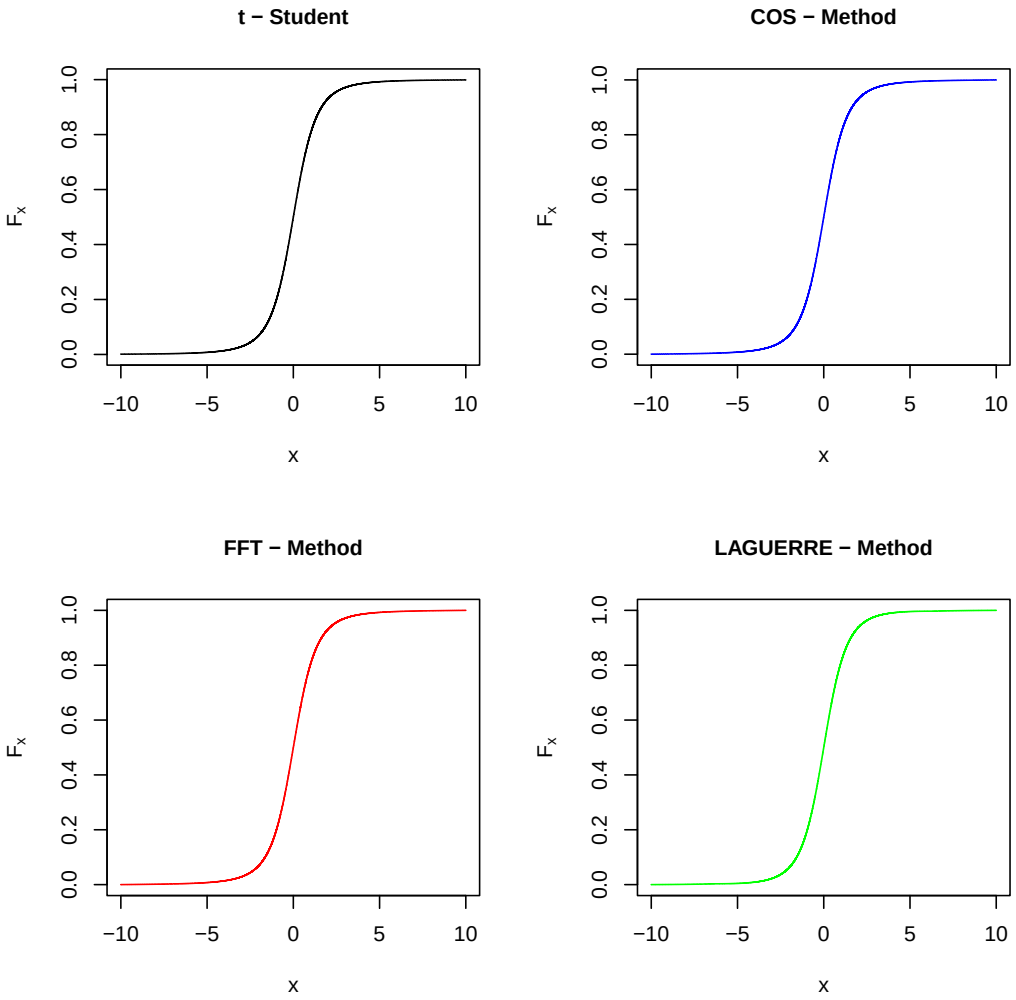

Figure 2: Cumulative distribution function comparison for a Student-t random variable with $\nu = 3$.

Table 1: Summary comparison of the distance between each method available in yuima and the cumulative distribution function obtained using the R function 'pt'.

Metric	COS	FFT	LAG
RMSE	2.10e-02	2.10e-02	0.064000
Max	3.20e-02	3.20e-02	0.085000
Min	7.88e-05	2.18e-05	0.000497
sec.	1.47e+00	4.00e-02	1.200000

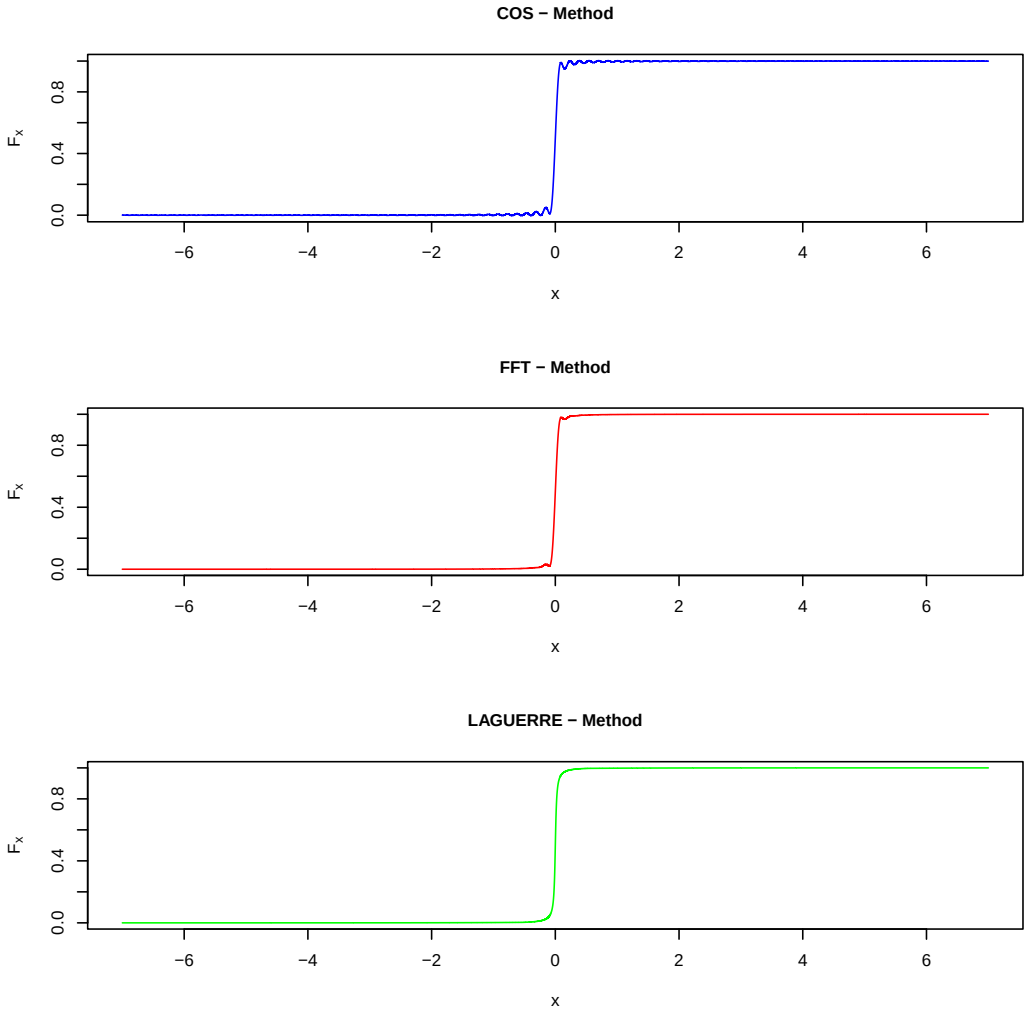

Figure 3: Cumulative distribution function comparison for a Student-t Lévy increment with $\nu = 3$ and $h = 0.01$.

4 Numerical examples

This section presents a series of numerical examples demonstrating the practical application of newly developed classes and methods within the Student-t Regression model. Specifically, we showcase the simulation and estimation of models where the regressors are determined by deterministic functions of time in the first example. The second example introduces integrated stochastic regressors. Lastly, we perform an analysis using real data in the final example.

