Using Databases to Implement Algorithms: Estimation of Allan Variance Using B$^+$-tree Data Structure ⋆

Satya Prasad Maddipatla * **Rinith Pakala** ** **Hossein Haeri** **
Cindy Chen ** **Kshitij Jerath** ** **Sean Brennan** *

* *The Pennsylvania State University, University Park, PA 16802, USA (e-mail: szm888@psu.edu, sbrennan@psu.edu).*

** *University of Massachusetts Lowell, Lowell, MA 01854, USA (email: rinith_pakala@student.uml.edu, hossein_haeri@uml.edu, cindy_chen@uml.edu, kshitij_jerath@uml.edu).*

Abstract: Many algorithms for data processing use batched or recursive methods of computation where the resulting manipulation of data has a topological structure similar to a tree; examples include FFTs, windowed averaging, and many core image processing methods (SIFT, HOG, SURF, etc.). As data sets requiring these methods grow, deploying such algorithms on single or multiple processors can be challenged by accessing data from storage into the processor, for example, data from databases. This work develops and explains a method of using a database organizational structure itself to implement data manipulations (grouping, addition, averaging, etc.) that have intentional algorithm behavior, specifically, the implementation of Allan VARiance (AVAR) using B$^+$-tree data structures is developed.
AVAR is a key algorithm to determine the computational limits on data accuracy imposed by real-world noise sources. Unfortunately, the typical time required to compute AVAR increases quickly with the quantity of the time-series data. In the previous work, the authors proposed Fast Allan VARiance (FAVAR) algorithms inspired by the FFT to improve computation speed by up to four orders of magnitude. These FAVAR algorithms apply to data sets manageable on a computer (MB to GB in memory) but can be difficult to deploy on "big data", e.g., data sets that cannot fit in main memory. Notably, B$^+$-trees index one-dimensional data similar to FAVAR, i.e., using data aggregations that scale in size as a function of tree order. This paper utilizes the similarity in B$^+$-trees and FAVAR to use the database's operational process to automatically deploy an algorithm to estimate AVAR using a B$^+$-tree for data corrupted by common noise types - white noise and random walk. Comparing AVAR estimates to algorithm-targeted B$^+$-tree data calculations, the results match within 95% confidence bounds.

Keywords: Allan variance, Big Data, Database, B$^+$-tree, fast Allan variance, Estimation.

1. INTRODUCTION

Many algorithms for data processing use batched or recursive methods of computation where the resulting manipulation of data has a topological structure similar to a tree; examples include FFTs, windowed averaging, and many core image processing methods (SIFT, HOG, SURF, etc.). As data sets requiring these methods grow, deploying such algorithms on single or multiple processors can be challenged by accessing data from storage into the processor, for example moving data in or out of databases. This work develops and explains a method of using a database organizational structure itself to implement data manipulations (grouping, addition, averaging, etc.) that have intentional algorithm behavior, specifically, the implementation of Allan VARiance (AVAR) using B$^+$-tree data structures is demonstrated.

Allan VARiance (AVAR) is a key method to characterize the stability of data over test dimensions, for example over time. It was first developed to quantify the frequency stability of atomic clocks, see Allan (1966); Barnes et al. (1971). In addition, it is used to quantify the stability of oscillators and lasers as in Giles et al. (1989); Abramski et al. (1990). It is also used to characterize noise in inertial sensors and other MEMS as in Malkin (2016); Jerath et al. (2018). Typical AVAR algorithms calculate changes in means between differently-sized groupings of data. They thus are useful in many data aggregation processes: to select the appropriate window length or timescales for estimating the moving average of a signal as in Haeri et al. (2021), to find the minimum variance of a signal, to determine optimal data reduction in Sinanaj et al. (2022), or to estimate the change in variance of a signal with complex noise contributions as a function of the number of collected data points in Galleani and Tavella (2008, 2015).

⋆ This material is based upon work supported by the National Science Foundation under grant no. CNS-1932509 "CPS: Medium: Collaborative Research: Automated Discovery of Data Validity for Safety-Critical Feedback Control in a Population of Connected Vehicles."

