The Journal of Supercomputing (2020) 76:2022–2038
https://doi.org/10.1007/s11227-019-02838-1

Enhancing the context-aware FOREX market simulation using a parallel elastic network model

Antonio V. Contreras[1] · Antonio Llanes[3] · Francisco J. Herrera[1] · Sergio Navarro[1] · Jose J. López-Espín[2] · José M. Cecilia[3]

Published online: 6 April 2019

Abstract
Foreign exchange (FOREX) market is a decentralized global marketplace in which different participants, such as international banks, companies or investors, can buy, sell, exchange and speculate on currencies. This market is considered to be the largest financial market in the world in terms of trading volume. Indeed, the just-in-time price prediction for a currency pair exchange rate (e.g., EUR/USD) provides valuable information for companies and investors as they can take different actions to improve their business. The trading volume in the FOREX market is huge, disperses, in continuous operations (24 h except weekends), and the context significantly affects the exchange rates. This paper introduces a context-aware algorithm to model the behavior of the FOREX Market, called parallel elastic network model (PENM). This algorithm is inspired by natural procedures like the behavior of macromolecules in dissolution. The main results of this work include the possibility to represent the market evolution of up to 21 currency pair, being all connected, thus emulating the real-world FOREX market behavior. Moreover, because the computational needs required are highly costly as the number of currency pairs increases, a hybrid parallelization using several shared memory and message passing algorithms studied on distributed cluster is evaluated to achieve a high-throughput algorithm that answers the real-time constraints of the FOREX market. The PENM is also compared with a vector autoregressive (VAR) model using both a classical statistical measure and a profitability measure. Specifically, the results indicate that PENM outperforms VAR models in terms of quality, achieving up to 930× speed-up factor compared to traditional R codes using in this field.

Keywords FOREX simulation · Trading · Context-aware · Big data · Bioinspired computing · Parallel computing

Antonio Llanes
allanes@ucam.edu

Extended author information available on the last page of the article

1 Introduction

The foreign exchange (FOREX) market is one of the main financial markets over the world. It daily manages \$4 billion of US dollars (USD) regarding the Bank for International Settlements (BIS) [2]. This market has traditionally been limited to large international banks, but new technologies are democratizing the access to this market [32]. Both the high liquidity and the flexibility of the schedule make FOREX market very attractive for the particular investor. The difficulty of manipulating the price due to the high liquidity in this market makes it even more attractive for any type of investor, because it is very complicated for either banks or large funds to take control on the price. In addition, the high volume of the contracts and the availability of datasets comprising historical time series provide an interesting framework for scientific and financial research [24].

Moreover, the FOREX market has several characteristics that make its simulation in silicon-based architectures very challenging. Firstly, the trading volume is huge as previously commented. The amount of information generated is unmanageable with real-time constraints looming on the horizon [18]. This problem is even more important since the FOREX market is continuously operating (24 h per day except weekends). Thus, parallelism is mandatory to deal with such big data problem. Secondly, a variety of factors drastically affect exchange rates. These factors include context-aware information. For instance, the quotation EUR/USD (EURUSD) is 1.1634, that is the price of the Euro in US dollars (i.e., 1 euro = 1.1634 dollars). This price is intrinsically affected by other quotations like GBP/USD or GBP/EUR, leading to a fully connected network of influence.

The setting of this paper is the efficient market hypothesis (EMH) postulated by Fama [8, 9] and Samuelson [27] in an environment of acceptance of the efficiency of the capital markets [20]. The EMH defends that the market is efficient and it is not affected by any new event or information because it absorbs them by correcting itself [23]. In 1970, Fama introduced a classification of EMH based on three forms of efficiency: weak, semi-strong and strong [7]. The weak form of EMH assumes that the current prices of financial assets include all the historical financial information. The semi-strong form defends that the prices of financial assets incorporate the weak form of EMH plus all the information available on the market at any moment, with prices changing fast to incorporate new information on the market. The strong form of EMH implies the weak and semi-strong forms of EMH, depending on all the public information available, and also the private information of a financial asset [31].

The literature concerning both testing the EMH and forecasting foreign exchange rates has focused on applying two types of methods. Firstly, econometric analysis has been traditionally applied for many years in order to tackle with this task. Vector autoregressive (VAR) models [19] that were advocated by Sims [30] have been used extensively in order to explain financial markets and testing the efficient market hypothesis [22, 28] as well. VAR models exploit the fact that the contemporaneous realization of a group of variables which are simultaneously related could be explained not only to the won past of a variable but also to that of all other variables. VAR models are very plausible in finance due to the random character of financial markets so that the best predictor of these markets is past values of their quotations. Regarding other methods in which econometric time series analysis has been applied, they can be studied in-depth in [11, 17], or in [26] and [1], where the latter is applied to the FOREX market.

Other methods widely used in the literature for forecasting the FOREX market are machine learning methods [6, 21]. Those methods are well suited for the FOREX-like simulation as they "teach" computers to recognize hidden patterns that may influence in increasing or decreasing the exchange prices. Applying machine learning in financial markets makes mandatory to solve not only a problem in databases, but it also has to deal with artificial intelligence in order to learn how to decide at each moment. Several authors have proved modeling financial markets through several machine learning techniques like neural networks [12, 32], support vector machines [13], fuzzy neural networks [16] and genetic algorithms [3, 25].