4.1 Model with Deterministic Regressors

In this example, we consider two deterministic regressors and the dynamics of the model have the following form:

$$Y_t = \mu_1 \cos(5t) + \mu_2 \sin(t) + \sigma J_t \tag{24}$$

with the true values $(\mu_1, \mu_2, \sigma_0, \nu_0) = (5, -1, 3, 3)$. The estimation of the model (24) has been investigated in Masuda et al. (2024) where the empirical distribution of the studentized estimates has been discussed. To use the simulation method in **yuima** we have to write the dynamics of the regressors $X_{1,t} := \cos(5t)$ and $X_{2,t} =:$, that is:

$$\mathrm{d}\begin{bmatrix} X_{1,t} \\ X_{2,t} \end{bmatrix} = \begin{bmatrix} -5\sin(5t) \\ \cos(t) \end{bmatrix} \mathrm{d}t \tag{25}$$

with the initial condition:

$$X_{1,0} = 1,\ X_{2,0} = 0.$$

In the following, we show how to implement this model in **yuima**. In this example, we set $h_n = 1/50$, and thus the number of the observations n over unit time is 50. We also set the terminal time of the whole observations $T_n = 50$ and that of the partial observations $B_n = 15$.

```
R> library(yuima)
##########
# Inputs #
##########
# Inputs for Fourier Inversion
R> method_Fourier <- "FFT"; up <- 6; low <- -6; N <- 2^17; N_grid <- 60000

# Inputs for the sample grid time
R> Factor <- 1
R> Factor1 <- 1
R> initial <- 0; Final_Time <- 50 * Factor; h <- 0.02/Factor1
```

We notice that the variable `Factor1` controls the step size of the time grid. Indeed for different values of this quantity, we have a different value for h for example `Factor1 = 1, 2, 4, 7.2` corresponds `h = 0.02, 0.01, 0.005, 1/365`.

The first step is the definition of an object of `yuima.th-class` using the constructor `setLaw_th`. This object contains the random number generator, the density function, the cumulative distribution function and the quantile function for constructing the increments of a Student-t Lévy process.

```
#####################################
# Example 1: Deterministic Regressors #
#####################################

R> mu1 <- 5; mu2 <- -1; scale <- 3; nu <- 3 # Model Parameters
```

```
# Model Definition
R> law1 <- setLaw_th(method = method_Fourier, up = up, low = low,
  N = N, N_grid = N_grid) # yuima.th

R> class(yuima_law)
[1] "yuima.th"
attr(,"package")
[1] "yuima"
```

The next step is to define the dynamics of the regressors described in (25). This set of differential equations is defined in **yuima** using the standard constructor `setModel`. Once an object containing the mathematical description of the regressors has been defined, we use `setLRM` to obtain the `yuima.LevyRM`. In the following, we report the command lines for the definition of the Student Lévy Regression Model in (24).

```
R> regr1 <- setModel(drift = c("-5*sin(5*t)", "cos(t)"), diffusion = matrix("0",2,1),
     solve.variable = c("X1","X2"), xinit = c(1,0)) # Regressors definition

R> Mod1 <- setLRM(unit_Levy = law1, yuima_regressors = regr1) # t-Regression Model

R> class(Mod1)
[1] "yuima.LevyRM"
attr(,"package")
[1] "yuima"
```

Using the object `Mod1` we simulate a trajectory of the model in (24) using the **yuima** method `simulate`.

```
# Simulation
R> samp <- setSampling(initial, Final_Time, n = Final_Time/h)
R> true.par <- unlist(list(mu1 = mu1, mu2 = mu2, sigma0 = scale, nu = nu))

R> set.seed(1)

R> sim1 <- simulate(Mod1, true.parameter = true.par, sampling = samp)
```

Figure 4 reports the simulated sample paths for the regressors $X_{1,t}$, $X_{2,t}$ and the Student Lévy Regression model Y_t.

The next step is to study the behaviour of the two-step estimation procedure described in Section 2. To run this procedure, we use the method `estimation_LRM` and then we initialize randomly the optimization routine as shown in the following command lines.

```
# Estimation
R> lower <- list(mu1 = -10, mu2 = -10,  sigma0 = 0.1)
R> upper <- list(mu1 = 10, mu2 = 10, sigma0 = 10.01)

R> start <- list(mu1 = runif(1, -10, 10), mu2 = runif(1, -10, 10),
+    sigma0 = runif(1, 0.01, 4))

R> Bn <- 15*Factor

R> est1 <- estimation_LRM(start = start, model = sim1, data = sim1@data,
+    upper = upper, lower = lower, PT = Bn)

R> class(est1)
```

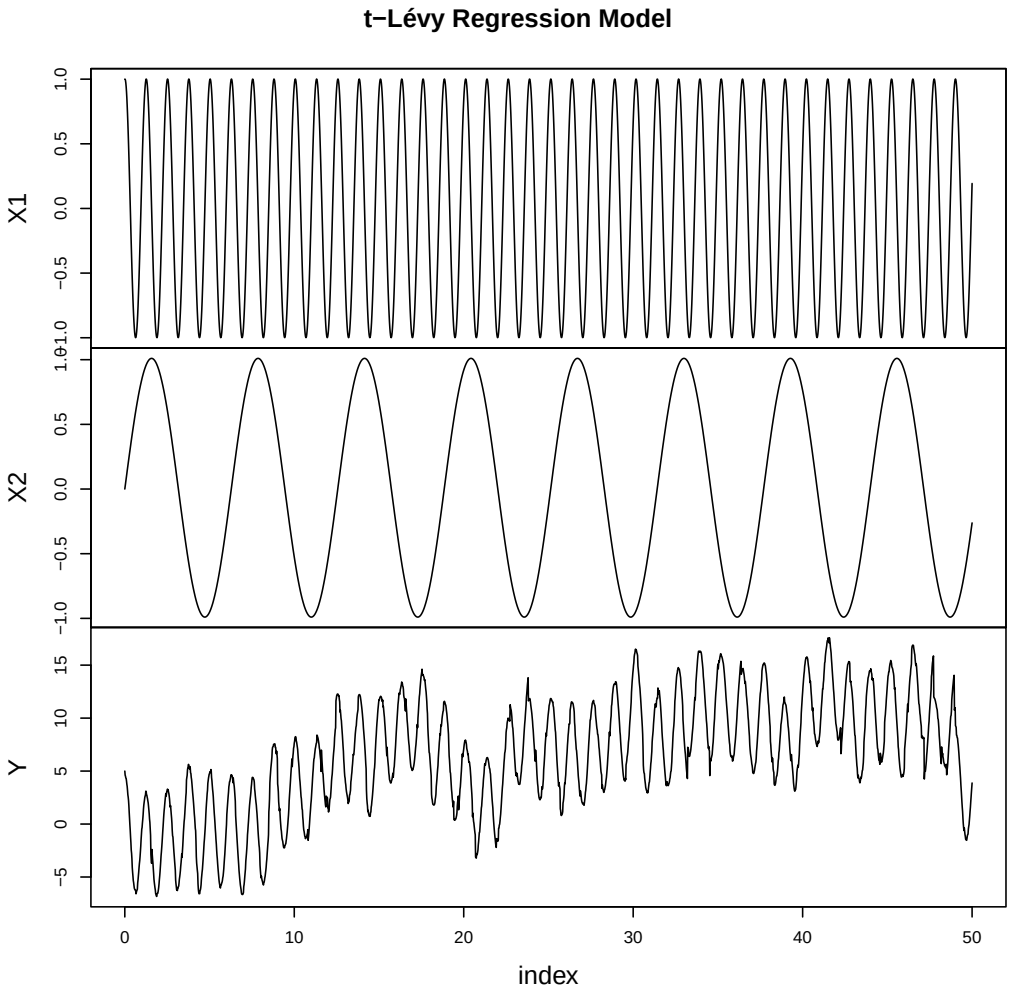

Figure 4: Simulated Trajectory of the Student Lévy Regression model defined in eqrefeq:Regr1.

```
[1] "yuima.qmle"
attr(,"package")
[1] "yuima"

R> summary(est1)
Quasi-Maximum likelihood estimation

Call:
estimation_LRM(start = start, model = sim1, data = sim1@data,
    upper = upper, lower = lower, PT = Bn)

Coefficients:
        Estimate Std. Error
mu1     5.029555 0.03538355
mu2    -1.128905 0.18026088
sigma0  2.425147 0.12523404
nu      2.735344 0.47457435

-2 log L: 3236.535 73.3833
```

