Fifteenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2023)

Debuffering Timestamped ADS-B Records for Kinematic Applications

Zhuoxuan Cao, David J. Lovell
Department of Civil and Environmental Engineering and
Institute for Systems Research
University of Maryland
College Park, MD, USA
zcao1235@umd.edu, lovell@umd.edu

Aishwarya Bokil
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH, USA
bokil.6@buckeyemail.osu.edu

Seth B. Young
Department of Civil, Environmental, and Geodetic Engineering and
Center for Aviation Studies
The Ohio State University
Columbus, OH, USA
young.1460@osu.edu

***Abstract*—Because the Automated Dependent Surveillance – Broadcast (ADS-B) data format does not include explicit timing information, messages are typically time-stamped upon receipt, after they have passed through a number of buffering steps associated with demodulating and decoding the signal. These queues push apparent message arrival times later in time, and more importantly, affect the inter-arrival times between messages, which can then have downstream impacts when trying to use time-based kinematic data such as velocities and accelerations. This paper demonstrates how this buffering process manifests itself, and introduces a queueing based debuffering algorithm to ameliorate the situation. Two forms of validation are offered, one based on the numerical integration of rate-of-climb data to produce elevation, and one based on implicit timestamp differences derived from the native ADS-B data stream.**

Keywords-ADS-B, aviation data, aircraft tracking

I. INTRODUCTION

In January 2020, the Federal Aviation Administration required [1] that aircraft operating in certain airspace within the United States be equipped with Automated Dependent Surveillance - Broadcast (ADS-B) technology [2, 3]. Since this mandate, approximately 60% of the United States fleet of registered civil use aircraft have been so equipped. This technology has provided the nation's air traffic control system with an additional effective and accurate means of aircraft flight tracking.

ADS-B can use the same Mode S transponders that answer when interrogation messages are sent from ground-based air traffic control facilities. The kinematic information broadcast by the transponder is more precise than what can be inferred from a primary radar system. In addition, ADS-B provides the ability for aircraft to broadcast in a "squitter", or unsolicited mode (not as the result of an interrogation). This mode is important because these regular broadcasts can be observed not just on the ground, but by other aircraft flying in the vicinity, to help them form better situational awareness. As a side benefit, ADS-B data can not only be received by other aircraft in flight, but also by the general public, including researchers interested in leveraging ADS-B data to optimize air traffic system performance.

Specifically, aircraft equipped with ADS-B "out" technology broadcast the aircraft's position in three dimensions (latitude, longitude, and altitude), telemetry (ground speed, track, and rate of climb), and the aircraft's identifying information (the aircraft's unique ICAO identification number and in some circumstances the tail number or call sign (such as airline and flight number). These data are transmitted in the form of ADS-B "messages" approximately 2 times per second. Any aircraft equipped with ADS-B "in" technology, along with ground-based receiver stations within line-of-sight and range of the transmitting aircraft may receive and interpret these messages. This provides increased safety benefits to other

aircraft in terms of enhanced traffic awareness and to air traffic control in providing an added layer of surveillance.

Beyond the safety benefits that ADS-B provides in real time, hobbyists and researchers may receive ADS-B data for their own real time and archival use. The research community, for example, is seeing increasing uses of archival ADS-B data for other kinds of analyses. For example, Sun et al. [4] demonstrate algorithms based on machine learning and fuzzy logic for stitching together individual location records into contiguous flights, and then segmenting those flights into different phases. Cao and Lovell [5] did something similar, but focused on distinguishing operations types, such as touch-and-go flights, individual landings or takeoffs, etc. Mitkas et al. [6] focus on operations that are particular to airports with significant flight training activities. Sun et al. [7] use a year's worth of data from one ADS-B data crowdsourcing platform (the OpenSky Network) to gain insights into trends in ADS-B equipage, fleet management, and other statistics. Schultz et al. [8] analyzed the ADS-B data from Zurich airport and extracted essential parameters for airport performance, including taxi times, runway occupancy times, and ground trajectories. This team further extended their research to other small/medium airports and proved the important role ADS-B plays in airport decision-making [9]. Proud [10] used ADS-B data for the detection of go-around operations.

Some of the applications are aimed directly at assessing, or improving, airport capacity. Powell et al. [11] posit that ADS-B data can be used in support of shrinking distances between consecutive landing aircraft to improve capacity. Mitkas et al. [12,13] use ADS-B data to measure runway occupancy times, arrival and departure velocities, taxiway usage, and other parameters important for capacity estimation at small general aviation airports. Capacity estimation can be informed by identifying specific flight operations, and several studies have demonstrated processes to do this with ADS-B data [4,14].

