Evaluation of Satellite Clock and Ephemeris Error Bounding Predictability for Integrity Applications

Xinwei Liu, Rebecca Wang, Juan Blanch, Todd Walter

Stanford University GPS Lab

ABSTRACT

The Advanced Receiver Autonomous Integrity Monitoring (ARAIM) concept requires a conservative characterization of the nominal pseudorange error bounds. This paper explores the behavior of the Gaussian bounding parameters of the user projected clock and ephemeris errors. In a previous study [4], we quantified the inherent uncertainties in the bounding parameters by using the bootstrap probability distributions for the maximum projected clock and ephemeris errors. This paper presents a single set of parameters that captures the inherent variability of the bounding parameters. In addition, we reduce the computational cost by using the direct bounding method. We then partition the data for different observable conditions and examine the differences among the bounding parameters to determine whether the current bounding behavior is representative of future data. Our study shows that the means of the Gaussian bounding are small, most of the standard deviations are below the user range accuracy, and the near-fault data points affect the stability of the bounding parameters for different observable conditions.

I. INTRODUCTION

The Advanced Receiver Autonomous Integrity Monitoring (ARAIM) concept relies on characterizing errors from the space segment, the user segment, and the control segment. With the introduction of dual-frequency signals, the satellite clock and ephemeris nominal error characterization will become more critical after eliminating the ionospheric error bound. The nominal errors are the error values less than a constant multiplied by the User Range Accuracy (σ_{URA}). To ensure the integrity is met, we need to guarantee that we are conservatively bounding over the nominal errors (error less than 4.42 times σ_{URA} for GPS). We use an algorithm to calculate a conservative Gaussian bound over the errors that is stable through convolution [2] [3]. The bounding algorithm first computes a symmetric unimodal (SU) distribution to bound over the nominal errors, then uses a Gaussian distribution to bound over the SU distribution. In doing so, we generate two bounding parameters, $bias$ as the Gaussian mean and σ as the Gaussian standard deviation.

Satellite nominal clock and ephemeris errors bounding parameters have inherent variability with different satellites, time, satellite block, σ_{URA}, and the age of data or time since the last upload (TSLU). Our previous work investigated this variability by examining the bounding parameters over time [4]. To further explore this effect, we applied the bootstrap method to explore the inherent variability in the bounding parameters applied mainly to the GPS constellation, which allows us to compute probability distributions of the bounding parameters [4]. Our previous study estimated the amount of data required for a relatively stable characterization of the constellation errors. Furthermore, we found that the near-fault data points affect the bounding parameter stability [4].

The bootstrap method measures our uncertainty in the bounding distributions by generating a probability distribution of the bounding parameters. However, we also need to provide the user with a single set of Gaussian bounding parameters that incorporates the uncertainty in the bounding process. To solve this problem, we develop a method for computing a single set of Gaussian bounding parameters that accounts for its probability distribution. We introduce the mathematical formulation in section II.

With this new development, we seek to utilize the User Projected Error(UPE) to compute the bootstrap statistics. In [4], we applied the bootstrap method to the maximum projected error(MPE). However, MPE values are generally bimodal and thus lead to artificially large bias due to this bimodal effect [5]. By using UPE, we can eliminate the bimodal effect of the MPE. To apply the bootstrap algorithm to UPE, we reduce the computation time by replacing the original bounding algorithm with a direct bounding algorithm, which reduces the computation cost of the UPE bootstrap process. We elaborate on the UPE direct bounding bootstrap algorithm and the justification in section III.

In this paper, we also expand our study of variability to different satellites, satellite blocks, σ_{URA}, and TSLU to see how bounding parameters changes with the above variables. We elaborate on the different partitions in section IV. These partitions give us insights into whether the current error bounding parameter values are representative of the future error bounding parameter behavior. The experiment results are presented in section IV with the GPS data from 2008 to 2021.

II. BOOTSTRAP SINGLE GAUSSIAN BOUNDING PARAMETER

In our previous study, we utilize the bootstrap method to generate possible alternative error histories to explore the data's inherent variability. As a result, we obtain the probability distribution of the bounding parameters. Depending on the spread of the distribution, we can estimate the stability of the parameters. However, the probability distributions are hard to use; thus, we present a way to formulate a single parameter that captures all the variabilities explored in the bootstrap process in this section.

Our error bounding distributions are Gaussian distributions with certain means and standard deviations equal to σs as $error\ bound \sim \mathcal{N}(bias, \sigma)$. The probability bound of the absolute value of an error larger than some bounding value L, using the Gaussian bound as its probability, can be written as $2(1 - Q(\frac{L-bias}{\sigma}))$ where Q is the CDF of the Gaussian distribution. Following the bootstrapping step, we obtain the probability distribution for the set of bounding parameters $bias$ and σ as $p_b(bias, \sigma)$. Then equation 1 can be used to express the probability of the absolute value of the error ε larger than L

$$P(|\varepsilon| > L) \leq 2 \times (1 - Q(\frac{L - bias}{\sigma})) \tag{1}$$

From the bootstrap process, we obtained the probability distribution of the bounding parameters. Each set of bounding parameter value $(bias, \sigma)$ has a probability of $p_b(bias, \sigma)$. In order to capture this probability distribution or the variabilities of the parameters, we apply marginalization. We rewrite the expression in the following form

$$P(|\varepsilon| > L) \leq \int_{bias,\sigma} 2 \times (1 - Q(\frac{L - bias}{\sigma}))p_b(bias, \sigma)d(bias, \sigma) \tag{2}$$