It is also possible to construct the dataset without the method `simulate`. This result can be achieved by constructing an object of `yuima.th-class` that represents the increment of the Student-t Lévy process on the time interval with length h.

```
# Dataset construction without YUIMA
R> time_t <- index(get.zoo.data(sim1@data)[[1]])
R> X1 <- zoo(cos(5*time_t), order.by = time_t)
R> X2 <- zoo(sin(time_t), order.by = time_t)
R> law1_h <- setLaw_th(h = h, method = method_Fourier, up = up, low = low,
     N = N, N_grid = N_grid)

R> print(c(law1_h@h,law1@h))
[1] 0.02 1.00
```

The object `law1_h` refers to the Student-t Lévy increment over an interval with length 0.02 as it can be seen looking at the slot `...@h`.

```
R> set.seed(1)
R> names(nu)<- "nu"
# Simulation a t-Levy process Z_t.
R> Z_t <- zoo(cumsum(c(0,rand(law1_h, Final_Time/h,nu))), order.by = time_t)
# Sample path a scaled t-Levy process J_t
R> J_t <- Z_t/sqrt(nu)
# Sample path of a t-Levy regression model
R> Y <- mu1 * X1 + mu2 * X2 + scale * J_t
R> data1_a <- merge(X1, X2, Y)
```

We observe that, as highlighted in Section 3.1, the **yuima** method `rand` returns the increments of a t-Lévy process Z_t. To obtain the trajectory of the scaled t-Lévy process J_t in (24), the process Z_t needs to be rescaled by the factor $\sqrt{\nu}$. The estimation is performed by means of the method `estimation_LRM` as in the previous example.

```
R> est1_a <- estimation_LRM(start = start, model = Mod1, data = setData(data1_a),
+    upper = upper, lower = lower, PT = Bn)
R> summary(est1_a)
Quasi-Maximum likelihood estimation
```

```
Call:
estimation_LRM(start = start, model = Mod1, data = setData(data1_a),
    upper = upper, lower = lower, PT = Bn)

Coefficients:
        Estimate Std. Error
mu1     4.987778 0.03642823
mu2    -1.164138 0.18558422
sigma0  2.495930 0.12888927
nu      2.407683 0.41221590

-2 log L: 3232.784 85.32438
```

Using the result stored in the `summary` we can construct a confidence interval using the command lines reported below.

```
R> info_sum <- summary(est1_a)@coef
R> alpha <- 0.025
R> Confidence_Int_95 <- rbind(info_sum[,1]+info_sum[,2]*qnorm(alpha),
+     info_sum[,1]+info_sum[,2]*qnorm(1-alpha),unlist(true.par),
+     info_sum[,1])
R> rownames(Confidence_Int_95) <- c("LB","UB","True_par","Est_par")
R> print(Confidence_Int_95, digit = 4)
           mu1      mu2  sigma0     nu
LB       4.916  -1.5279   2.243  1.600
UB       5.059  -0.8004   2.749  3.216
True_par 5.000  -1.0000   3.000  3.000
Est_par  4.988  -1.1641   2.496  2.408
```

Based on the aforementioned examples, the estimated value for the σ parameter deviates from its true value. To further investigate this observation, we conducted a comparative analysis using the three integration methods discussed in Section 3.1. This exercise was repeated for three different values of $h = 0.01, 0.005$ and $\frac{1}{365}$. To ensure a fair comparison among the Laguerre, Cosine Series expansion, and Fast Fourier Transform methods, we set the argument `N = 180`. The obtained results are presented in Table 2.

Table 2: Estimated parameters for $h = 0.01, 0.005, 1/365$ and different integration methods. The number of points for the inversion of the characteristic function is $N = 180$. The parenthesis shows the asymptotic standard error. The last row reports the seconds necessary for the simulation of a sample path.

	Laguerre Method			COS Method			FFT Method		
Parameter	Lag_0.01	Lag_0.005	Lag_1_365	COS_0.01	COS_0.005	COS_1_365	FFT_0.01	FFT_0.005	FFT_1_365
$\hat{\mu_1}$	4.968	5.019	4.976	4.939	5.065	4.875	4.934	5.069	4.876
	(0.029)	(0.020)	(0.014)	(0.041)	(0.049)	(0.054)	(0.042)	(0.049)	(0.054)
$\hat{\mu_2}$	-1.065	-1.044	-0.914	-1.192	-1.195	-0.555	-1.179	-1.178	-0.528
	(0.146)	(0.104)	(0.074)	(0.210)	(0.247)	(0.277)	(0.213)	(0.248)	(0.277)
$\hat{\sigma_0}$	2.770	2.794	2.657	3.993	6.654	10.010	4.055	6.674	10.010
	(0.101)	(0.072)	(0.051)	(0.145)	(0.172)	(0.192)	(0.148)	(0.172)	(0.193)
$\hat{\nu}$	3.462	3.281	3.067	2.827	2.223	2.227	5.749	8.310	15.175
	(0.615)	(0.580)	(0.538)	(0.492)	(0.377)	(0.378)	(1.063)	(1.571)	(2.940)
sec.	2.07	2.02	2.07	1.10	1.24	1.34	0.15	0.24	0.39

In the Laguerre method, we observe that reducing the step size h leads to improved estimates of ν, however looking at the asymptotic standard error, the estimates for σ_0 seem to maintain the bias. As for the remaining methods, the estimates exhibit unsatisfactory performance due to the imposed restriction of `N = 180` (that is the maximum value allowed for the Laguerre quadrature in the evaluation of the density in (8)). This outcome is not unexpected, given the Laguerre method's ability to yield a valid distribution even with a

relatively small value of N, as demonstrated in Figure 3. Nevertheless for the COS and the FFT, it is possible to enhance the results by increasing the value of the argument N that yields a more accurate result in the simulation of the sample path for the model described in (24).

Table 3: Estimated parameters for $N = 1000, 5000, 10^4$ in the COS and the FFT methods. The step size h is $1/365$. The parenthesis shows the asymptotic standard error. The last row reports the seconds necessary for the simulation of a sample path.

	COS Method			FFT Method		
Parameter	COS_1000	COS_5000	COS_10000	FFT_1000	FFT_5000	FFT_10000
μ_1	4.968	4.975	4.975	4.968	4.975	4.975
	(0.034)	(0.016)	(0.016)	(0.035)	(0.306)	(0.016)
μ_2	-0.887	-0.909	-0.909	-0.886	-0.908	-0.908
	(0.174)	(0.082)	(0.082)	(0.177)	(0.022)	(0.082)
σ_0	3.300	2.964	2.964	3.363	2.963	2.964
	(0.064)	(0.057)	(0.057)	(0.065)	(0.057)	(0.057)
ν	2.713	2.765	2.765	3.279	2.760	2.702
	(0.470)	(0.480)	(0.480)	(0.579)	(0.479)	(0.468)
sec.	3.49	14.83	33.70	0.38	0.44	0.45

Table 3 shows the estimated parameters for the varying value of N and $h = 1/365$. As expected increasing the precision in the quadrature improved estimates for both methods. Notably for $N \geq 5000$, all estimates fall within the asymptotic confidence interval at the 95% level.

