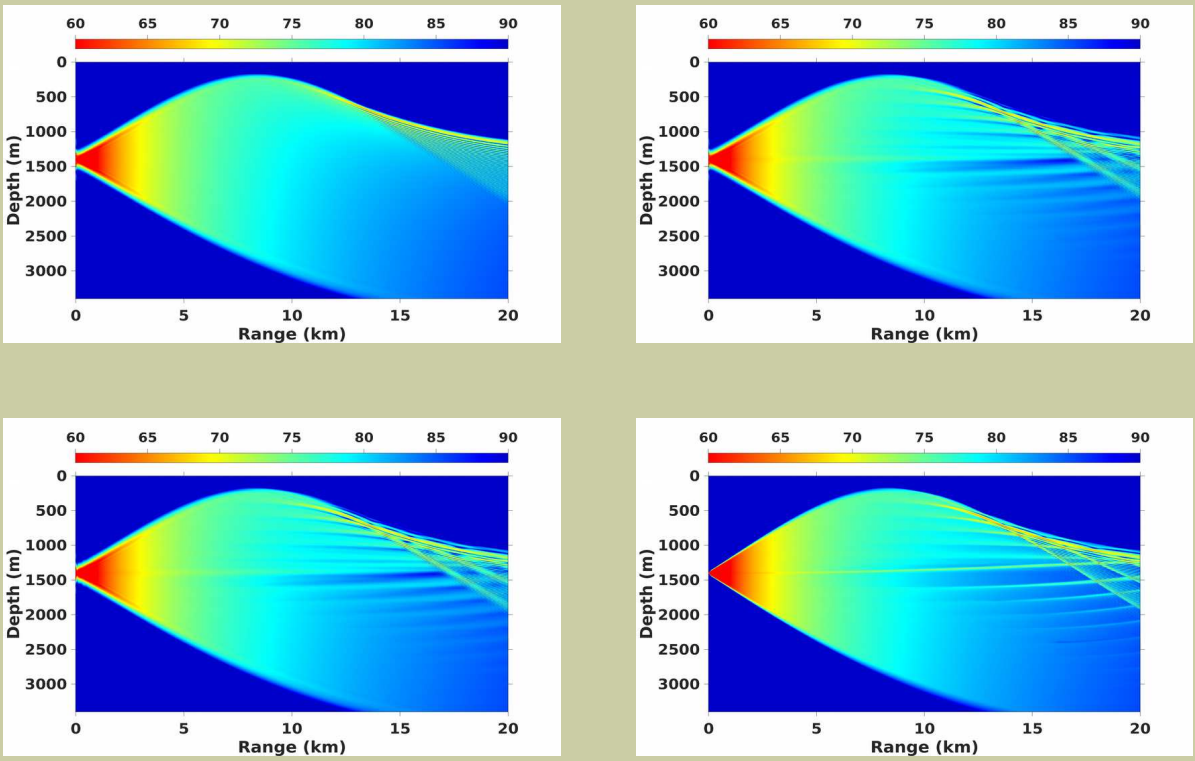
HLS REPORT
Report No. HLS-2021-01

HEAT, LIGHT, AND SOUND RESEARCH, INC.

A Robust Interpolant for use in Acoustic Propagation Modeling



John C. Peterson, Michael B. Porter

October 2021

www.hlsresearch.com

A robust interpolant for use in acoustic propagation modeling

John C. Peterson* and Michael B. Porter†

Heat, Light and Sound Research, Inc.,

12625 High Bluff Drive, Suite 211, San Diego, CA 92130-2054

(Dated: October 6, 2021)

Abstract

This article introduces an interpolation algorithm that is a hybrid of the cubic spline and the monotone piecewise cubic Hermite interpolating polynomial. The interpolant has several properties that make it an attractive choice for the representation of sound speed profiles in acoustic propagation models. It has been shown that when interpolants do not have a continuous first and/or second derivative, the associated acoustic model predictions can exhibit a variety of artifacts such as false or omitted caustics. The first derivative of our interpolant is continuous everywhere. Our interpolant also automatically respects the monotonicity of the underlying data in the sense that it will be monotone (increasing or decreasing) on any interval where the data is monotone. This property avoids the oscillations between data points that can be a source of frustration when using interpolants such as cubic splines. Finally, the interpolant will also possess a continuous second derivative everywhere, except at those data points where monotonicity considerations take priority. In many cases, the interpolant will coincide exactly with the cubic spline.

*Electronic address: jcp@hlsresearch.com

†Electronic address: porter@hlsresearch.com

I. INTRODUCTION

One issue encountered in both analytical and numerical acoustic propagation modeling is how to represent the sound speed profile. In most practical situations, the values of the profile are known only at a number of discrete depths as a result of experimental measurements. However, all methods for solving propagation problems require the knowledge of the sound speed at *arbitrary* depths within the water column. In the case of ray-based methods, the first and second derivatives of speed with respect to depth are also required at arbitrary depths. This issue has historically been addressed using interpolation. A function is produced that is defined for any depth within the water column, and its sound speed values agree exactly with the measured speeds at the depths associated with the data points. The interpolant that is presented in this article was specifically designed with this application in mind.

