The Brownian Integral Kernel:
A New Kernel for Modeling Integrated Brownian Motions (Supplementary Material)

Béla H. Böhnke[1][0000−0002−3779−3409] bela.boehnke@kit.edu, Edouard Fouché[2][0000−0003−0157−7648], and Klemens Böhm[1][0000−0002−1706−1913]

[1] Karlsruhe Institute of Technology (KIT) IPD, Kaiserstraße 12, Karlsruhe, 76131, Baden-Würtemberg, Germany
[2] Siemens Deutschland – Work originated while at KIT, Werner-von-Siemens-Straße 1, München, 80333, Bayern, Germany

Abstract. This document contains the supplementary materials to our work *The Brownian Integral Kernel: a New Kernel for Modeling Integrated Brownian Motions*. Here, we feature additional experiments, and visualizations useful for experts to obtain an intuition of our *Brownian integral kernel* (BKI). Further, we perform an ablation study, present the detailed proof by derivation of the BKI, and analyze its runtime. In addition to this document, we offer a Python implementation[3] of our kernel, compatible with the widely adopted GPy framework for Gaussian Process modeling. Together with the implementation, we publish all the code and data needed to reproduce our experiments.

Keywords: Integral Measurements· Learning from Aggregated Data· Integrated Brownian Motion· Gaussian Process Regression· Kernels

1 Notation

Random Variables $X, Y, Z, ...$ are denoted in uppercase, with their realizations $x, y, z, ...$ in corresponding lowercase. $X \sim \mathcal{N}(0, 1)$ signifies that X follows a standard normal distribution. As a shorthand, one can also write $x \leftarrow \mathcal{N}(0, 1)$ to express that a realization is drawn from a normal distribution.

Random Functions $F : x \mapsto F(x)$ associate each value x_i of a variable $x \in \mathbb{R}$ with a random variable $F(x_i)$ [11]. As such, a random function extends the notion of random variables to a continuous function space. *Sample Functions* $f(x) \leftarrow F(x)$ are functions $f : x \mapsto f(x)$ drawn from a random function $F(x)$.

Stochastic Processes S or random processes are random functions $S : t \mapsto S(t)$ where $t \in \mathbb{T}$ is interpreted as time, i.e., $\mathbb{T} = \mathbb{R}_+$.

[3] git: https://github.com/bela127/Brownian-Integral-Kernel

Brownian Motions $B : t \mapsto B(t)$ are stochastic processes characterized by random increments: $\delta B(\delta t) = B(t + \delta t) - B(t)$, these increments follow the normal distribution $\delta B(\delta t) \sim \mathcal{N}(0, \delta t)$. Brownian motion occurs in many scenarios where many small, random, and independent changes lead to a random overall change [11].

Gaussian Processes (GPs) are stochastic processes such as:
$GP(x) \sim \mathcal{N}\left(m(x), v(x)\right)$, illustrating that the process $GP(x)$ comes from a normal distribution with mean $m(x)$ and variance $v(x)$ depending on x [11]. One often uses GPs for applications other than time series modeling, which is why x is used instead of t. For instance, [8] uses a GP as a random function, i.e., as a probabilistic prior over smooth functions, and to model functions with associated uncertainty.

Kernel Functions $k(x, x')$, also called covariance functions, can define a GP instead of using a mean and variance function. Here, x, x' are two points from the regime of the GP, and $k(x, x')$ returns the covariance of the GP according to these given points.

The Brownian Kernel $k_{ff'}(t, t') = v_b \cdot min(t, t')$, with points in time t and t' has a variance parameter v_b. It translates to the drift speed of the resulting Brownian motion [11], i.e., how fast a time series diverges from a given starting point.

Primitive Functions , also called antiderivative, commonly use upper case letters. However, to avoid overlap in notation with random functions, we will use calligraphic letters $\mathcal{F}$ to denote primitive functions. In addition, we will use index notation to denote partly integrated functions, which is required for multivariate functions. For example:
$k(t, t') = k_{ff'}(t, t')$ is the original function, $k_{\mathcal{F}f'}(t, t') = \int_s^e k_{ff'}(t, t')\delta t$ is the one time integration and $k_{\mathcal{F}\mathcal{F}'}(t, t') = \int_s^e \int_{s'}^{e'} k_{ff'}(t, t')\delta t'\delta t$ the full (two-time) integration. Indexes of a primitive function $\mathcal{F}_{t,t'}$ (denoted with a calligraphic letter) are simply used for naming and do not have any special meaning if not otherwise specified in the text.

2 Extended Discussion of Related Work

Our work focuses on modeling time-continuous processes that contain integrators and on generating data from such learned models with Gaussian process regression. Such Gaussian processes (GPs) are the typical approach for this application [3], see Algorithm 2. A key strength of GPs is their use of flexible kernel functions, which allow for modeling complex, non-linear relationships in continuous data without explicit parametric assumptions, see step 1 in Algorithm 2. After training (step 2), GPs provide a posterior distribution over functions, making it possible not only to predict outcomes (step 3) but also associated uncertainty and to generate new realistic (continuous) data (step 4). This capability is particularly valuable for simulations and uncertainty quantification. Furthermore, GPs automatically adjust their complexity based on the data, making them robust to overfitting in small datasets and scalable to larger ones, unlike methods that require fixed parametric structures, manual tuning, or large datasets [5].

