This Manuscript is a post-print (author's version) of the following article:

Andrew Skelton, Allan R. Willms, 2014. *An Algorithm for Continuous Piecewise Linear Bounding of Discrete Time Series Data*, BIT. Numerical Mathematics 54 (4), pp. 1155-1169.

An algorithm for continuous piecewise linear bounding of discrete time series data

Andrew Skelton · Allan R. Willms

Received: date / Accepted: date

Abstract An algorithm for enclosing a given set of time series data inside a continuous piecewise linear band of varying height subject to certain constraints is presented. The band is defined by two piecewise linear curves that lie above and below the data respectively. Segments of these curves are constrained to start and end at one of the data points, and those whose slope does not lie between its neighbours' slopes are required to be at least as wide as a user-specified value. The algorithm yields a band which accurately preserves the general trends of the data, while enclosing the inherent measurement noise. This band is typically obtained in $O(n \log K)$ time, where n is the number of data points and K is the number of linear segments. The algorithm is described and its capabilities are tested on four data sets. Comparisons are made with alternative algorithms.

Keywords continuous representation · piecewise linear representation · time series · algorithm · parameter estimation

Mathematics Subject Classification (2000) 65D10 · 65G40

1 Introduction

Important tasks when working with discrete time series data include describing general trends and features of the data, interpolating data values between recorded data points and reducing the number of parameters required to describe the data set. To

This work was supported by an Ontario Graduate Scholarship and a Natural Sciences and Engineering Council of Canada Discovery Grant.

University of Guelph,
Guelph, Ontario, Canada
E-mail: skeltona@uoguelph.ca, awillms@uoguelph.ca

conduct these tasks, continuous representations of discrete time series data are required. Continuous representations are used in applications such as pattern recognition [14], medicine [15] and real world agents [5]. The application will often dictate the properties required of such a representation.

There has been considerable research conducted on representing discrete data with continuous functions. These functions are usually piecewise polynomials and are often linear, in which case we will refer to such functions as piecewise linear representations (PLRs). Algorithms are typically designed to compute representations that are good, or even optimal, with respect to some error tolerance [14] or metric such as the L_2 [2, 7] or L_∞ [13, 21] norm. Many algorithms also require the representation to satisfy some given constraints, such as a limit on the number of segments [2, 4, 7, 13, 14, 21] or some prescribed geometric behaviour [12, 16].

In some applications, it is necessary to obtain an upper and lower estimate at any given time, requiring two continuous representations: one bounding the data above and the other below. We are motivated in particular by the need for continuous upper and lower bounds of discrete data for use in a parameter range reduction algorithm for ordinary differential equation models [20, 19]. Parameter estimation in this context is typically accomplished by selecting a set of initial parameter values, numerically integrating the model equations and comparing the result to the time series data. A cost function such as least squares is used to measure the suitability of these parameters. Using estimates of how the cost function varies with the parameters, new parameter values are chosen until the cost function is hopefully minimized.

It is often the case that little is known a priori about the parameter values, thus making an initial selection difficult. If the initial selection must be made from a very large region of parameter space, it is possible that this selection will result in a system that cannot be numerically integrated over the observation window. Additionally, if the cost function has multiple local minima, or large flat regions in parameter space, it may be difficult for the procedure to converge to a reasonable minimum. To combat this problem, interval parameter estimation techniques (see also [8, 9, 11, 17, 18]) can be used to dramatically reduce the size of the parameter search space. Such techniques require interval representations (that is, a lower and upper bound) of the data set at any time point.

This paper presents an algorithm that approximates discrete data with two independent PLRs. These curves and the area between them will be referred to collectively as a band. The algorithm is designed to output a band which accurately retains the trends of the data to a user-specified tolerance while also retaining a certain user-specified minimal level of error in the data measurements. *All* data points must be contained in the band, and each individual segment for both boundaries of the band must begin and end at a data point. In addition, “inflection segments”, segments whose slopes are either greater than or less than both adjoining segments, are constrained to be at least some user-specified value $t_{\min}$ wide. This condition was applied with success in [16]. It is this final constraint that prevents the band from collapsing to a trivial one of zero area that connects all points. The value $t_{\min}$ needs to be chosen large enough to prevent the curves from following the local effects of noise, but small enough to allow for identification of the global trends in the data.

