Neural Augmented Kalman Filtering With Bollinger Bands for Pairs Trading

Amit Milstein, Guy Revach, Haoran Deng, Hai Morgenstern, and Nir Shlezinger

Abstract—Pairs trading is a family of trading techniques that determine their policies based on monitoring the relationships between pairs of assets. A common pairs trading approach relies on describing the pairwise relationship as a linear Space State (SS) model with Gaussian noise. This representation facilitates extracting financial indicators with low complexity and latency using a Kalman Filter (KF), which are then processed using classic policies such as Bollinger Bands (BB). However, such SS models are inherently approximated and mismatched, often degrading the revenue. In this work, we propose KalmanNet-aided Bollinger bands Pairs Trading (KBPT), a deep learning aided policy that augments the operation of KF-aided BB trading. KBPT is designed by formulating an extended SS model for pairs trading that approximates their relationship as holding partial co-integration. This SS model is utilized by a trading policy that augments KF-BB trading with a dedicated neural network based on the KalmanNet architecture. The resulting KBPT is trained in a two-stage manner, which first tunes the tracking algorithm in an unsupervised manner independently of the trading task, followed by its adaptation to track the financial indicators to maximize revenue while approximating BB with a differentiable mapping. KBPT thus leverages data to overcome the approximated nature of the SS model, converting the KF-BB policy into a trainable model. We empirically demonstrate that our proposed KBPT systematically yields improved revenue compared with model-based and data-driven benchmarks over various assets.

Index Terms—Pairs trading, quantitative strategies, Kalman filter, KalmanNet, model based deep learning.

I. INTRODUCTION

Quantitative methods constitute the fundamental mathematical framework for analysis and prediction in financial markets [2], [3]. A common type of quantitative method is algorithmic trading [4], which deals with decision-making carried out by an agent (i.e., a trader) to maximize a cumulative reward, most commonly achieving a high Profit and Loss (PNL) balance in the market. Quantitative trading schemes typically comprise two main stages: the agent first tracks a stochastic process describing the prices of the assets of interest to extract useful trading indicators. Then, these financial indicators are used as a basis for decision-making by setting a trading policy [5], [6], [7].

Quantitative trading requires a decision-making mechanism given application time constraints, i.e., a trading policy that outputs a position based on the trading indicators. Such policies are typically based on indicators obtained as statistical predictions of an asset price [8]. A popular classical policy is the Bollinger Bands (BB) [9], which is based on the intuition that if the price is much less than its mean, it will rise back to a normal level, and thus, one should long this asset. Because this method is not linear, it hedges the risk by constraining the investment.

Classical trading schemes such as BB work well for single stationary (and specifically, mean-reverting) processes [10]. It is therefore sought-after to look for stationary assets, though some schemes only look for the weaker condition of mean reverting, e.g., using the Ornstein–Uhlenbeck formula [11]. Accordingly, algorithmic tracking of financial processes is typically based on imposing a model on their temporal evolution [12]. A common approach imposes simple linear stochastic stationary model [13], often based on autoregressive and moving average models [14]. While assets are rarely stationary in real markets, their differences and spread (i.e., linear combination) are sometimes faithfully captured as stationary. Thus, such techniques are commonly adopted in pairs trading [4], [15]. The spread evolution and its relationship with the assets pair is often described using a Space State (SS) model [16], [17], [18], enabling tracking with a Kalman Filter (KF) [14, Ch. 10]. A core challenge with combining financial policies with algorithmic tracking based on such statistical models is that they typically require strong assumptions and prior financial knowledge.

For instance, to utilize the KF for spread tracking, one has to faithfully capture the pairs trading as a linear Gaussian SS model. Such models often fail to capture complicated patterns of real-world financial assets, leading to poor trading policies.

In recent years, there has been a growing interest in using model-agnostic deep learning to overcome the drawbacks of classic model-based methods. Deep learning systems are used to capture the time evolution of financial assets [19], extract features for trading [20], and determine trading policies [21].
In this work, we propose KalmanNet-aided Bollinger bands Pairs Trading (KBPT), a pairs trading algorithm that combines SS model-based trading policies with deep learning tools, based on model-based deep learning methodology [33], [34], [35]. KBPT is derived by proposing a novel SS model representation for pairs trading obtained from assuming partial co-integration [17] combined with an autoregressive prior imposed on the spread.

As opposed to previous SS model-based trading policies that utilize, e.g., KF with BB for setting the position, thus implicitly assuming that the SS model is Gaussian and accurate, we design our policy to mainly cope with the approximated nature of the SS model and its expected non-Gaussianity. This is achieved by having KBPT preserve the flow of KF-BB trading, retaining its structured modeling and interpretability while augmenting the KF with a trainable RNN following the recently proposed KalmanNet [36]. The resulting neural augmentation computes the Kalman gain (KG) using a dedicated RNN that learns the underlying stochasticity and, therefore, leverages data to track the spread in partially known and non-Gaussian SS models. Such usage of trainable architectures for learning the KG draws inspiration from previous designs of RNNs for tuning control systems, where the optimal gain of a controller may vary over time [37], [38], [39], [40], as well as in applications in renewable energy [41], [42], and speech enhancement [43], [44].

We propose a dedicated training scheme for KBPT that learns the pairs trading policy from sequences of past assets pairs. The learning method is based on a two-stage procedure, where we first train KalmanNet separately from the trading task as a form of pretraining. There, we overcome the fact that there is no ground-truth spread value by leveraging the interpretable architecture of KalmanNet, and particularly its internal prediction of the next observation which follows from the KF flow, for unsupervised learning [45]. Then, we train the overall trading policy, combining the neural augmented KalmanNet with a customized BB mapping that is differentiable, such that the tracking algorithm learns to produce features that are most useful in the sense of maximizing the PNL rather than accurately tracking the prices. By that, we gain the ability to cope with modeling mismatch, as the resulting architecture converts the model-based trading algorithm into a trainable discriminative model [46] that is trained end-to-end to maximize the PNL as a cumulative reward.

Our empirical study compares KBPT with both model-based trading and deep RL-based policies for various asset pairs. There, we demonstrate the individual gains of each of the ingredients of KBPT, including the usefulness of the extended SS model underlying KBPT, as well as the superiority of the proposed hybrid algorithm in systematically achieving higher PNL compared with all considered benchmarks. Our work extends upon its preliminary findings reported in [1] in the proposal of the new partially co-integrated SS model, the incorporation of a dedicated accumulated reward loss and the two-stage training methods, as well as in the extensive discussion, derivation, and experimental evaluations.

The rest of this paper is organized as follows: Section II covers preliminaries in model-based trading and formulates the problem; Section III describes the different SS models in pairs trading and presents our proposed model; Section IV details our proposed hybrid KBPT policy along with its learning procedure; Section V presents the empirical study of KBPT, contrasting it with both model-based and data-driven policies; while Section VI provides concluding remarks.

This paper uses boldface lower-case letters for vectors, e.g., x, and boldface uppercase letters for matrices, e.g., for X. We denote the step function as \( \mathbb{1}(\cdot) \), with \( \mathbb{1}(t) = 1 \) for \( t > 0 \) and \( \mathbb{1}(t) = 0 \) for \( t \leq 0 \), while \( E\{\cdot\} \) is the notation for stochastic expectation. We use the term stationary process to refer to a stochastic process that is stationary in the wide sense. For consistency, the prices of all assets are given in USD.

II. PRELIMINARIES AND PROBLEM FORMULATION

In this section, we formulate the considered model for pairs trading. To that aim, we first review necessary preliminaries in quantitative trading in Section II-A, and recall the BB policy in Section II-B. These preliminaries are then used to formulate the problem in Section II-C.

A. Trading Formulation

Trading strategies refer to the determining of investment policies based on the monitoring of financial assets. Accordingly, trading strategies can be generally divided into two stages: (i) tracking of the assets into financial indicators; and (ii) the trading policy that is based on these indicators [7].