Algorithm 2 Generic Use-Case of a Gaussian Processes

1: **Input:** Training data $\mathbf{t}_{\text{train}}$, $\mathbf{y}_{\text{train}}$, Test data $\mathbf{t}_{\text{test}}$
2: **Output:** Predicted mean $\mu(\mathbf{t}_{\text{test}})$, uncertainty $\sigma(\mathbf{t}_{\text{test}})$, generated data $\mathbf{y}_{\text{gen}}$
3: **Step 1: Kernel Selection**
4: Choose a kernel function $k(\cdot, \cdot)$ (e.g., RBF, Matérn, our BIK, etc., or combination).
5: **Step 2: Training**
6: Train the Gaussian process model on $\mathbf{t}_{\text{train}}$, $\mathbf{y}_{\text{train}}$ using the kernel $k(\cdot, \cdot)$.
7: Learn hyperparameters θ of the kernel by maximizing the log marginal likelihood.
8: **Step 3: Prediction and Uncertainty Estimation**
9: For test points $\mathbf{t}_{\text{test}}$, compute the posterior mean $\mu(\mathbf{t}_{\text{test}})$ and variance $\sigma^2(\mathbf{t}_{\text{test}})$ using the trained model.
10: **Step 4: Data Generation from Posterior Distribution**
11: Sample from the posterior $\mathcal{N}(\mu(\mathbf{t}_{\text{test}}), \sigma^2(\mathbf{t}_{\text{test}}))$ to generate new data $\mathbf{y}_{\text{gen}}$.

There do exist extensions of Kalman filters for time-continuous modeling [12, 23]. However, the needed time-transition functions are challenging to drive and often problem-specific. This is why the work of [10] goes so far as using GPs as a noise model for Kalman filters. Which would, in turn, require a kernel like we proposed. Further, [20,26] have shown that such Kalman filters are equivalent to Gaussian processes under certain assumptions, i.e., it is possible to transform the kernel of a GP to a transition function for a Kalman filter or even use the kernel directly in an adapted Kalman filter implementation [19].

The closest related kernel is the *Radial Basis Function (RBF)* integral kernel, introduced by [4,21]. Independently of this work [13] also derived a line-RBF integral kernel, which calculates the covariance between lines and measurements. [6] claimed that there is no analytic solution to the line-RBF integral kernel (which was disproven by [13]) and provided an approximate solution by only solving a part of the integration and solving the missing part with numeric integration. While the RBF kernel is widely used because of its universal approximation

property [14], it has no relation to actual physical processes. This questions the usefulness of its integral version. The Brownian kernel, on the other hand, directly relates to physical processes such as load profiles or market behavior.

Obtaining an integral kernel from any kernel by pure numerical integration is possible and was done for classification [15]. Similarly, in [24] a mixture of Gaussian processes and numeric integration is used to learn from spatially integrated data. However, this approach is always an approximation, and the approximation error only decreases with a quadratic number of evaluation points. Coupled with the quadratic complexity of Gaussian process regression, this becomes computationally challenging for larger datasets. Fully analytic solutions, such as ours, are superior because they only require constant calculation time and provide an exact solution to the integral [13]. [25] uses Markov chain Monte Carlo to approximate the integral during training. However, such a method requires an adaptive training strategy, preventing one from using existing Gaussian process models and standard training algorithms. We provide a more detailed analysis of training complexity in our Supp.Mat. 9.

For smooth kernels, it is feasible to approximate the integral kernel by a finite spectral approximation on a Hilbert space [7, 22]. [17] uses this approximation to solve kernel line integrals associated with the problem of x-ray tomographic reconstruction. However, the Brownian kernel is discontinuous, which renders such an approximation intractable.

3 How Kernel Integration yields an Integrated Process

Theorem 1. *The BIK $k_{\mathcal{F}\mathcal{F}'}((s,e),(s',e'))$ yields the covariance between two integral measurements $\mathcal{B}(s,e)$ and $\mathcal{B}(s',e')$ taken from a Brownian motion $b(t)$:*

$$\mathrm{Cov}[\mathcal{B}(s,e),\mathcal{B}(s',e'))] = \mathrm{Cov}[\int_s^e b(t)\delta t, \int_{s'}^{e'} b(t')\delta t']$$

.

Theorem 1 implies that the double integral over the covariance $k_{ff'}(t,t')$ of a Brownian motion $b(t)$ is equal to the covariance between two integrated measurement regions $\mathcal{B}(s,e)$ and $\mathcal{B}(s',e')$, i.e., equal to the covariance of an integrated Brownian motion:

$$\mathrm{Cov}[\int_s^e b(t)\delta t, \int_{s'}^{e'} b(t')\delta t'] = \int_{s'}^{e'}\int_s^e \mathrm{Cov}[b(t),b(t')]\delta t'\delta t.$$

[4] features a respective proof for this relation which holds for integrals of any kernel, by decomposing the covariance as $\mathrm{Cov}[X,Y] = \mathbf{E}[XY] - \mathbf{E}[X]\mathbf{E}[Y]$. An additional proof can be obtained by interchanging the sequence of integration (Fubini) [18], [1]. This also guarantees that the resulting stochastic process is again a Gaussian process.