In this paper, we design a novel bioinspired algorithm to deal with FOREX market simulations in real time within the framework of the weak form of EMH. This algorithm is inspired in the biomolecule motions widely used to study large-scale dynamics in several structural biological scenarios [14]. In particular, we use elastic network models and normal model analysis [10] to simulate the currencies trading in the FOREX market like a biomolecule motions. The structure of a complex neural network [29] is represented as a fully connected graph $R(U, \varepsilon)$, where $U = \{U_1, U_2, U_3 .. \ U_N\}$ are the nodes and $\varepsilon = \{e_1, e_2, e_3, .. \ , e_M\}$ are the edges that connect pairs of nodes. In our case, the nodes are the currencies and the quotation between both is the edges. However, those edges are not static in any meaningful sense, and they are springs that are able to adapt to market situations.

With the purpose of testing the validity of the model proposed, the out-of-sample forecasting accuracy is compared with a classical VAR model. In order to make the comparison, the mean square error (MSE) metric is used to compare them in terms of quality as it is the most representative forecasting accuracy metric within the econometric sphere. Moreover, the profit factor (PF) [5] is compared between the two compete models with the purpose of analyzing the profitability of each model.

With this in mind, major contributions of this paper include the following:

1. We develop a novel bioinspired algorithm inspired by elastic network models to deal with FOREX market simulations in order to capture the context-aware information about different currency pair exchange rates. This model is called parallel elastic network model (PENM).
2. We parallelize this novel algorithm using OpenMP and MPI to meet real-time constraints of the FOREX market simulations.
3. An in-depth evaluation is carried out to search for the best algorithm configuration on large computing clusters.
4. We test whether the search for performance is translated into an actual benefit in the quality of the results. (Reductions in execution time do not necessarily mean a better affinity quality.)

5. We compare the out-of-sample accuracy between both the model proposed and a classical VAR model. The out-of-sample forecasting accuracy is compared using the mean square error and the profit factor.

The rest of the paper is structured as follows. Section 2 shows the parallel elastic network model (PENM) based on elastic network behavior. Section 3 includes the empirical results in which both comparison with VAR model and the experimental setup and results are presented. Section 4 summarizes the conclusions and gives some directions for future work.

2 Parallel elastic network model for the FOREX market simulation

This section introduces the parallel elastic network model (PENM) for the FOREX market simulation. Firstly, we briefly show the sequential baselines before the parallelization approach based on OpenMP and MPI is discussed. As previously explained, the FOREX market is based on the price variation for the quotations of representative currencies. These variations show the interaction of all stakeholders through small fluctuations caused by the interaction of market players. National banks act as entities to regulate the currency price. However, this regulation influences the price of other currencies that may be overall increased or decreased. Indeed, this behavior could be modeled as a random movement, which is actually the initial hypothesis for the PENM algorithm. The PENM simulates the FOREX market randomness nature through Brownian movements like macromolecules in dissolution.

With that in mind, we create the PENM model to reproduce this fully connected environment nature of the FOREX market. The PENM model translates the FOREX market into a complete graph where nodes are currencies (e.g., EUR, USD and GBP) and links between nodes are the currency quotations (e.g., EUR/USD). Inspired by elastic network models, links are actually springs that are modeled using Hookian potential to regulate the interaction between the values of quotes (see Fig. 1). Like the molecular behavior, whenever an atom (i.e., coin in our case) is excited, other atoms (coins) react looking for the overall system equilibrium. The values of all currencies whenever this equilibrium point is reached are exactly the PENM outputs. Indeed, the PENM algorithm is a context-aware application to predict the currency quotation, given by the relationship between groups of currencies for a given time period.

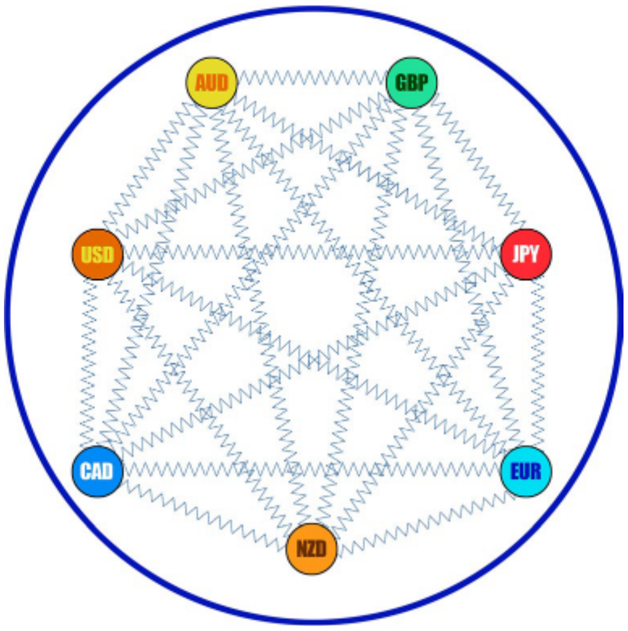

Fig. 1 The elastic network model as a fully connected graph, inspired in the macro-molecules connections