This paper is motivated by a project requiring database-mediated friction analysis for autonomous vehicles aggregating information over large road networks. The goal is to maintain a "lean" database by forgetting any data that become irrelevant, and by grouping data in time and spatial scales to minimize the variance in aggregated data. The amount of data representative of highly connected vehicle data sharing is quite large. A simulation of approximately 3000 connected vehicles sharing roadway information over a small town's roadway network (State College, Pennsylvania) generates about 900 million data measurements per hour during peak travel hours. Sinanaj et al. (2022) proposed a granular data reduction technique for temporal databases based on AVAR.

The insights into the combined usage of databases with algorithms whose data manipulations follow a tree-like structure motivated this study into methods of utilizing the database itself for computational algorithm implementations. Specifically, this work presents AVAR estimation using B^+-trees and its inherent similarity with the FAVAR algorithm. This article is organized as follows: Section 2 introduces tree data structures, particularly B^+-trees. Section 3 describes the FAVAR algorithm. Section 4 presents B^+-tree implementation of AVAR and its similarity with FAVAR. Finally, section 5 compares the AVAR estimated using the B^+-tree with those estimated using FAVAR.

2. TREE DATA STRUCTURES

The database format considered in this work is a *tree* data structure consisting of nodes and edges representing hierarchical data. Consistent with typical notation for database trees, a *node* is a structure that contains data and is connected to other nodes by *edges*. The *root node* is the topmost node with no parent node, and the *leaf nodes* are the bottom-most with no children. All the intermediate nodes are *internal nodes*, defined in (Knuth (1998a)). Tree structures provide a very efficient method of search, insert and delete operations on the data by reducing the time complexity of the algorithms see Knuth (1998b), and thus have a wide range of applications ranging from file systems to classification algorithms. Multiple tree data structures like Binary search trees in Nievergelt (1972), B-trees by Bayer and McCreight (1972); Graefe (2004), B^+-trees defined by Comer (1979); Jensen et al. (2004), and R-trees in Guttman (1984); Beckmann et al. (1990) are widely used in storage systems.

B^+-tree The specific type of tree structure utilized in this work is a B-tree, a generalized binary search tree allowing nodes to have more than two children, introduced in the historical work by Bayer and McCreight (1972). In a binary search tree, a key stored in each node divides the 'key space' into two pieces. Similarly, in an n-way search tree, each node containing $n-1$ keys divides the 'key space' into n pieces, with each subtree holding keys in those pieces. A B-tree of order n is characterized by properties presented in Knuth (1998b).

B^+-trees are like B-trees that store all the data at the leaf level, as proposed in Comer (1979). In addition to references to data records, leaf nodes have pointers to the next leaf node. All the internal nodes act as the pointers to the leaf-level data, and these pointers are hierarchically placed based on the order of the occurrence of search-keys in leaf data. The B^+-tree structure has particular benefit in this work as the algorithm implementation presented here requires multiple children and a balanced tree structure, both of which are easily implemented via B^+-trees.

3. FAST ALLAN VARIANCE

The AVAR of signal $y(t)$ is a measure of how the signal's variance changes across averaging intervals and is typically plotted as variance (y-axis) versus correlation time (x-axis). The AVAR function dependence on correlation time τ uses form originally defined in Allan (1966) and Barnes et al. (1971), given by equation (1).

$$\sigma_A^2[\tau] = \frac{1}{2}\mathbb{E}\left[\left(\bar{y}_t - \bar{y}_{t-\tau}\right)^2\right] \tag{1}$$

where $\sigma_A^2[\tau]$ is AVAR, $\mathbb{E}$ is the expectation operator, and $\bar{y}_t$ is given by equation (2).

$$\bar{y}_t = \frac{1}{\tau}\int_{t-\tau}^{t} y(t')\,dt' \tag{2}$$

The expectation operator in equation (1) is approximated as a time average so that the AVAR of data $\{y_i\}\,(i = 1, 2, \ldots, N)$ as a function of correlation interval m can be estimated using equation (3) as in Allan (1987).

$$\sigma_A^2[m] = \frac{1}{2(N-2m)}\sum_{l=2m+1}^{N}\left(\bar{y}_l - \bar{y}_{l-m}\right)^2 \tag{3}$$

where $\sigma_A^2[m]$ is AVAR, N is the data length, and $\bar{y}_l$ is given by equation (4).