In our own studies, and apparently in those of others, it is possible to produce meaningful results even when relying on inexpensive hobbyist level radio receiver equipment. However, there are some important ramifications. A typical setup would include some kind of antenna (indoor or outdoor), connected through a bandpass filter to a demodulator dongle. A software defined radio (SDR) is then used on a computer (Raspberry Pis are popular for this purpose) to digitally decode the demodulated signal and convert it into what would be considered raw ADS-B messages. It is important to recognize that the original ADS-B data, as broadcast by the aircraft, do not contain any absolute or relative chronological timing information. Timing information can be critical for downstream analyses, however. For example, one cannot make proper use of derivative kinematics like velocities or accelerations, without knowing the intervals of time over which those values should be extended.

A typical solution to this problem is to timestamp the received ADS-B messages with the local machine's time (e.g. in POSIX time). However, there are two important ways that this can produce timestamps with erroneous intervals between them. First, the act of demodulating and decoding the data is itself a bandwidth-limited queue; hence results do not come out of the back end with the same intervals they entered the front end. Similarly, the local processing unit (e.g., the Raspberry Pi) is also capacity constrained. The messages that come out of this process are therefore buffered by at least two capacity-constrained mechanisms.

Another factor affecting the message stream is message collisions. We know that the original signals, with different type codes, are broadcast (usually) at approximately 2 Hz frequency, with some random dithering added to transmission times to help minimize the risk of message collisions from different aircraft [15]. Table I shows some details of the subset of type codes that pertain to kinematic information. In heavy traffic, there is still a distinct possibility of message garbling, which means that some of the intended messages will simply be lost. Again, there is no timing information contained in the message stream, nor is there anything like a sequence number, so it is impossible to be certain, at the receiving end, which messages, if any, were lost.

If neither the buffering nor the message collisions described above were to occur, then the applied timestamps would be accurate, and one would expect to see messages coming out the back end of the decoding process with inter-arrival time intervals averaging ½ second, with only very small deviations from this mean value. In practice, however, we almost always see something quite different. Figure 1 shows an example from a typical message stream of the inter-arrival times between consecutive decoded messages. The larger inter-arrival times all cluster closely around some multiple of 0.5 seconds, with the noise most likely representative of dithering. Clearly, however, there are also some very small intervals (less than 0.5 seconds) that cannot correspond to the original transmission intervals. The larger intervals are indicative of situations where intervening data must have been lost. The purpose of this paper is to introduce a new debuffering process that can be applied to mitigate against these buffering issues. The method itself can be applied in two different ways – both as part of the message decoding / amalgamation process, which is explained below, or after decoding. The process will be validated in two different ways, where some semblances of ground truth timing data can be constructed from the ADS-B message stream, and we show that analytical processes employing the debuffering algorithm are more accurate than those without it.

It should be noted that there are expensive hardware solutions to this problem. Advanced demodulation hardware sometimes comes with the ability to tag outgoing data with GPS timestamps. Relative time intervals inferred from this expensive hardware would be much closer to the intervals at which the radio frequency messages were actually received, which should be quite consistent with how they were broadcast. The methods of this paper, then, are intended to be applied in situations where the more readily available and affordable hobbyist equipment is used. It is important to remember that this hobbyist level equipment forms the backbone of most of the crowd-sourced ADS-B data repositories, so this is indeed an important application. Additionally, there is an increasing number of small businesses that are marketing ADS-B data products and analytical solutions to small airports, and whose hardware platform and software functionality may not be any more robust than the hobbyist level technology described here.

TABLE I. TRANSMISSION FREQUENCY BY TYPE CODE [12]

Messages	Type Codes	Ground (still)	Ground (moving)	Airborne
Surface position	5-8	0.2 Hz	2 Hz	-
Airborne position	9-18, 20-22	-	-	2 Hz
Airborne velocity	19	-	-	2 Hz

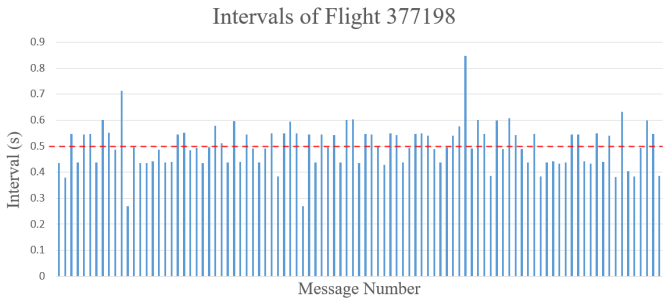

Figure 1 – Typical inter-arrival times of buffered messages

II. DEBUFFERING ALGORITHM

This section describes the general concept of the debuffering algorithm. We then introduce some detail as to at what stage in the decoding process this process should occur. The basic tenets of the debuffering algorithm are quite simple:

- Any timestamp assigned to an ADS-B message must, in fact, be later than the actual time at which that message reached the radio system. The extraneous time is due to the processing required for demodulation/decoding, plus any queueing delays in those systems while they were busy processing other messages.
- Any consecutive messages with timestamp differences less than ½ second were not transmitted at this interval, and their recorded time interval must instead be (improperly) reflective the throughput of the decoding process.