Algorithm 1 Elastic Network Model as applied to the Forex Market. Sequential baselines.

```
Require: Number of currencies $n \in \mathbb{N}$ , maximum trend $t \in \mathbb{N}$, $GX \in \mathbb{R}^{t\times\frac{n(n-1)}{2}}$,
    $time\_frame \in \mathbb{N}$ and $total\_time \in \mathbb{N}$.

Ensure: $prediction \in \mathbb{R}^{1\times\frac{n(n-1)}{2}}$
 1: Define $prices, price_0, price_1, weight\_mean, mean\_GX, sd\_GX \in \mathbb{R}^{1\times\frac{n(n-1)}{2}}$ and
    $potential_0, potential_1, E_0, jump \in \mathbb{R}$
 2: $tray := total\_time/time\_frame$
 3: $mean\_GX$ := calculate mean of $GX$
 4: $sd\_GX$ := calculate standard deviation of $GX$
 5: repeat
 6:     $step$:=0
 7:     $price_0 := GX_{t,:}$ $price_0 := GX_{t,:}$
 8:     while ($tray > step$) do
 9:         $weight\_mean$ := Price_Balance($GX$)
10:         $potential_0$:=Potential($price_0$, $weight\_mean$, $sd\_GX$)
11:         Obtain $jump$ using Monte Carlo method with Gaussian distribution
12:         $price_1 := jump + price_0$              ▷ Adding $jump$ to all the elements of $price_0$
13:         $potential_1$:=Potential($price_1$, $weight\_mean$, $sd\_GX$)
14:         if (Accept_Potential($potential_0, potential_1, E_0$)) then
15:             $price_0 := price_1$
16:             update $GX$ adding $price_1$ in position $t$
17:             $step++$
18:         end if
19:     end while
20:     $prices$ += $price_1$
21: until reach $maximum\_iterations$
22: $prediction := prices/maximum\_iterations$
23: return $prediction$
```

Algorithm 1 shows the PENM sequential baselines. It is based on the Monte Carlo simulation where different runs are executed to optimize the overall system.

This algorithm receives several input parameters that are described below:

- n is the number of currencies to be simulated (e.g., EUR, USD and GBP).
- t is the instant of time to be modeled.
- *GX* is the dataset of historical currency exchange rates up to the instant t. This dataset contains up to t different exchange rates sorted by time, from oldest to most recent time (i.e., t). Therefore, the *GX* is $n \times m$ matrix, where n is the number of links between the targeted currencies (i.e., $\frac{n(n-1)}{2}$) and m is the t historical records our algorithm uses to make the prediction.
- *time_frame* is the difference in minutes between two different quotes.
- *total_time* is the real FOREX market time to be simulated.
- E_0 is the refuse rate or threshold for the Monte Carlo method.

The algorithm repeats a maximum number of iterations (*maximum_iterations*) and excitation-and-return to the point of equilibrium process starting from a instant t. The PENM output will be the exchange currency prices for all pairs that will be achieved by averaging all predictions obtained in all iterations. With that in mind, for each iteration the $price_0$ trajectory will be calculated until the simulation reaches the *total_time*; i.e., the real FOREX market time to be simulated. For instance, *total_time* = 120 min means we are simulating two FOREX market hours. This *total_time* is discretized in *time_frame* minutes. The *time_frame* is difference in minutes between two different quotes. They are usually values of 5 min (m5), 15 min (m15), 60 min (h1), 1440 (d1). Therefore, each trajectory carries a number of steps (*tray*), *total_time*/*time_frame*. Moreover, other configuration parameters include the following. The *Refuse Rate* E_0 is the threshold for the Monte Carlo method. The mean and standard deviation of the *GX* dataset are calculated for each of the currency links to be predicted. These statistics are used to measure the molecule potency as well as prediction when calculating the *jump* value using the Monte Carlo method and Gaussian distribution function.

Algorithm 2 Parallel Elastic Network Model as applied to the FOREX Market.

```
Require: Number of currencies $n \in \mathbb{N}$ , maximum trend $t \in \mathbb{N}$, $GX \in \mathbb{R}^{t\times\frac{n(n-1)}{2}}$,
    $time\_frame \in \mathbb{N}$, $total\_time \in \mathbb{N}$ and $num\_threads \in \mathbb{N}$.

Ensure: $prediction \in \mathbb{R}^{1\times\frac{n(n-1)}{2}}$
 1: Define $prices, price_0, price_1, weight\_mean, mean\_GX, sd\_GX \in \mathbb{R}^{1\times\frac{n(n-1)}{2}}$ and
    $potential_0, potential_1, E_0, jump \in \mathbb{R}$
 2: $tray := total\_time/time\_frame$
 3: $mean\_GX :=$ calculate mean of $GX$
 4: $sd\_GX :=$ calculate standard deviation of $GX$
 5: IN PARALLEL each thread$_i$ from $i$ = 0, ..., $num\_threads$ - 1 DO
 6: repeat
 7:     $step$:=0
 8:     $price_0 := GX_{t,:}$ $price_0 := GX_{t,:}$
 9:     while ($tray > step$) do
10:         $weight\_mean :=$ Price_Balance($GX$)
11:         $potential_0$:=Potential($price_0$, $weight\_mean$, $sd\_GX$)
12:         Obtain $jump$ using Monte Carlo method with Gaussian distribution
13:         $price_1 := jump + price_0$        ▷ Adding $jump$ to all the elements of $price_0$
14:         $potential_1$:=Potential($price_1$, $weight\_mean$, $sd\_GX$)
15:         if (Accept_Potential($potential_0, potential_1, E_0$)) then
16:             $price_0 := price_1$
17:             update $GX$ adding $price_1$ in position $t$
18:             $step + +$
19:         end if
20:     end while
21:     $prices$ += $price_1$
22: until reach $maximum\_iterations/num\_threads$
23: END PARALLEL
24: $prediction := prices/maximum\_iterations$
25: return $prediction$
```