4 Proof of BIK Correctness

Theorem 2. *The Brownian integral kernel is positive-semidefinite:*

$$\sum_{j=1}^{n}\sum_{i=1}^{n} a_i a_j k_{\mathcal{F}\mathcal{F}'}((s_i,e_i),(s_j,e_j)) \geq 0$$

for any $s_i, e_i, s_j, e_j \in \mathbb{T} | s_i < e_i, s_j < e_j$ and for any $a_i, a_j \in \mathbb{R}$.

Proof. Theorem 2 directly follows from the derivation of the BIK as an integration of an existing kernel. The well-known Brownian kernel $k_{ff'}(t,t')$ is known to be positive-semidefinite [11]. Any integral of a kernel is again a kernel [1]. Therefore, the two times integral of $k_{ff'}(t,t')$, which results in the Brownian integral kernel, is positive-semidefinite. □

5 Details on used Data Sets

- For *Synth*, we draw data from a Gaussian process with a Brownian Kernel to obtain true Brownian motion behavior. To obtain integral measurements, we integrate this data numerically by summing over given equidistant time intervals. We generate 20 such time series, each with a length of 2.5k, 5k, 10k, and 20k time steps, and integrate over a window size $w \in [25, 50, 100, 200]$ steps, respectively. This results in the constant number of 1000 integral measurements for training.
- For *Load*, we use 17 consumer load profiles with one-minute resolution and a length of 7 days (time series of the electricity consumption of households) generated with a widely used data generator [16]. This generator simulates real-world households according to 400+ parameters, such as household size, gender and age of household members, employment, photovoltaic, and many more, and is steadily improved. It is accepted as close to real-world data in the electricity community as witnessed by the citations listed in [16]. Like smart meters, we aggregate the resulting one-minute profiles in 15-minute intervals.
- For *Real*, we use real-world load profiles provided by a private household, that are already aggregated in 15-minute intervals. Here no ground truth is available, and thus we only use it for evaluating the quality of the time series generated by the GP within Supp.Mat. 6.
- For *HIPE*, we use the High-resolution Industrial Production Energy (HIPE) data set [2]. It contains readings of ten machines and the main terminal of a power-electronics production plant.
- For *Stock*, we use daily stock market prices from [9]. Weekly aggregation provides us with integrated training data.

6 Providing Intuition about the BIK by A Comparative Visualization with BK

For domain experts, it is interesting to understand in which scenarios our BIK brings benefits. In this regard, we provide intuition by visualizing and discussing the similarities and differences between a Gaussian process model (*Brown*) with the Brownian kernel (BK) and a model (*Int_ Brown*) with our BIK. For brevity, we omit BNK from this visual comparison because the quantitative evaluation in Section 6 has shown that this kernel is nearly equivalent to the BK for integral measurements.

We examine the estimates and estimated variances as well as the synthesized data. For this, we visualize the fitted relation and its variance together with the integrated measurements and true underlying data, as well as generated load profiles that satisfy the conditions imposed by the observed integral measurements and the used kernel.

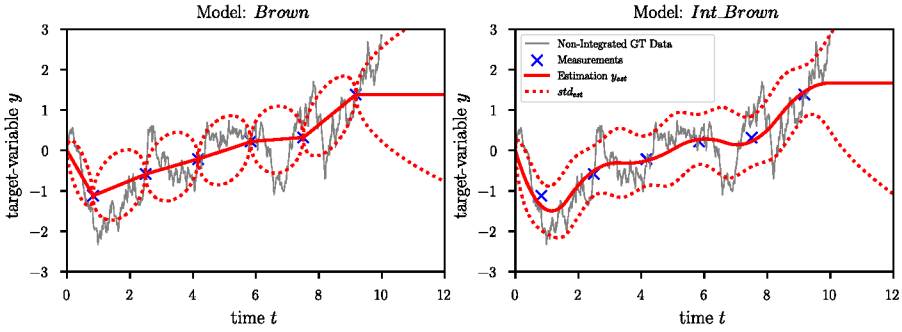


Fig. 1. Comparison between estimated variance v_{est} of the two kernels; 1a the standard Brownian kernel; 1b our new Brownian integral kernel.

Figure 1 illustrates the differences between the two models in estimation and estimation variance v_{est} (shown as a ▬ dotted line). We integrate Brownian motion data (*Synth* data), which gives us integral measurements as by definition 1 (marked as ✕). These measurements are inherently imprecise due to integration.

With integral measurements, the most likely function should be a smooth function. This is because integrating over time intervals acts like a sliding window smother. Comparing the fitted functions (depicted by the ▬ line) in terms of smoothness, the *Brown* model exhibits roughness, whereas the function modeled by *Int_ Brown* is smooth as expected.

Furthermore, in Figure 1a, the *Brown* model erroneously estimates a variance of zero at measurement locations, failing to account for the fact that the measurements are integrated. In contrast, *Int_ Brown* with our BIK in Figure 1b

acknowledges the drift that occurs during measurement periods. It incorporates this influence by modeling the resulting inaccuracy with a higher variance. The maximum variance between measurement locations remains roughly consistent for both models. This is expected for points far away from actual measurements because the underlying Brownian motion is the same.