Furthermore, we wish to express our probability distribution $P(|\varepsilon > L|)$ as a single Gaussian distribution CDF using the following equation

$$P(|\varepsilon| > L) \leq 2 \times (1 - Q(\frac{L - Bias}{\Sigma})) \tag{3}$$

Here, $Bias$ and Σ are the Gaussian bounding parameters. We equate the left hand side of equation 2 and 3 and formulate equation 4

$$\int_{bias,\sigma} 2 \times (1 - Q(\frac{L - bias}{\sigma}))p_b(bias, \sigma)d(bias, \sigma) = 2 \times (1 - Q(\frac{L - Bias}{\Sigma})) \tag{4}$$

Equating the two sides of the equation allows $Bias$ and Σ to account for the probability distribution on the left-hand side of the equation. In this way, $Bias$ and Σ are the new Gaussian distribution parameters that capture the inherent variability in the bounding parameters. The Gaussian CDF can be expressed as $Q(\frac{L-bias}{\sigma}) = \frac{1}{2}(1 + erf(\frac{L-bias}{\sigma\sqrt(2)}))$.The above expression can be simplified to

$$\frac{L - Bias}{\Sigma} = erf^{-1}(\int_{bias,\sigma} erf(\frac{L - bias}{\sigma\sqrt(2)})p_b(bias, \sigma)d(bias, \sigma)) \tag{5}$$

Knowing $p_b(bias, \sigma)$ provided by bootstrap simulation, the parameter $\frac{L-Bias}{\Sigma}$ can be computed numerically. This method gives us the relation between the three parameters L, $Bias$, and Σ, with the latter being the Gaussian bounding parameters that capture the inherent variability in the data set.

III. UPE DIRECT BOUNDING BOOTSTRAP ALGORITHM

One downside of using a statistical method such as bootstrap is the computational cost. Since the bounding procedure is repeated many times for bootstrap, we need to guarantee that each bounding step is computationally cheap. The most computationally expensive step is a procedure we refer to as the symmetric unimodal(SU) condition checking. This process stems from the Gaussian bounding algorithm. The bounding algorithm first computes an empirical CDF based on the distribution. Then it applies a symmetric unimodal (SU) bounding algorithm to produce a SU CDF distribution over the empirical CDF. Finally, the algorithm produces a Gaussian bound over the SU CDF. The SU bounding used in the algorithm relies on linear programming and is conditioned on excess mass. This bounding step is necessary as it guarantees stability through convolution for Gaussian bounding. [2] The bias we use has to satisfy the SU bounding conditions simultaneously for all the users [2]. The process is shown in figure 1.

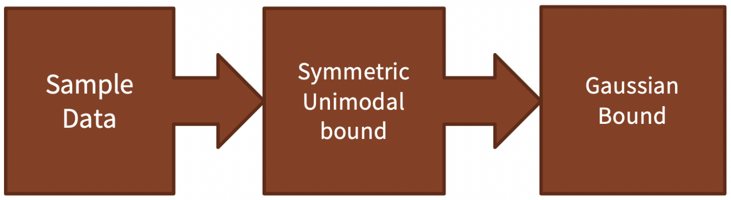


Figure 1: Bounding process achieved by two steps: the SU bounding and the Gaussian bounding

Since we are working with 200 users, we need to find the smallest bias that satisfies all users' SU conditions. We use the binary search to find the bias. We apply a bias to all 200 users each time to check if this bias satisfies the SU condition for all 200 users. Using binary search, we can find the smallest possible bias to achieve the SU condition for all the users.

We then use this bias as the input for the bounding algorithm, compute the σs that correspond to each user, and take the largest σ as the bounding parameter. This process is shown in figure2

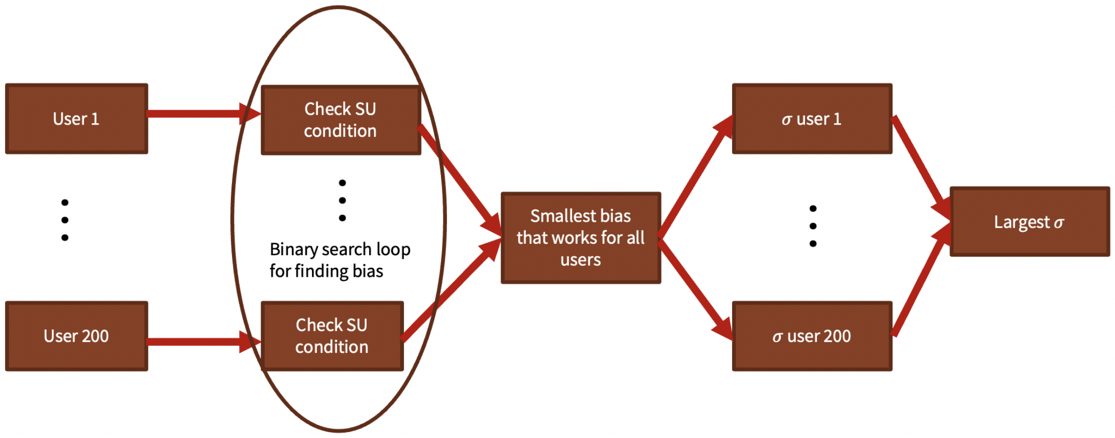


Figure 2: Process to find the smallest feasible bias and the corresponding largest σ among the users

This search step requires $O(log(n))$ computation step given the possible bias range length to be n. Each step checks the SU conditions 200 times since we have 200 users, making it the most computationally expensive step. This step is further applied to all 49 SVNs. The entire procedure is repeated 1000 times for bootstrap, which becomes time-consuming. To reduce computational time, we apply the direct bounding algorithm.