1.1 Background

Early work on data representation was conducted in Bellman [2] (1961) and in Gluss [7] (1962), in which the authors used dynamic programming methods to find the best (in an L_2 sense) piecewise polynomial with K pieces fit to a continuous function $f(t)$. In [2], the piecewise polynomial was not assumed to be continuous while in [7], the piecewise polynomial was constrained so that the end points of each segment coincided with original data points. The algorithm required the evaluation of a function on a given partition and ran in $O(N^2K)$ time, where N is the number of evaluation points in the partition. This method was also applied to discrete data sets. In this case, the solution is exact and the algorithm ran in $O(n^2K)$ time, where n is the number of data points.

In 1973, Douglas and Peucker [4] presented an algorithm designed to smooth a piecewise linear curve by reducing the number of points required to describe the original curve. Since the original curve can be thought of as a discrete data set, the resulting output is an effective PLR whose segments are constrained to lie on original data points. The algorithm's expected time complexity was $O(n \log n)$, but the worst-case performance was $O(n^2)$.

In Keogh et al. [10] in 2001, the authors conducted a short review of many types of PLR algorithms and sorted them by the following categories: (1) top-down algorithms, in which the data are subdivided and the PLRs are determined on these subsets until some stopping criteria is reached; (2) bottom-up algorithms, in which adjacent segments are merged again until a stopping criteria is reached; and (3) sliding window algorithms, in which the data are scanned and segments are created as data are processed. They also presented a bottom-up, sliding window $O(n^2/K)$ algorithm. Our algorithm can be considered a top-down, sliding window algorithm.

In 2007, Moreira and Santos [12] designed an algorithm to create a concave hull of a given data set. The smoothness of the output is adjusted with a user-defined input, k . They considered a k -nearest neighbours approach where an increase of k results in a smoother curve. Their algorithm was designed to ignore noisy outlying points and focus on the best geometric description of clusters of points. In this algorithm, it was possible for the hull to splice if the data was clustered in more than one group. The algorithm appeared to scale linearly with respect to the number of points while also scaling with respect to the user-defined value.

The work in this paper was originally an extension of the algorithm presented in 2010 by Szusz and Willms [16]. There the authors designed a top-down algorithm in which the Remez exchange algorithm [6] was used to find a preliminary PLR of the entire data set. The data set was then split at the largest error and PLRs were obtained by the Remez algorithm for each resulting subset. On each subset, a constant was added and subtracted from the PLR to create a band segment in which all data was contained. Each component PLR was expanded vertically to match the tallest band segment, and the segments were joined. These join points did not necessarily coincide with the original data. The output of the algorithm was a constant-height band containing all data points. The algorithm run time was bounded above by $O(nK)$, where K is the number of segments.

1.2 The problem

The goal of this algorithm is to determine two piecewise linear curves such that all data lie between the pair of curves and that the pair of curves retain the general trends and measurement error of the data up to a user-specified resolution. Each segment is restricted to begin and end at one of the original data points, and the piecewise linear curves are required to satisfy the condition that any “inflection segment”, as defined below, have width greater than some user-specified value $t_{\min}$. A detailed mathematical description of the problem is provided below.

Given a data set $D = \{D_j = (t_j, y_j)\}_{j=1}^n$, with the t_j strictly increasing, the distance and the slope between two points are defined respectively as

$$d(D_j, D_k) = |t_k - t_j|, \quad \text{and} \quad s(D_j, D_k) = \frac{y_k - y_j}{t_k - t_j}.$$

The task is to select from D subsets of upper and lower pivot points,

$$P^U = \left\{ P_i^U = (t_{n_i^U}, y_{n_i^U}) \right\}_{i=1}^{N^U}, \quad P^L = \left\{ P_i^L = (t_{n_i^L}, y_{n_i^L}) \right\}_{i=1}^{N^L}$$

that respectively define upper and lower piecewise bounding curves by linear interpolation between the pivots. The points D_1 and D_n are required to be in both P^U and P^L . Thus the upper curve, $B^U(t)$, defined on $[t_1, t_n]$, is the union of the linear segments

$$B_i^U(t) = s(P_i^U, P_{i+1}^U)(t - t_{n_i^U}) + y_{n_i^U}, \quad t_{n_i^U} \leq t \leq t_{n_{i+1}^U}, \quad 1 \leq i < N^U, \quad (1.1)$$

and the lower curve, $B^L(t)$ is defined similarly. This choice is constrained first of all by the requirement that the resulting curves bound the data:

$$B^L(t_j) \leq y_j \leq B^U(t_j), \quad 1 \leq j \leq n. \quad (1.2)$$

The two curves B^L and B^U are independent of one another. The algorithm first determines the set of upper pivots from the original data, and then reflects the data in the y -axis and obtains the set of lower pivots as upper pivots of these reflected data. Therefore, the description below just deals with determination of P^U , and for ease of readability when context is clear, the U/L superscript is dropped.