$$\bar{y}_l = \frac{1}{m}\sum_{i=l-m}^{l-1} y_i \tag{4}$$

which is easily recognizable as a local, windowed average of data. One can recognize that AVAR is, therefore, a measure of the variance of local averages relative to each other across different windows produced by a selected correlation time. Equation (5) defines the relationship between the correlation time τ, correlation interval m, and the sampling interval τ_0.

$$\tau = m\tau_0 \tag{5}$$

The computational time to estimate AVAR increases very quickly with the increasing quantity/length of data, but modifications to the AVAR algorithm can greatly reduce these computations. Dynamic Allan VARiance (DAVAR) is an extension of AVAR for changing signals, and examples include the study of changing characteristics of atomic clock behavior in the Galileo satellite system. In turn, with the work from Galleani (2010), DAVAR decreases the computation time by exploiting the fact that the DAVAR is a moving AVAR. Maddipatla et al. (2021) presents the FAVAR algorithms which increase the computation speed of AVAR up to four orders of magnitude by exploiting the significant similarities between the calculation of the Fast Fourier Transform (FFT) and AVAR. Specifically, the results utilize the power-of-2 structure of the FFT to identify a similar structure in AVAR calculations. To illustrate the similarities, note that the FFT evaluates the Fourier transform using a recursive process of multiplication followed by addition, thereby considerably decreasing the computation time (Cooley and Tukey (1965); Brigham and Morrow (1967)).

Estimating AVAR using the FAVAR algorithm is similar to FFT, as demonstrated in Maddipatla et al. (2021) and Fig. 1. Level-0 corresponds to the correlation interval $m = 2^0$, and Level-1 corresponds to the correlation interval $m = 2^1$. Similarly, Level-i corresponds to the correlation interval $m = 2^i$. 'Base data' at Level-i is defined such that AVAR at Level-i or correlation interval $m = 2^i$ is estimated as the mean square of the differences of 'Base data'. As described here, FAVAR involves recursive steps of averages, i.e., addition followed by division in power-of-2 groupings. 'Base data' at Level-0 is the same as the signal. 'Base data' at Level-1 is the mean of consecutive points in the 'base data' at Level-0. 'Base data' at Level-2 is the average of two points separated by 2^1 indices in the 'base data' at Level-1. Similarly, 'base data' at Level-i is the average of two points separated by 2^{i-1} indices in the 'base data' at Level-$i-1$, as demonstrated in line 8 of algorithm 1 in Maddipatla et al. (2021).

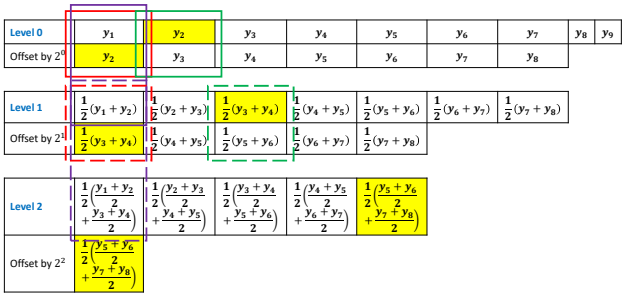

Fig. 1. Fast Allan VARiance algorithm.

At Level-0, AVAR is the mean square of the difference between successive points in the 'base data.' Furthermore, at Level-1, AVAR is the mean square of the difference between points separated by 2^1 indices in the 'base data.' Similarly, at Level-i, AVAR is the mean square of the difference between points separated by 2^i indices in the 'base data' as in line 14 in algorithm 1 in Maddipatla et al. (2021).

4. B$^+$-TREE IMPLEMENTATION OF AVAR

B$^+$-tree inherently organizes data like the FAVAR algorithm. In FAVAR, data are grouped in sizes of power-of-2 as described in the above section, whereas the grouping size in the B$^+$-tree is a function of the tree order. In a B$^+$-tree, all data are referred to by keys in leaf nodes. A leaf node in a B$^+$-tree of order n contains a minimum of $\frac{n}{2}$ keys and a maximum of $n-1$ keys. So a leaf node in a B$^+$-tree refers to a minimum of $\frac{n}{2}$ and a maximum of $n-1$ data records. In AVAR estimation, the data grouping size is equivalent to the AVAR correlation interval. On average, leaf nodes in a B$^+$-tree typically group data in sizes of $\frac{n}{2}$ which means the minimum possible correlation interval is $\frac{n}{2}$. In contrast, internal nodes in level L group data in $\frac{n}{2}\left(\frac{n}{2}+1\right)^L$ sizes because each internal node typically has $\frac{n}{2}+1$ children. At the leaf level, $L = 0$. The value of L increments by one at each increasing level, traversing from the leaf to the root level.