With this in mind, we propose pushing assigned timestamps backwards in time, through a first-in, first-out (FIFO) queue that runs in reverse time, with a capacity of 2 messages/second (this is the transmission rate common to most of the type codes in ADS-B messages). Doing this is not guaranteed to push the messages back to their correct arrival times, but it is our hypothesis that, en masse, they will be closer to the truth than they were with the buffering included. In broad strokes, this process is illustrated in Figure 2.

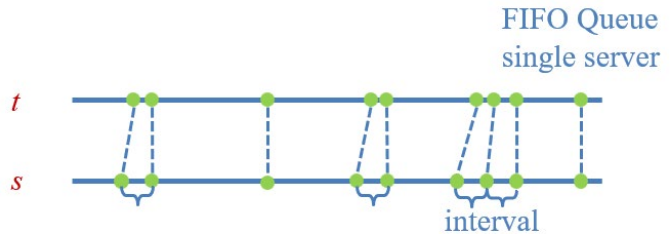

Figure 2 – Illustration of debuffering algorithm

In this figure, the top timeline represents the times at which received ADS-B messages were timestamped, and it is understood that several of the intervals between messages are less than 0.5 seconds. Working from right to left, those events are pushed through a FIFO queue with capacity 2 messages/second, yielding the timestamps shown in the bottom timeline. If t_i and s_i are the buffered and debuffered timestamps, respectively, of message i, then the FIFO queue is invoked by considering the messages in decreasing chronological order, and applying the following recursion:

$$s_i = \min\{t_i, s_{i+1} - 0.5\}. \quad (1)$$

There is an additional detail that can influence the choice of how to deploy the debuffering algorithm. ADS-B messages come in many different type codes, each containing only part of the overall set of data that is of interest at each time step. For example, airborne position messages are encoded using a Compact Position Reporting (CPR) format. This format allows high resolution position information to be transmitted with fewer bits.

If the receiver has no information about a starting absolute position (this is always the case with anonymous ground receiver equipment), then two messages must be combined in order to properly decode the aircraft's latitude and longitude [15]. Additionally, it is common usage to construct a data stream where each line of data contains a single sample of each of the statistics of interest, and hence messages must be amalgamated. The most popular open source software for decoding 1090 ES ADS-B messages is pyModeS [16]. Its mechanism for amalgamating messages relies, in some sense, on the buffering described above to have happened. Essentially, message timestamps are rounded to the nearest second, and then combined. If any duplicate information is included, only the most recent is kept. This amalgamation process is a very practical solution for buffered data, but it retains the inaccuracy of the buffered timestamps. A conundrum, then, is that if the messages are debuffered as described above, then the resulting data stream will not have enough messages in close temporal proximity to allow the rounding process to group them together into complete messages. As such, using the debuffering algorithm to improve timestamp accuracy also predicates a need to use an alternate approach to message amalgamation. Also, knowing that messages of different type codes are broadcast independently of each other, each with its own frequency, this is also an opportunity to consider stratifying the messages by type code, and applying the debuffering process to each separately. Thus, we have constructed two separate variants of the debuffering algorithm, each considering the amalgamation and stratification processes differently:

1. Amalgamate the data using the rounding process in [16], and debuffer complete messages. This variant is particularly useful when working with archival data where the complete messages have been saved, but the original raw ADS-B messages have been discarded.
2. Stratify the raw messages by type code in the decoder, debuffer them separately, and then amalgamate using a different process, described in the next section. The advantage of this method is that it is conceptually more

consistent with the reconstruction of the original timeline of message transmission, and so it should provide more accurate results, when the original raw ADS-B messages are available.

III. Message Amalgamation

The first 5 bits of the message section of an ADS-B message indicate the message type. In particular, in order to properly decode position, from its CPR format, a pair of odd/even position messages is required. Here we should note that there are two types of position, i.e., surface position and airborne position. For the surface position messages, the altitude and rate of climb are defaulted to be 0. Surface position messages can also describe the movements and ground track, which are also known as surface velocity. Then, the surface and airborne velocities are decoded by the software. Unlike surface velocity, airborne velocity messages are separate from airborne position messages. The rounding method of amalgamation relies on the premise that, once message timestamps are rounded to integer seconds, there will be the necessary odd/even position records, velocity records, etc., in the same group, to allow them to be merged to form a complete record. There can be excess messages of various types encountered during this process, and in this case older versions are dropped. Hence, this message has the added disadvantage that data are being lost; data that would have had a higher chance of being retained if the timestamps had been pushed back closer to the truth. When we use this rounding process and debuffer the resulting complete records, we call this "post-decoder debuffering".