Once the input data are defined, the following operations are carried out to obtain the quote for all currencies. Firstly, we calculate the weighted average of the dataset (*GX*) by means of the function *Price_Balance*(). Then, the Monte Carlo procedure calculates the new state of the system. Specifically, the Monte Carlo method returns a random number which is the length of the "*jump*" to the new state of each atom according to the applied distribution function; in this case we use Gaussian distribution; i.e., $price_1$ is the new state of the molecule which is $jump + price_0$. Once the new status is obtained, we calculate the potential of both $price_0 and price_1$ with the function *Potential*(). If *Accept_Potential*($potential_0$,$potential_1$) is true, then the system goes toward a point of equilibrium and the new $price_1$ value is accepted. Otherwise, a new potential is calculated taking into account the previously calculated and the rejection rate E_0. If this new potential is greater than a random value, then it is taken; otherwise, it is discarded and the whole process of this step has to be repeated using the *jump* again to calculate $price_1$ using the Monte Carlo method and the potential calculation.

When $price_1$ is taken, it means that the step has been completed correctly and we move on to the next step, so $price_0$ is updated by taking the value of $price_1$.

In addition, as for the calculation of a new element the previous values are taken into account the new value $price_1$ added to the GX at the end of the file. (It is the most recent record from now on.) As previously explained, GX has t records; therefore, the oldest record in GX is removed. Finally, the final prediction is calculated through the mean of all the final stages accumulated in *prices*, providing as an output $prediction := prices/maximum\_iterations$.

2.1 Parallelization strategies for the PENM

This section introduces the parallelization strategy of the ENM algorithm presented in Sect. 2 for a heterogeneous cluster based on multicore CPUs. First, our algorithm sets an MPI process for each node in the cluster where our simulation is executed. As previously explained, the t parameter determines the number of records stored in the GX dataset. This parameter could be randomly selected to better adjust the input parameters. In this case, an MPI process will deal with a random t, proceeding with the execution of the ENM algorithm in parallel. This is a consensus approach to the comparison of classifications between all ENM executions performed. The supporting data structures are created in each node to avoid communication overhead. Then, each MPI process creates as many OpenMP threads as cores are available at the node, which is easily obtained by querying the CPU properties at runtime using `omp_get_num_procs()`.

Algorithm 2 shows the parallelization strategy we use to leverage heterogeneous nodes with shared memory multiprocessors. Indeed, the main bottleneck in the ENM computation is found at the outer loop (i.e., iterations loop) where a number of different trajectories are executed in parallel. However, the trajectory calculation is that the new states of the molecule, $price_1$, are not always taken as there is a conditional operation that establishes whether or not it is taken on the basis of potential. This situation also has the effect that the loop used for the full calculation of a molecule trajectory is a "*while*" loop. These two issues preclude parallelization in that code segment. However, if we level up and parallelize iterations, we would include in the parallelization the whole calculation loop of the path that contains the most computational floating point operations and would also eliminate dependencies by parallelizing a *for* loop.

We have even developed another strategy to improve overall performance. This strategy is based on the casuistry of the problem, and how this fits perfectly with the concept of vectorization. Today's high-performance processors, being extensively equipped with vector units, fit perfectly to get the most out of vectorization. There are several options when vectorizing the code of an algorithm. This work avoids the use of low-level instructions, like assembly language, in order not to lose portability between different computer architectures. Instead of using low-level instructions, we make use of the capabilities of current compilers to automatically vectorize some loops (enabled with the optimization level `O3`), but additionally, we can make the compiler's task easier by adding some hints and changes on the code.

- *Data alignment* The way we use the memory is very important because of the number of times that the memory is accessed. The default way memory is allocated in C is in the form of Array of Structures, *AoS*; however, in the form of Structure of Arrays, *SoA*, it usually fits much better for *SIMD* data processing. Data alignment is also crucial, and aligned memory allocations must be forced by specific functions such as *aligned_alloc()* instead of usual *malloc()* for data alignment on the heap (to a multiple of 64 bytes). In this way, we use *"void *aligned_alloc(size_t alignment, size_t size);"* function to align the heap memory to page boundary.
- *#pragma GCC ivdep*: Use `#pragma ivdep` before a loop for telling the compiler to ignore vector dependencies, which would prevent simultaneous consecutive iterations of the following loop with SIMD from being executed simultaneously. Summarizing, using the `#pragma ivdep` involves:
  - Assume no dependencies.
  - Compiler may vectorize loops that it would otherwise think are not vectorizable.