In the AVAR implementation of this work using a B$^+$-tree, the keys represent time and leaf nodes point to friction estimates for time-domain friction data. This work used a B$^+$-tree of order 4, as this is the simplest tree with non-infinite variance at the lowest nodes, because with a tree of order 2 there may be branches partially filled and containing only one data point making within-branch variance calculations impossible. However, order 2 could work for AVAR implementation if special consideration is given to the likely conditions that, at the bottom-most level, there will be nodes with 1 data record, resulting in undefined variance for the means of the node.

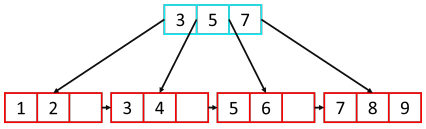

Fig. 2. B$^+$-tree of order 4 with keys from 1 to 9.

While the brevity of the paper does not allow complete explanation of the B$^+$-tree formation, illustrations are provided for those who are somewhat familiar already with B$^+$ or similar trees. Figures 2, 3, and 4 demonstrate data insertion into a B$^+$-tree of order 4. Figure 3 shows the B$^+$-tree in Fig. 2 after inserting key 10. As the new key 10 is more than 9, it is inserted into the rightmost leaf node. As the leaf node in a B+ tree of order 4 cannot contain more than three keys, it splits into two leaf nodes, one with keys 7, 8 and the other with keys 9, 10. Key 9 is inserted into the root node as the leaf node splits. As the total number of keys in the root node will be more than 3, the root node splits into two internal nodes, creating a new root node with key 7.

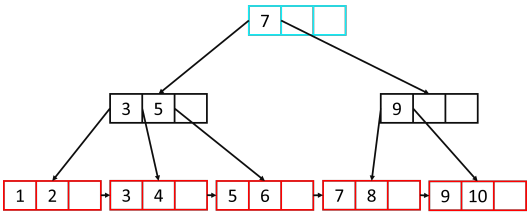

Fig. 3. B$^+$-tree of order 4 with keys from 1 to 10.

Figure 4 shows the B$^+$-tree in Fig. 3 after inserting key 11. As before, the key is inserted into the rightmost leaf node. The number of keys in the rightmost leaf node equals three, so the insertion is complete. To visualize data insertion and deletion in a B+ tree, see `https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html`. Moreover, irrespective of the order of the data received by the database, B$^+$-tree inherently sorts the data. For example, if the data is received as two packets $(1, 2, 7, 8)$ and then $(3, 4, 5, 6)$, all the data will be sorted after receiving both packets. However, the AVAR estimate from the first packet of 4 data points may not accurately represent the first four $(1, 2, 3, 4)$ data points or even the eight data points until the ordered data is inserted. In other words, the database implementation of AVAR has a very useful property in that it will automatically sort out-of-order data entries, but to match AVAR behavior, the database must contain the same data as used in typical AVAR calculations.

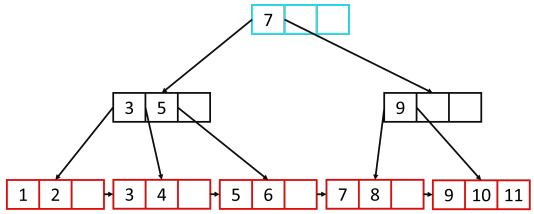

Fig. 4. B$^+$-tree of order 4 with keys from 1 to 11.

