Implementation of Non-Gaussian Motion Models Within Stone Soup

Zhen Yuen Chong
Department of Engineering
University of Cambridge
Cambridge, UK

Henry Pritchett
Cyber and Information Systems
Defence Science and Technology Laboratory
Salisbury, UK

Qing Li, Runze Gan, Yaman Kındap
Department of Engineering
University of Cambridge
Cambridge, UK

Simon Godsill
Department of Engineering
University of Cambridge
Cambridge, UK

***Abstract*—In recent years, state-space models for highly manoeuvrable objects have been proposed based on non-Gaussian, continuous time, jump-based Lévy processes, the so-called Lévy state-space model [1]–[4]. In these models, the standard Brownian motion driving process for continuous time processes is replaced with a heavy-tailed non-Gaussian alternative. This retains all the flexibility of its Gaussian counterpart in terms of possible dynamical model structures and operations with irregular time stamps or heterogeneous data sources. These models aim to operate in areas such as surveillance of irregularly moving drones or people, and tracking wildlife or biological data. Implementation is relatively straightforward since the Kalman filters of the Brownian motion case can be replaced in the non-Gaussian case by *mixtures* of Kalman filters within a *marginalised* particle filtering framework [5]. While the Stone Soup tracking software environment includes both Kalman filtering and generic particle filtering, it does not currently allow the combination of these tasks within a marginalised particle filtering framework. We discuss the significant challenges involved in incorporating these models and algorithms into Stone Soup, and present initial simulation results for the new software.**

***Index Terms*—non-Gaussian stochastic process, sequential Monte Carlo, non-linear filtering, particle filter, Lévy process, stochastic differential equation.**

I. Introduction

The Stone Soup framework [6] is a robust platform dedicated to the development and testing of advanced tracking algorithms. Through the adoption of Object-Oriented Programming (OOP) principles, it achieves a modular design that allows for interchangeable components, akin to the modularity found in machinery parts. This facilitates the seamless integration of various algorithms within a structured framework, including those for fusion, tracking, and sensor management. OOP principles, such as inheritance, permit Stone Soup to enhance its functionalities while keeping the code base accessible and neat, essential for encouraging collaboration and innovation within the tracking and state estimation community.

Most dynamic models in tracking applications, such as the linear Gaussian model emphasised in [7], [8], rely on analytical tractability. Presently, Stone Soup extensively supports systems modelled by standard Brownian motion driving processes. However, these models' assumption of normally distributed state transitions becomes problematic when dealing with abrupt and random changes, e.g. when an object is manoeuvring or attempting to avoid localisation. Here, the Lévy state-space model [1] is implemented in Stone Soup, as a generic framework for extending linear Gaussian models into the non-Gaussian and heavy-tailed domain. Our implementations are more general than previous tracking implementations [2], which were limited to the canonical α-stable class of models. Here, we incorporate additional non-Gaussian Lévy processes including the normal Gamma (NG), normal tempered-stable (NTS), and other combinations as yet unexplored within the tracking paradigm. Moreover, the proposed implementations include the possibility of skewness in the dynamical models, a feature not explored in [2], which may be of use in learning the intent and long-term behaviour of an object over time.

For inference purposes, the so-called *marginalised* or Rao-Blackwellised particle filter method [5] is first implemented, a methodology that should find general application in Stone Soup for tracking models that are not fully non-linear and Gaussian. Here, the focus is on a marginalised particle filter implementation of the Lévy process models discussed above, leading to a complete implementation of this rich class of models within Stone Soup's environment [9]. All codes presented herein are available in a forked repository of Stone Soup and should be used as a reference until official pull requests have been made and merged into the official repository. Note that, the implementation described here is not final and may be subject to future changes.

The structure of this paper is as follows. Sections II and III provide a summary of the theoretical foundations and algorithmic details for the marginalised particle filter and the Lévy state-space model, respectively. Section IV elaborates on the design choices and challenges involved in incorporating the marginalised particle filter and the Lévy state-space model into Stone Soup. Section V provides examples demonstrating the application of the proposed components via Stone Soup's existing display functionalities. The paper concludes

with Section VI and highlights areas for further research and improvement.

II. Marginalised Particle Filtering

A key element not yet incorporated into Stone Soup is the implementation of a *marginalised*, or Rao-Blackwellised, particle filter (see [5] Section II.G), implemented in the conditionally linear Gaussian model case. Introducing this functionality will significantly enhance the range of models explored within Stone Soup. The underlying idea is that in many applications, a portion of the state vector can be well-modelled as linear and Gaussian, conditional upon the remaining non-linear (and possibly non-Gaussian) states. Particle filtering is thus confined to the non-linear states, with the linear components being updated in a closed form via Kalman filtering. This approach generally results in enhanced performance due to the consequent reduction in the dimensionality of the problem.

The marginalised particle filtering approach has become fairly standard, as detailed in the literature [5], [10], [11], complete with fully specified equations and procedural steps. An overview is offered here, given that the intricacies of this approach form a crucial aspect of its implementation in Stone Soup. The general framework partitions the state vector into linear and non-linear parts, denoted by $\boldsymbol{x}^L(t)$ and $\boldsymbol{x}^N(t)$ respectively. The linear Gaussian component is specified accordingly over the time interval δ_t by

$$\begin{aligned} \boldsymbol{x}^L(t+\delta_t) &= \mathbf{F}(\delta_t)\,\boldsymbol{x}^L(t) + \boldsymbol{m}(\delta_t) + \boldsymbol{u}^L(t+\delta_t)\,, \\ \boldsymbol{y}(t) &= \mathbf{C}\boldsymbol{x}^L(t) + \boldsymbol{v}^L(t)\,, \end{aligned} \tag{1}$$

where $\boldsymbol{u}^L, \boldsymbol{v}^L$ are independent, zero-mean Gaussian noise with covariances $\mathbf{Q}(\delta_t)$ and $\mathbf{R}$ respectively. $\mathbf{F}(\delta_t), \mathbf{C}$ are the transition and measurement matrices respectively, while $\boldsymbol{m}(\delta_t)$ is a mean offset in the transition equation. All five terms may depend on the non-linear state $\boldsymbol{x}^N(t)$ and time.

Now, suppose that observations arrive at increasing times t_n, where $n = 0, 1, \ldots$, and define $\boldsymbol{x}_n := \boldsymbol{x}(t_n)$, $\boldsymbol{y}_n := \boldsymbol{y}(t_n)$ etc. At $n = 0$, the conditional prior $\boldsymbol{x}_0^L \sim \mathcal{N}(\boldsymbol{\mu}_0(\boldsymbol{x}_0^N), \mathbf{P}_0(\boldsymbol{x}_0^N))$ is assumed.