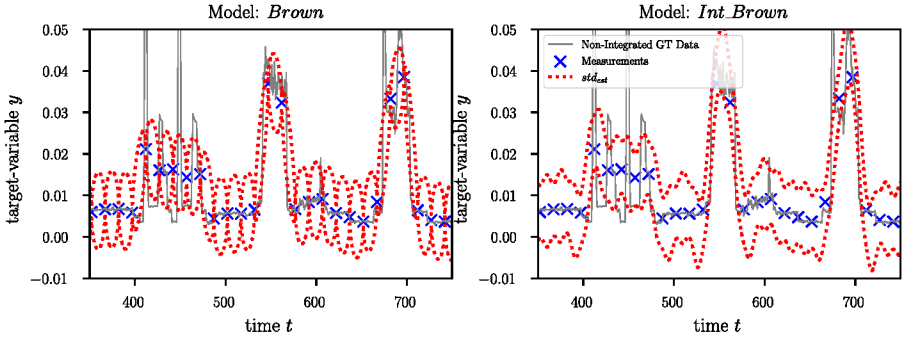


Fig. 2. Comparison using *Load* Data; 2a the Brownian kernel; 2b our new Brownian integral kernel.

In Figure 2, we observe similar patterns in the synthetic load profiles, which are integrated over 15-minute intervals (marked as ×). Additionally, we display the high-resolution ground truth (GT) load profile with a one-minute resolution (depicted by the — line). Upon comparing the actual load profile with the predicted one, it is evident that the true profile seldom aligns precisely with the integral measurement points. The *Int_Brown* model accurately captures this, exhibiting high variance in the vicinity of these locations.

In Figure 3, we see the ability of the GPs to synthesize data partially conditioned on measurements. Both models (*Brown* Figure 3a, and *Int_Brown* Figure 3b) are conditioned on the data presented in Figure 2. With *Brown*, all the generated time series pass through the measurement points (marked as ×), which is incorrect. Integrating the generated data gives random values that are not equal to the observed measurements. In contrast, *Int_Brown* correctly generates data that does not necessarily intersect with the measurement locations. The integration of this data, however, yields the correct measured value. Therefore, the data generated with our Brownian integral kernel reflects reality more accurately.

In Figure 4, we trained the models on real-world data (*Real*) and generated potential load profiles. The behavior of load profiles produced by our *Int_Brown* model, as shown in Figure 4b, is plausible, as integrating them again yields the measured data, this is not the case for data generated with *Brown*.

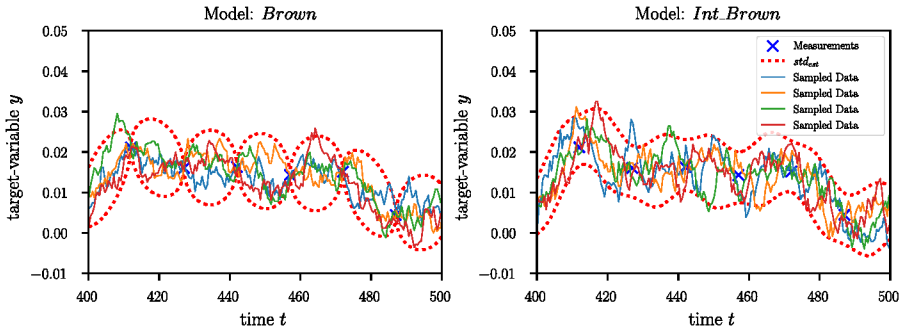


Fig. 3. Data generation from prior conditioned on *Load*; 3a data generated with Brownian kernel passes through measurement locations which is wrong; 3b data generated with our Brownian integral kernel. It does not need to go through measurement locations, but its integral is guaranteed to be the same as what was measured.

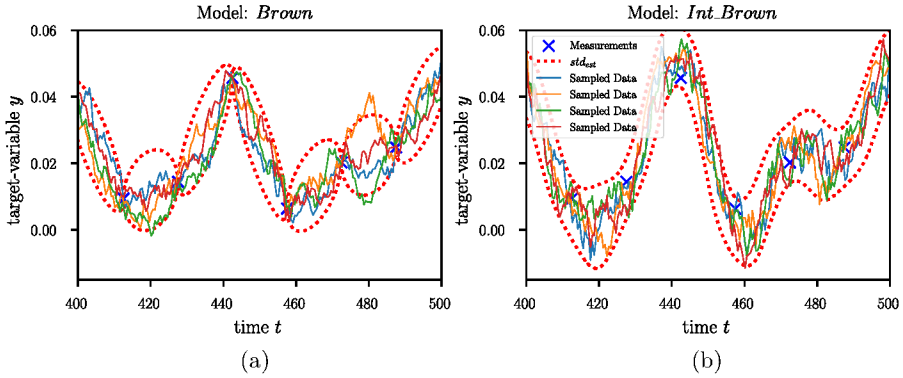


Fig. 4. Data generated from a model conditioned on real-world data (*Real*); 4a integrating data from the Brownian kernel does not reflect real behavior; 4b data from our Brownian integral kernel is much more likely in the real world, and integrating the data results in the measured data.

7 Ablation Study of Integral Window Size and Data Variance

In the following, we will examine how properties of the underlying physical process impact the estimation and the estimated uncertainty. First, we will discuss how the integral window size, i.e., the time constant of the integrator of a physical process, reflects on estimations. Secondly, we will examine the difference between Brownian motions with different variances – equivalent to different drift speeds of the Brownian motion.