4.1 Allan variance estimation using B$^+$-tree

To estimate AVAR of regularly sampled data, a B$^+$-tree's operation is studied and customized from bottom to top, i.e., from the leaf to the root level. B$^+$-tree organizes regularly sampled data irregularly across leaf nodes. For example, the rightmost leaf node of the B$^+$-tree in Fig. 5 refers to three data records, whereas all the other leaf nodes refer to only two data records. So estimating the AVAR of regularly sampled data using a B$^+$-tree is equivalent to estimating the AVAR of irregularly sampled data. Haeri et al. (2021) presented AVAR estimation of irregularly sampled data using equation (6).

$$\sigma_A^2[\tau] = \frac{1}{2\sum_S w_t} \sum_S w_t \left(\bar{y}_t - \bar{y}_{t-\tau}\right)^2 \tag{6}$$

where the summation is performed on a finite set of instances $t \in S$. $\bar{y}_t$ and w_t are given by equations (7a) and (7b) respectively.

$$\bar{y}_t = \begin{cases} \dfrac{1}{|C_t|} \displaystyle\sum_{y_{t_i} \in C_t} y_{t_i} & |C_t| \neq 0 \\ 0 & |C_t| = 0 \end{cases} \tag{7a}$$

$$w_t = |C_t|\,|C_{t-\tau}| \tag{7b}$$

where $C_t = \{y_{t_i} : t - \tau \leq t_i < t\}$ and $|C_t|$ is cardinal number of the set C_t.

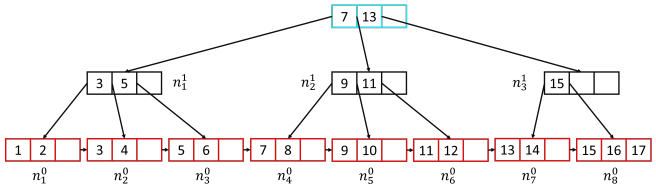

Fig. 5. B$^+$-tree of order 4 with keys from 1 to 17 to demonstrate AVAR estimation.

For this work, two leaf nodes are said to be successive if one points to another. Two internal nodes are said to be successive if the rightmost leaf node in the subtrees of an internal node point to the leftmost leaf node in the subtrees of another. Based on the formulation in equation (6) to estimate AVAR of irregularly sampled data, AVAR estimation of regularly sampled data using a B$^+$-tree involves three steps:

(1) At each level, estimate the mean of data and the number of data records referred by a node and all its children.
(2) Evaluate the difference in the average of data referred by successive nodes and their children.
(3) AVAR is the weighted mean of squares of differences evaluated in the above step.

AVAR of regularly sampled data at level L or correlation interval $\frac{n}{2}\left(\frac{n}{2}+1\right)^L$ estimated using a B$^+$-tree is given by equation (8).

$$\sigma_A^2[L] = \frac{1}{2w^L} \sum_{i=1}^{N_L-1} w_i^L w_{i+1}^L \left(\bar{y}_i^L - \bar{y}_{i+1}^L\right)^2 \tag{8}$$

where N_L is the number of nodes at level L, $\bar{y}_i^L$ is the mean of data pointed by node i at level L and all its children, w_i^L is the number of data records referred by node i at level L and its children. w^L is given by equation (9).

$$w^L = \sum_{i=1}^{N_L-1} w_i^L w_{i+1}^L \tag{9}$$

The above methodology of estimating AVAR using a B$^+$-tree is demonstrated using an example in Fig. 5. Figure 5 shows a B$^+$-tree of order 4 with 17 data records where n_i^L indicates node i in level L. At the leaf level, the mean of the data and the number of data records referred to by nodes n_7^0 and n_8^0 are given by equations (10a) and (10b), respectively.

$$\bar{y}_7^0 = \frac{1}{w_7^0}\left(y_{13} + y_{14}\right) \qquad w_7^0 = 2 \tag{10a}$$

$$\bar{y}_8^0 = \frac{1}{w_8^0}\left(y_{15} + y_{16} + y_{17}\right) \qquad w_8^0 = 3 \tag{10b}$$

Similarly, the mean of the data and the number of data records referred to by node n_3^1 are given by equation (11). The estimation of averages happens recursively from the leaf to the root level.

$$\bar{y}_3^1 = \frac{1}{w_3^1}\left(w_7^0 \bar{y}_7^0 + w_8^0 \bar{y}_8^0\right) \qquad w_3^1 = w_7^0 + w_8^0 \tag{11}$$

After estimating the averages, successive differences are calculated, followed by the weighted mean of squared differences as AVAR. AVAR at level $L = 1$ for a B$^+$-tree in Fig. 5 is given by equation (12).

$$\sigma_A^2[1] = \frac{1}{2w^1}\left[w_1^1 w_2^1 \left(\bar{y}_1^1 - \bar{y}_2^1\right)^2 + w_2^1 w_3^1 \left(\bar{y}_2^1 - \bar{y}_3^1\right)^2\right] \tag{12}$$