The choice of pivots, P , is further constrained to satisfy the inflection interval width condition: each segment of the piecewise bounding curve B must either have width greater than or equal to some user specified $t_{\min}$, or have a slope that lies between its neighbours' slopes. In other words, each segment is either sufficiently wide or it is not an inflection segment. Formally, one of the following conditions must therefore be satisfied on $B_i(t)$

$$d(P_i, P_{i+1}) \geq t_{\min}, \quad 1 < i < N - 1, \quad (1.3a)$$

$$s(P_{i-1}, P_i) > s(P_i, P_{i+1}) > s(P_{i+1}, P_{i+2}), \quad 1 < i < N - 1, \quad (1.3b)$$

$$s(P_{i-1}, P_i) < s(P_i, P_{i+1}) < s(P_{i+1}, P_{i+2}), \quad 1 < i < N - 1. \quad (1.3c)$$

2 Algorithm

The algorithm selects an upper pivot set in three phases, each phase relaxing the choice of pivots by allowing an additional constraint from (1.3). The first phase simply takes the upper convex hull of the data, and therefore imposes (1.3b) on all points. The second phase allows pivots to satisfy either (1.3b) or (1.3a), and chooses pivots so that the curve fits tightly around “*hills*”, that is, intervals on which the data exhibit a concave-down trend. These peaks are separated by band segments satisfying the width requirement. In the third phase of the algorithm the focus is on fitting a tight band to “*valleys*”, intervals on which the data exhibit a concave-up trend. This is accomplished by relaxing the conditions to include (1.3c) as well. After both upper and lower pivot sets are selected, a final phase is employed, if necessary, to expand the band to the minimum user-defined height.

2.1 Phase 1: Upper Convex Hull

The algorithm first attempts to locate as many locally maximal points as possible by finding an *upper convex hull* using *Andrew’s monotone chain convex hull algorithm* [1]. Our implementation of the algorithm is a slight modification of the code presented in [3], and works by considering points from left to right. Each new point is added to the list of pivot points. Initially the pivot set is just the first point, $P = P_1 = D_1$. In turn, each point D_j , $j \geq 2$, is tested. If its addition to the pivot set $P = \{P_i\}_{i=1}^m$ causes a clockwise turn, that is, $s(P_{m-1}, P_m) > s(P_m, D_j)$, then D_j is added to P without further action. However, if its addition causes a counter-clockwise turn, $s(P_{m-1}, P_m) \leq s(P_m, D_j)$, then the last point, P_m is removed from the pivot set and this is done sequentially until either $m = 1$ or $s(P_{m-1}, P_m) > s(P_m, D_j)$, at which point D_j is added to the set. This procedure runs in $O(n)$ time and at the end of this phase the pivot set satisfies (1.3b) and includes the end points D_1 and D_n .

2.2 Phase 2: Segment Splitting

Given the set of pivots provided by Phase 1, this phase attempts to discover the peaks of hills not included in the convex hull by searching for new pivots whose segments satisfy (1.3a) or (1.3b). All pivot pairs (P_i, P_{i+1}) are initially flagged as *open*. The search starts with the leftmost pair of existing pivots. If the algorithm determines that no further pivots between this pair can be added, it flags the pivot pair as *closed* and moves to the next pair to the right. If the end of the pivot list is reached, the algorithm returns to the leftmost pair. Phase 2 terminates when each pair of pivots in P has been flagged as closed.

To process the region between the pivot pair P_i, P_{i+1} , all points strictly between the pivots, (i.e. all points (t_j, y_j) , such that $t_{n_i} < t_j < t_{n_{i+1}}$) are considered. Such points are called *interior points*, and the set of such interior points is denoted $\text{int}(P_i, P_{i+1})$. Each interior point is tested individually to determine if its inclusion in the list of pivots would produce a curve that continues to satisfy the inclusion condition (1.2).

Such points are denoted *potential pivots*. Once a list of potential pivots is obtained, one or more of these points is chosen for further consideration.