The alternative is what we call "in-decoder debuffering". Each message type is debuffered separately, before being decoded and merged into a complete record. We begin by debuffering and decoding surface and airborne position messages. Those decoded position messages then become the "anchors" on the debuffered timeline, onto which we attempt to attach any other missing information, such as velocities, to form complete records.

Because the surface position messages also contain velocity information, the timestamps will still be the same even if we debuffer them separately. Thus, for the surface velocity and surface position, we can directly merge them without any data loss. In terms of the airborne velocity and position, instead of rounding and dropping, for every position, we look for the nearest airborne velocity message that is within the range of 0.5 seconds. Figure 3 shows an example of this process and the various ways it can play out.

In this figure, the blue circles denote the positions on the timeline where the debuffered and merged position records fall. These need to be merged with velocity records, which are the red circles.

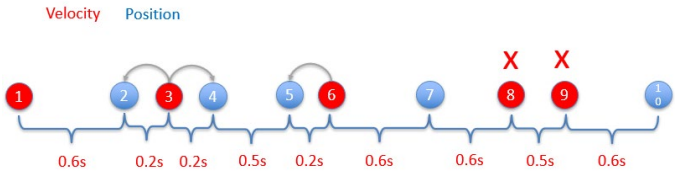

Figure 3 – Matching process for merging complete records

Each position record is mated with its nearest velocity record, so long as that velocity record is within 0.5 seconds. If there are no such nearby velocity records, then only the position is retained, as this is deemed the most important part of the message stream (and, in fact, velocities could be inferred from numerical differentiation of position wherever those data were missing). Any velocity records not near enough a position record to be merged are ignored. Presumably, this happens when the position record that should have been nearby was garbled. Another reason for prioritizing the position records over the velocity records, and in particular prioritizing their retention of their timestamps, is that position changes regularly as the airplane is flying, and velocity is less subject to change. Thus, adjusting a velocity record slightly in time should have very little impact on accuracy. This same presumption is also followed in the rounding process in the post-decoder debuffering variant.

An important advantage of the in-decoder variant is that all position records are retained (the rounding process in the post-decoder variant might erase some), and then matched with as many velocity records as possible. As a result, the total number of complete records generated using the in-decoder variant is larger. Figure 4 shows the extent to which additional data are retained using in-decoder debuffering. In that figure, the differences in height between the blue and orange bars represents the data loss for complete flights if timestamp rounding is used (e.g., in the post-decoder debuffer). The green and yellow bars show similar information when attention is limited to post-takeoff messages; clearly the impact of data loss is felt most for surface messages.

IV. Validation Case Study: Altitude Integration

Complete ADS-B messages contain both barometric altitude data, in units of feet, and rate-of-climb information, in units of feet per minute. At first glance, it might appear that there is redundancy here, as the altitude data should be the integral of the rate of climb data. However, as noted above, there are instances where the rate of climb data in an aircraft's message stream seem to be reasonable, while the altitude data do not. In such cases, it would make sense to construct an alternate version of the altitude data by integrating the rate of climb data [5]. The complication is that one needs to know what time intervals over which to perform the integration. One way to validate the debuffering algorithm is to perform this rate of climb integration on flights whose altitude data is also reliable. The time intervals can be inferred from the consecutive differences in timestamps. If debuffering is doing a good job, then the debuffered timestamps should better replicate the altitude data (taken to be the "truth") than the originally buffered timestamps would.

Examples of applying this idea to some individual flights are shown in Figures 5-7. The numerical integration is performed using the trapezoidal rule (because rates of climb can change quickly), but similar results can be found using Simpson's rules.

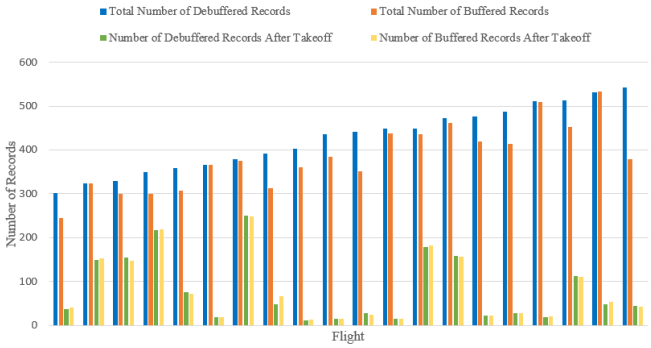

Figure 4 – Data retention comparison

Figure 5 shows a case of a single takeoff. The red line is the ground truth altitude data reported in the ADS-B data stream. The yellow line shows the results of integrating rate of climb to estimate altitude, based on the original buffered timestamps. Finally, the green line represents doing the same thing but with debuffered timestamps, and it is clearly more representative of the truth.