The non-linear states follow a more general dynamical model, though typically not necessarily Markovian,

$$\boldsymbol{x}_n^N \sim f\left(\boldsymbol{x}_n^N \mid \boldsymbol{x}_{0:n-1}^N\right), \quad \boldsymbol{x}_0^N \sim \pi_0\left(\boldsymbol{x}_0^N\right). \tag{2}$$

The linear part $\boldsymbol{x}_{0:n}^L$ is marginalised using the Kalman filter and algorithms such as sequential Monte Carlo may be applied directly in this marginal space with the target distribution $\pi_{0:n|0:n}\left(\boldsymbol{x}_{0:n}^N \mid \boldsymbol{y}_{0:n}\right)$. The standard Bayesian filtering recursions are then applied as prediction/correction steps,

$$\begin{aligned} &\pi_{0:n|0:n-1}\left(\boldsymbol{x}_{0:n}^N \mid \boldsymbol{y}_{0:n-1}\right) \\ &\quad = \pi_{0:n-1|0:n-1}\left(\boldsymbol{x}_{0:n-1}^N \mid \boldsymbol{y}_{0:n-1}\right) f\left(\boldsymbol{x}_n^N \mid \boldsymbol{x}_{0:n-1}^N\right), \\ &\pi_{0:n|0:n}\left(\boldsymbol{x}_{0:n}^N \mid \boldsymbol{y}_{0:n}\right) \\ &\quad \propto p\left(\boldsymbol{y}_n \mid \boldsymbol{y}_{0:n-1}, \boldsymbol{x}_{0:n}^N\right) \pi_{0:n|0:n-1}\left(\boldsymbol{x}_{0:n}^N \mid \boldsymbol{y}_{0:n-1}\right). \end{aligned} \tag{3}$$

Note in particular that the marginal model is *non-Markovian*, so the required likelihood term above satisfies $p\left(\boldsymbol{y}_n \mid \boldsymbol{y}_{0:n-1}, \boldsymbol{x}_{0:n}^N\right) \neq p\left(\boldsymbol{y}_n \mid \boldsymbol{x}_n^N\right)$. It is obtained through the Prediction Error Decomposition of the Kalman filter as

$$p\left(\boldsymbol{y}_n \mid \boldsymbol{y}_{0:n-1}, \boldsymbol{x}_{0:n}^N\right) = \mathcal{N}\left(\boldsymbol{y}_n; \boldsymbol{\mu}_{\boldsymbol{y}_n}, \mathbf{P}_{\boldsymbol{y}_n}\right), \tag{4}$$

where

$$\boldsymbol{\mu}_{\boldsymbol{y}_n} = \mathbf{C}\boldsymbol{\mu}_{n|0:n-1}\,, \quad \mathbf{P}_{\boldsymbol{y}_n} = \mathbf{C}\mathbf{P}_{n|0:n-1}\mathbf{C}^T + \mathbf{R}\,. \tag{5}$$

The terms required in this formula are obtained from the standard Kalman filtering recursions

$$\begin{aligned} \boldsymbol{\mu}_{n|0:n-1} &= \mathbf{F}\boldsymbol{\mu}_{n-1|0:n-1} + \boldsymbol{m}_n\,, \\ \mathbf{P}_{n|0:n-1} &= \mathbf{F}\mathbf{P}_{n-1|0:n-1}\mathbf{F}^T + \mathbf{Q}\,, \\ \boldsymbol{\mu}_{n|0:n} &= \boldsymbol{\mu}_{n|0:n-1} + \mathbf{K}_n\left(\boldsymbol{y}_n - \mathbf{C}\boldsymbol{\mu}_{n|0:n-1}\right), \\ \mathbf{P}_{n|0:n} &= \left(\mathbf{I} - \mathbf{K}_n\mathbf{C}\right)\mathbf{P}_{n|0:n-1}\,, \\ \mathbf{K}_n &= \mathbf{P}_{n|0:n-1}\mathbf{C}^T\left(\mathbf{C}\mathbf{P}_{n|0:n-1}\mathbf{C}^T + \mathbf{R}\right)^{-1}. \end{aligned} \tag{6}$$

For notational convenience, the dependence of the terms $\boldsymbol{m}, \mathbf{F}, \mathbf{C}, \mathbf{Q}, \mathbf{R}$ on the non-linear states have been omitted, e.g. $\mathbf{F}(\delta_t) = \mathbf{F}$.

For each Monte Carlo particle, the marginalised particle filter computes $p\left(\boldsymbol{y}_n \mid \boldsymbol{y}_{0:n-1}, \tilde{\boldsymbol{x}}_{0:n}^{N,(k)}\right)$ and stores the associated sufficient statistics, i.e. $\boldsymbol{\mu}_{n|0:n}^{(k)}$, $\mathbf{P}_{n|0:n}^{(k)}$, and the self-normalised weights $\omega_n^{(k)}$,

$$\omega_n^{(k)} = \frac{\tilde{\omega}_n^{(k)}}{\sum_{i=1}^{N_\mathcal{K}} \tilde{\omega}_n^{(i)}}, \quad k = 1, \ldots, N_\mathcal{K}\,, \tag{7}$$

where

$$\tilde{\omega}_n^{(k)} = \omega_{n-1}^{(k)} \frac{p\left(\boldsymbol{y}_n \mid \boldsymbol{y}_{0:n-1}, \tilde{\boldsymbol{x}}_n^{N,(k)}\right) f\left(\tilde{\boldsymbol{x}}_n^{N,(k)} \mid \tilde{\boldsymbol{x}}_{0:n-1}^{N,(k)}\right)}{q_n\left(\tilde{\boldsymbol{x}}_n^{N,(k)} \mid \tilde{\boldsymbol{x}}_{0:n-1}^{N,(k)}, \boldsymbol{y}_{0:n}\right)}. \tag{8}$$

In the bootstrap filter, the state transition density $f(\cdot \mid \cdot)$ is used as the proposal density $q_n(\cdot \mid \cdot)$, simplifying (8) to

$$\tilde{\omega}_n^{(k)} = \omega_{n-1}^{(k)}\, p\left(\boldsymbol{y}_n \mid \boldsymbol{y}_{0:n-1}, \tilde{\boldsymbol{x}}_n^{N,(k)}\right). \tag{9}$$

Resampling is then carried out in the standard way, as required. Finally, the posterior density can be obtained at each time step from the filtered non-linear state samples $\left\{\tilde{\boldsymbol{x}}_{0:n}^{N,(k)}\right\}$ as a Rao-Blackwellised Monte Carlo approximation,

$$\pi_{n|0:n}\left(\boldsymbol{x}_n^L \mid \boldsymbol{y}_{0:n}\right) \approx \sum_{k=1}^{N_\mathcal{K}} \omega_n^{(k)} p\left(\boldsymbol{x}_n^L \mid \tilde{\boldsymbol{x}}_{0:n}^{N,(k)}, \boldsymbol{y}_{0:n}\right), \tag{10}$$

which corresponds to a weighted Gaussian mixture approximation whose mean and covariance may be computed using the standard formulas.