5. RESULTS

A summary analysis comparing the accuracy of AVAR estimated using the B$^+$-tree data structure against the FAVAR algorithm is presented here for three signals: white noise, random walk, and white noise added to a random walk. Considering the main memory requirement and computation time of the FAVAR algorithm, a data length of 2^{18} is used for demonstration. Furthermore, a B$^+$-tree of order 4 is used, as described in the section 4. Figure 6 shows white noise with a power spectral density of $0.0004\frac{Unit^2}{s}$ synthesized at $50Hz$ as described in Jerath et al. (2018). AVAR of white noise is estimated using B$^+$-tree by equation (8). It is compared against the AVAR estimated using the FAVAR algorithm developed in Maddipatla et al. (2021). AVAR has a chi-squared distribution, so the confidence bounds depend on the confidence coefficient and degrees of freedom and are estimated by equation (13) (Galleani (2011)).

$$\frac{\nu[m]}{\chi_2^2}\sigma_A^2[m] \leq \mathbb{E}\left[\sigma_A^2[m]\right] \leq \frac{\nu[m]}{\chi_1^2}\sigma_A^2[m] \tag{13}$$

where p is the confidence coefficient, and we obtain χ_1^2 and χ_2^2 from the cumulative distribution function $F\left(\chi^2\right)$ of the chi-squared probability distribution so that

$$F(\chi_1^2) = \frac{1-p}{2} \tag{14a}$$

$$F(\chi_2^2) = \frac{1+p}{2} \tag{14b}$$

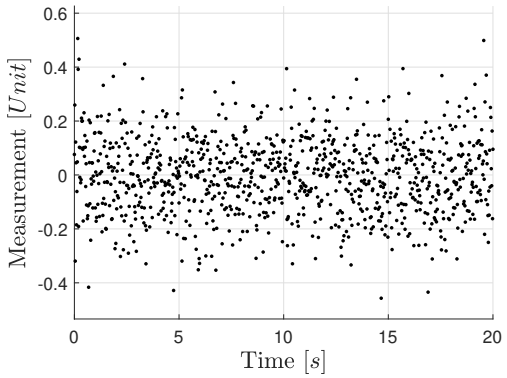

Fig. 6. White noise with a power spectral density of $0.0004\frac{Unit^2}{s}$ synthesized at a frequency of $50Hz$.

Figure 7 shows that the AVAR estimate using B$^+$-tree lies within the 95% confidence lower (LB) and upper (UB) bounds of the FAVAR estimate.

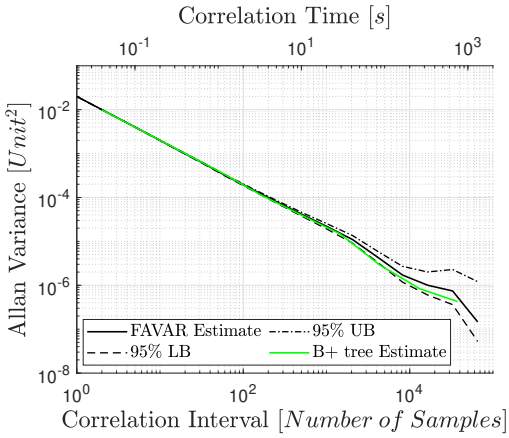

Fig. 7. AVAR of white noise estimated using B$^+$-tree lies within the confidence bounds of the FAVAR estimate.

Random walk, a second signal as an example with a random walk coefficient of $0.02\frac{Unit}{\sqrt{s}}$ synthesized at a frequency of $50Hz$, is shown in Fig. 8. Figure 9 reiterates that AVAR estimated using a B$^+$-tree data structure lies within the 95% confidence bounds of AVAR estimated using FAVAR.

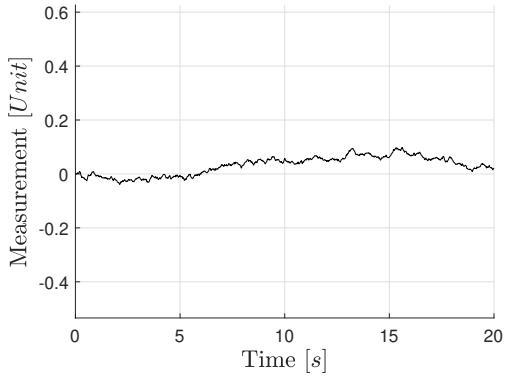

Fig. 8. Random walk with a random walk coefficient of $0.02\frac{Unit}{\sqrt{s}}$ synthesized at a frequency of $50Hz$.