The essential idea behind this algorithm is that we neglect the SU bounding and directly apply the Gaussian bounding to the empirical CDF. In this way, we no longer need to check for SU conditions. Thus, we can remove the binary search. We refer to this method as the direct bounding method. We refer to the original bounding method as the two-step bounding method.
One shortcoming of the direct bounding algorithm is that it relies on the user to provide the $bias$ value. Here, we recommend that the user pick the median of the given data set. Specifically, we compute the median for the "right" and "left" empirical CDF and pick the larger one. We do not elaborate on the bounding algorithm further. The detail of the bounding process can be found in the previous study "Gaussian Bounds of Sample Distributions for Integrity Analysis" [3].

In order to justify this simplification, we check experimentally to ensure that given the same $bias$ value, the two methods produce similar σs. The results are shown in the experimental section. Based on the provided plots, the σ values generated by the two methods are close. In other words, we could substitute the two-step bounding with the direct bounding without affecting the bootstrap result for the given data set severely. This replacement is not always valid, but it appears to hold for the data sets we are analyzing.

IV. THREE PARAMETERS COMPARISON AND ALGORITHM

In the previous sections, we have established the mathematical tools for computing the single bounding parameter that accounts for the variability in the bounding parameters and the bootstrap computation simplification utilizing the direct bounding algorithm. We can compute the following three sets of bounding parameters given a set of data: the two-step bounding parameters, the direct bounding parameters, and the single bootstrap bounding parameters.

1. Two-step bounding algorithm

The first parameter is computed by applying the two-step Gaussian bounding algorithm. Taking the example of computing the bounding parameter corresponding to one SVN, we show the algorithm in 1

Algorithm 1: Two Step Gaussian Bounding for One SVN

```
Result: Two Step Gaussian Bounding $bias$ and $\sigma$ for one SVN
Take the 200 user data corresponding to the SVN;
for all the users do
    Compute the right and left medians of the empirical CDF;
    Take the largest of the two values to be the lower bound for bias;
    Compute the right and left maximum value for the empirical CDF;
    Take the smallest of the two values to be the higher bound for bias;
    Gather the lower and higher bound for bias give the bias range for this particular user;
end
Compute the union of the ranges for all users;
while upper bias bound and lower bias bound difference smaller than a small value $\epsilon$ do
    Set bias to be the median of the bias range;
    Check the SU condition for all users corresponding to the bias;
    if all users satisfy the SU condition then
        Set the current bias as the upper bias bound;
    else
        Set the current bias as the lower bias bound;
    end
end
Get the final bias after the while loop condition is met;
for all the users do
    Compute the $\sigma$ corresponding to the bias using the two-step bounding algorithm;
end
Take the largest of the $\sigma$s to be the bounding parameter
```

2. Direct bounding algorithm

The direct bounding algorithm is a simplification of the two-step bounding algorithm. The algorithm is shown in Algorithm 2 In our experiment, rather than computing the bias, we directly use the bias computed from the two-step bounding to compare σ

Algorithm 2: Direct Gaussian Bounding for One SVN

```
Result: Direct Gaussian Bounding $bias$ and $\sigma$ for One SVN
Take out the 200 user data corresponding to the SVN;
for all the users do
    Compute the right and left medians of the empirical CDF;
    Take the largest of the two values to be the bias;
end
for all the users do
    Compute the $\sigma$ corresponding to the bias using the direct bounding algorithm;
end
Take the largest of the $\sigma$s to be the bounding parameter
```

values with the same $bias$.

3. Bootstrap bounding algorithm

The bootstrap algorithm combines the bootstrap process, the direct bounding algorithm, and the overall bounding algorithm shown in section II. We first divide the error data into units. Then we do a sample with replacements on the units, getting a new set of data that is the same size as the original data set. We apply the direct bounding algorithm to the new data set. This process is repeated 1000 times, which gives us 1000 $bias$ and σ values. Let the number of bootstrap resamples be N. We discretize equation 4 and set the p_b for each set of bounding parameter to be $\frac{1}{N}$. We get equation 6 through some algebraic manipulation.We then apply equation 6 to compute the single set of bounding parameters that captures the bootstrap probability distribution.

$$\frac{L - BIAS}{\Sigma} = Q^{-1}\left(\frac{1}{N}\sum_{i=1}^{N} Q\left(\frac{L - \text{bias}(i)}{\sigma(i)}\right)\right) \tag{6}$$