Potential pivots are classified by the effect they have on the band, were they to be added to the list of pivots. A potential pivot A is said to satisfy the *distance condition on the left or right* if $d(P_i, A)$ or $d(A, P_{i+1})$, respectively, is greater than $t_{\min}$. The point A is said to satisfy the *slope condition on the left*, if the inclusion of A in the pivot list causes (1.3b) to be satisfied on the *left-adjacent segment* $B_{i-1}(t)$. A is similarly said to satisfy the *slope condition on the right* if (1.3b) holds on the *right-adjacent segment* $B_{i+1}(t)$ were A to be added to the pivot list. All potential pivots are classified with respect to these four conditions. Figure 2.1 shows the regions in which potential pivots may be found on the sample set $\{P_i, P_{i+1}\}$ during Phase 2.

Fig. 2.1 Searching for potential pivots in $\text{int}(P_i, P_{i+1})$ in Phase 2.

Region **I** is defined as the region where all points satisfy the distance condition on both the left and right. A potential pivot in this region may be immediately and successfully added to the list of pivots. Points in Region **II** fail the slope condition on the same side that they fail the distance condition. Points in Region **III**, however, do satisfy the slope condition on the side that they fail the distance condition. After all potential pivots have been classified, they are considered by region in the following priority.

If at least one potential pivot is found in Region **I**, a subset of the potential pivots in this region is chosen to add to the list of pivots. This choice is made according to Procedure 2.1 in order to maximize the likelihood that the points chosen are maximal points of undiscovered peaks.

Procedure 2.1 Repeat until all potential pivots have been included or excluded.

1. Include the potential pivot in Region **I** with the highest y-value.
2. Exclude all potential pivots whose inclusion would violate (1.2) or (1.3a).

If two data points have identical y-values, but cannot both be selected without violating a banding condition, the point with a t -value closest to the horizontal midpoint of the entire data set is selected.

If no potential pivots are found in Region **I**, then the algorithm determines which potential pivot is the greatest impediment to further progress. For each potential pivot

C_k found in Regions **II** or **III**, define the *slope difference* to be

$$SD_k = \begin{cases} s(P_i, P_{i+1}) - s(P_i, C_k) & \text{if } d(P_i, C_k) < d(C_k, P_{i+1}), \\ s(C_k, P_{i+1}) - s(P_i, P_{i+1}) & \text{otherwise.} \end{cases} \quad (2.1)$$

The potential pivot that minimizes SD_k is chosen. In Figure 2.1, C_1 has the smallest slope difference and is clearly the most severe obstacle to further progress. The region in which this chosen point is located determines further action as follows.

1. *The chosen point is found in Region II.* In Figure 2.1, it is clear that the point labelled C_1 cannot currently be added to the list of pivots. This point, however, is preventing potential pivots being located in $\text{int}(C_1, P_{i+1})$. This is a serious problem if the distance $d(C_1, P_{i+1})$ is particularly large. If this is the case, the problem may be overcome by replacing P_i with C_1 , as illustrated in Figure 2.2. If the left-adjacent pivot pair P_{i-1}, P_i has not been flagged as closed, then there exists a possibility that further work will place point C_1 in Region **III**. The algorithm waits until progress has ceased on all adjacent intervals before proceeding with a pivot replacement. Each pivot pair is flagged as either closed, open or a pending pivot replacement. If our current pivot pair is contained entirely in a region in which the only options available are pending pivot replacements that would result in narrower pairs, then we may attempt a pivot replacement.

Since C_1 is a potential pivot, such a replacement may be safely performed without violating (1.2). If both (1.3a) and (1.3b) are satisfied when P_i is replaced by C_1 , and if $d(C_1, P_{i+1}) \geq 2t_{\min}$, (there is a possibility of finding a Region **I** potential pivot), then it is advantageous to perform such a replacement. The pivot pair C_1, P_{i+1} is then flagged open, and the pivot pair P_{i-1}, C_1 is flagged as closed. Otherwise the pivot pair (P_i, P_{i+1}) is flagged as closed.

Note that if the candidate pivot C_1 was located on the right hand portion of Region **III**, a similar set of conditions must be satisfied for it to replace pivot P_{i+1} .

Fig. 2.2 Processing potential pivots found in Region **II**. The point labelled C_1 was identified and a pivot replacement was determined to be advantageous. We then replace pivot P_i with C_1 .