1) Tracking: A crucial part of any trading scheme is constantly evaluating and analyzing the financial markets, individual securities, or sectors. Information such as price movements, volatility, liquidity, volume, momentum, and market breadth is valuable for making informed decisions in the trading market. Using this financial data, one can derive financial indicators that enable the trader to get insight into potential entry and
exit points, assess risks, and ultimately optimize the investment strategy. Quantitative financial indicators can include technical indicators (e.g., moving averages, relative strength index) [47], fundamental indicators (e.g., earnings per share, price-to-earnings ratio), or macroeconomic indicators (e.g., GDP growth rate, inflation rate) [48].

To formulate this mathematically, we use \( d_t \) to denote the financial information (e.g., assets’ price) at time \( t > 0 \). A financial tracker, denoted \( \varphi \), is a mapping of all the financial data accumulated until time \( t \) into financial indicators \( z_t \), i.e.,

\[
\varphi : \{d_t\}_{t \leq \tau} \mapsto z_t.
\]

The financial indicator should provide sufficient information for the policy to dictate the current decision, as detailed next.

2) Policy: The policy component of a trading scheme, denoted by \( \pi \), refers to the rules, guidelines, and principles that govern the decision-making process and the execution of trades. The policy component, in general, may encompass both quantitative and qualitative aspects: Quantitative aspects can involve specific parameters, thresholds, or algorithms based on financial indicators or other mathematical models. Qualitative aspects consider factors such as market conditions, investor sentiment, news events, or expert judgment.

The policy is the last step of the trading scheme and outputs the recommended actions for the trader to take to optimize profits. We refer to the return of each trade transaction as the reward. In quantitative trading, the action at time \( t \), denoted \( p_t \), is determined using a trading policy \( \pi \) based on the current indicator \( z_t \) as well as past actions and indicators, namely,

\[
\pi : \{z_t\}_{t \leq \tau}, \{p_s\}_{\tau < t} \mapsto p_t.
\]

We henceforth focus on settings where

A1 The information \( d_t \) represents the price of an asset.

A2 The actions correspond to long/short decisions on \( d_t \), i.e., holding positive or negative quantities, respectively.

The action space in A2 indicates that \( p_t \) encapsulates open and close decisions. We formulate this by writing \( p_t = [op_t, cp_t] \), where \( op_t \in \{-1, 0, 1\} \) is the open position policy that signals if to short, hold or long the asset, respectively; and \( cp_t \in \{0, 1\} \) is the close position policy, which gets the value 1 when an existing open position (e.g., from time \( t - 1 \)) needs to be closed. Otherwise, if a position needs to remain open or there is no open position, it gets the value of 0. The order in which positions are taken involves checking whether the closing criteria are met and then checking if one should open one. We say that \( p_t \) is an active position if \( op_t = \pm 1 \).

3) Reward: Under A1-A2, one can mathematically formulate the reward accumulated for an active position. To that aim, let \( t_i^\varphi \) be the time the \( i \)th active position is taken and \( t_i^\pi \) the time it is closed. Accordingly, the reward obtained for the \( i \)th activity of policy \( \pi \) with financial tracker \( \varphi \), denoted by \( r_i^{\varphi, \pi} \), is computed based on the difference in the asset price over the activity period and whether it was long or short via

\[
r_i^{\varphi, \pi} = op_{t_i^\varphi} \cdot (d_{t_i^\varphi} - d_{t_i^\pi}).
\]

The reward in (3) can be positive or negative, i.e., profit or loss, respectively.
trading strategies using RL [31], [49], [50], one is often interested in obtaining instantaneous rewards. This is achieved by closing a position after a single time step (though it can then be re-opened and treated as a new active position, yielding an additional transaction cost, i.e., friction [18], which we omit for simplicity). Such an operation results in

\[
c_{pt} = \mathcal{U} \left( \text{op}_{t-1} \right).
\]

Alternatively, one can determine the close position based on the indicator, allowing a cumulative reward where a position can be held over multiple time steps. In this case, the closing of a position is a function of the indicator \( z_t \). For instance, one can decide to close a currently open position if \( z_t \) has crossed the middle or opposite band [4], [51], [52], i.e.,

\[
c_{pt} = \mathcal{U} \left( -z_{t-1} \cdot z_t \right) \cdot \left( 1 - \mathcal{U} \left( \tau_{cp,t} - \tau_{op,t} \right) \right).
\]

A comparison between the cumulative and instantaneous approaches is schematically illustrated in Fig. 2, where only the middle and upper Bollinger Bands are drawn for clarity. As depicted in Fig. 2, the instantaneous trading policy effectively makes a profit based on the difference between the asset price when it crosses the upper band for the second time and the asset’s price the following day. On the other hand, the profit made by the cumulative trading policy is the price difference between the upper band and the MA of the asset. This also directly affects the behavior of the PNL function such that under modeling assumptions detailed in the sequel, the cumulative PNL is expected to exhibit a non-decreasing behavior. A more detailed analysis of the PNL behavior is provided in Section V-D.

There are several advantages in using BB for determining trading policies that go beyond the profit-loss reward and stem from the simplicity and interpretability of the scheme. First, the bands adjust to the volatility of the asset’s price, enabling it to represent it visually. In addition, BB can be used to determine price reversal signals - when the price of the asset goes above the upper or below the lower band, it can indicate that the asset is being overbought or oversold, respectively, and therefore suggest that a trend reversal is likely to occur. Moreover, BB can identify trends - if the price is constantly above the middle band, it can suggest an uptrend or, similarly, a downtrend if it is below the middle band [51].

C. Problem Formulation

We consider the design of a trading strategy for the task of pairs trading. Here, the financial information accumulated at each time instance \( t \) corresponds to a pair of assets denoted \( \alpha_t \) and \( \beta_t \). There is no underlying assumption of stationarity on the considered assets. We aim to jointly design a financial tracker \( \varphi \) along with a policy mapping \( \pi \) that maximizes the expected PNL. The PNL is defined as the sum of all rewards accumulated up to time \( t \), i.e.,

\[
P_{\varphi,\pi} = \sum_{i: t_{cp,i} \leq t_{op,i}} r_{i,\varphi,\pi}.
\]

Accordingly, the trading strategy design problem is formulated by identifying

\[
\varphi^*, \pi^* = \arg \max_{\varphi,\pi} \mathbb{E} \{ P_{\varphi,\pi} \}.
\]

For design purposes, we are given access to a data set comprised of past financial information measurements corresponding to \( n_t \) past time indices. This dataset is given by

\[
D = \{ \alpha_t, \beta_t \}_{t=1}^{n_t}.
\]

To leverage established BB techniques, which inherently rely on underlying stationarity, we exploit pairs trading modeling, where we extend the notions of Co-Integration (CI) and Partially Co-Integration (PCI) proposed in [16], [17], as detailed in the following Section III. This enables a hybrid model-based/data-driven design, as stated in Section IV.

III. PAIRS TRADING AS A STATE SPACE MODEL

The price of a single asset is rarely described as stationary and is typically a chaotic stochastic process. However, tracking the inherent statistical relationship between two assets is often much easier than tracking each asset individually and, consequently, can be used for investment decisions. This is the reasoning behind pairs trading and its general form, which consists of more than two assets [14, Ch. 10].