3 Empirical results

3.1 Comparison between ENM and VAR model

This section compares the forecasting accuracy results between both PENM and vector autoregressive (VAR) model. First of all, the structure of the VAR model was defined on the methodology proposed by [19]. It is focused on Schwarz lag length criterion in order to minimize collinearity. The optimal lag length obtained by this criterion was 7 so a stationary unrestricted VAR(7) model was estimated. The main reason to select this seven currencies is the published transaction volume at the last BIS triennial report of 2016 (Bank for International Settlements, see Table 1) [4], obviously excluding blocked currencies.

The data available are from 01.01.2015 23:00 to 07.28.2017 20:00 so that the frequency of the data analyzed has a spread of five minutes (5 min). The forecasting strategy adopted was the followed by many researchers as [22] who were one of the pioneer works using this methodology. Firstly, a subsample within the dataset was selected with the purpose of model identification and parameter optimization. This period is from 01.01.2015 23:00 to 07.28.2017 17:00. Secondly, we carry out the forecasting strategy of sequential estimation generating the remaining observations. In other words, we predict from 1 to 36 out-of-sample horizons. In addition, the average and the median of the out-of-sample periods are computed.

The most representative seven currency pairs in the FOREX market were selected in order to conduct the analysis. Table 1 shows these selected currencies in which we carried out the comparative study through the 21 currency pairs stem from all possible combinations between the seven currencies. Table 2 shows the quality results obtained for the more representative out-of-sample periods. The forecasting accuracy is evaluated by the mean square error (MSE) computed across the 21 currency pair

Table 1 Currency and abbreviation

Currency	Abbreviation
Australian dollar	AUD
Canadian dollar	CAD
Euro	EUR
British pound	GBP
Japanese yen	JPY
New Zealand dollar	NZD
American dollar	USD

Table 2 Forecasting accuracy (MSE) and profitability (PF) measures for both PENM and VAR models

Forecast horizons	Out-of-sample period	PENM		VAR	
		MSE	PF	MSE	PF
$H = 1$	07/28/2017 17:05	**2.0E−06**	**2.550**	2.07E−06	2.374
$H = 6$	07/28/2017 17:30	**4.1E−05**	7.917	5.26E−05	**6.281**
$H = 12$	07/28/2017 18:00	**2.5E−05**	6.097	3.42E−05	**3.212**
$H = 24$	07/28/2017 19:00	**3.3E−05**	**23.177**	5.99E−05	1.547
$H = 30$	07/28/2017 19:30	6.4E−05	**6.471**	**6.38E−05**	1.354
$H = 36$	07/28/2017 20:00	9.1E−05	**16.075**	**6.97E−05**	1.224
$H = 1$ to $H = 36$	Average	**4.267E−05**	**10.381**	4.705E−05	2.665
$H = 1$ to $H = 36$	Median	**3.700E−05**	**7.194**	5.628E−05	1.961

combinations, and the profitability is measured by profit factor (PF) [5] which are defined as follows:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=0}^{n}\left\{\hat{y}_{i,t+h} - y_{i,t+h}\right\}^2$$

where n is the number of currency pairs analyzed and h the forecast horizon predicted. $\hat{y}_{i,t+h}$ and $y_{i,t+h}$ are the actual and forecast values, respectively

$$\mathrm{PF} = \frac{\text{gross profits}}{\text{gross losses}}.$$

The mean square error (MSE) is a metric widely used in econometrics. It is a temporary (and cumulative) rather than a transverse measure, i.e., the MSE is increasing as greater prediction horizons are compared (for example, being the EURUSD in $H = 1$ the MSE is 1, for $H = 2$, the MSE is the value of $H = 1 +$ MSE in $H = 2$). Since the main objective in our case is to compare the goodness of the PENM model, we propose the MSE in a transversal way, i.e., for each time horizon evaluated, the MSE is calculated for each of the currencies. For example, for $H = 1$, the MSE would be the MSE of $H = 1$ for the EURSD + the MSE of $H = 1$ for EURGBP +· . up to the quoted price predicted number 21. Therefore, this does not imply that for $H = 2$, the MSE is greater than that of $H = 1$ since we do not accumulate the values temporarily. (Each time horizon is independent for the calculation of the MSE.) Taking into account what has been said before about the PF, this is also calculated in a transversal way and being independent of the ECM, there is no dependence between the two metrics.

Regarding the forecasting accuracy, PENM outperforms the VAR model in the closer forecast horizons, whereas VAR model is more accurate in the further horizon. Nevertheless, the average MSE is slightly larger for VAR model than PENM. Moreover, the median suggested the same results. Hence, we summarized that the forecasting accuracy of PENM is relatively similar to VAR model and sometime even better. Furthermore, the profitability of PENM was clearly larger than VAR model. Except for two of the six forecast horizons analyzed, PENM widely outperforms VAR model. Based on both average and median, profit factor of PENM was almost four times the profit factor of VAR model. Thus, PENM is more profitable than VAR model.

3.2 Computational analysis

This section shows an experimental evaluation for the FOREX market model running on a cluster based on Intel CPUs. First of all, we briefly introduce the hardware and software environment where the experiments are performed. Afterward, we carry out an analysis of the performance and quality of our application comparing with well-established methods in the literature like VAR.