This example suggests a possible strategy for selecting the optimal N for the grid defined in (20), particularly when $h = \frac{1}{365}$ (commonly used for daily data). The strategy consists of the following steps:

- Estimate the model using Laguerre quadrature with the maximum allowable polynomial order (180).
- Using the estimates from the previous step as an initial guess, re-estimate the model using the FFT method with $N = 500$. If the resulting estimates are sufficiently close to those obtained in the previous step, stop.
- Otherwise, using the Laguerre estimates as the initial guess, double the number of intervals in the grid in (20) and re-estimate the model using the FFT. Repeat this process until two successive estimates are sufficiently close. The Euclidean norm can be used to determine whether the estimates are sufficiently close.

4.2 Model with integrated stochastic regressors

In this section, we consider two examples whose regressors are stochastic. Moreover, to satisfy the regularity conditions, the regressors are supposed to be an integrated version of stochastic processes.

Example 1 In this example, we consider the following continuous time regression model with a single regressor:

$$Y_t = \mu \int_0^t X_u du + \sigma J_t, \quad J_1 \sim t_\nu, \tag{26}$$

with the true values $(\mu_0, \sigma_0, \nu_0) = (-3, 3, 2.5)$, and the process X is supposed to be the Lévy driven Ornstein-Uhlenbeck process defined as:

$$dX_t = -X_t dt + 2dZ_t,$$

where the driving noise Z is the Lévy process with $Z_1 \sim \text{NIG}(1, 0, 1, 0)$. The normal inverse Gaussian (NIG) random variable is defined as the normal-mean variance mixture

of the inverse Gaussian random variable, and the probability density function of $Z_t \sim \mathrm{NIG}(\alpha, \beta, \delta t, \mu t)$ is given by

$$x \mapsto \frac{\alpha\delta t \exp\{\delta t\sqrt{\alpha^2-\beta^2} + \beta(x-\mu t)\} K_1(\alpha\psi(x;\delta t,\mu t))}{\pi\psi(x;\delta t,\mu t)}$$

where $\alpha^2 := \gamma^2 + \beta^2$ and $\psi(x;\delta t,\mu t) := \sqrt{(\delta t)^2 + (x-\mu t)^2}$. More detailed theoretical properties of the NIG-Lévy process are given for example in **?**.

For the simulation by **yuima**, we formally introduce the following system:

$$d\begin{bmatrix} X_{1,t} \\ X_{2,t} \end{bmatrix} = \begin{bmatrix} X_{2,t} \\ -X_{2,t} \end{bmatrix} dt + \begin{bmatrix} 0 \\ 2 \end{bmatrix} dZ_t, \tag{27}$$

with the initial condition:

$$X_{1,0} = 0,\ X_{2,0} = 0.$$

The process $X_{1,t}$ corresponds to the regressor $\int_0^t X_u du$ in the regression model (26). Due to the specification of the new **yuima** function, we will additionally construct the "full" model:

$$Y_t = \mu_1 X_{1,t} + \mu_2 X_{2,t} + \sigma J_t, \quad J_1 \sim t_\nu, \tag{28}$$

and simulate the trajectory of Y with $\mu_2 = 0$. After that, we will estimate the parameters (μ_1, σ, ν) by the original model (26). From now on, we show the implementation of this model in **yuima**. The sampling setting is unchanged from the previous example code: $h_n = 1/50, n = 1/h_n = 50, T_n = 50, B_n = 15$.

```
R> library(yuima)
##########
# Inputs #
##########
# Inputs for Fourier Inversion
R> method_Fourier = "FFT"; up = 6; low = -6; N = 2^17; N_grid = 60000

# Inputs for the sample grid time
R> Factor <- 1
R> Factor1 <- 1
R> initial <- 0; Final_Time <- 50 * Factor; h <- 0.02/Factor1
```

The role of the variable `Factor1` is the same as in the previous section. By using the constructor `setLaw_th`, we define `yuima.th-class` in a similar manner. After that, we construct the "full" model (28) by the constructors `setModel` and `setLRM`. A trajectory of Y and its regressors can be simulated by the YUIMA method `simulate`.

```
#####################################################
# Regressor: Integrated NIG-Levy driven OU process #
#####################################################
R> mu1 <- -3; mu2 <- 0;  scale <- 3; nu <- 2.5 # Model Parameters

# Model Definition
R> lawILOU <- setLaw_th(method = method_Fourier, up = up, low = low, N = N,
+    N_grid = N_grid) # yuima.th

R> regrILOU <- setModel(drift = c("X2", "-X2"), jump.coeff = c("0", "2"),
+    solve.variable = c("X1","X2"), xinit = c(0,0), measure.type = "code",
+    measure = list(df = "rNIG(z, 1, 0, 1, 0)")) # Regressors definition

# t-Regression model
```