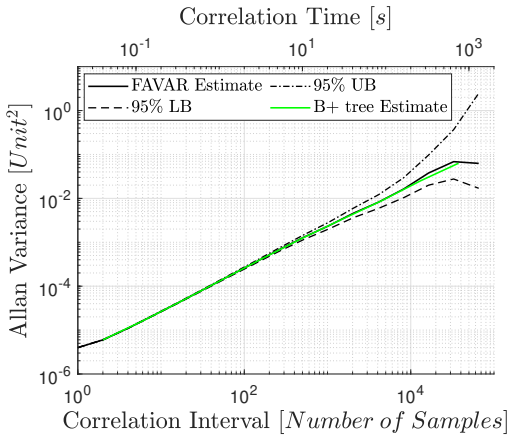

Fig. 9. AVAR of random walk estimated using B$^+$-tree lies within the confidence bounds of the FAVAR estimate.

Figure 10 concludes that AVAR estimated using B$^+$-tree data structure lies within the 95% confidence bounds of AVAR estimated using FAVAR with a signal synthesized by adding white noise in Fig. 6 to a random walk in Fig. 8.

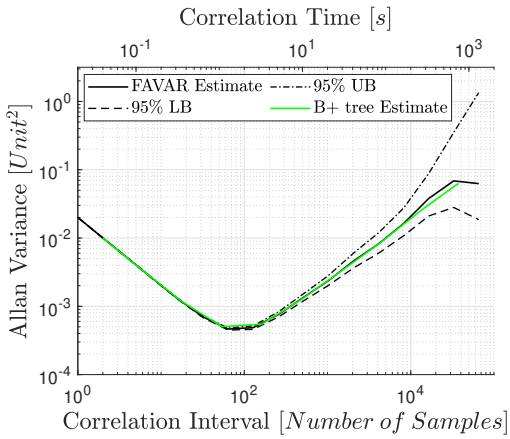

Fig. 10. AVAR of white noise added to random walk estimated using the B$^+$-tree lies within the confidence bounds of the estimate using FAVAR.

6. CONCLUSION

This paper shows how B^+-tree data structure can organize data inherently similar to the FAVAR algorithm enabling the estimation of AVAR of large datasets with low computational costs. B^+-tree implementation of AVAR presented in this work is within the 95% confidence bounds of the estimate using FAVAR. The accuracy of AVAR estimation using B^+-trees is demonstrated using three common noise signals: white noise, random walk, and white noise added to a random walk. These idealized signals are chosen for clarity in presentation, and there are no obvious algorithmic challenges in extending the work to ordered arbitrary signals. In the case of unordered signals, the AVAR estimate using B^+-tree will have errors for a brief duration. However, once the database receives all the data, B^+-tree inherently sorts them and provides accurate AVAR estimates. Similar to errors that can arise with irregularly-sampled AVAR implementations, the tree implementation will also have errors at the lowest levels because of the unequal number of children between nodes at the same level. Future work can explore the magnitude of the estimation error caused by the asymmetry in the tree structure.

REFERENCES

Abramski, K., Colley, A., Baker, H., and Hall, D. (1990). Offset frequency stabilization of rf excited waveguide co/sub 2/ laser arrays. *IEEE Journal of Quantum Electronics*, 26(4), 711–717. doi:10.1109/3.53388.

Allan, D.W. (1966). Statistics of atomic frequency standards. *Proceedings of the IEEE*, 54(2), 221–230. doi:10.1109/PROC.1966.4634.

Allan, D.W. (1987). Time and frequency (time-domain) characterization, estimation, and prediction of precision clocks and oscillators. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 34(6), 647–654. doi:10.1109/T-UFFC.1987.26997.

Barnes, J.A., Chi, A.R., Cutler, L.S., Healey, D.J., Leeson, D.B., McGunigal, T.E., Mullen, J.A., Smith, W.L., Sydnor, R.L., Vessot, R.F.C., and Winkler, G.M.R. (1971). Characterization of frequency stability. *IEEE Transactions on Instrumentation and Measurement*, IM-20(2), 105–120. doi:10.1109/TIM.1971.5570702.