Here, N is the bootstrap time, which is 1000 in our case. Each bootstrap result generates a set of $bias(i)$ and $\sigma(i)$. We then take the Gaussian CDF Q and sum the results. After dividing the result by N, we take the inverse of Gaussian CDF, which gives us the left-hand side of the equation. With the given relation for L, $Bias$, and Σ, we need to guarantee that the bounding parameters $Bias$ and Σ work for all possible bounding value L, and we need to eliminate one extra degree of freedom in the equation.

a) parameter selection

To eliminate the extra degree of freedom, we choose to set $Bias$ to be the same as the $bias$ computed from the two-step bounding process using the original data before bootstrapping. To guarantee that the bounding parameters work for all L values, we loop through different Ls and compute their corresponding Σs. The largest Σ is used as the overall bootstrap bounding parameter. The range of L we pick here is from the $bias$ computed using the two-step bounding process applied to all the data to the maximum error value. The algorithm is shown in 3

Algorithm 3: Bootstrap Overall Bounding for One SVN

```
Result: Bootstrap Overall Bounding $bias$ and $\sigma$ for one SVN
Take one SVN error data with 200 users as the original sample;
Designate the length of sampling units;
Divide the data into units by time periods; each consists of 200 user data;
for 1000 times do
    Sample with replacements on the units, producing resamples with the same size as the original sample;
    Apply direct bounding algorithm to the new sample, compute and store the $bias$ and $\sigma$;
end
Designate the $L$ range from the $bias$ computed using the two-step bounding algorithm applied to the original sample to the
  largest error data value from the original sample;
Set the $Bias$ to be the $bias$ computed using the two-step bounding algorithm applied to the original sample;
for all the possible $L$ values do
    Compute the $\Sigma$ value using equation 6
end
Use the largest $\Sigma$ as the bounding parameter
```

In this way, we have obtained three sets of bounding parameters, the two-step bounding, the direct bounding, and the bootstrap bounding. Here, the $bias$ values are set to be the same for comparison purposes for all the bounding methods.

V. DATA PARTITION AND EXPERIMENTAL RESULTS

With the bounding algorithm, we can produce a single set of bounding parameters that captures the inherent variability. Given the available data, we further explore how the bounding parameters vary with different observable conditions. In this sense, we can evaluate the stability over different observable condition values. The observable conditions we are interested in are time period, satellite vehicle number (SVN), satellite blocks, user range accuracy, and the age of data. This section elaborates on how we partition the data and its motivation.

1. Time period

The first variable is the time period. Given the time series of the satellite clock and ephemeris errors, we want to know how it evolves with time. In other words, we wish to evaluate the stability of the bounding parameters over time. Here, we take the error data for a certain time window. We then slide the time window to get the corresponding data. For example, we set the time window for GPS to be 3 years. We take the data from 2008 to 2010 and compute the corresponding bounding parameters. We then slide the time window by 1 year, take the data from 2009 to 2011 and compute the bounding parameters again. We repeat the process until we get to the year 2021. Particularly, we aggregate all satellites together. The process is shown in figure 3 If we observe similar bounding parameter values corresponding to each period, we can draw some preliminary conclusions

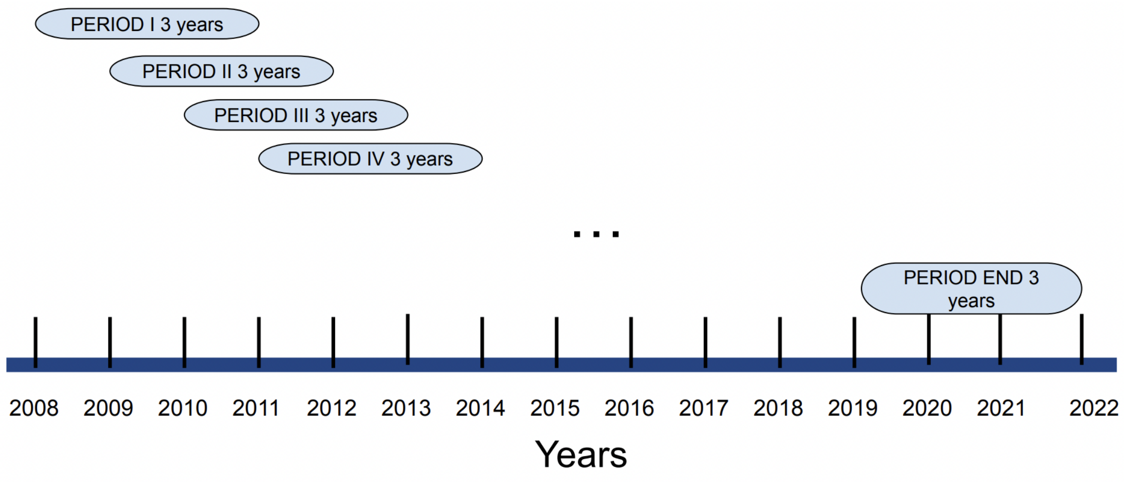


Figure 3: Time period division

regarding the stability of the bounding parameters over time and use this information to predict future error-bounding behavior.

2. Satellite vehicle number(SVN)

The second variable is the satellite vehicle number. SVN is one of the observable conditions available to the users. For a given SVN, we seek to provide a valid upper bound corresponding to the SVN condition such that the bounding parameters can be provided to the user if the observable condition is specified. In addition, we want to know if the given set of SVN error data is representative of SVN error data that are yet to be collected. To make rough predictions of the future error data, we need to explore how the bounding parameters vary with different SVNs and whether the bounding parameters are stable for different SVNs. To achieve this evaluation, we partition the error data by SVN. Here, for each satellite, we take the entire error history from 2008 to 2021 and apply the bounding algorithm. We set the $bias$ to be the same for all the SVNs and compute the different SVN's σ bounding values.