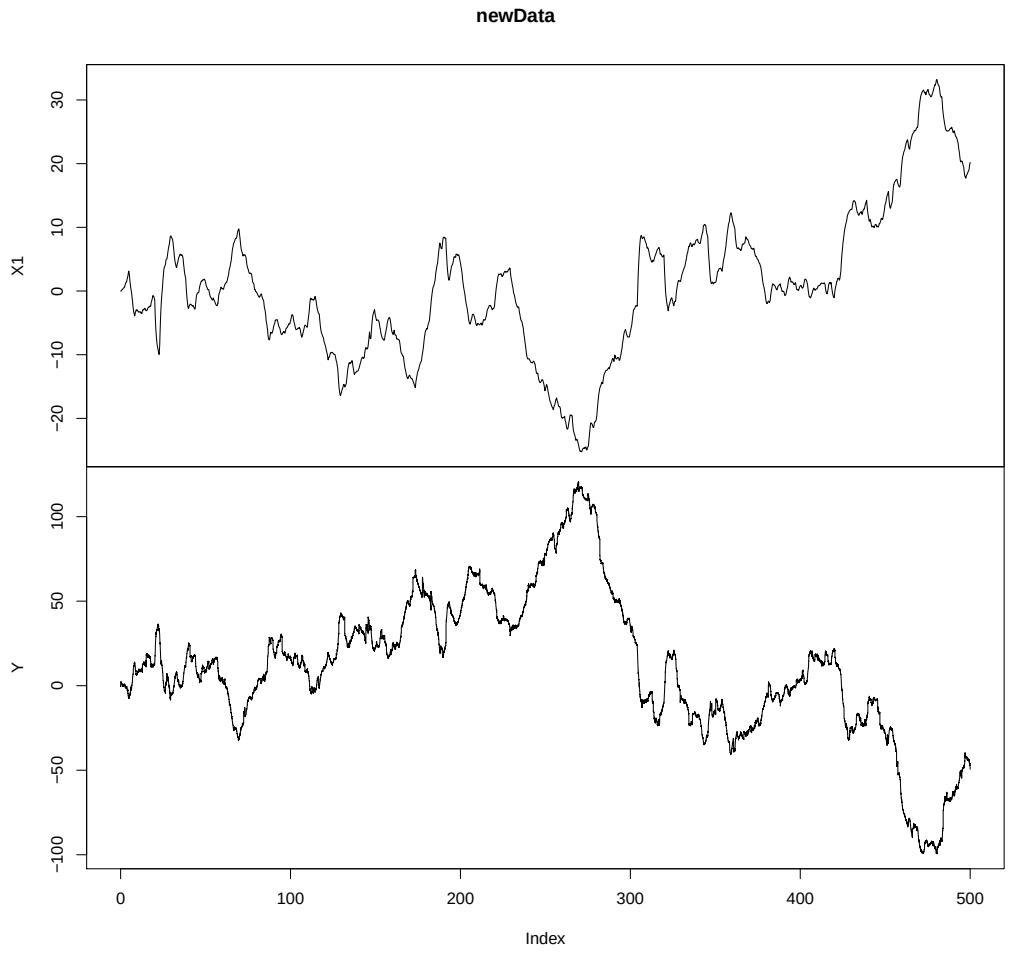

Figure 5: Simulated Trajectory of the Student Lévy Regression model defined in eqrefeq:yuex1

```
R> ModILOU <- setLRM(unit_Levy = lawILOU, yuima_regressors = regrILOU)

# Simulation
R> sampILOU <- setSampling(initial, Final_Time, n = Final_Time/h)
R> true.parILOU <- unlist(list(mu1 = mu1, mu2 = mu2, sigma0 = scale, nu = nu))
R> set.seed(1)
R> simILOU <- simulate(ModILOU, true.parameter = true.parILOU, sampling = sampILOU)
```

Next we extract the trajectories of Y and X_1 from the `yuima.LevyRM-class`: `simILOU` and construct the new model for the estimation of the parameters. Figure 5 shows the simulated sample paths for the regressor and the response process Y.

```
# Data extraction
R> Dataset <- get.zoo.data(simILOU)
R> newData <- Dataset[-2]
R> newData <- merge(newData$X1, newData$Y)
R> colnames(newData) <- c("X1","Y")
R> plot(newData)

# Define the model for estimation
R> regrILOU1 <- setModel(drift = c("0"), diffusion = matrix(c("0"),1,1),
+    solve.variable = c("X1"), xinit = c(0)) # Regressors definition
```

```
# t-Regression model
R> ModILOU1 <- setLRM(unit_Levy = lawILOU, yuima_regressors = regrILOU1)

# Estimation
R> lower1 <- list(mu1 = -10,  sigma0 = 0.1)
R> upper1 <- list(mu1 = 10, sigma0 = 10.01)
R> startILOU1 <- list(mu1 = runif(1, -10, 10), sigma0 = runif(1, 0.01, 4))

R> Bn <- 15*Factor
R> Data1 <- setData(newData)
R> estILOU1 <- estimation_LRM(start = startILOU1, model = ModILOU1, data = Data1,
+    upper = upper1, lower = lower1, PT = Bn)
R> summary(estILOU1)
Quasi-Maximum likelihood estimation

Call:
estimation_LRM(start = startILOU1, model = ModILOU1, data = Data1,
    upper = upper1, lower = lower1, PT = Bn)

Coefficients:
        Estimate Std. Error
mu1    -2.959621 0.02112734
sigma0  2.778659 0.03208519
nu      2.404843 0.13018410

-2 log L: 69711.32 854.3838
```

Example 2 In this example, we consider the following regression model:

$$Y_t = \mu_1 \int_0^t V_{1,u} du + \mu_2 \int_0^t V_{2,u} du + \sigma J_t, \quad J_1 \sim t_\nu \tag{29}$$

where the process $X = (V_1, V_2)$ satisfy

$$dV_{1,t} = \frac{1}{\epsilon}(V_{1,t} - V_{1,t}^3 - V_{2,t} + s)dt, \tag{30}$$

$$dV_{2,t} = (\gamma V_{1,t} - V_{2,t} + \alpha)dt + \sigma' dw_t, \tag{31}$$

with

$$(\epsilon, s, \gamma, \alpha, \sigma') = (1/3, 0, 3/2, 1/2, 2),$$

and standard Wiener process w. We set the true values

$$(\mu_{1,0}, \mu_{2,0}, \sigma_0, \nu_0) = (8, -4, 8, 3).$$

The process V is the so-called stochastic FitzHugh-Nagumo process which is a classical model for describing a neuron; V_1 expresses the membrane potential of the neuron and V_2 represents a recovery variable. Its theoretical properties such as hypoellipticity, and Feller and mixing properties are well summarized in León and Samson (2018). The paper also provides a nonparametric estimator of the invariant density and spike rate. Similarly, other integrated degenerate diffusion can be considered as regressors. Its ergodicity for the regularity conditions is studied for example in Wu (2001). For the statistical inference for degenerate diffusion processes, we refer to Ditlevsen and Samson (2019) and Gloter and Yoshida (2021).

For the implementation of the regression model (29) on YUIMA, we formally consider

the following dynamics:

$$d\begin{bmatrix} V_{1,t} \\ V_{2,t} \\ V_{3,t} \\ V_{4,t} \end{bmatrix} = \begin{bmatrix} V_{3,t} \\ V_{4,t} \\ 3(V_{3,t} - V_{3,t}^3 - V_{4,t}) \\ \frac{3}{2}V_{3,t} - V_{4,t} + \frac{1}{2} \end{bmatrix} dt + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 2 \end{bmatrix} dw_t, \tag{32}$$

with the initial condition:

$$V_{1,0} = 0,\ V_{2,0} = 0,\ V_{3,0} = 0,\ V_{4,0} = 0.$$

The first and second elements correspond to the regressor in (29). As in the previous example, we first simulate data by the "full" model defined as:

$$Y_t = \mu_1 V_{1,t} + \mu_2 V_{2,t} + \mu_3 V_{3,t} + \mu_4 V_{4,t} + \sigma J_t, \quad J_1 \sim t_\nu, \tag{33}$$

with $\mu_3 = \mu_4 = 0$, and after that, we extract the simulated data and estimate the parameters based on the original regression model (29). Below we show how to implement on **yuima**. In this example, we set $h_n = 1/200, n = 1/h_n = 200, T_n = 1000, B_n = 300$. The values of them are controlled by the variables `Factor` and `Factor1` in the example code.

```
R> library(yuima)
##########
# Inputs #
##########
# Inputs for Fourier Inversion
R> method_Fourier = "FFT"; up = 6; low = -6; N = 2^17; N_grid = 60000

# Inputs for the sample grid time
R> Factor <- 20
R> Factor1 <- 4
R> initial <- 0; Final_Time <- 50 * Factor; h <- 0.02/Factor1

#############################################################
# Regressor: Integrated stochastic FitzHugh-Nagumo process #
#############################################################

R> mu1 <- 8; mu2 <- -4; mu3 <- 0; mu4 <- 0;  scale <- 8; nu <- 3 # Model Parameters
# Model Definition
R> lawFN <- setLaw_th(method = method_Fourier, up = up, low = low,
+    N = N, N_grid = N_grid) # yuima.th

R> regrFN <- setModel(drift = c("V3", "V4", "3*(V3-V3^3-V4)", "1.5*V3-V4+0.5"),
+    diffusion = matrix(c("0", "0", "0", "2"),4,1),
+    solve.variable = c("V1","V2", "V3", "V4"),
+    xinit = c(0,0,0,0)) # Regressors definition

R> ModFN <- setLRM(unit_Levy = lawFN, yuima_regressors = regrFN) #  t-regression model

# Simulation
R> sampFN <- setSampling(initial, Final_Time, n = Final_Time/h)
R> true.parFN <- unlist(list(mu1 = mu1, mu2 = mu2, mu3 = mu3, mu4 = mu4,
+    sigma0 = scale, nu = nu))
R> set.seed(12)
R> simFN <- simulate(ModFN, true.parameter = true.parFN, sampling = sampFN)
```