In Figure 6, we highlight a single flight that takes off and climbs (the taxiing portion of the trajectory is omitted from the figure), leveling off at just over 1000 feet, where it dwells for a short period, and then climbs again to just over 2000 feet. The integration step is inaccurate at the beginning of this exercise, because the timestamps were buffered, and this error remains for the rest of the trajectory. It is notable that in Figure 6, an incoherent section, which is caused by buffered timestamps, exists at the beginning of the takeoff operation. The same phenomenon can be found in many other takeoffs in KOSU, while the buffered timestamps in landings tend to be uniformly distributed along the time axis. A possible reason is that the layout of KOSU runways could lead to a higher signal density at the takeoff end, especially the area below 1500ft near KOSU. Both the original trajectory and the integrated trajectory are improved by applying debuffering, and thus a much better match is produced, as shown in Figure 7.

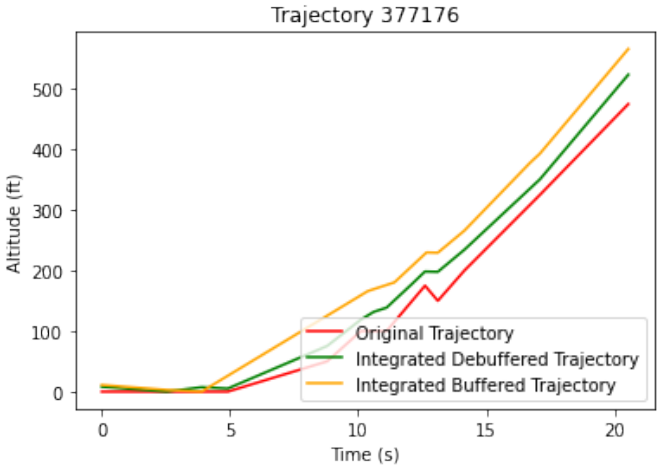

Figure 5 – Altitude accuracy improvement with debuffering

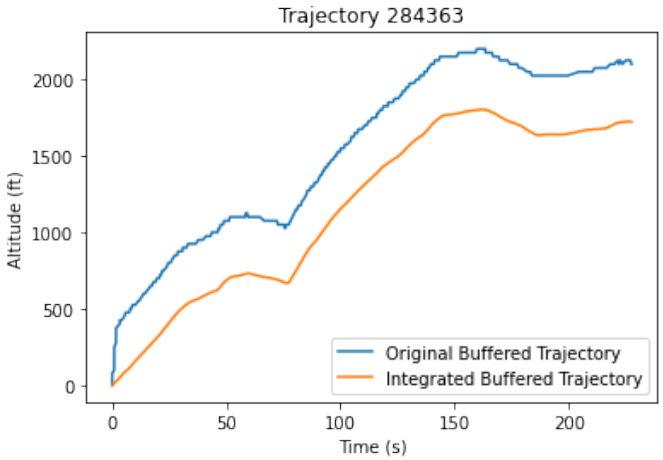

Figure 6 – Altitude integration with buffered data

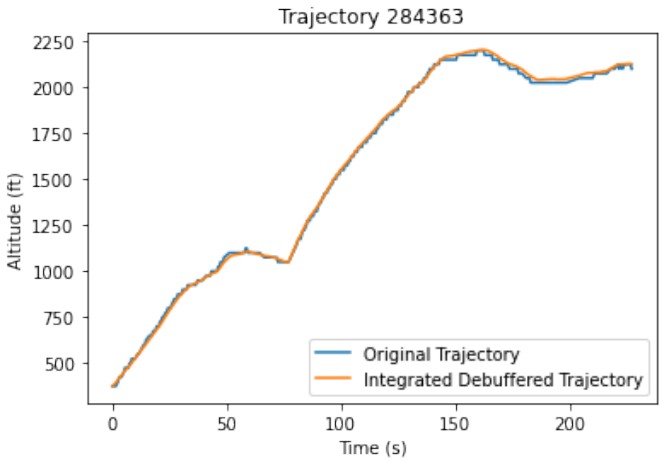

Figure 7 – Altitude integration with debuffered data

For a more comprehensive test, we selected 100 flights to test this integrated altitude comparison. Of these flights, 75 are single takeoffs, and the rest are touch-and-go flights. We deem the recorded altitude in the received messages as the ground truth. We construct two cases for each flight: first, we integrate the rates of climb using the buffered timestamps, and compare to the altitude data (assessed at those same buffered timestamps); second, we perform the same operation with the debuffered timestamps. The error metric in each case is the final vertical distance (in units of feet) of altitude deviation between the true and the integrated altitude. We call this measurement the "drift" because it represents the final extent by which the two profiles have deviated.