Hardware and software environment: Tests have been performed on a cluster where two servers are available. The former, called Heterolistic, has 4 Intel (R) Xeon (R) processors E5-2650 CPU v4 running at 2.20 GHz with 12 cores each processor, in total 48 cores CPU and 128 GB of RAM. The latter, called Hertz, is four Intel Xeon X7550 processors running at 2 GHz and plugged into a quad-channel motherboard endowed with 128 Gigabytes of DDR3 memory.
Regarding the software environment used, the compiler has been GCC version 5.4.0. OpenMP 4.0 technology has been used to make the parallelization in shared memory, and OpenMPI 3.0 has been used for the parallelization with message passage. GNU Scientific Library v2.4 has also been used for numerical calculations.
Datasets: The datasets on exchange rates of different currencies are collected from the OANDA FOREX Trading and Exchange Rates Services online database.[1] The time frame established is 5 min, and we deal with up to 7 currencies: AUD, CAD, EUR, GBP, NZD USD, JPY. As the relationship AUD – CAD = CAD – AUD, there are up to 21 different pairs.

[1] https://www.oanda.com/.

Table 3 Number iterations, execution time in seconds non-vectorized version, execution time in seconds for vectorized version, and the speed-up between vectorized versus non-vectorized version

Num. iterations	Ex.time non-vect. (s)	Ex.time vect. (s)	% Speed-up
1	0.56	0.46	21.83
2	1.12	0.91	23.73
4	2.21	1.79	23.27
8	4.38	3.57	22.81
16	8.94	7.12	25.45
32	17.57	14.23	23.46
64	35.11	28.48	23.30
128	69.68	56.96	22.34
256	139.68	113.63	22.92
512	278.38	227.5	22.37

First of all, the vectorization impact is analyzed by isolating a single core to avoid OpenMP and MPI overheads and to show the impact of vector optimizations. The following experiments have been carried out: executions for 1, 2, 4, 8, 16, 32, 64, 128, 256, 512 iterations for 21 currency pairs for predictions with a horizon of 60 min, using single-threaded non-vectorized and vectorized versions; these results are shown in Table 3. Let us remind the reader that the function where the vectorization has been used is in the price_equilibrium function as it is the function with a better layout for vectorization.

Moreover, the potential bottlenecks have been identified by using the toplev [15] tool. This tool implements a top-down mechanism to analyze the performance in the Linux operating systems of the modern Intel CPUs. There are two types of tools like this one, counting and sampling; specifically toplev uses the performance counters of the CPU to count events. In our case, we run the toplev application for both non-vectorized and vectorized execution including two configuration options *longdesc* to provide a long description of the bottleneck found and $single - thread$ to ensure that the execution is performed on a single thread. This executions were run with 512 iterations with a time frame of 5 min, with 21 currency pairs, and with a horizon of 60 min. As expected, our bottleneck is a back-end bound problem where complex calculations and data access make slow down the computation. As it is an event-counting software, the higher the value obtained, the higher the performance of the application. In addition to detecting the bottleneck in the back end, the non-vectorized execution score is 58.99 and the vectorized version 69.26, which is translated in a performance improvement of 17.09%, which is actually similar to the speed-up reported (20%×).

Table 4 shows the execution time in seconds for the FOREX market simulations using our parallel elastic neural model (PENM) approach under different conditions. Firstly, different programming languages are under evaluation. The R programming language is our baseline version as in this field is widely extended this statistical environment. However, this environment is not well suited for high-performance simulations. Actually, the C counterpart version is up to two orders of magnitude faster than R version. Moreover, the different OpenMP

Table 4 Execution time in seconds for the parallel elastic neural model (PENM) approach using different programming languages and configurations

Currency pairs	3	6	12	21
R Prog	1764	3349	7178	13765
C Prog	15.87	30.74	66.75	112.66
C-OMP (2th)	7.87	15.51	31.20	56.62
C-OMP (4th)	4.30	8.34	16.38	31.20
C-OMP (8th)	2.43	4.62	9.89	21.76
C-OMP (16th)	1.47	2.94	8.03	16.69
C-OMP (32th)	0.87	1.89	6.18	14.88
C-OMP (48th)	0.85	2.03	5.81	14.80
V-OMP (48th)	0.98	2.20	6.44	13.38

The simulation is carried out with a different number of currency pairs. Moreover, R prog. is the R version of ENM, C Prog. is the C version, C-OMPs are the parallel counterpart version with different numbers of threads (i.e., th.), and V-OMP is the vectorized version with aligned data

configurations, having up to 48 threads running in parallel, are evaluated to look for the optimal parallel configuration. The best OpenMP configuration is 48 threads that matches with the number of cores of our computing platform as long as the simulation size is big enough (beyond 6 currency pairs). The speed-up factor of our best OpenMP configuration reaches up to 930× compared to R version and 7.6× compared to C single-threaded counterpart version. The vectorized version also contributes in this way; the more data we have to compute, the better is our results, achieving up to 1.10× of speed-up against the previous best implementation. Our results lead to two main conclusions. First, R is not suitable for real-time constrained simulations like those within the FOREX Market. The FOREX market simulations have strict time constraints as the predicted results are only valid within a concrete time frame. The only migration to C offers significant performance gains. However, typical time frames in the FOREX market are under 5 min, and single-threaded C version is not able to defeat this constraint by a wide margin. With the parallelization, we are able to simulate up to 21 currency pairs in less than 14 s.