An implementation of the marginalised particle filter requires the specification of the model matrices $\mathbf{F}(\cdot)$, $\mathbf{C}(\cdot)$, $\mathbf{Q}(\cdot)$, $\mathbf{R}(\cdot)$ and $\boldsymbol{m}(\cdot)$ which may all be functions of the non-linear state $\boldsymbol{x}_n^N$, and the non-linear state transition density $f(\cdot \mid \cdot)$. In the next Section, the specific modelling choices for the Lévy state-space model are summarised. In this case, the non-linear states correspond to the latent pairs, $\{\Gamma_i, V_i\}$, which

TABLE I
SUMMARY OF MODEL PARAMETERS. $\theta > 0$ IS THE DAMPING COEFFICIENT AND $\eta > 0$ SETS THE STRENGTH OF THE RESTORATION FORCE. IN THE EQUILIBRIUM REVERTING MODEL (ERV), ρ CORRESPONDS TO THE TARGET'S DESTINATION POSITION [2].

Models	$\boldsymbol{x}$	$\mathbf{A}$	$\boldsymbol{b}$	$\boldsymbol{h}$	$\boldsymbol{g}$
Langevin	$\begin{bmatrix} x \\ \dot{x} \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 0 & -\theta \end{bmatrix}$	$\mathbf{0}$	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$	-
Singer	$\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -\theta \end{bmatrix}$	$\mathbf{0}$	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	-
ERV	$\begin{bmatrix} x \\ \dot{x} \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ -\eta & -\theta \end{bmatrix}$	$\eta \begin{bmatrix} 0 \\ \rho \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$	-
Latent destination	$\begin{bmatrix} x \\ \dot{x} \\ l \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ -\eta & -\theta & \eta \\ 0 & 0 & 0 \end{bmatrix}$	$\mathbf{0}$	$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

are proposed from their prior distribution in a Bootstrap-style marginalised particle filter, with weight updates as in (9).

III. THE LÉVY STATE-SPACE MODEL

The specific form of the Lévy process dynamical model here is summarised, based on results in [1], [2], [4], [9], [12], [13]. The model is a continuous-time linear system, as represented by the stochastic differential equation (SDE) [2], [14]

$$d\boldsymbol{x}(t) = \mathbf{A}\boldsymbol{x}(t)dt + \boldsymbol{b}dt + \mathbf{H}d\boldsymbol{W}(t) + \mathbf{G}d\boldsymbol{\beta}(t)\,, \tag{11}$$

where $\boldsymbol{b}$ is a known input vector used for destination-aware models, $\{\boldsymbol{\beta}(t)\}$ is a Brownian motion that drives Gaussian disturbances such as latent destinations, and $\{\boldsymbol{W}(t)\}$ is a Lévy process that drives the non-Gaussian behaviour of the object's motion. The goal is to implement a model in M spatial dimensions (e.g. $M = 2$ for planar motion). For now, however, consider just a single spatial dimension, with subsequent explanations on scaling to M dimensions.

The general structure of the model allows a wide range of linear model structures to be incorporated [15], see Table I for the model classes implemented here, as in [2].

The solution to (11) is obtained via stochastic integration,

$$\boldsymbol{x}(t+\delta_t) = \boldsymbol{F}(\delta_t)\boldsymbol{x}(t) + \boldsymbol{\zeta}(\delta_t) + \boldsymbol{I}_\beta(\delta_t) + \boldsymbol{I}_W(\delta_t)\,, \tag{12}$$

where

$$\begin{aligned} \boldsymbol{F}(\delta_t) &= e^{\boldsymbol{A}\delta_t}, \quad \boldsymbol{\zeta}(\delta_t) = \int_0^{\delta_t} e^{\boldsymbol{A}(\delta_t-u)}\boldsymbol{b}du\,, \\ \boldsymbol{I}_\beta(\delta_t) &= \int_0^{\delta_t} e^{\boldsymbol{A}(\delta_t-u)}\boldsymbol{g}d\beta(u) := \int_0^{\delta_t} \mathbf{g}(\delta_t,u)d\beta(u)\,, \\ \boldsymbol{I}_W(\delta_t) &= \int_0^{\delta_t} e^{\boldsymbol{A}(\delta_t-u)}\boldsymbol{h}dW(u) := \int_0^{\delta_t} \mathbf{f}(\delta_t,u)dW(u)\,. \end{aligned} \tag{13}$$

The first three terms in (13) can be computed explicitly. Note that $\boldsymbol{I}_\beta(\delta_t)$ is a Brownian-driven stochastic integral and has a closed-form solution [16],

$$p\left(\boldsymbol{I}_\beta(\delta_t)\right) = \mathcal{N}\left(0,\, \mathbf{S}_\beta(\delta_t)\right)\,, \tag{14}$$

where σ_β^2 is the Brownian motion variance and

$$\mathbf{S}_\beta(\delta_t) = \sigma_\beta^2 \int_0^{\delta_t} \mathbf{g}(\delta_t,u)\mathbf{g}(\delta_t,u)^\top du\,. \tag{15}$$

A closed-form solution does not, however, exist for the Lévy stochastic integral, $\boldsymbol{I}_W(\delta_t)$. Instead, a generalised shot-noise representation [17] is used to solve $\boldsymbol{I}_W(\delta_t)$.

The driving Lévy processes considered here can be broadly categorised into two major classes, namely the Normal σ-Mean (NσM) and the Normal Variance-Mean (NVM) mixture process, which include a wide range of heavy-tailed possibilities including the α-stable, NG and NTS processes [4], [18].

A. Generalised Shot Noise Representation of the Lévy Integral

The generalised shot noise formulation of the Lévy stochastic integral [17], [18], defined on a time axis $\tau \in [0, T]$, is

$$\boldsymbol{I}_W(\delta_t) = \sum_{i=1}^{\infty} H\left(\Gamma_i,\, U_i\right)\mathbf{f}\left(\delta_t, V_i\right) - \boldsymbol{q}_{i,\delta_t}\,, \tag{16}$$

where:

- $\{\Gamma_i\}$ are event times of a Poisson process with rate δ_t/T, or equivalently a unit rate Poisson process whose epochs are multiplied by T/δ_t,
- $\{V_i\} \overset{\text{i.i.d.}}{\sim} \mathcal{U}(0, \delta_t)$,
- $\{U_i\} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1)$,
- $\boldsymbol{q}_{i,\delta_t}$ is a centring (compensator) term required for series convergence only in certain cases of the NσM model.