There are several frameworks for capturing statistical relationships in pairs trading. These include monitoring distances [53], adopting a CI model [16], as well as viewing pairs trading as a stochastic control setup [54]. A large volume of research is dedicated to comparing these statistical models and their usefulness for pairs trading [18]. In this work, we adopt the PCI approach [17], which extends the CI model. The PCI model is suitable for tackling (11) due to its ability to capture both long-lasting and transient company shocks and their effect on the asset price and its flexibility in describing temporal correlations in the pairwise relationship [17]. Moreover, as shown in the design and analysis of trading policies [55], [56], trades of
smaller magnitude impact the market much less. Therefore, we operate under the assumption that the act of buying or selling assets by the trader does not exert any discernible impact on the intrinsic pricing dynamics of the assets involved, i.e., the magnitude of trades is not large enough to impact the market. This is a common assumption that is taken in many pair trading schemes [4], [17], [54]. From a signal processing perspective, impacting the market turns the framework from a ‘tracking’ model to a ‘control’ model. Our focus is on trades that are not of such magnitude.

To describe this model, we first review the notion of CI and how it gives rise to a SS model for pairs trading in Section III-A. Then, in Section III-B, we review the extended PCI model and present our SS representation for pairs trading that follows from the PCI model. The proposed extended SS model is used to derive KBPT in Section IV.

A. Co-Integration in Pairs Trading

We next review the notion of CI and how it is specialized for pairs trading.

1) Co-Integration: To define CI, we first recall the definition of integration order (see, e.g., [14, Ch. 2.6]):

Definition 1 (Integration Order): A time series is called integrated of order $p$, denoted as $I(p)$, if the time series obtained by taking the difference of the time series $p$ times is weakly stationary while taking the difference of the time series $p - 1$ times yields a series that is not weakly stationary.

Definition 1 refers to the basic statement of integration order. A more general definition considers weighted differences, stated as follows:

Definition 2 (Weighted Integration Order): A time series $x_t$ is said to be weight integrated with order $p$, and written as $x_t \sim W1(p)$, if there exists a $(p + 1) \times 1$ vector $w$ such that $w^T x_t$ is a stationary time series, with $x_t \triangleq [x_t, x_{t-1}, ..., x_{t-p}]$.

Definition 2 is clearly a generalization of Definition 1. Specifically, if we set $w$ as a vector of $(p + 1)$ ones, then Definition 1 is obtained. For brevity, in the remainder of the paper, when we say that a series is integrated, we refer to the general case of Definition 2.

The definition of CI generalizes Definition 2 by considering multivariate time sequences as follows:

Definition 3 (Co-Integration): A $(p + 1) \times 1$ multivariate time sequence $x_t = [x_t^{(1)}, x_t^{(2)}, ..., x_t^{(p+1)}]$ is said to obey a CI model of order $p$, denoted as $x_t \sim CI(p)$, if there exists a $(p + 1) \times 1$ vector $w$ such that $w^T x_t$ is stationary.

Definition 3 clearly specializes Definition 2, as for a time series $x_t$ with integration order $p$, one can equivalently define the co-integrated multivariate time sequence $x_t$ by having its $\tau$th entry, denoted $x_t^{(\tau)}$, set to $x_t^{(\tau)} = x_{t-\tau+1}$ for every $\tau \in \{1, ..., p + 1\}$.

2) Co-Integration in Pairs Trading: CI is often used to model pairs trading. In this framework, the two assets $\alpha_t$ and $\beta_t$ are assumed to obey a CI(1) model (see Def. 3). In particular, we define the spread time sequence $s_t$ via

$$s_t = \beta_t - h \cdot \alpha_t - \mu,$$

and look for $h$ and $\mu$ such that $s_t$ is a zero-mean stationary time series. We refer to $h$ as the hedge ratio and $\mu$ as the equilibrium value. These parameters are typically estimated by rewriting (13) as

$$\beta_t = s_t + h \cdot \alpha_t + \mu,$$

from which $h$ and $\mu$ can be recovered from a set of pairs as in (12) via, e.g., least-squares [57]. Once $h$ and $\mu$ are estimated, one can apply statistical tests to check for stationarity, e.g., Dicky-Fuller test [58] or Johansen test [59].

3) Co-Integration SS Model: An inherent limitation with the above form of CI modeling of pairs trading follows from the fact that $h$ and $\mu$ are assumed to be static. In practice, they might drift, and the resulting model may not be reliable enough to serve as a basis for designing trading policies. However, the CI framework in (14) can be used to form a SS model that supports tracking of the hedge ratio and equilibrium values using, e.g., the KF.

The SS model for pairs trading with the CI framework assumes that $h$ and $\mu$ are first-order Markov processes

$$h_t = h_{t-1} + \epsilon^h_t, \quad \mu_t = \mu_{t-1} + \epsilon^\mu_t,$$

where $\epsilon^h_t$ and $\epsilon^\mu_t$ follow an i.i.d. Gaussian distribution. Accordingly, the relationship between the assets pairs in (14) is replaced with

$$\beta_t = s_t + h_t \cdot \alpha_t + \mu_t.$$

The representation in (15) gives rise to a SS model as follows. The latent state vector is $x_t = [h_t, \mu_t]^T$, the observation is set to be $y_t = \beta_t$, and the underlying dynamics of the SS model are

$$x_t = F_t \cdot x_{t-1} + e_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_{t-1} + e_t,$$

$$y_t = g_t^T \cdot x_t + v_t = [\alpha_t \ 1] x_t + v_t.$$

In (16b), $v_t$ is modeled as white Gaussian observation noise, and the measurement noise is given by $e_t \triangleq [\epsilon^h_t, \epsilon^\mu_t]^T$. Designing trading policies based on (16) requires estimating the parameters as a form of system identification, e.g., the second-order moments of the noise signals $v_t$ and $e_t$, which can be done via maximum likelihood-based estimation [17], expectation maximization [60], autocovariance least-squares [61], or the flexible least squares method [62].

Formulating the evolution of the hedge ratio and equilibrium value as a linear Gaussian SS via (16) allows the tracking of these parameters using the KF. The KF is comprised of prediction and update stages, respectively, given as

$$\hat{x}_{t|t-1} = F_t \cdot \hat{x}_{t-1}, \quad \tilde{y}_{t|t-1} = g_t^T \cdot \hat{x}_{t|t-1},$$

$$\hat{x}_t = K_t \cdot \Delta y_t + \hat{x}_{t|t-1}, \quad \Delta y_t = y_t - \tilde{y}_{t|t-1}.$$
and thus, it describes the difference between the posterior and the prior. Using the fact that $y_t = \beta_t s_t$ and substituting the time-varying relationship between $\alpha_t$ and $\beta_t$ from (18), we obtain that
\[
\Delta y_t = s_t + \left( (h - \bar{h}_{t-1}) + (\mu_t - \bar{\mu}_{t-1}) \right).
\] (19)

From (19), it follows that $\Delta y_t$ is an estimate of the value of the spread corrupted by an additive term that represents the estimation error at time $t$. Thus, $\Delta y_t$ can be used as an equivalent pairwise asset, from which one can extract a valuable indicator for BB as in (4), i.e.,
\[
z_t = \frac{\Delta y_t}{\sigma^y_t}.
\] (20)

Here, $\sigma^y_t$ is the STD of $\Delta y_t$ (which is also tracked by the KF for computing the KG).

### B. Partial Co-Integration in Pairs Trading

While the CI model for pairs trading gives rise to a simple KF-aided trading policy, it is limited in the sense that it does not incorporate any temporal statistical model on the spread time sequence, which is at the core of the financial indicator. Such a temporal statistical model is incorporated by extending the notion of CI into PCI [17]. In the following, we first recall the PCI definition, after which we review its associated SS model proposed in [17], and then further extend it to propose the SS model for pairs trading used in our derivation in Section IV.

1) Partial Co-Integration: An extended definition of CI is that of PCI [17]:

**Definition 4 (Partial Co-Integration):** A multivariate time sequence $\tilde{x}_t$ is said to obey a PCI model of order $p, q$ denoted $\tilde{x}_t \sim \text{PCI}(p, q)$, if there exists a vector $\tilde{w}_t$ such that the time series $\tilde{w}^T \tilde{x}_t$ is a combination of scalar time sequences that are integrated in time with orders $q$ and $p$. Namely, it can be decomposed into $\tilde{w}^T \tilde{x}_t = \mu_t + \bar{s}_t$ where $\mu_t \sim \text{WI}(q)$ and $\bar{s}_t \sim \text{WI}(p-q)$.