3. Satellite blocks

The third variable is satellite block. For GPS, the satellites are divided into several blocks. The users can choose which satellite block they are interested in. With the satellite block being another observable condition, we need to provide the users with a valid upper bound for each block. We take the entire time span for each block and aggregate all the satellites in the block to compute the error bounding parameters. We repeat this process for all the blocks and compare among the blocks.

4. User range accuracy (URA)

The fourth variable is the URA, which is another observable condition. Following the reasoning for the other two observable conditions, we want to provide the valid upper bound for a given URA value. We compute the error bounding parameters for each URA error data point. Each error data point corresponds to a particular URA, and the users have the freedom to choose which URA they are interested in. We take the entire time span of data and divide the errors by different URA values. Each set contains all 200 users' data points corresponding to that URA. Here, we aggregate the error data for entire time period and for all satellites.

5. Time since last update (TSLU)

Finally, we repeat the process for the age of data or TSLU by dividing the error data by different TSLU and computing their corresponding bounding parameters. If relative consistency is achieved for each variation's error bounding parameters, we can conclude that high stability exists in the data and draw conclusion regarding the data yet to be collected.

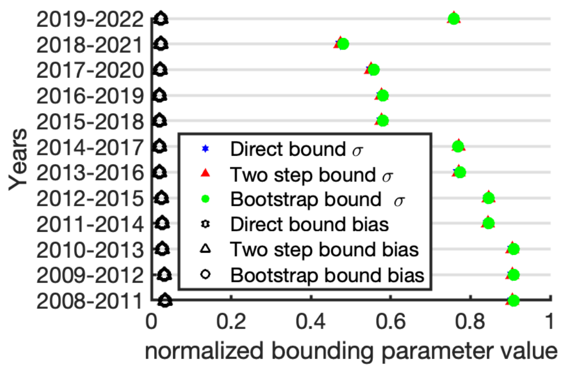
6. Near-fault data elimination

In the previous study [4], we found that the near-fault data points affect the stability of the error bounding parameters. In this study, we explore this effect by regarding the near-fault data points as faulted and evaluate their impact on the bounding parameter stability with respect to different observable condition values.

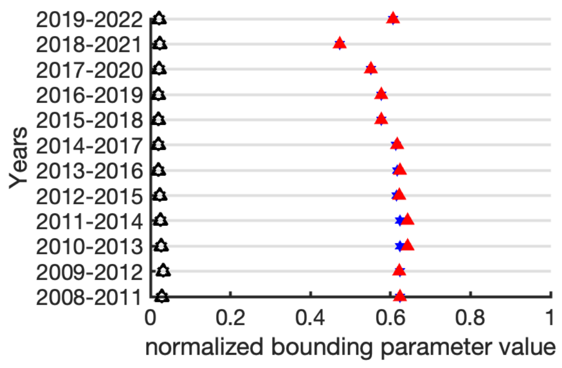
7. Experiment and result

This subsection shows the experiment results for the three types of bounding parameters computed for different partitions. We are given the GPS satellite clock and ephemeris error data normalized by σ_{URA} from 2008 to 2021 for each SVN. We are further given the URA and TSLU data corresponding to each epoch and satellite. For a single SVN, we are given 200 users projected errors, the error vector projected to the user line of sight. There are 200 users evenly distributed around the globe.

We first compute the two-step, direct, and bootstrap bounding parameters for time period variation with a 3 years time window and slide every 1 year. The data is computed for GPS satellite clock and ephemeris nominal UPE from 2008 to 2021. We set the $bias$ for the three types of bounding to be the same for better σ comparisons. The results are shown in figure 4



(a) nominal threshold with $4.42 \times \sigma_{URA}$



(b) threshold with $3 \times \sigma_{URA}$

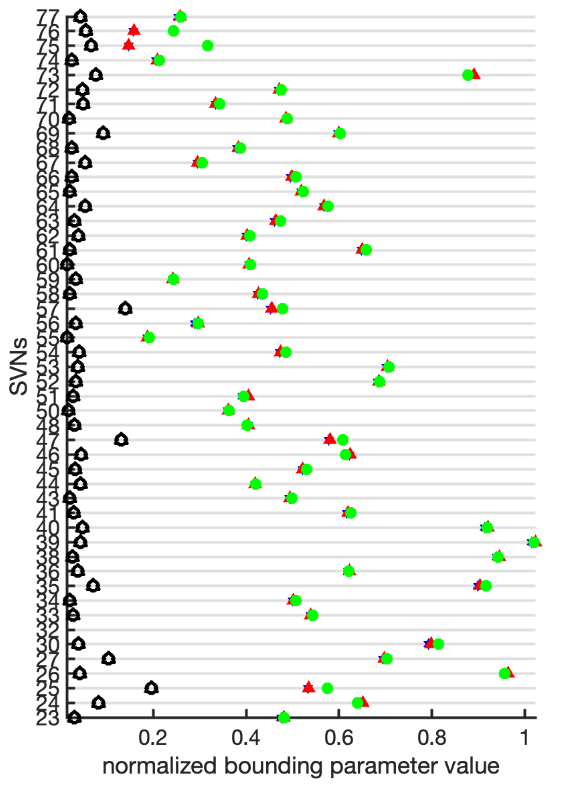
Figure 4: Bounding parameter results for direct, two-step and bootstrap bounding processes varying in time. The blue dots are the direct bounding σs computed using the designated 3 years of data. The red dots are the two-step bounding σs computed using the designated 3 years of data, and the green dots are the bootstrap bounding σs computed using the abovementioned method. The black circles are the biases. Since we set them to be the same for three bounding methods, they overlap.