$H(\gamma, u)$, which is non-increasing in γ, determines the type of Lévy process being modelled. The two generic choices are the NσM and the NVM mixture processes, given by

$$H\left(\Gamma_i,\, U_i\right) = \begin{cases} Z_i\mu_W + Z_i\sigma_W U_i, & \text{N}\sigma\text{M}\,, \\ Z_i\mu_W + \sqrt{Z_i}\sigma_W U_i, & \text{NVM}\,, \end{cases} \tag{17}$$

where $Z_i = h(\Gamma_i)$ is a specific non-increasing function that determines which Lévy process applies. In the α-stable case, for example, the NσM form is used with

$$h(\gamma) = \gamma^{-1/\alpha}, \quad 0 < \alpha < 2. \tag{18}$$

The NTS and NG processes are obtained using the NVM form and rejection sampling to generate the samples $h(\Gamma_i)$ indirectly, see [3], [9], [18] for full details. In general the form of $h(\cdot)$ required is defined by the Lévy measure of the jump process Z, defined as

$$Q_Z(dz) := \begin{cases} \alpha z^{-1-\alpha}dz, & \alpha\text{-stable}\,, \\ \alpha z^{-1-\alpha}\exp\left(-\beta z\right)dz, & \text{tempered-stable}\,, \\ \nu z^{-1}\exp\left(-\beta z\right)dz, & \text{Gamma}\,. \end{cases} \tag{19}$$

In practical implementations, the infinite series is approximated by truncating to a finite number of terms as

$$\boldsymbol{I}_W^c(\delta_t) = \sum_{i:\Gamma_i \leqslant c} \left[H\left(\Gamma_i,\, U_i\right)\mathbf{f}\left(\delta_t, V_i\right)\right] - \bar{\boldsymbol{q}}_{\delta_t}^c\,, \tag{20}$$

where the centring term $\bar{\boldsymbol{q}}^c_{\delta_t}$ may be directly computed as $\mathbb{E}\left[\sum_{i:\Gamma_i \leqslant c} H\left(\Gamma_i,\, U_i\right) \mathbf{f}\left(\delta_t, V_i\right)\right]$ when the expectations exist,

$$\bar{\boldsymbol{q}}^c_{\delta_t} = \mu_W \frac{\delta_t}{T} \mathbb{E}\left[h(\Gamma_i)\right] \mathbb{E}\left[\mathbf{f}\left(\delta_t, V_i\right)\right] , \tag{21}$$

where in the α-stable case with $1 < \alpha < 2$, $\mathbb{E}\left[h(\Gamma_i)\right] = \frac{\alpha}{\alpha-1} c^{1-\frac{1}{\alpha}}$ [1]. In all other cases considered in this work, no centring term is required and thus $\bar{\boldsymbol{q}}^c_{\delta_t} = 0$.

Given the form of $\boldsymbol{I}^c_W$ in (17) and (20), the truncated series representation of the stochastic integral is Gaussian when conditioned on the latent variable pairs corresponding to the time interval t to $t+\delta_t$, denoted $\{\Gamma_i, V_i\}^c_{\delta_t}$, i.e.

$$p\left(\boldsymbol{I}^c_W(\delta_t) \mid \{\Gamma_i, V_i\}^c_{\delta_t}\right) = \mathcal{N}\left(\mu_W \boldsymbol{m}^c_W(\delta_t),\, \sigma^2_W \mathbf{S}^c_W(\delta_t)\right) , \tag{22}$$

where

$$\begin{aligned} \boldsymbol{m}^c_W(\delta_t) &= \sum_{i:\Gamma_i \leqslant c} \left[h\left(\Gamma_i\right) \mathbf{f}\left(\delta_t, V_i\right)\right] - \bar{\boldsymbol{q}}^c_{\delta_t} , \\ \mathbf{S}^c_W(\delta_t) &= \begin{cases} \displaystyle\sum_{i:\Gamma_i \leqslant c} h\left(\Gamma_i\right)^2 \mathbf{f}\left(\delta_t, V_i\right) \mathbf{f}\left(\delta_t, V_i\right)^\top , & \text{N}\sigma\text{M} , \\ \displaystyle\sum_{i:\Gamma_i \leqslant c} h\left(\Gamma_i\right) \mathbf{f}\left(\delta_t, V_i\right) \mathbf{f}\left(\delta_t, V_i\right)^\top , & \text{NVM} . \end{cases} \end{aligned} \tag{23}$$

Finally, the error $\boldsymbol{r}_c(\delta_t)$ due to the finite truncation of the series representation is approximated as

$$\boldsymbol{r}_c(\delta_t) = \boldsymbol{I}_W(\delta_t) - \boldsymbol{I}^c_W(\delta_t) . \tag{24}$$

The results of [1], [2], [4] show that a Gaussian approximation to $\boldsymbol{r}_c(\delta_t)$ is appropriate as c becomes large for most cases of the NσM and NVM models (excluding the normal Gamma case, which is non-Gaussian). Its distribution may be characterised approximately as

$$\boldsymbol{r}_c(\delta_t) \approx \hat{\boldsymbol{r}}_c(\delta_t) \sim \mathcal{N}\left(\boldsymbol{\mu}^c_{\boldsymbol{r}}, \mathbf{C}^c_{\boldsymbol{r}}\right) . \tag{25}$$

The mean is zero in the compensated case for the α-stable ($1 < \alpha < 2$) and computed in other cases as

$$\boldsymbol{\mu}^c_{\boldsymbol{r}} = \mu_W M^{(1)}_{Z_\epsilon} \frac{1}{T} \int_0^{\delta_t} \mathbf{f}(\delta_t, u) du .$$

The covariance is given by

$$\boldsymbol{C}^c_{\boldsymbol{r}} = \begin{cases} \left(\mu^2_W + \sigma^2_W\right) M^{(2)}_{Z_\epsilon} \boldsymbol{\Psi}(\delta_t), & \text{N}\sigma\text{M} , \\ \left(\mu^2_W M^{(2)}_{Z_\epsilon} + \sigma^2_W M^{(1)}_{Z_\epsilon}\right) \boldsymbol{\Psi}(\delta_t), & \text{NVM} , \end{cases} \tag{26}$$

where

$$\boldsymbol{\Psi}(\delta_t) = \frac{1}{T} \int_0^{\delta_t} \mathbf{f}\left(\delta_t, u\right) \mathbf{f}\left(\delta_t, u\right)^\top du . \tag{27}$$

Here, $M^{(n)}_{Z_\epsilon}$ corresponds to the n-th moment of the underlying subordinator process that generates the jumps Z_i in (17),

$$M^{(n)}_{Z_\epsilon} = \int_0^{h(c)} z^n Q_Z(dz), \quad n \geqslant 1 \tag{28}$$

where Q_Z is the Lévy measure of $Z(t)$ in (19). These are all finite except for the $n = 1$ case of the compensated stable form with $1 < \alpha < 2$.