Note that any multivariate time sequence (Definition 3) is also PCI and is obtained when $q = p = 0$, which results in $\tilde{w}^T \tilde{x}_t$ being a stationary time series.

2) Partial Co-Integration in Pairs Trading: As suggested in [17], the PCI model can be used to generalize the CI model for pairs trading. In this case, it is assumed that $[\beta_t, \alpha_t] \sim \text{PCI}(1, 1)$, i.e.,
\[
\begin{align*}
\beta_t &= s_t + h \cdot \alpha_t + \mu_t, \quad (21a) \\
\alpha_t &= \rho s_{t-1} + \epsilon^w_t, \quad (21b)
\end{align*}
\]
where
\[
\begin{align*}
\mu_t &= \mu_{t-1} + \epsilon^\mu_t. \quad (21c)
\end{align*}
\]

The PCI model for pairs trading preserves the notions of spread ($s_t$) and hedge ratio ($h$) - where in this model, the hedge ratio is constant unlike in the CI model which it was modeled as time-varying. It introduces the parameter $\rho \in (-1, 1)$ as an autoregressive coefficient, while $\epsilon^w_t, \epsilon^\mu_t$ are stationary signals, often modeled as mutually independent Gaussian white noises [17].

The PCI framework in (21) introduces two main degrees of freedom compared with the CI model in (14): (i) First, the CI model does not explicitly capture the temporal correlation in the spread time sequence $s_t$, where the PCI model incorporates such correlation via its autoregressive modeling and the parameter $\rho$; (ii) PCI captures temporal variations in the equilibrium value $\mu_t$, by modeling it as a random walk.

3) Partial Co-Integration SS Model: The SS model proposed in [17] for using the PCI representation in (21) for trading policies sets the latent state vector to be $x_t = [\alpha_t, s_t, h_t]^T$, the observation to be $y_t = [\beta_t, \alpha_t]^T$, and the underlying dynamics of the SS model are
\[
\begin{align*}
y_t &= H_t x_t = \begin{bmatrix} h & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} x_t, \quad (22a) \\
x_t &= F_t x_{t-1} + q_t \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \rho & 0 \end{bmatrix} x_{t-1} + \begin{bmatrix} \epsilon^s_t \\ \epsilon^\alpha_t \\ \epsilon^\beta_t \end{bmatrix}. \quad (22b)
\end{align*}
\]

In (22), $q_t$ is a multivariate white Gaussian noise.

Note that the SS model in (22), which is the one suggested in [17], differs from the CI-based SS model of (16) not only in its usage of the PCI model but also in the fact that the spread is considered in the state vector, while the hedge ratio is assumed to be static. Accordingly, one can track the state using the KF (given in (17a)-(17b)) respectively, and use the tracked spread to form the indicator used for BB trading as
\[
z_t = \frac{\bar{s}_t}{\sigma^s_t}. \quad (23)
\]

where $\sigma^s_t$ is the STD of the spread.

4) Proposed Partial Co-Integrated SS Model: While the PCI model in (22) offers additional degrees of freedom in capturing temporal dependencies in the spread, it is associated with two main drawbacks: (i) the hedge ratio is modeled as being static, which, as stated before, might not always be the case; and (ii) the model assumes that all the noises are Gaussian, i.i.d. and mutually independent. In financial data, this is rarely the case [14, Ch. 3]. Accordingly, we propose an alternative SS model for pairs trading, which shares the improved temporal degrees of freedom of the PCI model while accounting for the dynamic nature of the hedge ratio without imposing a specific distribution on the stochasticity.

Our proposed SS Model is similar to the CI SS model in (16). At the same time, we also add the assumed autoregressive behavior of the spread in (21b). Specifically, our latent space vector is $x_t = [h_t, \mu_t, s_t]^T$, the observation is $y_t = \beta_t$, and the dynamics of the systems are modeled as
\[
\begin{align*}
x_t &= F_t x_{t-1} + e_t = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & \rho \end{bmatrix} x_{t-1} + e_t, \quad (24a) \\
y_t &= g^T_t x_t + v_t = \begin{bmatrix} \alpha_t & 1 & 1 \end{bmatrix} x_t + v_t. \quad (24b)
\end{align*}
\]

It is emphasized that we do not impose a specific distribution on the noise vectors $e_t$ and $v_t$, unlike in (16) and (22), where Gaussian noise are assumed.
IV. NEURAL AUGMENTED TRADING POLICY

We are interested in exploiting the SS model representation in (24) for formulating a trading policy following established approaches combining KF and BB policies, e.g., [17]. However, the application of such methodologies gives rise to the following challenges:

C1 modeling Stochasticity: KF inherently assumes Gaussian noise. This is unlikely to hold in financial data, and the noise tends to follow a heavy-tailed distribution [63].

C2 modeling Mismatches: the models for $F_t$ and $g_t$ in (24), which follow from the PCI, are an approximation of the complex real-world dynamics governing pairs trading. Therefore, a model mismatch is inevitable.

C3 State Tracking: Tracking based on KF is inherently designed to provide an accurate estimate of its state vector $\mathbf{x}_t$. Here, $\mathbf{x}_t$ is comprised of model parameters that do not necessarily represent a physical quantity but are rather intermediate variables used for trading policy. One is really interested in obtaining a representation that is most useful for pairs trading in the PNL sense. However, the conventional KF is tailorly designed to optimize tracking the state, which is here assumed model parameters, though this might be sub-optimal for maximizing the PNL.

To enable pairs trading under C1–C3, we propose a hybrid model-based/data-driven algorithm based on neural augmentation of KF-BB pairs trading. We present the proposed KBPT and its training procedure in Sections IV-A-IV-B, respectively, and a discussion is provided in Section IV-C.

A. KalmanNet-Aided Bollinger Bands Pairs Trading

1) High Level Design: We design our pairs trading scheme to preserve the interpretable and reliable operation of the combination of KF with BB trading under our proposed SS model in (24). To cope with challenges C1-C2, we augment the operation of the KF, and specifically its computation of the KG, with a RNN, while preserving the operation of the KF. Accordingly, the prediction and update stages of KalmanNet are given by

\[
\hat{x}_{t|t-1} = F_t \cdot \hat{x}_{t-1}, \quad \hat{y}_{t|t-1} = g_t^T \cdot \hat{x}_{t|t-1}, \quad (25a)
\]

\[
x_t = K_t(\theta) \cdot \Delta y_t + \hat{x}_{t|t-1}, \quad \Delta y_t = y_t - \hat{y}_{t|t-1}. \quad (25b)
\]

The sole difference between (25) and (17) is that here in the update stage in (25b) utilizes the KG computed by the RNN, instead of computing it from knowledge of the underlying architecture of the RNN, while preserving the operation of the KF. Accordingly, the innovation produced by KalmanNet is passed to the indicator extractor which outputs its $Z$-score, denoted $z_t$, via

\[
z_t = \frac{\Delta y_t}{\hat{\sigma}_t}, \quad (26)
\]

In (26), $\hat{\sigma}_t$ is the empirical STD of the innovation process calculated using a rolling window. In our experimental study reported in Section V, we used a window size of 80 samples based on empirical trials. The architecture of KalmanNet with the indicator extractor is depicted in Fig. 4.