For the single takeoff flights, we start the experiment at the point of takeoff, which is identified using an algorithm developed in [17]. The touch-and-go flights all start and end at zero feet altitude above ground level (AGL). Altitude integration is more challenging for such flights, because it is our experience that the rate of climb data are biased towards positive rates of climb. Thus, even though the actual flight profile would suggest that the troughs of the altitude profile should be very close to 0 feet AGL, the integrated data show their altitudes increasing over time, contributing to the final value of drift. This is ameliorated somewhat, but not entirely, by the debuffering

process. Other possible explanations include differences in "visibility" of the transmitting antenna on the aircraft in climbing and descending attitudes, which could also depend on the receiver antenna placement. At this point, this phenomenon is not completely understood, and more inquiry is required.

In Table II, we show a sample of 38 out of these 100 flights. Any row with a positive sign for the value of the improvement indicates that the drift derived from debuffered timestamps is better (i.e., less) than what would have been obtained with the original buffered timestamps. For these results, all of the debuffered drifts are calculated using in-decoder debuffered data.

TABLE II. IMPROVEMENT IN DRIFT METRIC

Flight ID	Debuffered Drift (ft)	Buffered Drift (ft)	Improvement (ft)
356015	-29.11	-72.56	43.45
356185	42.87	48.80	5.93
356930	182.90	349.12	166.21
380627	-2.23	-30.43	28.20
283714	-7.43	-197.40	189.97
282936	-3.79	-20.38	16.58
284363	26.14	-377.91	351.77
271063	-5.21	-0.65	-4.55
282983	-30.04	-39.14	9.10
149569	-0.24	-0.87	0.63
284168	20.24	25.31	5.07
284451	-38.95	-57.16	18.22
283423	243.98	247.98	4.00
282676	139.93	142.03	2.10
283123	106.60	106.58	-0.02
283900	-70.10	-63.77	-6.33
270408	-12.18	-56.22	44.04
284314	-48.43	-23.00	-25.43
275489	7.19	-33.34	26.14
282259	36.81	24.76	-12.05
283207	5.02	8.47	3.45
378372	-11.75	1.03	-10.72
376508	-64.25	-70.18	5.93
376662	-29.39	-30.75	1.37
378598	-237.54	-246.57	9.03
376660	-10.64	-16.63	5.99
376792	49.36	72.40	23.04
377324	-300.37	-275.52	-24.84
377198	62.66	71.84	9.18
377754	-21.67	-27.36	5.69
377547	-107.07	-102.62	-4.45
378488	-16.98	-14.25	-2.72
377035	-37.27	48.30	11.03
377752	25.87	48.51	22.63
376918	-33.63	-72.12	38.49
377176	48.56	91.04	42.48
381389	-50.25	-13.72	-36.52
378880	-69.37	-84.71	15.34

From the takeoff flights, we found that 52 of the 75 tested takeoffs have smaller debuffered drifts. The average improvement is 40.77 feet. Only 2 cases have similar drifts in the buffered and debuffered trajectories. For the rest of the cases, the buffered drifts and debuffered drifts have an average difference of 12.63 feet, which is even smaller than the altitude resolution of 1090 MHz devices (25 feet). Moreover, 12 of the 15 cases with buffered drifts over 100 feet have smaller debuffered drifts. This result indicates that debuffering has the potential to ease drifts for flights with large initial integration deviation. As for the touch-and-go flights, 80% of the flights' drifts are improved by debuffering. Compared to takeoffs, the touch and goes' drifts are much larger; some of them are over 1000 feet, while the others are over 100 feet. The reason is that longer flight durations will cause more drifts to accumulate.

V. VALIDATION CASE STUDY: CORRELATION OF IMPLICIT AND EXPLICIT TIMESTAMP DIFFERENCES

Despite the ADS-B message stream not containing explicit time information, it is possible to approximate the elapsed time between consecutive messages using the other telemetry data in those messages. For message i, we denote the latitude lat_i, longitude lon_i, and speed v_i. If we are willing to assume that the acceleration is constant between two consecutive messages (this is not an onerous assumption, as aircraft tend not to exhibit large values of jerk), then the mean velocity between the two instances is

$$v_m \cong \frac{v_{i+1}-v_i}{2}. \quad (2)$$

The elapsed time between the two messages can then be estimated by

$$\Delta t \cong \frac{d(lat_i,lon_i,lat_{i+1},lon_{i+1})}{v_m}, \quad (3)$$

where $d(lat_i, lon_i, lat_{i+1}, lon_{i+1})$ is a distance measurement based on the two pairs of latitude and longitude. We call this estimate of the time interval the "implicit" time interval for this pair of messages. For the short time intervals expected in this application, the Haversine distance formula should be sufficiently accurate, and that is what was used to produce the following results. It should be noted that, of course, these position data are collected by the aircraft with some error distribution. If those errors are strongly correlated between consecutive measurements, then this distance estimate will be quite accurate; otherwise, it is less so.