7.1 Impact of the Integral Window Size

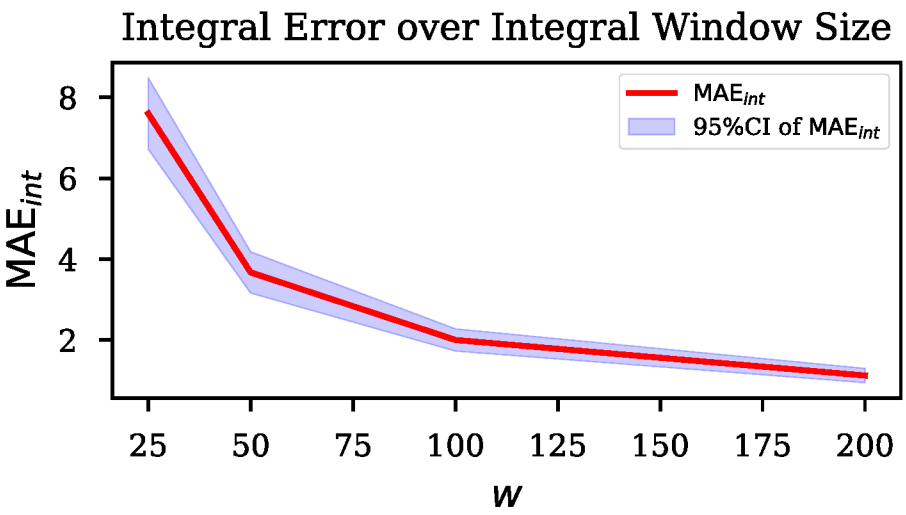


Fig. 5. Comparison of how changing the integral window size changes the integration error of generated data.

An observant reader may notice that the MAE_{int} decreases with increasing window size w. See Figure 5. We examined this behavior in depth and found that it is an artifact of the metric calculation rather than our kernel.

We calculate the metric over a fixed number of generated points in time, equal to the points provided by the original ground truth data, i.e., w points. We then sum up the values at these points to approximate the integral measurement and calculate the error in relation to the ground-truth integral. However, as the generated data is continuous, using a fixed number of points will always introduce an error in the integral approximation. The smaller the number of points, the greater the error and the larger the approximation variance (refer to the blue region in Figure 5). Indeed, in our experiments, we found that maintaining the

same start and end times for integration while increasing the number of points within the interval reduces the metric error. The error approaches zero for large w, which aligns with our expectations for an exact kernel like ours – integrating the data generated with the kernel yields the ground-truth integral value.

7.2 Impact of Data Variance

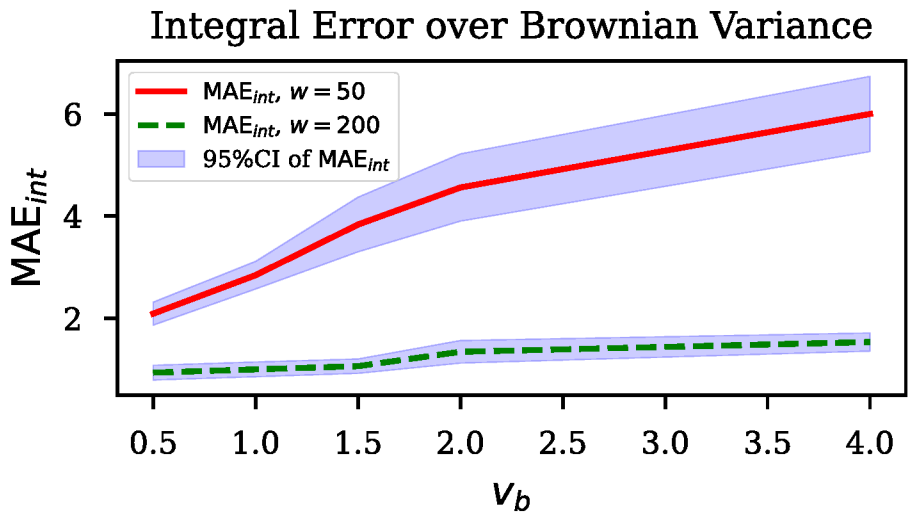
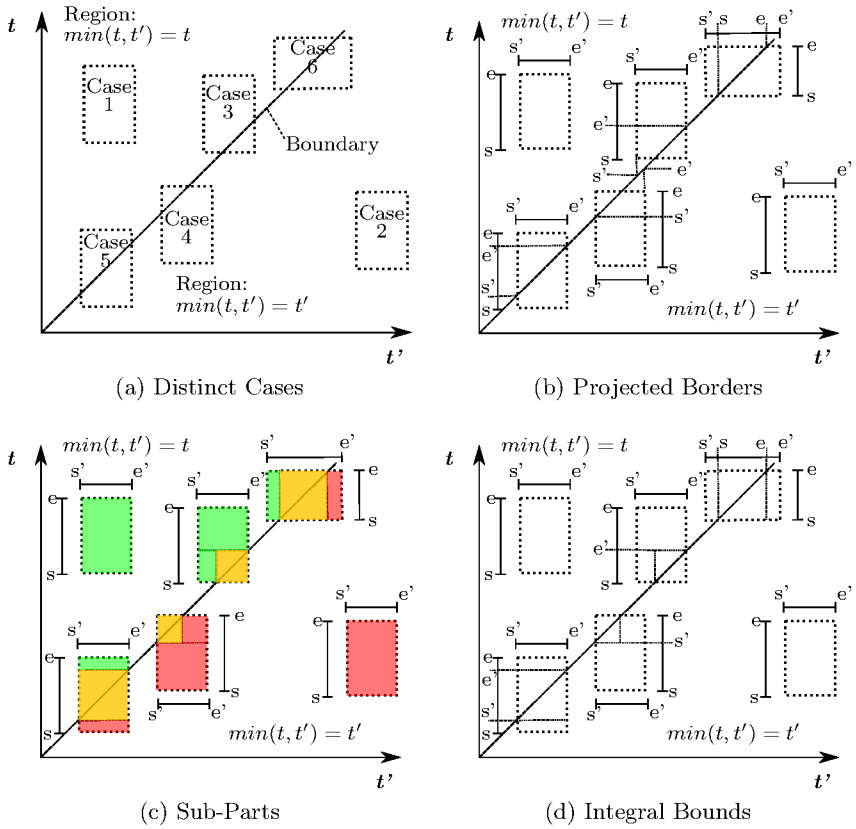