The indicator in (26) is selected for its ability to facilitate stable learning of the overall trading policy based on the PNL objective. In BB-based pairs trading policies, the indicator is often chosen to correspond to the spread. For instance, in the trading policy based on the CI model, described in Section III-A3, the indicator was chosen to be $\Delta y_t$, as it encapsulates the spread, as can be seen by the derivation in (19). Also, in the trading policy based on the PCI model of [17], described in Section III-B3, the spread is explicitly tracked as $s_t$, and this feature is therefore is used as the indicator. Our formulation in (26) combines these approaches. Even though our proposed method is influenced by the PCI model, we use the $Z$-score of $\Delta y_t$ as the indicator. This is because, upon further numerical evaluations on different pairs, we consistently observed that using $\Delta y_t$ as the indicator allows achieving better and more stable learning. Specifically, the usage of this indicator was shown to facilitate training KalmanNet to maximize the overall

![Fig. 3. KBPT pipeline.](image-url)
multiplied by $\zeta$ based on the current hedge ratio, i.e., is the position taken on the spread trade policy on $z$.

PNL, yielding learned features that most suit the subsequent BB-based policy.

The final component of our trading policy executes the BB trade policy on $z_t$. Its output is the position to take at time $t$ denoted as $p_t$ based on (7) for opening and (9) for closing. The resulting overall trading policy is summarized as Algorithm 1.

4.) Reward: The BB policy is stated in Section II-B for a single asset. Here, since we consider pairs trading, an open position for the $i$th transaction at time $t^i_c$ implies open positions on $\alpha, \beta$ in the sum of 18. Its division between the assets is based on the current hedge ratio, i.e., $\hat{h}_{i|^c}$. Similarly, when the position is closed at time $t^i_c$, we close the open positions on $\alpha, \beta$ according to the hedge $\hat{h}_{i|^c}$. Accordingly, the reward of each asset in such a transaction is the sum of the rewards in $\alpha$ and in $\beta$, i.e.,

$$r_{i^c,\pi}^\phi = r_{i^c,\pi,\beta}^\phi + r_{i^c,\pi,\alpha}^\phi.$$  \hspace{1cm} (27)

To formulate the individual reward terms in (27), recall that $\alpha_{D_{ij}}$ is the position taken on the spread. Consequently, to set the position taken with respect to $\beta$ and $\alpha$ individually it should be multiplied by $\zeta_{i^c} \triangleq \text{sign}(\beta_{i^c} - \hat{h}_{i|^c}\alpha_{C^c})$ and $-\zeta_{i^c}$, respectively, to determine what position is taken on each individual asset given a position on the spread. The resulting reward terms are thus

$$r_{i^c,\pi,\beta} = \frac{\beta_{i^c}}{1 + |\hat{h}_{i^c}|} - \frac{\beta_{i^c}}{1 + |\hat{h}_{i^c}|} \cdot \alpha_{D_{ij}} \cdot \zeta_{i^c},$$  \hspace{1cm} (28a)

$$r_{i^c,\pi,\alpha} = \frac{|\hat{h}_{i^c}|\alpha_{i^c}}{1 + |\hat{h}_{i^c}|} - \frac{|\hat{h}_{i^c}|\alpha_{i^c}}{1 + |\hat{h}_{i^c}|} \cdot \alpha_{D_{ij}} \cdot \zeta_{i^c}.$$  \hspace{1cm} (28b)

As in [4], [64], we consider pairs trading in which the hedge ratio $h$ is allowed to change over time. Accordingly, one is likely to have that $\hat{h}_{i|^c} \neq h_{i|^c}$, implying that the trader may open and close with different quantities of each of the assets. In Section IV-C, we further discuss this property.

**B. Training Procedure**

The trading policy is stated in Algorithm 1 for a given parameterization $\theta$. The tuning of these parameters is carried out based on a data set comprised of a sequence of measured asset pairs $D$ given in (12). To train $\theta$, we use a two-step training procedure, which first trains KalmanNet separately from the trading task for stability and then adapts the overall architecture based on the PNL, thus tackling C3. These two steps are detailed next.

1) Task-Ignorant Training: This first step acts as a warm start. It tunes $\theta$ to be good initial values for training the overall policy to maximize the PNL in Step 2. To that aim, we train only KalmanNet with the objective of optimizing the tracking of the state vector $x_t$ with stochastic gradient descent.

In its original formulation in [36], KalmanNet is trained in a supervised manner, i.e., using multiple trajectories of observations and their corresponding gt states. Such a scheme cannot be applied here as the state $x_t$ comprises modeling parameters for which one cannot provide gt. To overcome this challenge, we follow the unsupervised learning procedure proposed in [45], where the training loss is computed based on the predicted measurements and not on the estimated state. The data set

---

**Algorithm 1: Neural Augmented (KBPT) at Time t**

Init: RNN parameters $\theta$

Input: Previous estimate $\hat{x}_{t-1} = [\hat{h}_{t-1}, \hat{\mu}_{t-1}, \hat{s}_{t-1}]^\top$; Assets $\beta_t, \alpha_t$.

KalmanNet:
1) Predict $\hat{y}_{t|t-1}$ via (25a);
2) Update estimate and compute $\Delta y_t$ via (25b);
3) Calculate feature $z_t$ via (26);
4) Compute $p_t$ via (7) and (9) with $z_t$ as argument.

**Fig. 4.** KalmanNet with indicator extraction for the BB trade policy.
using backpropagation, converting Algorithm 1 into a discriminated as Algorithm 3.

The second training step is summarized as Algorithm 2. The training (step 1) procedure is summarized as Algorithm 2.

2) Training End-to-End: In this second and last step, we train KalmanNet in an end-to-end manner based on the overall system task of pairs trading with full gradient descent. We adapt the 

\[ \hat{\theta} \] to maximize the PNL, namely, our loss function is

\[ L_2(\theta) = -\sum_{i=0}^{N} r_i^{\gamma,\pi}. \]  

(30)

In (30), \( r_i^{\gamma,\pi} \) is the \( i \)th reward given in (27) and \( N \) is the number of active position taken.

A core challenge to optimize the trainable parameters with respect to (30) stems from the non-differentiable nature of the policy in (7) and (9). This prevents conventional deep learning optimizers based on gradient-descent, from being used. We overcome this challenge by approximating (7) and (9) with a differentiable mapping during training. Accordingly, we replace the unit step function with a parametric surrogate [65], \( \hat{u}_p(\cdot) \), i.e., we compute the positions using (7) and (9), while taking the gradient of (30) assuming their surrogate approximations

\[ \text{op}_t = \left( \hat{u}_p(-1-z_t) - \hat{u}_p(z_t-1) \right) \cdot \hat{u}_p(\tau_{op,t} - \tau_{op,t}), \]  

(31)

and

\[ \text{cp}_t = \hat{u}_p(-(z_t-1) \cdot z_t) \cdot (1 - \hat{u}_p(\tau_{op,t} - \tau_{op,t})), \]  

(32)

respectively. Following [65], we set the surrogate approximation \( \hat{u}_p(\cdot) \) to be the cumulative distribution function of a zero-mean Gaussian random variable with variance \( \gamma^2 \), i.e.,

\[ \hat{u}_p(x) = \int_{x=\infty}^{x} \frac{1}{\sqrt{2\pi\gamma^2}} \exp \left\{ -\frac{\tau^2}{2\gamma^2} \right\} d\tau. \]  

(33)

The differentiable nature and the simple derivative of the Gaussian function in (33) enables computing the gradients of (30) using backpropagation, converting Algorithm 1 into a discriminative trainable model [46]. The second training step is summarized as Algorithm 3.

C. Discussion

KBPT jointly exploits both the approximated SS model representation in (24) along with a principled augmentation of deep learning techniques. This hybrid model-based/data-driven framework utilizes the strength of each one the approaches – both classical techniques based on SS models as well as data-driven deep learning tools. On one hand, we have partial domain knowledge, which we approximate using the PCI model of the system dynamics. This lets us maintain the interpretability and transparency of the algorithm, which is crucial, especially when one wants to invest based on such an algorithm. On the other hand, we use data and deep learning capabilities to overcome challenges elegantly C1–C3.