The benefit of parallelization is having in the outer loop as shown in Algorithm 2. Figure 2 shows the scalability of our best configuration approach (OMP 48-thread version) when the number of iterations is incremented. We empirically demonstrate that the PENM algorithm converges in 256-iterations on average. It is noteworthy to highlight that *x*-axis is in logarithmic scale which may cause a visual effect of logarithmic growing along with the iterations instead of linear which is actually the real behavior.

Finally, the number of different random execution of *t* records is fully independently to each other. Therefore, the simulation can be carried out in different nodes in parallel in a map reduce fashion. Table 5 shows the execution time of 256-iterations on a cluster using up to 4 nodes using MPI + OpenMP version against execution time of the MPI + OpenMP *vectorized* version.

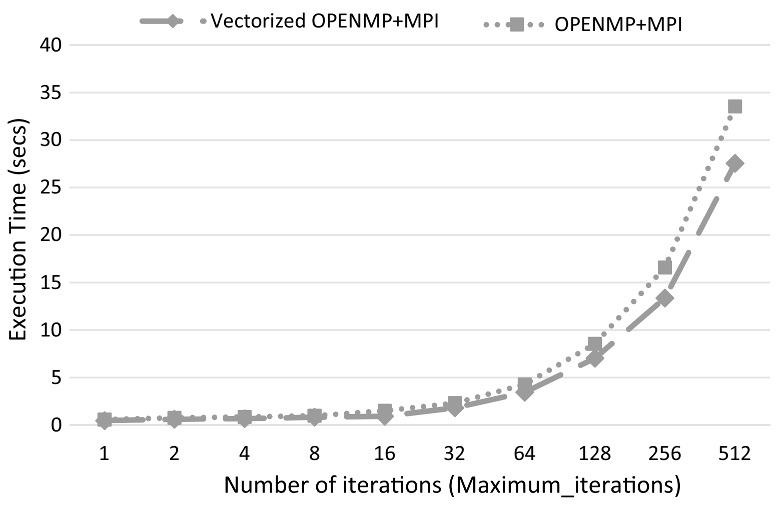

Fig. 2 Execution times in seconds for 48-threads OpenMP version, varying the number of iterations

Table 5 Number of computing node (first column), number of iterations per node (second column), execution time in seconds for MPI + OPENMP version (third column) and MPI + OPENMP + Vectorization (fourth column)

Node	Num. iterations	Ex. time MPI + O.	Ex. time MPI+ O.V.
0	52	3.760	3.291
1	51	3.498	3.236
2	51	3.636	3.174
3	51	3.693	3.064
4	51	3.966	3.255
Total ex. (s)	256	18.553	16.023

The overall execution time is determined by the slowest node

4 Conclusions and future work

The FOREX market is one of the most important markets in the world. The simulation of price fluctuation for currency pair exchange rate provides valuable information for investors. Indeed, this simulation need to be both (1) context-aware as the exchange rates for each different coin are drastically affected by other exchange rates, and (2) real-time requirements as the knowledge is valid as far as it is obtained before the reality happens. This paper introduces a context-aware bioinspired algorithm to model the behavior of up to 21 currencies' exchange rates, called parallel elastic network model (PENM). It is inspired by the behavior of macromolecules in dissolution. As bioinspired algorithm, its computation is massively parallel by its definition, and thus, this parallelism is exploited in a computer-based cluster using OpenMP and MPI.

The empirical results lead to two different conclusions. Firstly, the PENM approach outperforms traditional vector auto-regression models according to the forecasting accuracy. Secondly, the PENM is massively parallel by its definition and this feature enables the concurrent simulation of up to 21 currency pairs' exchange rates at the same time in less than 14 s. Therefore, the PENM algorithm is able to make prediction less than 1-minute time frame. Actually, traditional r codes developed on this field take up to 13,765 s which means 1030× speed-up factor, comparing results obtained with MPI + OpenMP and aligning data.

Parallel codes for the FOREX market are still at a relatively early stage, and we acknowledge that we have tested a relatively simple parallel infrastructure. But, with many other types of computing architectures still to be explored, this field suggests to provide a potentially and promising fruitful area of research. Indeed, it is expected to get even higher speed-up factors on other architectures like GPUs whenever the problem size keeps growing and larger simulations are executed. Moreover, we may guess that the benefits of our approach would also increase when using future heterogeneous architectures endowed with thousands of cores and eventually grouped into heterogeneous clusters to lift performance into unprecedented gains where parallelism is called to play a decisive role.

Acknowledgements This work was jointly supported by the Fundación Séneca (Agencia Regional de Ciencia y Tecnología, Región de Murcia) under Grant 20813/PI/18 and by the Spanish MEC and European Commission FEDER under Grants TIN2016-78799-P and TIN2016-80565-R (AEI/FEDER, UE).