Algorithm 1 Single iteration of the marginalised particle filter

Input: $\{\Gamma_i, V_i\}^{c,(k)}_{0:n-1}$, $\omega^{(k)}_{n-1}$, $\{\boldsymbol{\mu}, \mathbf{P}\}^{(k)}_{n-1|0:n-1}$, $\boldsymbol{y}_n$
Output: $\{\Gamma_i, V_i\}^{c,(k)}_{0:n}$, $\omega^{(k)}_n$, $\{\boldsymbol{\mu}, \mathbf{P}\}^{(k)}_{n|0:n}$

```
if Resampling then
    Resample particles
    Set weights ω^(k)_{n-1} = 1/N_K
for k = 1, ..., N_K do
    Sample {Γ_i, V_i}^{c,(k)}_{δ_t}
    Assign {Γ_i, V_i}^{c,(k)}_{0:n} = {Γ_i, V_i}^{c,(k)}_{δ_t} ∪ {Γ_i, V_i}^{c,(k)}_{0:n-1}
    Compute {μ, P}^(k)_{n|0:n} with y_n via (6)
    Compute self-normalised weights ω^(k)_n via (7)
```

B. Conditionally Gaussian State-Space Form

The various components of the model, including the Gaussian residual approximation, may now be put together in the form of (1) to be implemented with the marginalised particle filter,

$$\boldsymbol{x}(t+\delta_t) \approx \boldsymbol{F}(\delta_t)\boldsymbol{x}(t) + \boldsymbol{\zeta}(\delta_t) + \boldsymbol{I}_\beta(\delta_t) + \boldsymbol{I}^c_W(\delta_t) + \hat{\boldsymbol{r}}^c(\delta_t) . \tag{29}$$

Hence, from the conditionally Gaussian form of all of the random terms which are independent of one another,

$$\begin{aligned} &p\left(\boldsymbol{x}(t+\delta_t) \mid \boldsymbol{x}(t), \{\Gamma_i, V_i\}^c_{\delta_t}\right) \\ &\qquad = \mathcal{N}(\boldsymbol{F}(\delta_t)\boldsymbol{x}(t) + \boldsymbol{m}(\delta_t), \mathbf{Q}(\delta_t)) , \end{aligned} \tag{30}$$

where

$$\begin{aligned} \boldsymbol{m}(\delta_t) &= \boldsymbol{\zeta}(\delta_t) + \mu_W \boldsymbol{m}^c_W(\delta_t) - \bar{\boldsymbol{q}}^c_{\delta_t} + \boldsymbol{\mu}^c_{\boldsymbol{r}} , \\ \mathbf{Q}(\delta_t) &= \mathbf{S}_\beta(\delta_t) + \sigma^2_W \mathbf{S}^c_W(\delta_t) + \mathbf{C}^c_{\boldsymbol{r}} . \end{aligned} \tag{31}$$

Finally, the linear state in the discrete-time state-space model (1) is identified as $\boldsymbol{x}^L_n := \boldsymbol{x}(t_n)$ and the non-linear state as the collection of latent jump variables in the time interval t to $t+\delta_t$, denoted as $\boldsymbol{x}^N((t, t+\delta_t]) := \{\Gamma_i, V_i\}^c_{\delta_t}$. With the model now fully specified, the pseudo-code for a single iteration of the marginalised particle filter is presented in Algorithm 1.

C. Multidimensional Filtering

Thus far, the models are specified for a tracking entity in one dimension. For spatial tracking in two or more dimensions, this is straightforwardly achieved for the models here. The latent variables $\{\Gamma_i, V_i\}^c_{\delta_t}$ are treated as common to all spatial dimensions, since they may be interpreted as small or large changes in the dynamics of objects at random times, which would not happen independently across the spatial dimensions. However, conditional on the latent variables, independent linear dynamics according to (31) are generated for each spatial dimension. For each particle at time t_n, the incremental likelihoods (4) are independently computed for each spatial dimension. These are then combined via multiplication to give the overall weight update required by the particle filter in (9). A similar scheme was proposed in [2].

IV. Implementation in Stone Soup

The multidimensional dynamical models introduced here are coordinate-uncoupled Gaussian models when conditioned

on $\{\Gamma_i, V_i\}_{\delta_t}^{c}$. Hence, $H(\cdot)$ from the shot noise representation of the driving Lévy process $\boldsymbol{W}(t) \in \mathbb{R}^M$ can be written as

$$H\left(\Gamma_i, \boldsymbol{U}_i\right) = \begin{cases} Z_i \boldsymbol{\mu}_W + Z_i \boldsymbol{\Sigma}_W^{1/2} \boldsymbol{U}_i, & \text{N}\sigma\text{M}\,, \\ Z_i \boldsymbol{\mu}_W + \sqrt{Z_i} \boldsymbol{\Sigma}_W^{1/2} \boldsymbol{U}_i, & \text{NVM}\,, \end{cases} \tag{32}$$

where $\boldsymbol{U}_i \sim \mathcal{N}(0, \mathbf{I}) \in \mathbb{R}^M$ is a vector of independent standard Brownian motion, and $\boldsymbol{\Sigma}_W$ is the positive definite, diagonal covariance matrix. The models defined herein extend the works of [2] where only isotropic α-stable Lévy process models are supported. This allows for asymmetric Lévy driving processes for simulating biased trajectories, e.g. financial returns where positively skewed distributions are commonly found [16].

A. Driver Class and Sub-classes

To extend support for models driven by Lévy processes, a new `Driver` base class has been introduced as presented in Fig. 1. This class of transition models decouples the deterministic and non-deterministic elements, thereby allowing for the specification of the noise process driving any given model.

The sub-classes, namely `NonGaussianDriver` and `GaussianDriver` derive from the `Driver` class. The `NonGaussianDriver` defines a series of additional methods for the computation of the series representation of the required stochastic integral in (20), mainly:

- `latents(...)` samples the latent pairs $\{\Gamma_i, V_i\}_{\delta_t}^{c}$.
- `hfunc(...)` specifies the function $h(\cdot)$ in (18).

In practice, any driver, i.e. `AlphaStable` that inherits from either `GaussianDriver` or `NonGaussianDriver` are initialised independently and subsequently injected into the chosen model as a dependency. Upon injection, the model is bounded to the driver instance via the `register_model(...)` method. By utilising dependency injection, the selection of the driving noise process is rendered flexible, eliminating the necessity to define a new class for every model and noise driver pair.

An added advantage of this design is the flexibility it offers in terms of having either independent or shared latent variables across coordinate axes. For axes with independently sampled latents, the trajectory of the moving target typically exhibits significant jumps parallel with either axis, which may be useful in certain applications. In such instances, a new driver must be instantiated for each axis. More commonly though, the latents are shared, i.e. the same latents are generated once and shared across all axes. This is achieved by injecting the same driver instance into every transition model representing each axis, a process facilitated by maintaining a list of models, `models`, bound to the driver instance. Among these, one model, by default the first bound model, is designated as the `master`. This model is responsible for generating and caching the required latents, with subsequent requests for latents by non-master models returning the cached values. For simplicity, this cache is implemented through a pair of attributes, `_Gammas` and `_Vvs`, corresponding to $\{\Gamma_i, V_i\}_{\delta_t}^{c}$.

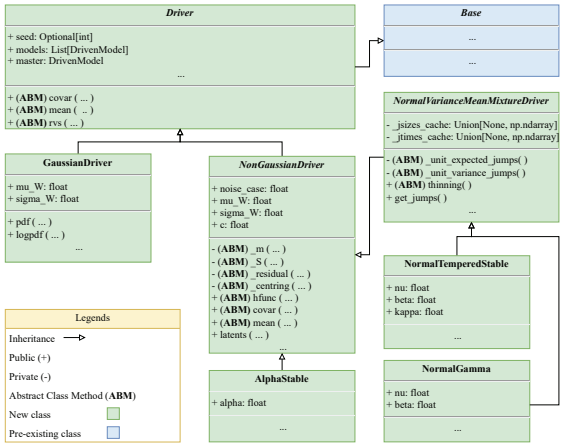


Fig. 1. Simplified class diagram for the `Driver` component and sub-classes showing their key attributes and methods.