Bayer, R. and McCreight, E.M. (1972). Organization and maintenance of large ordered indexes. *Acta Informatica*, 1(3), 173–189. doi:10.1007/BF00288683.

Beckmann, N., Kriegel, H.P., Schneider, R., and Seeger, B. (1990). The r*-tree: An efficient and robust access method for points and rectangles. SIGMOD '90, 322–331. Association for Computing Machinery, New York, NY, USA. doi:10.1145/93597.98741.

Brigham, E.O. and Morrow, R.E. (1967). The fast fourier transform. *IEEE Spectrum*, 4(12), 63–70. doi:10.1109/MSPEC.1967.5217220.

Comer, D. (1979). Ubiquitous b-tree. *ACM Comput. Surv.*, 11(2), 121–137. doi:10.1145/356770.356776.

Cooley, J. and Tukey, J.W. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19, 297–301.

Galleani, L. (2010). The dynamic allan variance ii: a fast computational algorithm. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 57(1), 182–188. doi:10.1109/TUFFC.2010.1396.

Galleani, L. (2011). The dynamic allan variance iii: Confidence and detection surfaces. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 58(8), 1550–1558. doi:10.1109/TUFFC.2011.1982.

Galleani, L. and Tavella, P. (2008). Detection and identification of atomic clock anomalies. *Metrologia*, 45, S127. doi:10.1088/0026-1394/45/6/S18.

Galleani, L. and Tavella, P. (2015). The dynamic allan variance iv: characterization of atomic clock anomalies. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 62(5), 791–801. doi:10.1109/TUFFC.2014.006733.

Giles, A., Jones, S., Blair, D., and Buckingham, M. (1989). A high stability microwave oscillator based on a sapphire loaded superconducting cavity. In *Proceedings of the 43rd Annual Symposium on Frequency Control*, 89–93. doi:10.1109/FREQ.1989.68841.

Graefe, G. (2004). Write-optimized b-trees. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, 672–683.

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. SIGMOD '84, 47–57. Association for Computing Machinery, New York, NY, USA. doi:10.1145/602259.602266.

Haeri, H., Beal, C.E., and Jerath, K. (2021). Near-optimal moving average estimation at characteristic timescales: An allan variance approach. *IEEE Control Systems Letters*, 5(5), 1531–1536. doi:10.1109/LCSYS.2020.3040111.

Jensen, C.S., Lin, D., and Ooi, B.C. (2004). Query and update efficient b+-tree based indexing of moving objects. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, 768–779.

Jerath, K., Brennan, S., and Lagoa, C. (2018). Bridging the gap between sensor noise modeling and sensor characterization. *Measurement*, 116, 350–366. doi:https://doi.org/10.1016/j.measurement.2017.09.012.

Knuth, D.E. (1998a). *The Art of Computer Programming, Volume 1: (3rd Ed.) Fundamental Algorithms*. Addison Wesley Longman Publishing Co., Inc., USA.

Knuth, D.E. (1998b). *The Art of Computer Programming, Volume 3: (2nd Ed.) Sorting and Searching*. Addison Wesley Longman Publishing Co., Inc., USA.

Maddipatla, S.P., Haeri, H., Jerath, K., and Brennan, S. (2021). Fast allan variance (favar) and dynamic fast allan variance (d-favar) algorithms for both regularly and irregularly sampled data. *IFAC-PapersOnLine*, 54(20), 26–31. doi:https://doi.org/10.1016/j.ifacol.2021.11.148. MECC 2021.

Malkin, Z. (2016). Application of the allan variance to time series analysis in astrometry and geodesy: A review. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 63, 582–589.

Nievergelt, J. (1972). Binary search trees and file organization. SIGFIDET '72, 165–187. Association for Computing Machinery, New York, NY, USA. doi:10.1145/800295.811490.

Sinanaj, L., Haeri, H., Maddipatla, S.P., Gao, L., Pakala, R., Kathiriya, N., Beal, C., Brennan, S., Chen, C., and Jerath, K. (2022). Granulation of large temporal databases: An allan variance approach. *SN Computer Science*, 4(1), 7. doi:10.1007/s42979-022-01397-2.