In this example, both `Factor` and `Factor1` are larger than the previous example, and hence it takes a higher computational load than the previous one. Figure 6 shows the

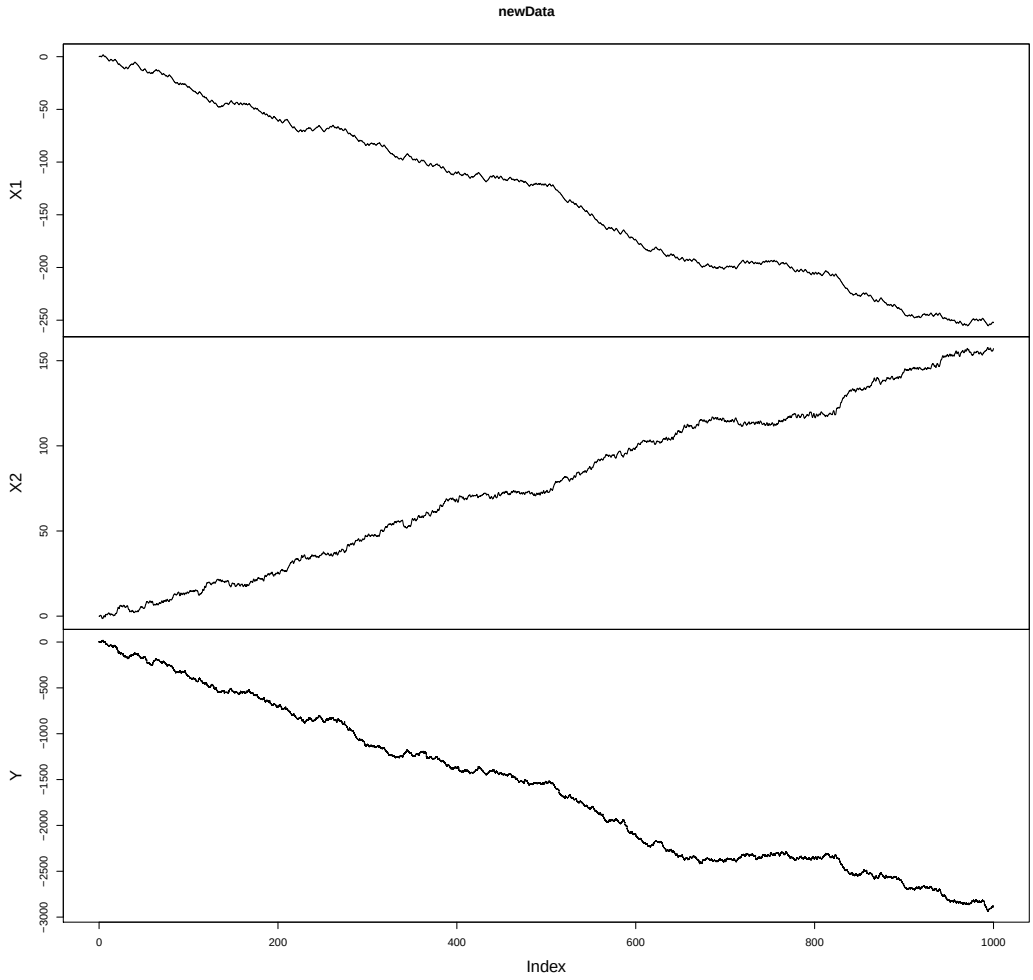

Figure 6: Simulated Trajectory of the Student Lévy Regression model defined in eqrefeq:yuex2.

simulated sample paths for the regressor and the Student Lévy Regression model Y. Now we move on to the estimation phase.

```
R> Dataset <- get.zoo.data(simFN)
R> newData <- Dataset[-c(3,4)]
R> newData <- merge(newData$X1,newData$X2, newData$Y)
R> colnames(newData) <- c("X1","X2", "Y")
R> plot(newData)

# Define the model for estimation
R> regrFN1 <- setModel(drift = c("0", "0"), diffusion = matrix(c("0", "0"),2,1),
+    solve.variable = c("X1", "X2"), xinit = c(0,0)) # Regressors definition
R> ModFN1 <- setLRM(unit_Levy = lawFN, yuima_regressors = regrFN1) # t-Regression model.

# Estimation
R> lower1 <- list(mu1 = -10, mu2 = -10,  sigma0 = 0.1)
R> upper1 <- list(mu1 = 10, mu2 =10, sigma0 = 10.01)
R> startFN1 <- list(mu1 = runif(1, -10, 10), mu2 = runif(1, -10, 10),
+    sigma0 = runif(1, 0.01, 4))

R> Bn <- 15*Factor
R> Data1 <- setData(newData)
```

```
R> estFN1 <- estimation_LRM(start = startFN1, model = ModFN1, data = Data1,
+    upper = upper1, lower = lower1, PT = Bn)
R> summary(estFN1)
Quasi-Maximum likelihood estimation

Call:
estimation_LRM(start = startFN1, model = ModFN1, data = Data1,
    upper = upper1, lower = lower1, PT = Bn)

Coefficients:
        Estimate Std. Error
mu1     8.041838 0.04895791
mu2    -4.041518 0.04495255
sigma0  7.712157 0.04452616
nu      3.020972 0.11837331

-2 log L: 403960.9 1291.516
```

4.3 Real data regressors

In this section, we show how to use the **yuima** package for the estimation of a Student Lévy Regression model in a real dataset. We consider a model where the daily price of the Standard and Poor 500 Index is explained by the VIX index and two currency rates: the YEN-USD and the Euro Usd rates. The dataset was provided by `Yahoo.finance` and ranges from December 14th 2014 to May 12th 2023. We downloaded the data using the function `getSymbol` available in `quantmod` library. To consider a small value for the step size h we estimate our model every month $h = 1/30$. We report below the code for storing the market data in an object of `yuima.data-class`

```
R> library(yuima)
R> library(quantmod)
R> getSymbols("^SPX", from = "2014-12-04", to = "2023-05-12")
# Regressors
R> getSymbols("EURUSD=X", from = "2014-12-04", to = "2023-05-12")
R> getSymbols("VIX", from = "2014-12-04", to = "2023-05-12")
R> getSymbols("JPYUSD=X", from = "2014-12-04", to = "2023-05-12")

R> SP <- zoo(x = SPX$SPX.Close, order.by = index(SPX$SPX.Close))
R> Vix <- zoo(x = VIX$VIX.Close/1000, order.by = index(VIX$VIX.Close))
R> EURUSD <- zoo(x = `EURUSD=X`[,4], order.by = index(`EURUSD=X`))
R> JPYUSD <- zoo(x = `JPYUSD=X`[,4], order.by = index(`JPYUSD=X`))
R> Data <- na.omit(na.approx(merge(Vix, EURUSD, JPYUSD, SP)))
R> colnames(Data) <- c("VIX", "EURUSD", "JPYUSD", "SP")
R> days <- as.numeric(index(Data))-as.numeric(index(Data))[1]

# equally spaced grid time data
R> Data <- zoo(log(Data), order.by = days)
R> Data_eq <- na.approx(Data, xout = days[1] : tail(days,1L))
R> yData <- setData(zoo(Data_eq, order.by = index(Data_eq)/30)) # Data on monthly basis
```

We decided to work with log-price (see `Data` variable in the above code) to have quantities defined on the same support of a Student-t Lévy process, i.e.: the real line. Since the estimation method in **yuima** requires that the data are observed on an equally spaced grid time, we interpolate linearly to estimate possible missing data to get a log price for each day. Figure 7 shows the trajectory for each financial series.