The left figure is the temporal variation of the bounding parameters for nominal errors normalized by σ_{URA}. As we can observe from the plots, the bias values are small, with the largest value being 0.034. All the σs are below 0.91. The three processes produce σ with similar values, with the largest difference being 0.0084. The standard deviation of the two-step σ bounding parameter value is 0.16.

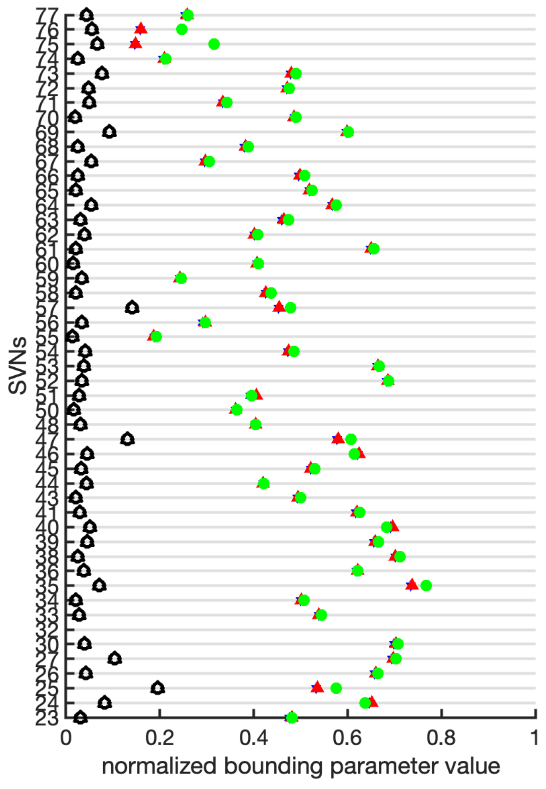
The right figure is the temporal variation of the bounding parameters for nominal errors normalized by σ_{URA} excluding the near-fault data points by eliminating the normalized error data points larger than 3. This plot does not contain the bootstrap data points. As we can observe from the plots, the bias values are small, with the largest value being 0.031. All the σs are below 0.64. The three processes produce σ with similar values, with the largest difference being 0.019. In addition, the bounding parameter σ becomes more stable for different time periods. The standard deviation of the two-step σ bounding parameter value is 0.048. In other words, the σ parameter's variation for time decreases and becomes more stable by regarding the near-fault data points as faulted.

From the result, we can make 5 observations. First, the biases are small. Second, all σ values are below 1. Third, if we regard the near-fault data points as faulted, the bounding parameter σ becomes more stable. Since the σ parameter is stable through time, it might be possible for us to make predictions of the future data and regard the past data as relatively representative. Fourth, the two-step bounding σ and the direct bounding σ are close to each other, which makes it possible to use the direct bounding method for bootstrap for the sole purpose of computational cost reduction. Fifth, the small differences among the three bounding methods indicate that there might be a possibility that we could potentially substitute the bootstrap and the two-step bounding processes with the direct bounding process. The bootstrap bounding gives the bounding parameters that take into account the inherent variability, and the two-step bounding guarantees that the bounding is stable through convolution [3]. Although applying the two-step bounding algorithm is not computationally costly, this process is repeated for 200000 times, 200 times for 200 users, and 1000 times for 1000 bootstrap resamples.

We also computed similar bounding parameters for each SVN. Here we use 14 years of UPE data for GPS, each with 200 users. The results are shown in figure 5



(a) nominal threshold with $4.42 \times \sigma_{URA}$



(b) threshold with $3 \times \sigma_{URA}$

Figure 5: Bounding parameters for error data partitioned by SVN. On the left, we have the nominal error, and on the right, we use a lower threshold to explore the near-fault error data point effect

The normalized nominal error biases are below 0.14, with the largest bias associated with SVN 25, 47, and 57. The σ values are all below 1.02. The largest difference among the three bounding σs is 0.17 from SVN 75. The standard deviation of the two-step σ values is 0.22.After eliminating the near-fault data points, the largest bias is 0.20. The σ values are all smaller than 0.77. The largest difference among the three bounding σs is 0.1690 from SVN 75. The standard deviation of the two-step σ values is 0.16.Here, most error data have small biases, σ values are below 1, and the different bounding σs have small differences with a few exceptions. The standard deviation of the two-step σ values decreased after eliminating the near-fault data points.