Fig. 6. Comparison of how Brownian ground truth data with different drift speeds v_b changes the integration error of generated data.

The previous observation in Supp.Mat. 7.1 has shown us that our kernel gives exact solutions for the same ground truth but with different integration intervals. Now we will examine how changing the speed of the Brownian motion, i.e., the Brownian variance v_b will impact MAE_{int}. In Figure 6 we see that increasing v_b while keeping $w = 50$ constant increases MAE_{int}. This behavior is again coused by the metric calculation. The Brownian motion within the same interval now behaves more erratically, requiring more evaluation points to calculate the integral with the same precision. Increasing the evaluation points ($w = 200$) (while keeping training points the same) again leads to a declining error approaching zero for any v_b, which is the expected behavior for an exact kernel.

8 Proof for the Brownian Integral Kernel by Derivation



(a) Distinct Cases (b) Projected Borders

(c) Sub-Parts (d) Integral Bounds

Fig. 7. Integral case distinction; 7a The six distinct cases; 7b The projected integral borders; 7c Division of cases into (toned) sub-parts; 7d Bounds of sub-integrals

Proof. The function $min(t, t')$ is discontinuous, so the solution of the integral:

$$k_{\mathcal{F}\mathcal{F}'}((s, e), (s', e')) = v_b \cdot \int_{s'}^{e'} \int_{s}^{e} min(t, t') \delta t \delta t',$$

depends on the integration bounds s, e, s', e'. To prove Theorem 1, we need a case distinction to cover the different bound configurations. We visualize each bound configuration in Figure 7a. We do so by drawing a boundary into one quadrant where $min(t, t')$ switches from yielding t to yielding t'. This boundary is the linear function $t' = t$. In the region above the boundary, the function $min(t, t')$

yields t. In the region below, it yields t'. We then see that there are six cases of possible bound configurations (dotted boxes). Because $min(t, t')$ is symmetric to the boundary, we can project s, e, s', e' onto the other axis, receiving an ordering of the integral borders. This ordering uniquely identifies each of the six cases. See if-conditions of Theorem 1 and Figure 7b (thin lines).

We further divide these six cases into smaller partitions and find that three common sub-parts, highlighted in Figure 7c, give solutions for each partition. While the solution for the sub-parts is structurally similar, the integral bounds are different. This is why we solve each sub-part independently of the specific bounds by substituting the upper bounds with u, u' and the lower bounds with l, l'. We name the resulting three sub-integrals after the regions they are part of:
(1) Integrals with integration bounds u, u', l, l' in one region
$u, u', l, l' \in \{t \mid t < t'; t, t' \in \mathbb{T}\}$ are named $\mathcal{F}_{tt}$.
(2) Integrals with integration bounds in the other region
$u, u', l, l' \in \{t' \mid t' < t; t, t' \in \mathbb{T}\}$ are named $\mathcal{F}_{t't'}$ respectively.
(3) Integrals with bounds in both regions are named $\mathcal{F}_{tt'}$.
In Figure 7c the sub-integral $\mathcal{F}_{tt}$ has color green, $\mathcal{F}_{t't'}$ red and $\mathcal{F}_{tt'}$ orange.

Sub-parts $\mathcal{F}_{tt}$ and $\mathcal{F}_{t't'}$: The function $min(t, t')$ is continuous for these sub-parts. This allows us to directly integrate once, leading to:

$$t < t' : \mathcal{F}_t(t, l', u') = \int_{l'}^{u'} min(t, t')\delta t' = \int_{l'}^{u'} t\delta t' = |_{t'=l'}^{u'}\ t \cdot t' = t \cdot u' - t \cdot l' \tag{1}$$

$$t' < t : \mathcal{F}_{t'}(l', u') = \int_{l'}^{u'} min(t, t')\delta t' = \int_{l'}^{u'} t'\delta t' = |_{t'=l'}^{u'}\ \frac{1}{2}t'^2 = \frac{1}{2}u'^2 - \frac{1}{2}l'^2 \tag{2}$$