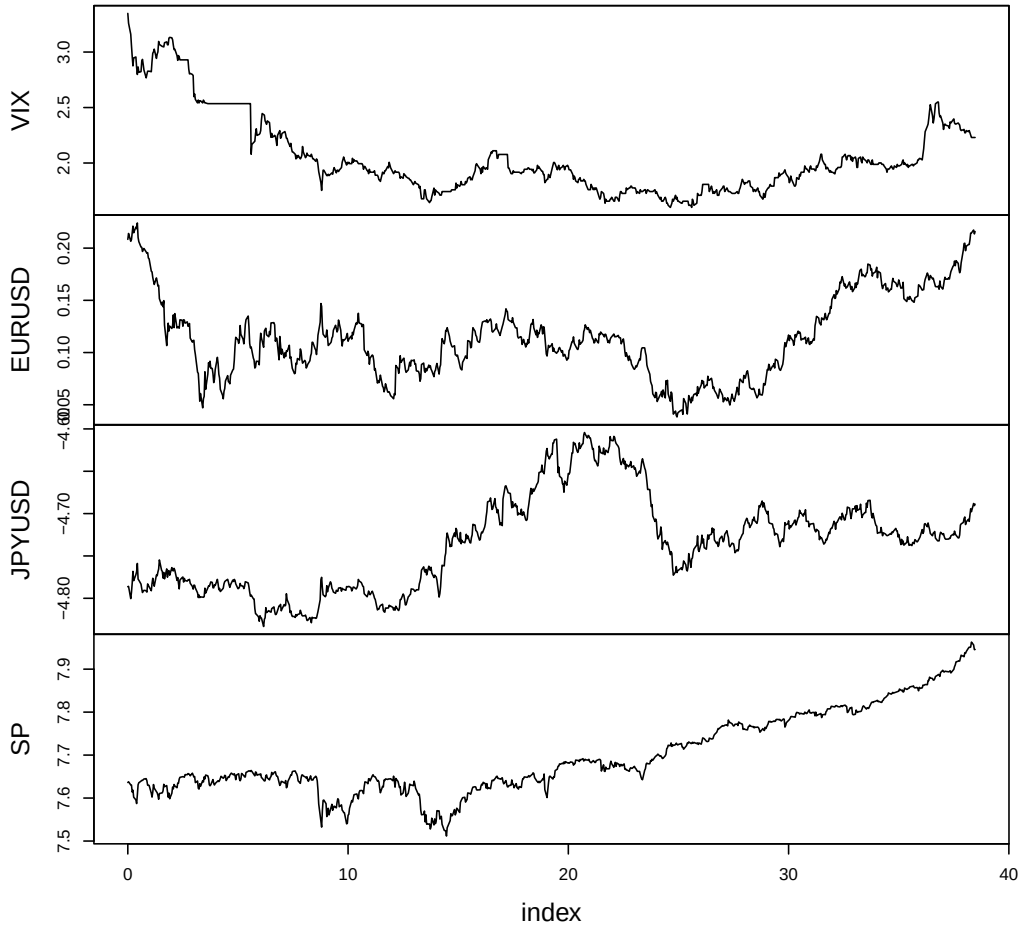

Figure 7: Financial data observed on a monthly equally grid time ranging from December 14th 2014 to May 12th 2023.

To estimate the model we perform the same steps discussed in the previous examples and we report below the code for reproducing our result.

```
# Inputs for integration in the inversion formula
R> method_Fourier <- "FFT"; up <- 6; low <- -6; N <- 10000; N_grid <- 60000

# Model Definition
R> law <- setLaw_th(method = method_Fourier, up = up, low = low, N = N,
+    N_grid = N_grid)
R> regr <- setModel(drift = c("0", "0", "0"), diffusion = matrix("0",3,1),
+    solve.variable = c("VIX", "EURUSD", "JPYUSD"), xinit = c(0,0,0))
Mod <- setLRM(unit_Levy = law, yuima_regressors = regr, LevyRM = "SP")

# Estimation
R> lower <- list(mu1 = -100, mu2 = -200, mu3 = -100,  sigma0 = 0.01)
R> upper <- list(mu1 = 100, mu2 = 100, mu3 = 100, sigma0 = 200.01)
R> start <- list(mu1 = runif(1, -100, 100), mu2 = runif(1, -100, 100),
+    mu3 = runif(1, -100, 100), sigma0 = runif(1, 0.01, 100))
R> est <- estimation_LRM(start = start, model = Mod, data = yData, upper = upper,
+    lower = lower, PT = floor(tail(index(Data_eq)/30,1L)/2))
R> summary(est)

Quasi-Maximum likelihood estimation

Call:
estimation_LRM(start = start, model = Mod, data = yData, upper = upper,
    lower = lower, PT = floor(tail(index(Data_eq)/30, 1L)/2))

Coefficients:
           Estimate  Std. Error
mu1     0.000335328 0.004901559
mu2    -0.062235188 0.033027232
mu3     0.016291220 0.039382559
sigma0  0.082031691 0.004859138
nu      3.062728822 0.616466913

-2 log L: -1506.067 48.17592
```

5 Conclusion

In this paper, we have presented classes and methods for a t-Lévy regression model. In particular, the simulation and the estimation algorithm have been introduced from a computational point of view. Moreover three different algorithms have been developed for computing the density, the cumulative distribution and the quantile functions and the random number generator of the t-Lévy increments defined over a non-unitary interval. These latter methods can be also used in any stochastic model available in **yuima** for the definition of the underlying noise.

Based on our simulated data, for the estimation of the degrees of freedom the Laguerre method seems to be more accurate. However, due to the restriction on the number of roots used in the evaluation of the integral, we notice a bias on the estimation of the scale parameter. Such bias seems to be observable also in the other implemented methods when the same number of points is used in the evaluation of the integral for the inversion of the characteristic function. However for the COS and the FFT methods, the numerical precision can be improved. In particular, it is possible to ensure that the estimates fall in the asymptotic confidence interval at 95% level even if the estimates of the degrees of freedom

in the simplest model (first example) seem to underestimate the true value. As concerns computational time, the FFT methods with a sufficiently large computational precision ($N \geq 5000$) and h sufficiently small seems to be an acceptable compromise.

Acknowledgment

This work is supported by JST CREST Grant Number JPMJCR2115, Japan and by the MUR-PRIN2022 Grant Number 20225PC98R, Italy CUP codes: H53D23002200006 and G53D23001960006.