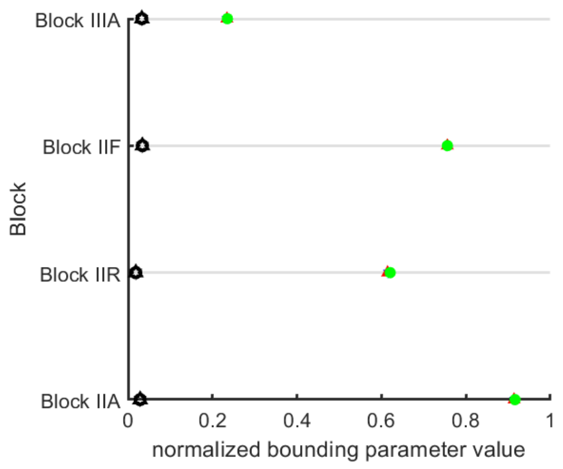
We now show the results for each satellite blocks for GPS in figure 6

Here, we observe similar behavior as in the case with varying time periods. For the nominal error, the largest bias is 0.033, the largest σ is 0.91, and the largest σ difference is 0.0084. The standard deviation of the two-step σ values is 0.29. For the error data after eliminating the near-fault data points, the largest bias is 0.033, the largest σ is 0.64, and the largest σ difference is 0.019. The standard deviation of the two-step σ values is 0.19.

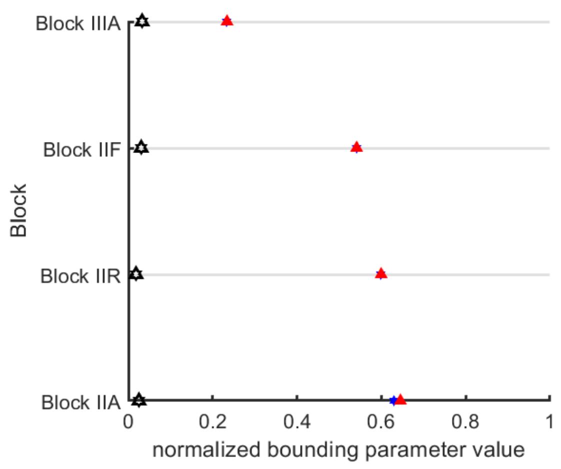
We then plot the data partition by URA in figure7

For this partition, for the nominal error, the largest bias is 0.094, the largest σ is 0.86, and the largest σ difference is 0.0081. The standard deviation of the two-step σ values is 0.27. For the error data after eliminating the 0.0081 near-fault data points, the largest bias is 0.072, the largest σ is 0.60, and the largest σ difference is 0.0081. The standard deviation of the two-step σ values is 0.21.

Finally, in figure 8, we show the error bounding parameters partitioned by the time since last upload.

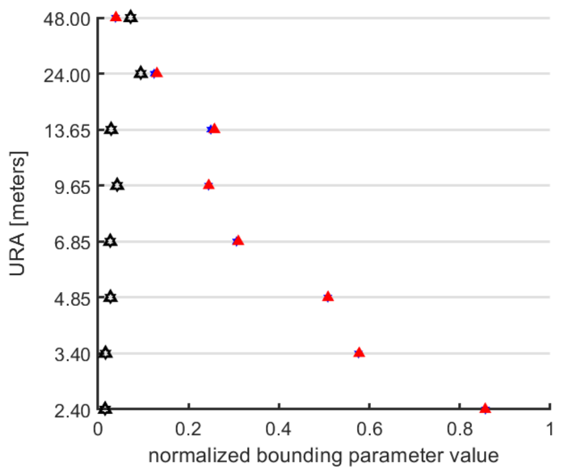


(a) nominal threshold with $4.42 \times \sigma_{URA}$

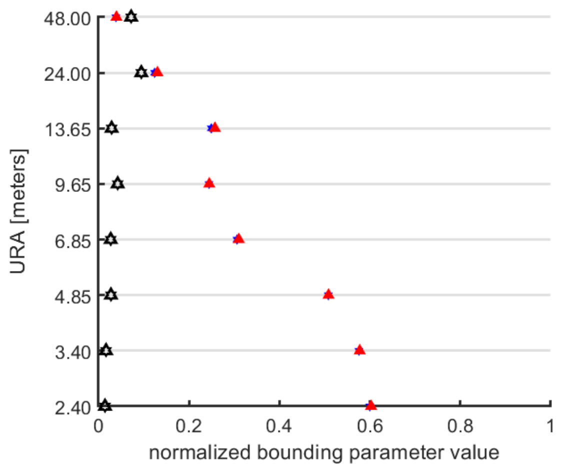


(b) threshold with $3 \times \sigma_{URA}$

Figure 6: Bounding parameters for error data partitioned by satellite blocks. On the left, we have the nominal error, and on the right, we use a lower threshold to explore the near-fault error data point effect