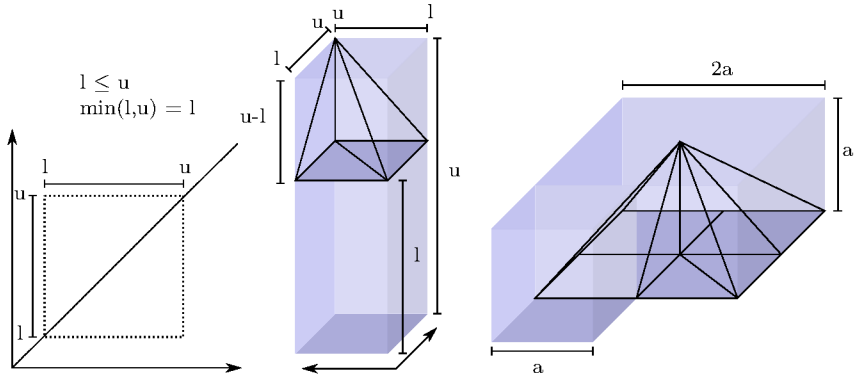
We finish the first part of the proof by integrating twice, which gives us:

$$t < t' : \mathcal{F}_{tt}((l', u'), (l, u)) = \int_{l}^{u}\int_{l'}^{u'} t\delta t'\delta t = |_{t=l}^{u}\ \frac{1}{2}t^2 \cdot u' - \frac{1}{2}t^2 \cdot l' = \frac{1}{2}u^2 \cdot u' - \frac{1}{2}u^2 \cdot l' - \frac{1}{2}l^2 \cdot u' + \frac{1}{2}l^2 \cdot l',$$

$$t' < t : \mathcal{F}_{t't'}((l', u'), (l, u)) = \int_{l}^{u}\int_{l'}^{u'} t'\delta t'\delta t = |_{t=l}^{u}\ \frac{1}{2}u'^2 \cdot t - \frac{1}{2}l'^2 \cdot t = \frac{1}{2}u'^2 \cdot u - \frac{1}{2}l'^2 \cdot u - \frac{1}{2}u'^2 \cdot l + \frac{1}{2}l'^2 \cdot l.$$

These are the Equations 1 and 2 of Theorem 1.

Sub-part $\mathcal{F}_{tt'}$: The third integral $\mathcal{F}_{tt'}$ needs a separate solution because it intercepts the region border and, thus, is discontinuous. We derive Equation 3



(a) Total Volume (b) Pyramid

Fig. 8. Geometric view on integrating; 8a Total volume; 8b subvolumes combined to pyramid.

by looking at $min(t, t')$ geometrically, see Figure 8. The volume of the geometry is equivalent to the two-times integration. We decompose the geometry into two subvolumes (sv), one cuboid (cub), and one more complex subvolume (com). We calculate the volume of the cuboid directly, see Figure 8a. By merging the geometry of the complex subvolume four times with itself, we obtain a pyramid, see Figure 8b. We now work out the integral bounds and obtain the solution to the double integral with the following system of equations:

$$\begin{cases} \mathrm{sv}_{com} = \frac{1}{4} V_{pyr} \\ V_{pyr} = \frac{1}{3} V_{cub} \\ V_{cub} = h \cdot w \cdot d \end{cases} \quad \begin{cases} \\ \Longrightarrow \\ \\ \end{cases} \quad \begin{cases} a{=}u-l \\ h{=}a \\ w{=}2a \\ d{=}2a \end{cases} \quad \begin{cases} \mathrm{sv}_{com} = \frac{1}{3}(u-l)^3 \\ \mathrm{sv}_{cub} = (u-l)^2 \cdot l \\ \mathcal{F}_{tt'}(l,u) = \mathrm{sv}_{com} + \mathrm{sv}_{cub} \end{cases},$$

which results in Equation 3 of Theorem 1:

$$\mathcal{F}_{tt'}(l,u) = \frac{1}{3}(u-l)^3 + (u-l)^2 \cdot l.$$

To combine the three subintegrals into the six original cases, we still need the integral bounds of the subintegrals corresponding to the original cases. Using the symmetric property of $min(t, t')$, we can obtain these bounds, see Figure 7d. With the matching integral bounds, we obtain the solution to the Brownian integral kernel, Theorem 1. □

9 Computational Complexity of the BIK

Concerning computational complexity, one has to distinguish between the complexity of the training algorithm without kernel and the complexity of the standalone kernel. We perform all experiments (for our kernel as well as the baseline kernels) with the standard GP model from the GPy framework[4], which has a complexity of $O(n^2)$. Depending on the use case and the needed accuracy, other less complex models can be employed to approximate GP. For example, one can use gradient-based models that allow batch training with a complexity of $O(n \cdot i)$, where i is the number of training iterations. Further, new work [19, 20, 26] has shown how to use GP kernels directly in an adapted Kalman filter implementation, thereby reducing the runtime to $O(n)$ as long as certain assumptions are met. Similarly, methods for training on time series can be used, like sliding windows that discard old measurements to keep n small, thereby losing information from old measurements. This is possible for all kernels, including ours.