As empirically shown in Section V, our hybrid policy allows us to surpass the performance of other fully model-based or fully data-driven frameworks. The fact that KBPT exploits the trainability of KalmanNet and learns its tracking mapping from data based on the PNL yields a tracking rule that does not necessarily give the most accurate tracking but is useful for trading, which is our primary objective. This gain of KBPT follows from its ability to learn to overcome the inherent mismatches of describing pairs trading as a SS model, as well as the need to estimate the model parameters. We achieve this by utilizing the fact that KalmanNet does not need to estimate the SS model parameters explicitly but instead learns to compute the KG such that the PNL is maximized, which is the primary goal. In addition, due to the interpretability of the framework, at each point in time, one can monitor the tracked latent state vector of the policy and understand why each action was taken.

Integrating model-based deep learning techniques with financial decision-making paves the way for many avenues for future exploration. Our derivation of Algorithm 1 does not account for the transaction cost of each buy/sell action. As this is often a significant factor in deciding if to perform such an action, one can potentially add a penalty term for each transaction made to the PNL function. In that way, the second step of the training (Algorithm 3) will optimize with regard to the transaction costs. Nonetheless, it is empirically shown in Section V that even without explicitly accounting for transaction cost, KBPT still typically achieves its improved PNL with fewer

---

Algorithm 2: Training Step 1

- **Init**: Randomly initialize \( \theta \), step size \( \eta_1 > 0 \)
- **Input**: Initial estimate \( \hat{x}_0 = [\hat{h}_0, \hat{\rho}_0, \hat{s}_0]^T \)
- **Data set** \( D \)

for each training epoch do

1. Divide \( D \) into \( Q \) batches \( \{D_q\}_{q=1}^Q \)

for each \( q \) do

1. for each \( t \) do

1. Predict \( \hat{y}_{t|t-1} \) via (25a)

2. Update estimate via (25b)

3. Compute \( L_{D_{q},1}(\theta) \) via (29)

4. Update \( \theta \leftarrow \theta - \eta_1 \nabla L_{D_{q},1}(\theta) \)

end for

end for

end for

end for

Algorithm 3: Training Step 2

- **Init**: Step size \( \eta_2 > 0 \)
- **Input**: Initial estimate \( \hat{x}_0 = [\hat{h}_0, \hat{\rho}_0, \hat{s}_0]^T \)
- **Data set** \( D \)
- Pre-trained parameters \( \theta \) from Step 1.

for each training epoch do

1. for each \( t \) do

1. Compute \( p_t \) via Algorithm 1 with \( \theta \)

2. Compute \( L_2(\theta) \) via (30)

3. Compute \( \nabla L_2(\theta) \) with (7) and (9) approximated as (31) and (32), respectively

4. Update \( \theta \leftarrow \theta - \eta_2 \nabla L_2(\theta) \)

end for

end for
number of trades compared with model-based and data-driven benchmarks. Another potential extension involves the usage of more than two assets. While we tackle the problem of pairs trading based on the statistical relation between two assets, one can generalize this approach and look for a relationship with multiple assets [14, Ch. 10].

Latency performance is another aspect of trading that can benefit from our hybrid model-based/data-driven design. Our experimental study reported in Section V considers daily trading, which means an action is performed once a day. This framework can be implemented in high-frequency trading, where a trader can perform hundreds of actions per day, and the latency of each step becomes a crucial factor. A core gain of such forms of model-based deep learning is their inference speed, which is often translated into much smaller latency than other deep learning models and sometimes also compared with model-based processing, see, e.g., [66]. Likewise, training will be faster compared to fully data-driven approaches, which is highly beneficial in a retraining regime. Therefore, we expect our framework to be well-suited for high-frequency trading, leaving this study for future work.

In addition, as discussed following (28), when the hedge ratio changes over time, one needs to sell/buy more/fewer assets than bought/sold when the position was opened. Such form of pairs trading of time-varying hedge ratio is not unique to our approach and was also considered in, e.g., [4], [64]. A natural approach to tackle this is to have the trader possess a number of $\alpha$ and $\beta$ assets from the start, i.e., not just during open positions. Since the assets are co-integrated, the temporal variations in the hedge ratio are expected to be minor. Therefore, the number of assets needed to begin with will be small. Moreover, one can encourage KalmanNet to produce low variations in its tracked hedge by regularization or even enforce it to be constant if starting with assets is to be avoided. We leave the exploration of this to future work.

Lastly, our derivation of KBPT and its numerical evaluation reported in the sequel is geared towards pairs that exhibit the PCI property. In practice, this assumption may be violated in some settings. Relaxing these assumptions gives rise to several issues that require further research. For once, from a theoretical perspective, the ability of a neural-augmented system to cope with such mismatches requires further research. Second, from a practical point of view, how can one still profit when trading with pairs that occasionally violate the integration modeling assumption. We leave this extension of our algorithm and its analysis to future work.

V. EMPIRICAL EVALUATION

In this section, we numerically evaluate KBPT, comparing it with both SS model-based and data-driven benchmarks. Our experimental study is comprised of four data sets (detailed in Section V-A), and considers four benchmark trading policies (discussed in Section V-B). In our numerical evaluations we first assess the trading procedure and the individual contribution of each of its stages in Section V-C, after which we compare the PNL of KBPT with the considered benchmarks in Section V-D.

A. Experimental Data

We consider four different data sets representing different asset pairs. Each data set is composed of in-sample data (training set) and out-of-sample data (test set), where a sample consists of the daily opening prices of the two assets. The data sets are taken from [67] in such a way that the out-of-sample set starts one trading day after the end of the in-sample set.

We particularly use the following assets pairs

1) **Swiss Frank - Euro (CHF - EURO)** - Consists of 2000 in-sample pairs, ending on 23/06/2019, and 944 out-of-sample pairs, starting from 24/06/2019.

2) **Australian Dollar - South African Rand (AUD - ZAR)** - Consists of 2000 in-sample pairs, ending on 20/07/2017, and 1500 out-of-sample pairs, starting from 21/07/2017.

3) **Canada ETF - Australia ETF Period A (EWC - EWA - A)** - Consists of 2000 in-sample pairs, ending on 29/01/2010, and 1500 out-of-sample pairs, starting from 01/02/2010.

4) **Canada ETF - Australia ETF Period B (EWC - EWA - B)** - Consists of 2000 in-sample pairs, ending on 26/01/2017 and a 1500 out-of-sample pairs, starting from 27/01/2010.

These pairs were specifically selected due to the fact that they were found to be suitable for pairs trading, in the sense that they exhibit the CI property, see e.g., [4], [14], [64], [68].

B. Benchmarks

We compare KBPT (Algorithm 1) with both model-based and data-driven policies, using the following benchmarks:

B1 KF with the CI SS model as detailed in Section III-A, followed by BB policy in (7) and (9).

B2 KF with the PCI SS model as detailed in Section III-B, followed by BB policy in (7) and (9).

B3 Neural augmented KF-BB policy based on the CI SS model, i.e., Algorithm 1 assuming the SS model of Section III-A instead of our proposed SS model of detailed in Section III-B.

B4 A deep RL trading policy based on the Double Deep Q Network (DDQN) architecture proposed in [31], comprised of an input layer of 10 features, 2 fully connected layers of 50 neurons, and a 3 neurons output layer.

Benchmarks B1 and B2 represent conventional model-based policy, wherein B2 is based on our proposed SS formulation. Benchmark B4 is purely data-driven, while B3 is a model-based/data-driven approach, which is based on our hybrid algorithm without incorporating the proposed SS formulation.