2. *The chosen pivot is found in Region III.* Such points satisfy the slope condition on the closer adjacent segment and as such can be conditionally included in the list of pivots and the smaller interior may be examined for Region **I** pivots. If the potential pivot is closer to P_i than P_{i+1} , then the chosen pivot is labelled L_1 and a new search for potential pivots in $\text{int}(L_1, P_{i+1})$ is begun. Otherwise, the

chosen pivot is labelled R_1 and a search is started in $\text{int}(P_i, R_1)$. Examples of such searches are illustrated in Figures 2.3 and 2.4. The shape of the band formed by the conditional pivots illustrate the importance of these points in the fitting of curves around hills of data. These points can only be permanently included if a Region I is eventually found in some subinterval. This allows the algorithm to explore valleys to the user-specified resolution.

This process continues with narrower regions until one of the following occurs.

- (a) Region I points are found. A subset of these points determined by Procedure 2.1 and all conditional pivots are added to the final pivot list.
- (b) Region II points are identified. All conditional pivots are discarded and the pivot pair is flagged as a pending pivot replacement.
- (c) The subinterval interior is empty, or the width between the two most recently added conditional end points is strictly less than $2t_{\min}$. All conditional pivots are discarded and the pivot pair is flagged as closed.

Fig. 2.3 Processing potential pivots found in Region III. The point labelled L_1 was found while searching in $\text{int}(P_i, P_{i+1})$ and was conditionally added to the pivot list. A search now begins in $\text{int}(L_1, P_{i+1})$. No potential pivots will be found in the white region, as such a point would have originally either been processed as a Region I point, or would have taken preference over L_1 as a Region II or III point.

The algorithm continues to process pivot pairs until all pairs have been flagged as closed. We then move to the next phase of the algorithm.

2.3 Phase 3: Concave Smoothing

At this point, the algorithm has identified all valleys to the user-specified resolution. Using the list of pivots obtained in Phase 2, the algorithm now attempts to obtain a better fit in the valleys of the data by allowing segments to satisfy any of the conditions (1.3a), (1.3b), or (1.3c).

Inflection segments are considered first, that is, segments $B_i(t)$ whose slope is less than (or greater than) the slopes of both adjacent segments, $B_{i-1}(t)$ and $B_{i+1}(t)$. In Figure 2.5, the points labelled A and B would have been recorded as failing the slope condition (1.3b) on the right while processing $\text{int}(P_i, P_{i+1})$. With the addition of slope condition (1.3c), A is now a valid pivot and can be added to the final list of pivots with L_1 and L_2 . B is also a valid point, but is passed in preference to A , whose inclusion creates a narrower inflection segment.

Fig. 2.4 Processing potential pivots found in Region III. A total of $n + m$ region III potential pivots have been found and conditionally added to the pivot list, the most recent being R_m . A new search is begun for potential pivots in $\text{int}(L_n, R_m)$. Point A may now be included in the list of pivots by also including the pivots $\{L_1, \dots, L_{n-1}, L_n, A, R_m, R_{m-1}, R_{m-2}, \dots, R_1\}$. In the absence of A , point B may be added by also including the pivots $\{L_1, \dots, L_{n-1}, L_n, B, R_{m-1}, R_{m-2}, \dots, R_1\}$.

Once all inflection segments have been processed and the pivot list has been updated, the algorithm focuses on *concave-up regions* of the band, that are comprised of three or more adjacent pivot pairs, whose corresponding segment slopes satisfy (1.3c). The union of solid line segments in Figure 2.6 connecting P_{i+1} to P_{i+4} represent a sample concave-up region. A process similar to Phase 1 is used: the algorithm searches for a concave hull of the data within this region that fits as tightly as possible to the data.

Working from left to right, a point is added to the pivot list if adding it makes a counter-clockwise turn, as defined in the discussion of Phase 1. If the slope of the segment connecting the two most recently added points is greater than the slope of the right-adjacent segment, in this case $s(P_{i+4}, P_{i+5})$, then this current configuration cannot be used. The algorithm then removes points backward, starting from the most recently added point, until this slope error no longer occurs.

Unlike the convex hull algorithm, this procedure is not guaranteed to produce the “best” concave hull with respect to the area under the curve. Nor is the procedure guaranteed to contain all the pivots in the original configuration. Further, the procedure will in general yield different results if the region is instead processed from right to left, see Figure 2.6. Therefore the algorithm considers potential concave hulls by processing in both directions and comparing these to the original configuration. In each case, the area under the curve is computed and whichever results in the smallest such area is chosen for the final pivot list. This phase terminates when all concave regions have been processed.