If an exact solution is needed, the standard GP model with complexity $O(n^2)$ must be used. For each of these n^2 calculations, the kernel is invoked once, which is why kernel complexity can become an issue when too large. Our analytically derived BIK achieves the lowest possible complexity of $O(1)$, which makes it ideal. In comparison, the work [6] has a complexity of $O(m)$ where m are the number of Montecarlo iterations used to approximate the integral. Here, m needs to be high to obtain good approximations. While full numerical integration like in [15] and [24] is possible for any kernel, it comes with a complexity of $O(m^2)$ because of the nested Montecarlo iterations. Coupled with the quadratic complexity of Gaussian process regression, this becomes computationally challenging for larger datasets.

References

1. Abrahamsen, P.: A Review of Gaussian Random Fields and Correlation Functions. Rapport 917 (1997)
2. Bischof, S., Trittenbach, H., Vollmer, M., Werle, D., Blank, T., Böhm, K.: Hipe: An energy-status-data set from industrial production. In: e-Energy. p. 599–603. ACM (2018)
3. Eric, S., Maarten, S., Andreas, K.: A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions. J. Math. Psychol. (2018)
4. Fariba, Y., T, S.M., Álvarez Mauricio: Multi-task learning for aggregated data using gaussian processes. In: Advances in Neural Information Processing Systems. Curran Associates, Inc. (2019)
5. Ferris, B., Hähnel, D., Fox, D.: Gaussian processes for signal strength-based location estimation. In: Robotics: Science and Systems (2006)
6. Hendriks, J.N., Jidling, C., Wills, A., Schön, T.B.: Evaluating the squared-exponential covariance function in gaussian processes with integral observations (2018)

[4] GPy: https://gpy.readthedocs.io

7. Jidling, C., Wahlström, N., Wills, A., Schön, T.B.: Linearly constrained gaussian processes. In: NIPS. vol. 30. Curran Associates, Inc. (2017)
8. Karvonen, T., Wynne, G., Tronarp, F., Oates, C., Särkkä, S.: Maximum likelihood estimation and uncertainty quantification for gaussian process approximation of deterministic functions. SIAM/ASA JUQ (2020)
9. Kumar, S.: 2019-2024 us stock market data (2024). https://doi.org/10.34740/KAGGLE/DSV/7553516
10. Kurtz, V., Lin, H.: Kalman filtering with gaussian processes measurement noise (2019)
11. Lindgren, G., Rootzen, H., Sandsten, M.: Stationary Stochastic Processes for Scientists and Engineers. T&F (2013)
12. Liu, F., Gao, Z., Yang, C., Ma, R.: Extended kalman filters for continuous-time nonlinear fractional-order systems involving correlated and uncorrelated process and measurement noises. International Journal of Control, Automation and Systems **18**, 2229–2241 (2020)
13. Longi, K., Rajani, C., Sillanpää, T., Mäkinen, J., Rauhala, T., Salmi, A., Haeggström, E., Klami, A.: Sensor placement for spatial gaussian processes with integral observations. In: UAI. vol. 124, pp. 1009–1018. PMLR (2020)
14. Micchelli, C.A., Xu, Y., Zhang, H.: Universal kernels. J. Mach. Learn. Res. (2006)
15. O'Callaghan, S., Ramos, F.: Continuous occupancy mapping with integral kernels. AAAI **25**, 1494–1500 (2011)
16. Pflugradt, N., Stenzel, P., Kotzur, L., Stolten, D.: Loadprofilegenerator: An agent-based behavior simulation for generating residential load profiles. Journal of Open Source Software **7**, 3574 (2022)
17. Purisha, Z., Jidling, C., Wahlström, N., Schön, T.B., Särkkä, S.: Probabilistic approach to limited-data computed tomography reconstruction. Inverse Problems **35**, 105004 (2019)
18. Rasmussen, C.E., Williams, C.K.I.: Gaussian processes for machine learning. Adaptive computation and machine learning, The MIT Press (2006)
19. Reece, S., Roberts, S.: An introduction to gaussian processes for the kalman filter expert. In: 2010 13th International Conference on Information Fusion. pp. 1–9 (2010)
20. Sarkka, S., Hartikainen, J.: Infinite-dimensional kalman filtering approach to spatio-temporal gaussian process regression. In: Lawrence, N.D., Girolami, M. (eds.) Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 22, pp. 993–1001. PMLR (2012)
21. Smith, M.T., Alvarez, M.A., Lawrence, N.D.: Gaussian process regression for binned data (2019)
22. Solin, A., Särkkä, S.: Hilbert space methods for reduced-rank gaussian process regression. Stat. Comput. **30**, 419–446 (2019)
23. Taghvaei, A., de Wiljes, J., Mehta, P.G., Reich, S.: Kalman Filter and Its Modern Extensions for the Continuous-Time Nonlinear Filtering Problem. Journal of Dynamic Systems, Measurement, and Control **140**(3), 030904 (11 2017)
24. Tanaka, Y., Tanaka, T., Iwata, T., Kurashima, T., Okawa, M., Akagi, Y., Toda, H.: Spatially aggregated gaussian processes with multivariate areal outputs. In: NEURIPS. vol. 32. Curran Associates, Inc. (2019)
25. Tanskanen, V., Longi, K., Klami, A.: Non-linearities in gaussian processes with integral observations. In: MLSP (2020)

26. Todescato, M., Carron, A., Carli, R., Pillonetto, G., Schenato, L.: Efficient spatio-temporal gaussian regression via kalman filtering. Automatica **118**, 109032 (Aug 2020)