All data-driven algorithms, i.e., KBPT as well as Benchmarks B3 and B4, were trained using the Adam optimizer [69], with hyperparameters chosen by empirical trials. For KalmanNet, we used Architecture 2 of [36] for the KG RNN. In the model-based policies, i.e. B1 and B2, we estimated the KF parameters, e.g., the variances of the noise signals, from data. We intentionally used the test set and not on the training set, giving a further

The source code and the complete set of hyperparameters used in our study is available at https://github.com/KalmanNet/KBPT_TSP.
advantage to the model-based approaches, in order to improve their performance and show that even under these conditions, they are still outperformed by our neural augmented design. Likewise was done on the DDQN model of B4, which we trained and evaluated on the test set, giving it a clear advantage over our neural augmented design. Following [31], we trained it based on the instantaneous reward, allowing us to obtain instantaneous loss for training.

C. Evaluation of Training Procedure

We commence our experimental study by numerically evaluating the contribution and usefulness of the different stages of the training procedure detailed in Section IV-B. We particularly focus on their effect on the tracking performance and overall trading PNL, as Step 1 (Algorithm 2) encourages accurate tracking of the assets, while its subsequent Step 2 (Algorithm 3) incorporates the PNL objective.

In Fig. 5 we evaluate KBPT applied to the EURO-CHF pair, comparing its PNL after trained only for tracking (Step 1) and after trained also for PNL (Step 2). Observing Fig. 5, we clearly see that the addition of Step 2 notably boosts the PNL, allowing us to successfully cope with the challenge in incorporating this key consideration noted in C3. We note that this improvement in PNL is not unique to our selection of the indicator in (26), and similar results are also obtained when using other indicators, e.g., (23). However, after extensive numerical evaluations, we systematically observed that our selected indicator results in stable training with a dominant increase in PNL, as exemplified in Fig. 5. It is emphasized, though, that training solely based on Step 2 (i.e., without preceding warm start via Step 1) makes learning much more challenging, and the training procedure often diverges.

To understand how training based on the PNL rather than for state estimation affects the tracking performance, we depict in Fig. 6 the difference between the estimated \( \hat{\beta}_t \) after each step and the real \( \beta_t \) (which is the EURO price in USD). It is observed in Fig. 6 that (unsupervised) training based on the state estimation objective in Step 1 yields an accurate tracking of the assets. However, training based on the PNL degrades the tracking performance, outputting a surrogate value that is most useful for trading in the sense of maximizing the PNL. While Fig. 6 focuses only on the EURO-CHF data, Table I reports that tracking degradation (in mean-squared error (MSE)) is systematically converted into improved PNL in all of the pairs after the second training step. This study demonstrates the validity of our proposed two-step training procedure and how it enables learning to track in a manner that contributes to the desired PNL objective.

D. Evaluation of PNL

We proceed to evaluate the PNL and its evolution in time as achieved by KBPT and compared with benchmarks B1-B4 detailed in Section V-B. We first consider the CHF-EURO pair and report the obtained PNL versus day index \( t \) in Fig. 7. Observing Fig. 7, we note that both trading policies following our model-based deep learning approaches, i.e., Algorithm 1 and B3, outperform both the black-box data-driven DDQN (B4) as well as the fully model-based benchmarks B1 and B2. Comparing the PNL of Algorithm 1 with B3, as well as that of B2 with B1, it is consistently shown that our proposed PCI SS model allows achieving improved PNL, for both hybrid model-based/data-driven policies as well as purely model-based ones.

Moreover, we observe in Fig. 7 (as well as in the following Figs. 8-10) that the DDQN approach, which relies on highly parameterized deep networks trained via RL, achieves similar
PNL values to that of the much simpler KF-BB policy. It is noted that this is due to (i) the DDQN uses a constant hedge ratio; (ii) RL is notably facilitated when provided with instantaneous rewards; hence, it is trained to provide policies based on an instantaneous reward. When the model-based KF-BB policy also determines its position based on the instantaneous rewards rather than the cumulative one, it is outperformed by the DDQN, as reported in Table II.

The findings reported in Fig. 7 are not unique to the CHF-EURO pair, and are also reproduced in Fig. 8. There, we report the PNL of a different pair, which is also from the foreign exchange market, i.e., AUD-ZAR. We systematically observe both the superiority of a hybrid model-based/data-driven design and the contribution of properly incorporating the expected variation profile of the spread via our SS model formulated in (24).

We demonstrate that the benefits of our design combining model-based deep learning with PCI -based SS modeling are also consistent over different markets. In Figs. 9-10, we report the PNL of EWC-EWA, which are two ETLs, under two different time periods. These results showcase that the benefits of our design in terms of imported PNL are consistent over time and over markets that are known to be faithfully described as obeying some level of co-integration. The jump in the PNL in Fig. 9, which is seen in all of the models, is arguably due to a global crash experienced by many stocks at the beginning of 2020, most likely due to the wide epidemic crisis. This caused a sudden jump in the hedge ratio estimation of all the models, which affected the trading policy.

| Pair       | B1  | B4  |
|------------|-----|-----|
| CHF-EURO   | 0.051 | 0.149 |
| AUD-ZAR    | 0.27 | 1.11 |
| EWC-EWA-A  | 0.62 | 1.09 |
| EWC-EWA-B  | 0.43 | 1.57 |

TABLE II
COMPARISON OF FINAL PNL WITH INSTANTANEOUS REWARD
TABLE III

| Pair          | Metric                  | B1  | B2  | B3  | B4  | KBPT |
|--------------|-------------------------|-----|-----|-----|-----|------|
| CHF-EURO     | Number of trades        | 137 | 146 | 193 | 557 | 57   |
|              | Annual return [%]       | 6.4 | 6.4 | 42  | 6   | 70.8 |
|              | Mean return per trade [%]| 0.12| 0.11| 1.72| 0.02| 3.11 |
|              | Average holding time per trade [days] | 5.4 | 5.0 | 12.6| 1   | 13.2 |
|              | Average time between returns [days] | 6.1 | 5.7 | 14.1| 1.6 | 15.0 |
| AUD-ZAR      | Number of trades        | 331 | 294 | 144 | 1054| 88   |
|              | Annual return [%]       | 26.9| 26.6| 157 | 26.4| 218.8|
|              | Mean return per trade [%]| 0.34| 0.38| 4.59| 0.10| 10.44|
|              | Average holding time per trade [days] | 3.6 | 4.1 | 8.9 | 1   | 14.9 |
|              | Average time between returns [days] | 4.2 | 4.8 | 9.8 | 1.4 | 16.4 |
| EWC-EWA-A    | Number of trades        | 335 | 322 | 282 | 1071| 275  |
|              | Annual return [%]       | 33.3| 32.6| 44.5| 25.9| 141.6|
|              | Mean return per trade [%]| 0.39| 0.41| 0.66| 0.10| 2.16 |
|              | Average holding time per trade [days] | 3.5 | 4.4 | 5.1 | 1   | 4.1  |
|              | Average time between returns [days] | 3.9 | 5.0 | 5.9 | 1.3 | 5.2  |
| EWC-EWA-B    | Number of trades        | 340 | 342 | 415 | 1126| 278  |
|              | Annual return [%]       | 40.7| 40.4| 42.8| 37.3| 83.5 |
|              | Mean return per trade [%]| 0.50| 0.49| 0.43| 0.14| 1.26 |
|              | Average holding time per trade [days] | 3.6 | 3.07| 3.2 | 1.45| 4.5  |
|              | Average time between returns [days] | 4.1 | 4.1 | 3.43| 1.3 | 5.1  |

We also consistently observe the non-decreasing monotonic behavior of the cumulative PNL in Figs. 7-10. This monotonicity is a direct byproduct of the usage of the cumulative PNL detailed in Section II-B, which opens and closes according to (7) and (9) respectively, and the return terms in (28). Specifically, a position is only closed when the spread crosses the equilibrium (as illustrated in Fig. 2); thus, a non-negative reward is guaranteed. When a position is opened but the spread has not yet crossed the equilibrium value, then the assets are held, and accordingly, the PNL will neither increase nor decrease. Another cause for the plateaus observed in some of the curves in Figs. 7-10 is the scenario where after closing a position, the spread remains between the top and the bottom thresholds of the BB policy. In such cases, no position is taken, and the PNL will not change.