References

1. Bahrepour M, Akbarzadeh-T MR, Yaghoobi M, Naghibi-S MB (2011) An adaptive ordered fuzzy time series with application to FOREX. Expert Syst Appl 38(1):475–485
2. Bank for International Settlements. https://www.bis.org/. Accessed 13 Feb 2013
3. Bhattacharyya S, Pictet OV, Zumbach G (2002) Knowledge-intensive genetic discovery in foreign exchange markets. IEEE Trans Evolut Comput 6(2):169–181
4. Bank of International Settlements (2016) Triennial central bank survey: foreign exchange turnover in April 2016, Basel
5. Caporale GM, Gil-Alana L, Plastun A (2017) Searching for inefficiencies in exchange rate dynamics. Comput Econ 49(3):405–432
6. De Grauwe P, Markiewicz A (2013) Learning to forecast the exchange rate: two competing approaches. J Int Money Finance 32:42–76
7. Fama E (1970) Efficient capital markets: a review of theory and empirical work. J Finance 25(2):383–417
8. Fama EF (1965) The behavior of stock-market prices. J Bus 38(1):34–105
9. Fama EF (1970) Efficient capital markets: a review of theory and empirical work. J Finance 25(2):383–417
10. Fuglebakk E, Reuter N, Hinsen K (2013) Evaluation of protein elastic network models based on an analysis of collective motions. J Chem Theory Comput 9(12):5618–5628
11. Hanssens DM, Parsons LJ, Schultz RL (2003) Market response models: econometric and time series analysis, vol 12. Springer, New York
12. Kamruzzaman J, Sarker RA (2003) Forecasting of currency exchange rates using ANN: a case study. In: Proceedings of the 2003 International Conference on Neural Networks and Signal Processing, 2003, vol 1. IEEE, pp 793–797

13. Kamruzzaman J, Sarker RA, Ahmad I (2003) SVM based models for predicting foreign currency exchange rates. In: Third IEEE International Conference on Data Mining, 2003. ICDM 2003, IEEE, pp. 557–560
14. Karplus M, McCammon JA (2002) Molecular dynamics simulations of biomolecules. Nat Struct Mol Biol 9(9):646–652
15. Kleen A (2015) Intel PMU profiling tools. https://github.com/andikleen/pmu-tools/tree/d70840ba. Accessed 15 Mar 2019
16. Kuo RJ, Chen C, Hwang Y (2001) An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network. Fuzzy Sets Syst 118(1):21–45
17. LeBaron B, Arthur WB, Palmer R (1999) Time series properties of an artificial stock market. J Econ Dyn Control 23(9):1487–1516
18. Li Q, Chen Y, Wang J, Chen Y, Chen H (2017) Web media and stock markets: a survey and future directions from a big data perspective. IEEE Trans Knowl Data Eng 30:381–399
19. Luetkepohl H (2009) Econometric analysis with vector autoregressive models. In: Belsley DA, Kontoghiorghes EJ (eds) Handbook of computational econometrics. Wiley, New York, pp 281–319
20. Makovskỳ P (2014) Modern approaches to efficient market hypothesis of FOREX—the central European case. Proc Econ Finance 14:397–406
21. Meade N (2002) A comparison of the accuracy of short term foreign exchange forecasting methods. Int J Forecast 18(1):67–83
22. Meese RA, Rogoff K (1983) Empirical exchange rate models of the seventies: do they fit out of sample? J Int Econ 14(1–2):3–24
23. Mockus J, Raudys A (2010) On the efficient-market hypothesis and stock exchange game model. Expert Syst Appl 37(8):5673–5681
24. Nassirtoussi AK, Aghabozorgi S, Wah TY, Ngo DCL (2014) Text mining for market prediction: a systematic review. Expert Syst Appl 41(16):7653–7670
25. Neely C, Weller P, Dittmar R (1997) Is technical analysis in the foreign exchange market profitable? A genetic programming approach. J Financial Quant Anal 32(4):405–426
26. Pincak R (2013) The string prediction models as invariants of time series in the FOREX market. Phys A: Stat Mech Appl 392(24):6414–6426
27. Samuelson PA (2016) Proof that properly anticipated prices fluctuate randomly. In: The World Scientific Handbook of Futures Markets, pp 25–38
28. Sarantis N, Stewart C (1995) Structural, VAR and BVAR models of exchange rate determination: a comparison of their forecasting performance. J Forecast 14(3):201–215
29. Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85–117
30. Sims CA (1980) Macroeconomics and reality. Econ: J Econ Soc. 48:1–48
31. Ţiţan AG (2015) The efficient market hypothesis: review of specialized literature and empirical research. Proc Econ Finance 32:442–449
32. Yao J, Tan CL (2000) A case study on using neural networks to perform technical forecasting of FOREX. Neurocomputing 34(1):79–98

Affiliations

Antonio V. Contreras[1] · Antonio Llanes[3] · Francisco J. Herrera[1] · Sergio Navarro[1] · Jose J. López-Espín[2] · José M. Cecilia[3]

Antonio V. Contreras
avicente@aitalentum.com

Francisco J. Herrera
fjherrera@aitalentum.com

Sergio Navarro
snavarro@aitalentum.com

Jose J. López-Espín
jlopez@umh.es

José M. Cecilia
jmcecilia@ucam.edu

[1] Artificial Intelligence Talentum, S.L., Edificio CEEIM, Campus Universitario de Espinardo, 30100 Murcia, Spain

[2] Center of Operations Research, Miguel Hernández University, Elche Campus, Elche, Alicante, Spain

[3] Polytechnic School, Catholic University of San Antonio of Murcia (UCAM), 30107 Murcia, Spain