Distinct from the `AlphaStable` driver that directly inherits from `NonGaussianDriver`, the NVM processes involve an additional level of abstraction via the `NormalVarianceMeanMixture`, providing the additional methods necessary for rejection sampling (as detailed in [9]). For example,

- `thinning(...)` computes the thinning probabilities.
- `get_jumps(...)` performs the rejection sampling and caches the subordinator jump sizes and times.
- `_unit_expected_jumps(...)` computes the first moment of the jump process (28).

A pair of additional attributes `_jsizes_cache` and `_jtimes_cache` is introduced to cache the subordinator jump sizes and times respectively post-thinning. This is done to support Lévy processes with conditionally non-zero mean Gaussian noise, and thus using the same subordinator jumps for separate invocations to both `mean(...)` and `covar(...)`.

B. DrivenModel Class and Sub-classes

Following the aforementioned modifications, a new abstract base class, `DrivenModel`, as presented in Fig. 2, is introduced to be inherited by all transition models that depend on externally injected driver components. To accommodate models that utilise both Gaussian and non-Gaussian processes, such as the latent destination model [2], the design choice was made for these models to accept either or both Gaussian and non-Gaussian noise drivers. In alignment with Stone Soup's framework, the `CombinedDrivenTransitionModel` has been designed to perform iterations through the transition models representing each coordinate axis, generating the necessary noise samples. Since the sharing of the latents pairs $\{\Gamma_i, V_i\}_{\delta_t}^{c}$ are managed by the driver component as outlined in Section IV-A, this eliminates the need for additional complexity in defining the transition model classes.

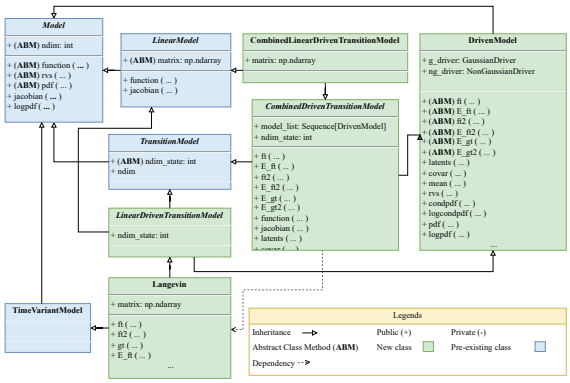


Fig. 2. Simplified class diagram for the `DrivenModel` component and sub-classes showing their key attributes and methods. The Langevin model is shown as an example.

Fig. 3. Simplified class diagram for the Rao-Blackwellised particle filter implementation showing the key attributes and methods of the classes involved.

Finally, the transition model class (e.g. `Langevin`) to be instantiated implements the model-specific abstract methods, mainly:

- `ft(...)` returns $\mathbf{f}(\delta_t, V_i)$ in the truncated series representation of the stochastic integral in (20).
- `E_ft(...)` returns the expectation of $\mathbf{f}(\delta_t, V_1)$ in the centring term in (21).

C. Marginalised or Rao-Blackwellised Particle Filter

Stone Soup currently implements a version of the Rao-Blackwellised particle filter (RBPF) which applies only to discrete state-space transition models. Hence, it is not appropriate for our (conditionally) linear, Gaussian context. Stone Soup currently also offers particle filters designed for Gaussian models, where a singular covariance matrix is shared among all particles. However, this approach does not extend naturally to the RBPF required here.

Therefore, our RBPF is introduced into Stone Soup as a new set of classes that inherits from the existing `ParticleState` class, named `RBParticleState`, as presented in Fig. 3 In the proposed RBPF, each particle possesses a unique covariance matrix, conditioned on $\{\Gamma_i, V_i\}_{\delta_t}^{c,(k)}$. Beyond the `state_vector` attribute, the `RBParticleState` defines the following:

- `covariance` represents a $M \times M \times N_{\mathcal{K}}$ tensor, with the first two dimensions denoting the covariance matrix for each particle k conditioned on $\{\Gamma_i, V_i\}_{\delta_t}^{c,(k)}$.
- `covar(...)` returns the covariance of the marginalised linear posterior.

The prediction and correction steps are carried out by the `RBParticlePredictor` and `RBParticleUpdater` classes respectively. Specifically, the `predict(...)` and `update(...)` methods are overridden to include the Kalman Filtering step in (6). Furthermore, to accelerate the RBPF algorithm, the Kalman filtering recursions of each particle can be executed in parallel by operating on the mean and covariance tensors directly.

V. Example Outputs from Stone Soup

A. The α-stable Langevin Model

The effectiveness of the new implementation is first demonstrated using the α-stable Langevin model to simulate synthetic data. The α-stable distribution here is parameterised by the α-stable law $S_\alpha(\sigma, \tilde{\beta}, \mu)$ with $\mu = 0$. The stability index α serves as a measure of the tail thickness of the distribution. A smaller α indicates heavier tails, implying a higher probability of extreme events. The scale parameter σ_W influences the scale of the jumps. The effect of varying α on the trajectories is depicted in Fig. 4. At $\alpha = 1.4$, significant large jumps occur. As α increases from 1 to 2, the process becomes less volatile until it reaches the Gaussian limit (Brownian motion) at $\alpha = 2$.

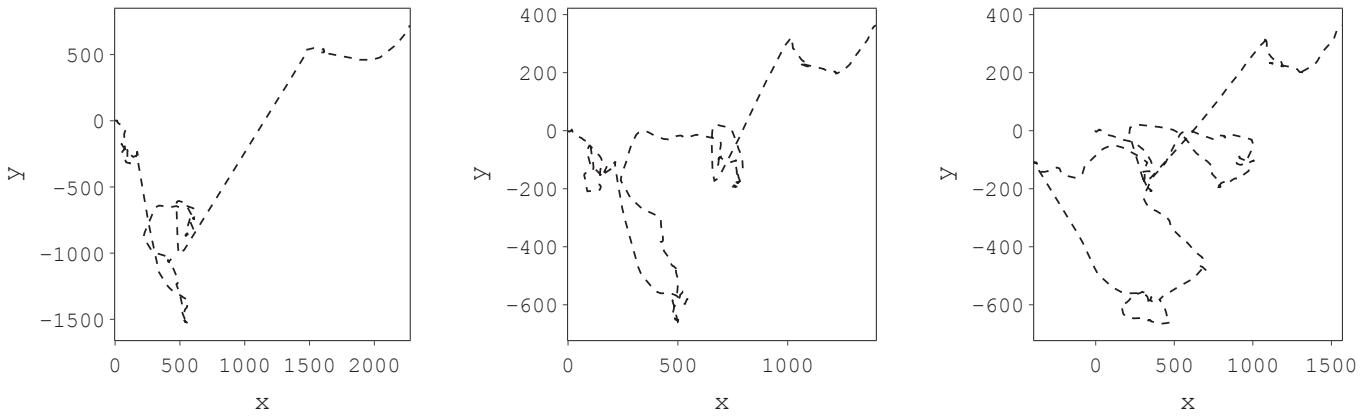
In the case of an asymmetric α-stable Lévy process, a positive skew is obtained by setting $\boldsymbol{\mu}_W > 0$, which may be advantageous for modelling motions biased towards any arbitrary direction. When $\boldsymbol{\mu}_W = \mathbf{1} \cdot \mu_W$, where μ_W is a scalar, jumps occur diagonally due to the uniform means in both coordinate axes. Modifying the individual means of each coordinate axis allows for the jump direction to be biased as desired. For instance, with $\boldsymbol{\mu}_W = [2, 0]^\top$, jumps are biased parallel to the horizontal axis, see Fig. 4c. The skewness parameter is given by [19]