This method of validation, then is to compare the values of Δt estimated from each pair of consecutive messages, with the differences in the recorded timestamps (either buffered or debuffered). The validation hypothesis here is that debuffering

is improving the accuracy of the timestamps if the sample correlation between the implicit timestamps and the debuffered timestamps is higher than that between the implicit timestamps and the buffered timestamps.

Figure 8 shows the results of this validation step for 50 representative flights. The blue bars represent the sample correlations between intervals formed from the original buffered timestamps, and the implicit intervals constructed from (3).The figure is organized so that the flights are sorted in increasing order of this statistic, to improve readability. The red bars show the correlation coefficients between time intervals formed from post-decoder debuffered data and the implicit time intervals. Clearly, there is a marked increase in accuracy from even this coarse method of debuffering. Not surprisingly, the magnitude of the improvement is reduced as the original correlations get higher, as correlation is clamped at 1.0. Finally, the green bars show the correlations between time intervals formed from in-decoder debuffered data and the implicit time intervals. In most cases, there is an incremental improvement from this approach. As before, the largest improvements occur when the buffered coefficients were lowest.

Compared to the drift measurement, we believe that this error metric measurement is more stable because it does not involve accumulative errors. Every pair of implicit and explicit time interval contains the information of only one pair of closely spaced data points. Therefore, implicit times and explicit times are independent from each other, so bias will not accumulate.

As part of a larger numerical experiment, we looked at all the flights without big data gaps in our database from two small airports, KOSU in Ohio and KFRG in New York. The reason for excluding these flights is that large time gaps create situations where the implicit timestamps are not reliable. The debuffering would be unaffected over a large time gap. For the 77 flights from KOSU, only one flight has a lower debuffered correlation coefficient than the buffered one. For all 384 flights from KFRG, 96.61% of the flights' correlation coefficient are improved by post-decoder debuffering. Among these flights, 100 were single takeoff operations, and 99 of them have better post-decoder coefficients than buffered coefficients. Additionally, 63 of them have better in-decoder debuffered coefficients than post-decoder. The average improvement from buffered to post-decoder debuffered coefficients was 25.8%, whereas the average improvement from buffered to in-decoder debuffering was 30.24%.

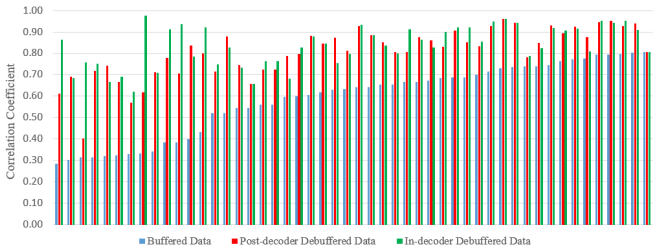

Figure 8 – Correlation coefficients between implicit and buffered/debuffered time intervals for flights from KFRG and KOSU

VI. Conclusions

This paper has articulated how a seemingly straightforward assignment of timestamps to received ADS-B messages in an inexpensive hardware/software configuration might yield timestamps whose intervals have been compressed due to buffering in the queues associated with demodulating and decoding the data. We present a simple FIFO queue debuffering algorithm to ameliorate this situation. This algorithm can be applied either after the messages have been decoded and amalgamated to form complete data records, or, preferably, as part of the internal decoding process, where messages of different type codes can be stratified and debuffered independently, which more faithfully represents the means by which they were transmitted.

The post-decoder method shows significant improvements over two validation techniques that rely on information in the ADS-B data stream as ground truth. In one case, we show that estimating altitude by numerical integration of rate-of-climb data is improved when debuffered timestamps are used, and in the second case we show that comparing relative differences of recorded timestamps to implicit timing information buried in the position and speed data also improves. In both cases, the in-decoder debuffering tends to do even better than the post-decoder method.

The in-decoder method is best, but can only be applied when original ADS-B messages are available, and they have not yet been amalgamated. In the case of archived data which have already been decoded, however, the post-decoder method still produces much better results than using buffered timestamps.