The results reported so far evaluated trading in terms of PNL. In practice, one is often interested in additional trading statistics, including the number of trades, annual return, mean return per trade, average holding time, and the average time between returns [17], [32]. In Table III, we summarize these trading statistics of the different pairs achieved by benchmarks B1-B4 and our KBPT. We show that our proposed trading policy performs fewer trades in the trading period but still achieves higher profit than other models. Although in our design, we did not explicitly take into account the transaction cost by penalizing each trade transaction, we still obtained a trading policy that outperforms other trading policies and requires fewer transactions to do so.

VI. CONCLUSION

We proposed KBPT, a hybrid model-based / data-driven trading policy that converts KF-BB pairs trading into a trainable architecture. It is based on an extended SS model obtained from assuming partial co-integration. KBPT utilizes the recent KalmanNet to learn to track while coping with the inherent mismatches in the underlying SS model. Training is done in a two-step manner, combining unsupervised learning of KalmanNet with approximating the BB policy with a differentiable mapping. Our empirical results show that the proposed policy notably improves KF-BB and deep RL policies while preserving interpretability, simplicity, and low latency.

REFERENCES

[1] H. Deng, G. Revach, H. Mogenstein, and N. Shlezinger, “KalmanBOT: KalmanNet-aided Bollinger bands for pairs trading,” in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), 2023, pp. 1–5.
[2] H. M. Markowitz, “Foundations of portfolio theory,” J. Finance, vol. 46, no. 2, pp. 469–477, 1991.
[3] E. F. Fama and K. R. French, “The cross-section of expected stock returns,” J. Finance, vol. 47, no. 2, pp. 427–465, 1992.
[4] E. Chan, Algorithmic Trading: Winning Strategies and Their Rationale, vol. 625, Hoboken, NJ, USA: Wiley, 2013.
[5] A. N. Akansu, S. R. Kulkarni, and D. M. Malioutov, Financial Signal Processing and Machine Learning. Hoboken, NJ, USA: Wiley, 2016.
[6] M. L. De Prado, Advances in Financial Machine Learning. Hoboken, NJ, USA: Wiley, 2018.
[7] G. Nuti, M. Mirghaemi, P. Treleaven, and C. Yingsaeree, “Algorithmic trading,” Computer, vol. 44, no. 11, pp. 61–69, 2011.
[8] N. Jegadeesh and S. Titman, “Returns to buying winners and selling losers: Implications for stock market efficiency,” J. Finance, vol. 48, no. 1, pp. 65–91, 1993.
[9] J. Bollinger, “Using Bollinger bands,” Stocks Commodities, vol. 10, no. 2, pp. 47–51, 1992.
[10] J. M. Poterba and L. H. Summers, “Mean reversion in stock prices: Evidence and implications,” J. Financial Econ., vol. 22, no. 1, pp. 27–59, 1988.
[11] R. A. Maller, G. Müller, and A. Szimayer, “Ornstein–Uhlenbeck processes and extensions,” in Handbook of Financial Time Series, 2009, pp. 421–437.
[12] K. Benidis, Y. Feng, and D. P. Palomar, “Optimization methods for financial index tracking: From theory to practice,” Found. Trends® Optim., vol. 3, no. 3, pp. 171–279, 2018.
[13] R. S. Tsay, Analysis of Financial Time Series. Hoboken, NJ, USA: Wiley, 2005.
[14] Y. Feng and D. P. Palomar, “A signal processing perspective on financial engineering,” Found. Trends® Signal Process., vol. 9, nos. 1–2, pp. 1–231, 2016.
[15] R. Elliott, J. Van Der Hoek, and W. Malcolm, “Pairs trading,” Quantitative Finance, vol. 5, pp. 271–276, Feb. 2005.
[16] H. Puspaningrum, “Pairs trading using cointegration approach,” Doctor of Philosophy thesis, School of Mathematics and Applied Statistics, Univ. Wollongong, 2012. [Online]. Available: https://ro.uow.edu.au/theses/3450
[68] J. Yu, “Cointegration approach for the pair trading based on the Kalman filter,” in Proc. Int. Conf. Bus. Admin. Data Sci. (BADS), Atlantis Press, 2022, pp. 633–642.

[69] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proc. Int. Conf. Learn. Representations (ICLR), 2015.

Amit Milstein received the B.Sc. (cum laude) degree in electrical and computer engineering from Ben-Gurion University (BGU) of the Negev, in 2021, and he is currently working toward the M.Sc. degree with BGU.

Guy Revach received the B.Sc. (cum laude) and M.Sc. degrees from Andrew and Erna Viterbi Department of Electrical & Computer Engineering, Technion – Israel Institute of Technology, Haifa, in 2008 and 2017, respectively. He completed the master’s thesis under the supervision of Prof. Nahum Shimkin on planning for cooperative multi-agents. Since 2019, he has been a Ph.D Candidate with the Institute for Signal and Information Processing (ISI), ETH Zürich, Switzerland, supervised by Prof. Dr. Hans-Andrea Loeliger. He is a Researcher with a proven industry track record as an Innovator and System Engineer. His research interests include intersection of machine learning with signal processing, explicitly combining sound theoretical principles from classical signal processing with state-of-the-art machine learning algorithms for tracking and detection problems. Before joining ETH Zürich, he worked with the Israeli wireless communication industry for over ten years, initially as a Real-Time Embedded Software Engineer and later as a Software Manager. He was the leading innovator behind state-of-the-art, software-defined radio (SDR) for wireless communication, which was game-changing and groundbreaking in size, weight, and power. As a System Engineer, he defined significant aspects of SDR requirements and architecture, including hardware, software, network, cyber defense, signal processing, data analysis, and control algorithms.

Haoran Deng received the B.Sc. degree in electrical engineering from Shanghai Jiao Tong University, in 2020 and the M.Sc. from ETH Zürich, in 2023. He is currently working as a quantitative trader in the Trading Industry.

Hai Morgenstern received the B.Sc. degree (summa cum laude), the M.Sc. degree (cum laude) and Ph.D. degree in electrical and computer engineering (ECE) from Ben-Gurion University (BGU) of the Negev, Israel, in 2012, 2013, and 2017, respectively. He spent six months as an Exchange Research Student with the Institute of Electronic Music (IEM), Graz, Austria, in 2011. In 2014, he was awarded the Chateaubriand Fellowship. He spent a year with the Institute de Recherche et Co-ordination Acoustique/Musique (IRCAM) in Paris, France, investigating acoustic MIMO systems for room acoustics analysis. He was awarded the short-term postdoctoral Negev grant in 2017 and investigated source localization and separation and processing methods for 3D room acoustics analysis. In 2019, he joined BeyondMinds and investigated deep neural networks with applications in speech and natural language processing, confidence, xAI, and monitoring. In 2020, he joined Ramon.Space, where he researches and develops satellite MIMO communication systems.

Nir Shlezinger received the B.Sc., M.Sc., and Ph.D. degrees in electrical and computer engineering from Ben-Gurion University, Israel, in 2011, 2013, and 2017, respectively. He is an Assistant Professor with the School of Electrical and Computer Engineering, Ben-Gurion University, Israel. From 2017 to 2019, he was a Postdoctoral Researcher with the Technion, and from 2019 to 2020, he was a Postdoctoral Researcher with Weizmann Institute of Science, where he was awarded the FGS Prize for outstanding research achievements. His research interests include communications, information theory, signal processing, and machine learning.