$$\tilde{\beta}_m = \frac{\mathbb{E}\left[|U_m|^\alpha \operatorname{sign} U_m\right]}{\mathbb{E}\,|U_m|^\alpha}, \quad U_m \sim \mathcal{N}(\mu_{W,m}, \sigma_{W,m}^2)\,, \quad (33)$$

in the α-stable case, and for the other Lévy processes considered here, $\mu_{W,m}$ similarly controls the skewness [20].

B. The Rao-Blackwellised Particle Filter

In line with the tutorial examples for the Kalman and particle filters in Stone Soup, a simplified, single-target tracking without clutter example is provided, using the same model from Fig. 4b. It is assumed that a linear sensor corrupted by white noise detects the position $\boldsymbol{y}_n = \mathbf{C}_n \boldsymbol{x}_n + \boldsymbol{v}_n$, such that



(a) $\alpha = 1.4$, $\boldsymbol{\mu}_W = \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix}$ (b) $\alpha = 1.8$, $\boldsymbol{\mu}_W = \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix}$ (c) $\alpha = 1.8$, $\boldsymbol{\mu}_W = \begin{bmatrix} 2.0 \\ 0.0 \end{bmatrix}$

Fig. 4. Simulated planar tracks using α-stable Langevin model with parameters $\theta = 0.15, c = 10$, and $\boldsymbol{\Sigma}_W = \mathbf{I}$ for $T = 500s$.

$\boldsymbol{v}_n \sim \mathcal{N}(\mathbf{0}, \sigma_v^2 \mathbf{I})$ and $\sigma_v = 3\sigma$ where σ is the scale of the resulting α-stable driving process [19],

$$\sigma = \left(\frac{\Gamma(2-\alpha)\cos(\pi\alpha/2)\mathbb{E}\,|U|^{\alpha}}{1-\alpha} \right)^{\frac{1}{\alpha}}, \quad U \sim \mathcal{N}(\mu_W, \sigma_W^2)\,. \tag{34}$$

For this setup of the RBPF, the Effective Sample Size (ESS) resampler from Stone Soup was utilised to avoid introducing excessive variance and correlation between particles [21]. The default threshold is used here [6], which resamples whenever

$$\left(\sum_{k=1}^{N_{\mathcal{K}}} \left(\omega_n^{(k)} \right)^2 \right)^{-1} < \frac{N_{\mathcal{K}}}{2}\,. \tag{35}$$

The initial states of the prior samples are drawn as

$$\boldsymbol{\mu}_0^{(k)} \sim \mathcal{N}([0,1,0,1]^{\top}, \mathbf{I}), \quad k \in \{1, \ldots, N_{\mathcal{K}}\}\,. \tag{36}$$

The RBPF's performance is evaluated by calculating the root-mean-squared-error (RMSE) between the inferred position $\hat{\boldsymbol{y}}$ and the true position vector $\boldsymbol{y}$,

$$RMSE = \sqrt{\frac{1}{T} \sum_{n=1}^{T} \|\hat{\boldsymbol{y}}_n - \boldsymbol{y}_n\|_2^2}\,, \tag{37}$$

for all time steps $n = 0, \ldots, T$. The total wall time scales linearly with the number of particles $N_{\mathcal{K}}$ and increases, while the RMSE decreases as anticipated. The standard particle filter from Stone Soup is used for comparison against the RBPF. Estimated tracks are depicted in Fig. 5. The RBPF outperforms the standard particle filter in terms of RMSE even when using fewer particles, inferring large jumps or straight-line tracks more accurately, owing to the use of Kalman filters on the model's conditional Gaussian structure.

Finally, an example of the RBPF on a Langevin model driven by an NG process where $\nu = \beta$ (also known as the Variance Gamma [22]) is provided in Fig. 6. As expected, the jumps generated by the NG process are less pronounced due to the variance being proportional to the underlying subordinator, while in the α-stable process, the variance takes the square of the subordinator process.

TABLE II
PERFORMANCE OF STANDARD PARTICLE FILTER AND RAO-BLACKWELLISED PARTICLE FILTER ON AN α-STABLE LANGEVIN MODEL WITH NOISY GAUSSIAN MEASUREMENTS FOR $T = 500s$ TIME STEPS.

Standard Particle Filter				
No. particles, $N_{\mathcal{K}}$	10000	5000	1000	500
RMSE	23.6962	23.3398	26.8375	34.4881
Wall time (s)	1.4904	0.8995	0.3715	0.3063

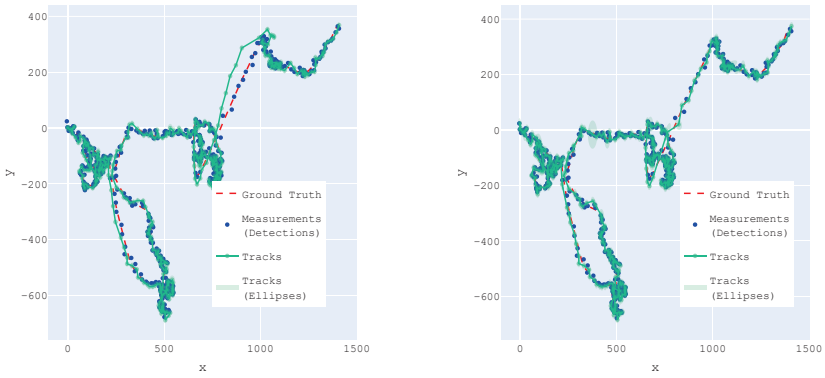
Rao-Blackwellized Particle Filter (RBPF)				
No. particles, $N_{\mathcal{K}}$	1000	500	100	5
RMSE	14.2551	14.3335	14.5501	14.8994
Wall time (s)	6.6260	3.0757	0.9231	0.2760

VI. Discussion and Conclusion

Currently, there are several proposals and works in progress aimed at enhancing the proposed framework for non-Gaussian Lévy process dynamical models, namely:

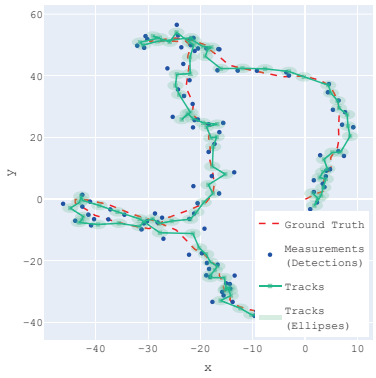
- The inclusion of the bias term $\boldsymbol{\mu}_W$ as an unknown within the state vector $\boldsymbol{x}$, and estimation of $\boldsymbol{\Sigma}_W$ as in [9].
- Evaluation on real datasets, combined with data association methods for multiple objects and object detection.