In our experience, there are cases where the data streams have large temporal gaps. This is likely due to occlusion of the antenna due to its placement in a building, or in some sense relative to the runway geometry. Any use of ADS-B data will benefit from using an antenna that is exterior mounted and has good lines of sight to the entire airfield. Of course, estimates of kinematic data over these longer time gaps are unreliable because one cannot presume that an aircraft's speed, velocity, heading, etc. change insignificantly over longer time periods.

It might be advisable to consider, when developing future replacements for the ADS-B communications protocol, incorporating some standardized timing information, if only on a subset of the transmissions, to allow some direct computation of lags to be measured between original transmission times and final decoding times. Additionally, it would be instructive to conduct direct experiments where the data from the transponder on a known aircraft were timestamped and logged inside the cockpit, and then that same aircraft's data were investigated after reception and decoding by the ADS-B receiver apparatus.

Acknowledgment

We thank Kent Duffy of the Federal Aviation Administration and Dr. Junzi Sun of the Delft University of Technology for their advice and technical guidance, and particularly Dr. Sun for the use of his open-source ADS-B decoding software. Some support for this work was provided by the Federal Aviation Administration under award number 693KA920F00175, titled

REFERENCES

[1] Final Rule, Automatic Dependent Surveillance-Broadcast (ADS-B) Out Performance Requirements to Support Air Traffic Control (ATC), 75 FR 30160 (May 28, 2010).

[2] RTCA DO-260B, Minimum Operational Performance Standards for 1090 MHz Extended Squitter Automatic Dependent Surveillance– Broadcast (ADS-B) and Traffic Information Services–Broadcast (TIS-B), December 2009.

[3] RTCA DO-282B, Minimum Operational Performance Standards for Universal Access Transceiver (UAT) Automatic Dependent Surveillance – Broadcast, December 2009.

[4] J. Sun, J. Ellerbroek, and J. Hoekstra, "Flight extraction and phase identification for large automatic dependent surveillance–broadcast datasets," Journal of Aerospace Information Systems, vol. 14(10), pp. 566-572, 2017.

[5] Z. Cao and D. Lovell, "Identifying aviation operation types using flight trajectories." In 10th International Conference for Research in Air Transportation (ICRAT), Tampa, FL, USA, 2022.

[6] D. Mitkas, D.J. Lovell, and S.B. Young, "Learning to recognize and stratify flight training activities at GA airports using ADS-B data." In: 8th OpenSky Symposium, virtual, 2020.

[7] J. Sun, X. Olive, M. Strohmeier, M. Schäfer, I. Martinovic, and V. Lenders, "OpenSky Report 2021: Insights on ADS-B Mandate and Fleet Deployment in Times of Crisis". In 2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC), pp. 1-10, 2021.

[8] M. Schultz, X. Olive, J. Rosenow, H. Fricke, and S. Alam, "Analysis of airport ground operations based on ADS-B data." In 2020 International Conference on Artificial Intelligence and Data Analytics for Air Transportation (AIDA-AT), pp. 1-9, 2020.

[9] M. Schultz, J. Rosenow, and X. Olive, "Data-driven airport management enabled by operational milestones derived from ADS-B messages." Journal of Air Transport Management, 99, 102164, 2022.

[10] S.R. Proud, "Go-around detection using crowd-sourced ADS-B position data." Aerospace, 7(2), 16, pp. 1-14, 2020.

[11] J.D. Powell, C. Jennings, and W. Holforty, "Use of ADS-B and perspective displays to enhance airport capacity." In 24th Digital Avionics Systems Conference, Vol. 1, pp. 4-D, 2005.

[12] D.Z. Mitkas, D.J. Lovell, S. Venkatesh, and S.B. Young, "Leveraging local ADS-B transmissions to assess the performance of air traffic at general aviation airports." In 14th USA/Europe Air Traffic Management Seminar (ATM), virtual, 2021.

[13] D.Z. Mitkas, D.J. Lovell, S.B. Young, and S. Venkatesh, "Developing capacity estimation metrics for airports accommodating smaller aircraft using locally collected Automated Dependent Surveillance-Broadcast data." Transportation Research Record 2676, pp. 285-295, 2022.

[14] X. Olive and L. Basora, "Detection and identification of significant events in historical aircraft trajectory data." Transportation Research Part C: Emerging Technologies 119, 102737, 2020.

[15] J. Sun, "The 1090 megahertz riddle: a guide to decoding mode s and ads-b signals." TU Delft OPEN Publishing, 2021.

[16] J. Sun, "The Python ADS-B/Mode-S Decoder." Github repository, https://github.com/junzis/pyModeS, visited 02/03/2023.

[17] A. Bokil and S.B. Young, "Development of QGIS tools to aid in airport operations modeling using ADS-B data." In 10th International Conference for Research in Air Transportation (ICRAT), Tampa, FL, 2022.