References

S. Asmussen and J. Rosiński. Approximations of small jumps of lévy processes with a view towards simulation. *Journal of Applied Probability*, 38(2):482–493, 2001. URL http://www.jstor.org/stable/3215901. [p58]

C. Berg and C. Vignat. On some results of Cufaro Petroni about Student t-processes. *Journal of Physics A: Mathematical and Theoretical*, 41(26):265004, 2008. doi: 10.1088/1751-8113/41/26/265004. [p57]

A. Brouste, M. Fukasawa, H. Hino, S. Iacus, K. Kamatani, Y. Koike, H. Masuda, R. Nomura, T. Ogihara, Y. Shimuzu, M. Uchida, and N. Yoshida. The yuima project: A computational framework for simulation and inference of stochastic differential equations. *Journal of Statistical Software*, 57(4):1–51, 2014. doi: 10.18637/jss.v057.i04. URL https://www.jstatsoft.org/index.php/jss/article/view/v057i04. [p56, 59]

L. V. Castro-Porras, R. Rojas-Martínez, C. A. Aguilar-Salinas, O. Y. Bello-Chavolla, C. Becerril-Gutierrez, and C. Escamilla-Nuñez. Trends and age-period-cohort effects on hypertension mortality rates from 1998 to 2018 in Mexico. *Scientific Reports*, 11(1):17553, 2021. URL https://pubmed.ncbi.nlm.nih.gov/34475436/. [p56]

J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90):297–301, 1965. doi: https://doi.org/10.2307/2003354. [p62]

G. Csárdi and J. Hester. *pak: Another Approach to Package Installation*, 2024. URL https://CRAN.R-project.org/package=pak. R package version 0.8.0. [p64]

N. Cufaro Petroni. Mixtures in nonstable Lévy processes. *J. Phys. A*, 40(10):2227–2250, 2007. ISSN 1751-8113. doi: 10.1088/1751-8113/40/10/001. URL http://dx.doi.org/10.1088/1751-8113/40/10/001. [p56, 57]

L. Devroye. On the computer generation of random variables with a given characteristic function. *Comput. Math. Appl.*, 7(6):547–552, 1981. ISSN 0097-4943. doi: 10.1016/0898-1221(81)90038-9. URL http://dx.doi.org/10.1016/0898-1221(81)90038-9. [p58]

S. Ditlevsen and A. Samson. Hypoelliptic diffusions: filtering and inference from complete and partial observations. *J. R. Stat. Soc. Ser. B. Stat. Methodol.*, 81(2):361–384, 2019. ISSN 1369-7412. doi: https://doi.org/10.1111/rssb.12307. [p75]

F. Fang and C. W. Oosterlee. A novel pricing method for European options based on Fourier-cosine series expansions. *SIAM Journal on Scientific Computing*, 31(2):826–848, 2009. doi: https://doi.org/10.1137/080718061. [p61]

F. Fang and C. W. Oosterlee. A Fourier-based valuation method for Bermudan and barrier options under Heston's model. *SIAM Journal on Financial Mathematics*, 2(1):439–463, 2011. doi: https://doi.org/10.1137/100794158. [p61]

G. Giner and G. K. Smyth. statmod: probability calculations for the inverse Gaussian distribution. *R Journal*, 8(1):339–351, 2016. doi: 10.32614/RJ-2016-024. URL https://doi.org/10.32614/RJ-2016-024. [p61]

A. Gloter and N. Yoshida. Adaptive estimation for degenerate diffusion processes. *Electron. J. Stat.*, 15(1):1424–1472, 2021. doi: 10.1214/20-ejs1777. URL https://doi.org/10.1214/20-ejs1777. [p75]

S. Haberman and A. Renshaw. On age-period-cohort parametric mortality rate projections. *Insurance: Mathematics and Economics*, 45(2):255–270, 2009. ISSN 0167-6687. doi: https://doi.org/10.1016/j.insmatheco.2009.07.006. URL https://www.sciencedirect.com/science/article/pii/S016766870900081X. [p56]

C. C. Heyde and N. N. Leonenko. Student processes. *Adv. in Appl. Probab.*, 37(2):342–365, 2005. ISSN 0001-8678. doi: 10.1239/aap/1118858629. URL http://dx.doi.org/10.1239/aap/1118858629. [p56, 57]

F. Hubalek. On simulation from the marginal distribution of a Student t and generalized hyperbolic lévy process. 2005. URL https://api.semanticscholar.org/CorpusID:19037661. [p58]

S. M. Iacus and N. Yoshida. Simulation and inference for stochastic processes with yuima. *A comprehensive R framework for SDEs and other stochastic processes. Use R*, 2018. URL https://link.springer.com/book/10.1007/978-3-319-55569-0. [p56]

J. R. León and A. Samson. Hypoelliptic stochastic FitzHugh-Nagumo neuronal model: mixing, up-crossing and estimation of the spike rate. *Ann. Appl. Probab.*, 28(4):2243–2274, 2018. ISSN 1050-5164. doi: 10.1214/17-AAP1355. URL https://doi.org/10.1214/17-AAP1355. [p75]

A. Loregian, L. Mercuri, and E. Rroji. Approximation of the variance gamma model with a finite mixture of normals. *Statistics & Probability Letters*, 82(2):217–224, 2012. doi: https://doi.org/10.1016/j.spl.2011.10.004. URL https://www.sciencedirect.com/science/article/pii/S0167715211003257. [p61]

T. Massing. Simulation of Student–lévy processes using series representations. *Computational Statistics*, 33(4):1649–1685, 2018. doi: 10.1007/s00180-018-0814-y. [p58]

H. Masuda, L. Mercuri, and Y. Uehara. Noise inference for ergodic Lévy driven SDE. *Electronic Journal of Statistics*, 16(1):2432–2474, 2022. doi: 10.1214/22-EJS2006. URL https://doi.org/10.1214/22-EJS2006. [p60, 64]

H. Masuda, L. Mercuri, and Y. Uehara. Quasi-Likelihood Analysis for Student-Lévy Regression. *Statistical Inference for Stochastic Processes*, pages 1–34, 2024. doi: https://doi.org/10.1007/s11203-024-09317-2. [p56, 57, 58, 59, 60, 67]

L. Mercuri, A. Perchiazzo, and E. Rroji. Finite mixture approximation of CARMA (p, q) models. *SIAM Journal on Financial Mathematics*, 12(4):1416–1458, 2021. doi: https://doi.org/10.1137/20M1363248. [p61]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2024. URL https://www.R-project.org/. [p64]

R. Singleton. An algorithm for computing the mixed radix fast Fourier transform. *IEEE Transactions on audio and electroacoustics*, 17(2):93–103, 1969. URL https://ieeexplore.ieee.org/document/1162042. [p62]

G. Smyth, L. Chen, Y. Hu, P. Dunn, B. Phipson, and Y. Chen. *statmod: Statistical Modeling*, 2023. URL https://cran.r-project.org/web/packages/statmod/index.html. R package version 1.5.0. [p61]

C. Sørensen. Modeling seasonality in agricultural commodity futures. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 22(5):393–426, 2002. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/fut.10017. [p56]

L. Wu. Large and moderate deviations and exponential convergence for stochastic damping Hamiltonian systems. *Stochastic Process. Appl.*, 91(2):205–238, 2001. ISSN 0304-4149. doi: 10.1016/S0304-4149(00)00061-2. URL https://doi.org/10.1016/S0304-4149(00)00061-2. [p75]

YUIMA Project Team. *yuima: The YUIMA Project Package for SDEs*, 2024. URL https://cran.r-project.org/web/packages/yuima/index.html. R package version 1.15.27. [p56, 62]

Hiroki Masuda
The University of Tokyo
Graduate School of Mathematical Sciences,
3-8-1 Komaba Meguro-ku Tokyo 153-8914, Japan
https://www.ms.u-tokyo.ac.jp/teacher_e/masuda_e.html
ORCiD: 0000-0002-5553-9201
hmasuda@ms.u-tokyo.ac.jp

Lorenzo Mercuri
University of Milan
Department of Economics, Management and Quantitative Methods
Via Del Conservatorio Milano 7, Italy.
https://www.unimi.it/en/ugov/person/lorenzo-mercuri
ORCiD: 0000-0003-0490-2952
lorenzo.mercuri@unimi.it

Yuma Uehara
Kansai University
Department of Mathematics, Faculty of Engineering Science, 3-3-35 Yamate-cho Suita-shi Osaka 564-8680, Japan
https://kugakujo.kansai-u.ac.jp/html/100000728_en.html
ORCiD: 0000-0002-7084-0457
y-uehara@kansai-u.ac.j