Fig. 2.5 Illustration of Phase 3 of the algorithm. Processing an inflection segment. Segment $B_i(t)$ is identified as an inflection segment. We are allowed to add to the final pivot list the points $\{L_1, B\}$ or the points $\{L_1, L_2, A\}$. We choose the latter as it reduces the width of the new inflection segment.

Fig. 2.6 Illustration of Phase 3 of the algorithm. Processing the concave region $[P_{i+1}, P_{i+4}]$. The solid line connects the original set of pivots. The dashed line in the plot on the left shows the concave hull obtained by considering points from left to right.

2.4 Phase 4: Band expansion

The final output of this algorithm tapers to a single point at the smallest and largest time values. There may also be a pivot contained in both the upper and lower band, thus creating a band of zero height at this time point. Our main use of this banding algorithm was in a parameter estimation algorithm for ordinary differential equations. It is highly unlikely that the measurement at any time point is exact, so the user may specify a minimum height $h_{\min}$ required at each time point.

The algorithm searches through the final band and identifies any segment end points at which the band height h is less than $h_{\min}$. The algorithm then travels left and right from this point and adds and subtracts $\frac{h_{\min} - h}{2}$ respectively to the upper and lower band at each segment end point until it reaches a segment with width at least $t_{\min}$. This ensures that the slope and distance conditions are still met. While this will

Fig. 3.1 Sample bands for current-clamp data from the pyloric dilator (PD) neuron. Horizontal axis measured in seconds. Vertical axis measured in millivolts. (a) $t_{\min} = 0.0180$, (b) $t_{\min} = 0.0400$.

remove the constraint that the start of each segment must coincide with a data point, it will allow for a more accurate analysis of the features of the data set. Note, however, that the t -values of the band still coincide with t -values of the data.

3 Results

The algorithm was first tested using three real time series data sets with uniformly distributed time values. They are described in [16] as follows. The first data set is voltage recordings from a pyloric dilator (PD) neuron of a spiny lobster, where the voltage in millivolts is measured regularly every 0.2 milliseconds for 2 seconds. These recordings were made by Jack Peck of Ithaca College. The second is echocardiogram (ECG) data from PhysioNet, with measurements in millivolts taken roughly every 0.008 seconds over a period of 5 seconds. The third data set is the average monthly exchange rate between the American and Canadian dollar from 1979 through 2009, obtained from the Pacific Exchange Rate Service. All three data sets and detailed references to their original sources are available on the second author's website at <http://www.uoguelph.ca/~awillms/TASLE/>. Figures 3.1-3.3 show sample bands obtained for specific values of $t_{\min}$.

In Figure 3.1a, it is clear that the fit of the band is excellent around the convex regions and inflection segments. The weakness of the algorithm is evident in the concave regions of the data where the fit of the band is not as tight, particularly in the lower curve. The algorithm does, however, deal well with the sudden change of direction of the data at the beginning of the time measurements. In Figure 3.1b, the lower curve fits tightly to the data, but the upper curve does not extend into any of the data valleys. Since each segment is constrained to begin and end at a data point, this type of behaviour is typical if the width of the valley is too small relative to $t_{\min}$. In this case, the concave regions of the upper curve are approximately 0.075 seconds apart (less than twice $t_{\min}$) whereas for the lower band they are approximately 0.085 seconds apart. In general, the algorithm is unable to detect data trends occurring within a distance less than twice $t_{\min}$.

Fig. 3.2 Sample bands for ECG data. Horizontal axis measured in seconds. Vertical axis measured in millivolts. (a) $t_{\min} = 0.0100$, (b) $t_{\min} = 0.0600$.

In Figure 3.2a, the band fits tightly around the sudden spikes in data. The band is wider around the spike for the larger $t_{\min}$ value in Figure 3.2b since both surrounding segments are inflection segments and their width is restricted by $t_{\min}$. When $t_{\min}$ is sufficiently large as in Figure 3.2b, it is evident that the lower curve cannot enter the valleys, thus the band cannot resolve that level of data detail given $t_{\min}$. In both cases, the band demonstrates a tight fit for the regions of generally horizontal data.

The band in Figure 3.3a shows an excellent fit for smaller time values, but the rapid fluctuations in the data near the right-end of the data set present some challenge to the algorithm as the intervals of increase and decrease are too narrow. Similarly, the band in Figure 3.3b is able to resolve general trends in the data, but the value of $t_{\min}$ is too high for the band to follow more local trends.