In summary, the marginalised particle filter has been introduced in Stone Soup as an efficient method for inference in scenarios where models are neither fully non-linear nor entirely non-Gaussian. This approach utilises components that inherit directly from the existing particle filter components within Stone Soup. Specifically, the Lévy process models, which are the focus here, exemplify a use case where a portion of the state vector remains Gaussian, conditional upon certain latent variables. The Lévy state-space model has



(a) (b)

Fig. 5. Standard and Rao-Blackwellised particle filters (RBPF) applied to the α-stable Langevin model using 1000 particles, with inferred tracks shown on the left and right figures, respectively. Model parameters are $\alpha = 1.8$, $\theta = 0.15$, $c = 10$, $\boldsymbol{\mu}_W = \mathbf{0}$, $\boldsymbol{\Sigma}_W = \mathbf{I}$. As shown, the RBPF infers the large jumps and straight-line tracks more accurately.



(a)

Fig. 6. Normal Gamma Langevin model with parameters $\nu = \beta = 2, \theta = 0.15, c = 10$, $\boldsymbol{\mu}_W = \mathbf{0}$, $\boldsymbol{\Sigma}_W = \mathbf{I}$, and measurement noise covariance $\mathbf{R} = 4\mathbf{I}$.

been developed as a versatile framework within Stone Soup, designed to facilitate modelling and inference across a range of non-Gaussian transition models. Currently, it supports both NσM and NVM driving processes, including the α-stable, normal Gamma, and normal tempered-stable Lévy processes. By adopting OOP principles and techniques such as dependency injection, the proposed framework allows for extension to additional classes of Lévy processes by varying the choice of $H(\cdot)$ within their respective shot noise representations.

Acknowledgements: The research of Godsill, Li and Gan is sponsored by the US Army Research Laboratory and the UK MOD University Defence Research Collaboration (UDRC) in Signal Processing under the SIGNeTS project. It is accomplished under Cooperative Agreement Number W911NF-20-2-0225. The views and conclusions in this document are of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory, the MOD, the U.S. Government or the U.K. Government. The U.S. Government and U.K. Government are authorised to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

REFERENCES

[1] S. Godsill, M. Riabiz, and I. Kontoyiannis, "The Lévy state space model," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, 2019, pp. 487–494.

[2] R. Gan, B. I. Ahmad, and S. J. Godsill, "Lévy state-space models for tracking and intent prediction of highly maneuverable objects," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 4, pp. 2021–2038, 2021.

[3] Y. Kindap and S. Godsill, "Point process simulation of generalised hyperbolic lévy processes," *Statistics and Computing*, Jan 2024.

[4] M. T. Costa, I. Kontoyiannis, and S. Godsill, "Generalised shot noise representations of stochastic systems driven by non-gaussian lévy processes," 2023.

[5] O. Cappé, S. J. Godsill, and E. Moulines, "An overview of existing methods and recent advances in sequential Monte Carlo," *Proceedings of the IEEE*, vol. 95, no. 5, pp. 899–924, 2007.

[6] S. Hiscocks, J. Barr, N. Perree, J. Wright, H. Pritchett, O. Rosoman, M. Harris, R. Gorman, S. Pike, P. Carniglia, L. Vladimirov, and B. Oakes, "Stone soup: No longer just an appetiser," in *2023 26th International Conference on Information Fusion (FUSION)*, 2023, pp. 1–8.

[7] R. A. Singer, "Estimating optimal tracking filter performance for manned maneuvering targets," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-6, no. 4, pp. 473–483, 1970.

[8] X. Rong Li and V. Jilkov, "Survey of maneuvering target tracking. part i. dynamic models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333–1364, 2003.

[9] S. Godsill and Y. Kındap, "Point process simulation of generalised inverse Gaussian processes and estimation of the Jaeger integral," *Statistics and Computing*, vol. 32, no. 1, p. 13, Dec 2021. [Online]. Available: https://doi.org/10.1007/s11222-021-10072-0

[10] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and computing*, vol. 10, no. 3, pp. 197–208, 2000.

[11] T. Schon, F. Gustafsson, and P.-J. Nordlund, "Marginalized particle filters for mixed linear/nonlinear state-space models," *IEEE Transactions on Signal Processing*, vol. 53, no. 7, pp. 2279–2289, 2005.

[12] S. Godsill and G. Yang, "Bayesian inference for continuous-time ARMA models driven by non-Gaussian Lévy processes," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 5, 2006, pp. V–V.

[13] Q. Li, R. Gan, J. Liang, and S. J. Godsill, "An Adaptive and Scalable Multi-Object Tracker Based on the Non-Homogeneous Poisson Process," *IEEE Transactions on Signal Processing*, vol. 71, pp. 105–120, 2023.

[14] R. Cont and P. Tankov, *Financial Modelling with Jump Processes*. Chapman & Hall/CRC, 2003.

[15] X. Rong Li and V. Jilkov, "Survey of maneuvering target tracking. part i. dynamic models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333–1364, 2003.

[16] B. Øksendal, *Stochastic Differential Equations: An Introduction with Applications (Universitext)*, 6th ed. Springer, Jan. 2014.

[17] J. Rosiński, "Series representations of Lévy processes from the perspective of point processes," in *Lévy Processes*, O. Barndorff-Nielsen, S. Resnick, and T. Mikosch, Eds. Birkhauser Boston, 2001, pp. 401–415.

[18] S. Godsill, I. Kontoyiannis, and M. Tapia Costa, "Generalised shot-noise representations of stochastic systems driven by non-gaussian lévy processes," *Advances in Applied Probability*, p. 1–36, 2024.

[19] T. Lemke, M. Riabiz, and S. J. Godsill, "Fully Bayesian inference for α-stable distributions using a Poisson series representation," *Digital Signal Processing*, vol. 47, pp. 96 – 115, 2015.

[20] O. E. Barndorff-Nielsen and N. Shephard, "Basics of Levy processes," no. 2012-W06, Jun. 2012, Economics Papers. [Online]. Available: https://ideas.repec.org/p/nuf/econwp/1206.html

[21] A. Doucet and A. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," *Handbook of Nonlinear Filtering*, vol. 12, 01 2009.

[22] D. B. Madan and E. Seneta, "The variance gamma (vg) model for share market returns," *Journal of business*, pp. 511–524, 1990.