(a) nominal threshold with $4.42 \times \sigma_{URA}$

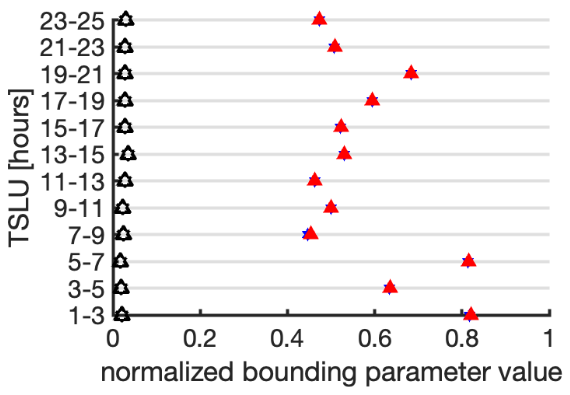


(b) threshold with $3 \times \sigma_{URA}$

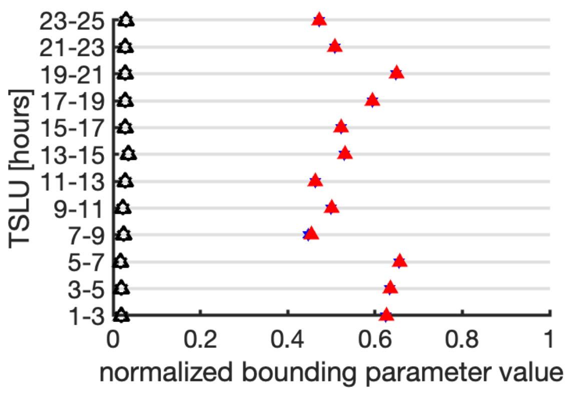
Figure 7: Bounding parameters for error data partitioned by user range accuracy. On the left, we have the nominal error, and on the right, we use a lower threshold to explore the near-fault error data point effect

For this partition, for the nominal error, the largest bias is 0.034, the largest σ is 0.82, and the largest σ difference is 0.0073. The standard deviation of the two-step σ values is 0.13. For the error data after eliminating the near-fault data points, the largest bias is 0.034, the largest σ is 0.66, and the largest σ difference is 0.0073. The standard deviation of the two-step σ values is 0.076.

The above results suggest that the bias value is small for most partitions. The normalized σs are below 1 except for SVN 39, the differences among the two-step bounding, direct bounding, and bootstrap bounding σs are small, with the largest being 0.17 for SVN 75, but overall the differences are smaller than 0.01. The standard deviation of the σ bounding parameter decreases after eliminating the near-fault data points. However, the small difference between the two-step and the direct bounding algorithm is due to the nature of the data being symmetric and unimodal. The two methods are not equivalent mathematically and will possibly produce different results given another set of data with different properties . The above results suggest that the near-fault data points could potentially increase the stability of the bounding parameters.



(a) nominal threshold with $4.42 \times \sigma_{URA}$



(b) threshold with $3 \times \sigma_{URA}$

Figure 8: Bounding parameters for error data partitioned by user time since last update. On the left, we have the nominal error, and on the right, we use a lower threshold to explore the near-fault error data point effect

VI. CONCLUSION

This study introduces a new way to capture the inherent variability in the satellite clock and ephemeris bounding parameters associated to the bootstrap algorithm. We then review the two-step bounding algorithm and reduced the computational cost by introducing the direct bounding algorithm. Furthermore, we present different ways to partition the error data. The result shows that for GPS satellite clock and ephemeris errors from 2008 to 2021, partitioned by time, SVN, satellite blocks, URA, and TSLU, the bias bounding parameter values are small, with a few exceptions. The σ values are all below 1 except for SVN 39. The near-fault data points affect the bounding parameter stability for the observable conditions. The differences among the σs produced by the two-step bounding algorithm, the direct bounding algorithm, and the bootstrap bounding algorithm are small.

ACKNOWLEDGEMENTS

We gratefully acknowledge the support of the FAA for funding this work.

REFERENCES

[1] Pullen, Sam, Lo, Sherman, Katz, Alec, Blanch, Juan, Walter, Todd, Katronick, Andrew, Crews, Mark, Jackson, Robert, "Ground Monitoring to Support ARAIM for Military Users: Alternatives for Rapid and Rare Update Rates," Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021), St. Louis, Missouri, September 2021, pp. 1481-1507.

[2] Blanch, Juan, Liu, Xinwei, Walter, Todd, "Gaussian Bounding Improvements and an Analysis of the Bias-sigma Tradeoff for GNSS Integrity," Proceedings of the 2021 International Technical Meeting of The Institute of Navigation, , January 2021, pp. 703-713.

[3] J. Blanch, T. Walter and P. Enge, "Gaussian Bounds of Sample Distributions for Integrity Analysis," in IEEE Transactions on Aerospace and Electronic Systems, vol. 55, no. 4, pp. 1806-1815, Aug. 2019, doi: 10.1109/TAES.2018.2876583.

[4] Liu, Xinwei, Blanch, Juan, Walter, Todd, "Investigation into Satellite Clock and Ephemeris Errors Bounding Uncertainty and Predictability," Proceedings of the 2022 International Technical Meeting of The Institute of Navigation, Long Beach, California, January 2022, pp. 252-272.

[5] Gunning, K., Walter, T., Gao, G., amp; Powell, J. D. (2021). Safety critical bounds for precise positioning for aviation and autonomy (dissertation).

[6] Hesterberg, Tim Monaghan, Shaun Moore, David Clipson, Ashley Epstein, Rachel Freeman, W York, Company. (2005). Bootstrap Methods and Permutation Tests. Introduction to the Practice of Statistics. 14.

[7] Victor Chernozhukov. Econometrics. Spring 2017. Massachusetts Institute of Technology: MIT OpenCouseWare, https://ocw.mit.edu/. License: Creative Commons BY-NC-SA.