For comparison, Figure 3.3 also includes plots that illustrate bands produced using the same $t_{\min}$ values and using the constant-height algorithm in [16]. Despite a relatively small value of $t_{\min}$, the band in Figure 3.3c fails to identify local trends in the data. The height of the band is constrained by the fit required for the rightmost region of data. It is also worth noting that the constant-height algorithm often displayed large ranges of $t_{\min}$ values for which the output band was unchanged, with large changes occurring when certain threshold $t_{\min}$ values were reached. For example, the band in Figure 3.3d is produced for all $t_{\min}$ values between 175 and 1000. The current algorithm typically displays a more gradual change in the band with respect to an increasing value of $t_{\min}$, indicating that the algorithm is more sensitive to the value of $t_{\min}$ and thus produces an output band more suitable for the users' needs.

We also tested the banding algorithm on data with irregularly spaced time measurements. We simulated data from a three-dimension HIV model published in [22]. Using their parameter values and initial conditions [22, pg. 792] we numerically solved the system to obtain a data set for T^* , the number of infected cells. We sampled 100 data points with time values logarithmically distributed from 10^{-2} to $10^{1.78}$. We then added Gaussian noise with mean zero and variance $0.01(\max\{T^*\} - \min\{T^*\})$. In Figure 3.4a, setting $t_{\min} = 0.4$ allowed a nice band fit for small t -values, but resulted in a zero height band for large t -values. In Figure 3.4b, setting $t_{\min} = 8.0$ resulted in a decent fit for large t -values, but was clearly insufficient for small t -values. By using

Fig. 3.3 Sample bands for monthly exchange rate data between the American and Canadian dollar. Horizontal axis measured in days. Vertical axis measured in U.S. dollars. The graphs on the top represent output from the algorithm, while the graphs on the bottom are bands produced from [16]. (a) and (c) $t_{\min} = 100.0$, (b) and (d) $t_{\min} = 500.0$.

$t_{\min} = 0.4$ and setting a minimum allowable height $h_{\min} = 1.5$, Figure 3.4c shows that an acceptable fit can be obtained for all t -values.

We conducted numerical experiments with a sample data set that suggested that the time complexity of the algorithm was $O(n \log N)$. The results of this experiment are available upon request. The results presented in this section, however, were typically obtained in less than 0.1 seconds on a 2.4 GHz Intel Core i5 MacBook Pro, even for the largest data sets considered. We have applied the algorithm presented in this paper to our parameter range reduction scheme. The speed at which this algorithm runs allowed provided us with the flexibility to fine tune the banding algorithm input parameters in order to output a useful and appropriate continuous band.

4 Discussion and Further Work

The algorithm described in this paper provides an efficient solution to the problem described in §1.2. The band produced by the algorithm fits tightly to the data, particularly around convex regions of data. The smoothness of the band depends on a user-defined $t_{\min}$ value, which defines a minimum length of segments whose slopes do not lie between their neighbours' slopes.

Fig. 3.4 Sample bands for the number of infected cells in a simulation of a three-dimensional HIV model. Horizontal axis measured in days. Vertical axis measured in number of cells. (a) $t_{\min} = 0.4$, minimum height = 0.0, (b) $t_{\min} = 8.0$, minimum height = 0.0, (c) $t_{\min} = 0.4$, minimum height = 1.5.

A significant strength of the algorithm is the relationship between the user-defined $t_{\min}$ value and the geometry of the final band. The algorithm is quite successful in detecting general data trends that occur in a time window wider than $2t_{\min}$, while ignoring trends occurring in narrower windows. The algorithm is successful in using the value of $t_{\min}$ to determine a band which satisfies the users' desired level of accuracy.

The fit of the band to the data around inflection and concave regions of data, while often optimal, was not as tight as other regions of data. While this fit can be improved by reducing the value of $t_{\min}$ (thus allowing a shorter inflection segment), we also considered various alternate definitions of $t_{\min}$. Any such alternative definition would have to ensure that the band does not spike around noise and continues to follow the data smoothly. The most difficult part of the algorithm was designing a procedure which can fit a band to concave regions of data. By choosing bands among three possible options, we improved the fit of the band to the data in these regions. A different definition of $t_{\min}$ however, may result in a better fit in such regions.

We were motivated in particular by ODE parameter identification schemes based on monotonic discretizations [20, 19]. Such schemes make use of the fact that data points lie in some known interval. Recent progress in the computer application of these schemes has indicated that more progress can be made on parameter range reduction when the data band exhibits a change in slope. It can be seen that the al-

gorithm presented here produces a band with more changes in slope than the original constant-height algorithm. Preliminary testing has indicated that this change in band shape results in a smaller valid parameter region when compared to a constant-height band with similar area. More comprehensive and detailed testing is currently being conducted and will be reported elsewhere.

References

1. Andrew, A.M.: Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters* **9**(5), 216–219 (1979)
2. Bellman, R.: On the approximation of curves by line segments using dynamic programming. *Commun. ACM* **4**(6), 284 (1961)
3. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: *Computational Geometry: Algorithms and Applications*, second edn. Springer-Verlag (2000). URL <http://www.cs.uu.nl/geobook/>
4. Douglas, D.H., Peucker, T.K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canadian Cartographer* **10**(2), 121–122 (1973)
5. Fitzgerald, W., Lemire, D., Brooks, M.: Quasi-monotonic segmentation of state variable behavior for reactive control. In: *Proc. of the Natl. Conf. on Artificial Intelligence*, Vol. 20, Part 3, pp. 1145–1150 (2005)
6. Fraser, W.: A survey of methods of computing minimax and near-minimax polynomial approximations for functions of a single independent variable. *JACM* **12**(3), 295–314 (1965)
7. Gluss, B.: Further remarks on line segment curve-fitting using dynamic programming. *Commun. ACM* **5**(8), 441–443 (1962). DOI <http://doi.acm.org/10.1145/368637.368753>
8. Granvilliers, L., Cruz, J., Barahona, P.: Parameter estimation using interval computations. *SIAM J. Sci. Comput.* **26**(2), 591–612 (2004)
9. Johnson, T., Tucker, W.: Rigorous parameter reconstruction for differential equations with noisy data. *Automatica* **44**(9), 2422–2426 (2008)
10. Keogh, E., Chu, S., Hart, D., Pazzani, M.: An online algorithm for segmenting time series. In: *Proc. of IEEE Intl. Conf. on Data Mining*, pp. 289–296. San Jose, CA (2001)
11. Lin, Y., Stadtherr, M.A.: Guaranteed state and parameter estimation for nonlinear continuous-time systems with bounded-error measurements. *Ind. Eng. Chem. Res.* **46**(22), 7198–7207 (2007)
12. Moreira, A., Santos, M.: Concave hull: a k-nearest neighbours approach for the computation of the region occupied by a set of points. *Proc. of the 2nd International Conference on Computer Graphics Theory and Applications* (2007)
13. Phillips, G.M.: Algorithms for piecewise straight line approximations. *The Computer J.* **11**(2), 211–212 (1968)
14. Ramer, U.: An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing* **1**(3), 244–256 (1972)
15. Shatkay, H., Zdonik, S.B.: Approximate queries and representations for large data sequences. In: *Proc. of 12th IEEE Intl. Conf. on Data Engineering*, pp. 536–545 (1996)
16. Szusz, E.K., Willms, A.R.: A linear time algorithm for near minimax continuous piecewise linear representations of discrete data. *SIAM J. Sci. Comput.* **32**, 2584–2602 (2010). DOI <http://dx.doi.org/10.1137/090769077>. URL <http://dx.doi.org/10.1137/090769077>
17. Tucker, W., Kotalik, Z., Moulton, V.: Estimating parameters for generalized mass action models using constraint propagation. *Math. Biosci.* **208**(2), 607–620 (2007)
18. Tucker, W., Moulton, V.: Parameter reconstruction for biochemical networks using interval analysis. *Reliab. Comput.* **12**, 389–402 (2006)
19. Willms, A., Szusz, E.: Parameter range reduction for ode models using monotonic discretizations. *Computational and Applied Mathematics* **247**, 124151 (2013)
20. Willms, A.R.: Parameter range reduction for ode models using cumulative backward differentiation formulas. *Journal of Computational and Applied Mathematics* **203**, 87–102 (2007)
21. Wilson, D.G.: Piecewise linear approximations of fewest line segments. In: *AFIPS Conference Proc. Vol. 40, the 1972 Spring Joint Computer Conference*, pp. 187–198. Montvale, NJ (1972)
22. Wu, H., Zhu, H., Miao, H., Perelson, A.: Parameter identifiability and estimation of hiv/aids dynamic models. *Bulletin of Mathematical Biology* **70**, 785–799 (2008)