While the mitigation of measurement noise or unwanted fine scale structure from sound speed profiles is an important problem, it should be noted that interpolation algorithms in general, (including ours), are not designed to directly address this issue. As noted above, interpolants by definition produce values that agree exactly with the measurements at the data points. In those situations where some form of noise mitigation is deemed necessary, it must be addressed first, using an approach designed for that purpose. Some commonly used algorithms for this purpose include decimation by simple averaging, approximation theory, (such as least squares polynomial regression), or maximum likelihood estimators when the error statistics can be characterized.

In the early days of acoustic propagation modeling, piecewise linear interpolation, or connecting the data points with straight line segments, was widely used. Its popularity was largely driven by the fact that it greatly simplified the mechanics of analytical solutions of the ray equations. While it is a convenient interpolant in many respects, it was later demonstrated that any interpolant that fails to have a continuous first and/or second derivative, (which is true of piecewise linear interpolation), will introduce artifacts into the associated propagation predictions. A discussion of these issues will be presented in section II.

The cubic spline algorithm was suggested as a solution to this problem and has since seen adoption in a number of propagation models. It always has a continuous first and second derivative, so it does avoid the modeling artifacts described above. The speed gradients at the data points that produce such a piecewise polynomial interpolant, (which are unique),

can sometimes introduce oscillations or "wiggles" in both the sound speed and speed gradient between the data points. An interpolant that introduces such oscillations on a depth scale that is finer than that of the measurements is probably not the *intended* profile, (although this clearly depends on the depth resolution of the measurements). In some cases, the oscillations can be large enough so as to introduce new local maxima or minima between the data points. Many solutions to the presence of these unwanted oscillations have been presented in the literature. These solutions ranged from general guidelines for mitigation of the phenomena by the introduction of additional data points to act as constraints, to entirely new interpolants such as the "splines under tension" algorithm.

Another interpolant that was designed to specifically address the issue of unwanted oscillations between the data points is the monotone piecewise cubic Hermite interpolating polynomial, or monotone PCHIP. It *automatically* respects the monotonicity of the underlying data. On any interval where the data is monotone (increasing or decreasing), the interpolant will also be monotone. The depths associated with a local maxima or minima of the interpolant corresponds exactly to those in the underlying data. Our new interpolant borrows heavily from monotone PCHIP, so we present a brief summary of that algorithm in section III.

The monotone PCHIP interpolant has a continuous first derivative everywhere, but the second derivative is generally not continuous at the data points. Our algorithm starts with the speed gradients at the data points from the cubic spline. It then adjusts those values at the data points where the conditions for monotonicity are violated. It has a continuous second derivative everywhere, except at those data points where monotonicity considerations take priority. It is more robust than the cubic spline as it always respects the monotonicity of the data, and largely avoids the problem of unwanted oscillations between the data points.

Because our interpolant *tries* to be a cubic spline everywhere, but does not always succeed, we have named it the monotone PCHIP ACS (almost a cubic spline) algorithm. The implementation details of our algorithm will be presented in section IV, and some examples and guidelines for using it in practice are presented in section V.

II. HISTORY OF INTERPOLATION METHODS FOR PROPAGATION MODELING

As noted in the introduction, the representation of the sound speed profile in propagation modeling algorithms has historically used some form of interpolation, usually piecewise polynomials. Consider a set of n sound speed measurements, namely a set of distinct, ordered depths $\{z_1 < z_2 < \cdots < z_n\}$ and corresponding sound speeds $\{c_1,\ c_2, \cdots,\ c_n\}$. The general form of a piecewise polynomial interpolant is given by

$$p(z) = \sum_{i=1}^{n-1} p_i(z), \tag{1}$$

where each polynomial $p_i(z)$ vanishes outside of the depth interval $[z_i, z_{i+1}]$ that it is associated with.

Much of the early work in propagation modeling was developed from an analytical perspective. There was a clear motivation to use a representation for the sound speed profile that would simplify the mechanics of analytical solutions of the ray equations. An obvious choice was to represent the underlying profile with layered media where each layer had a constant sound speed gradient or index of refraction gradient. From a mathematical perspective, this corresponds to piecewise linear interpolation, where the piecewise polynomials $p_i(z)$ are given by

$$p_i(z) = c_i + \Delta_i(z - z_i), \quad \text{where} \quad \Delta_i = \frac{c_{i+1} - c_i}{z_{i+1} - z_i}, \tag{2}$$

The primary advantage of piecewise linear interpolation is its relative simplicity, for both analytical and numerical approaches to propagation modeling. The behavior of the interpolant between data points is predictable, and does not require any special skills to envision.

However, there are a number of disadvantages of piecewise linear interpolation that became apparent over time. It often does not accurately represent the intended sound speed gradient between the data points, as it is constant valued over each data interval. We want an interpolant that faithfully represents both the sound speeds *and* the sound speed gradients, as the later is associated with refraction. The discrepancies in the speed gradient suggest that the profile produced by the piecewise linear interpolant is probably not the *intended* one.

This ramifications of this were described in detail by Pedersen[1], where rigorous mathematical arguments showed that ray-based methods can produce several anomalies in the predictions when piecewise linear interpolation is used. These include the introduction of false caustics, as well as missing caustics, (the omission of physically real caustics). It was also shown that when a ray becomes horizontal, (such as rays at turning points in a deep water sound channel), and is coincident with a data point, the predicted intensity approaches zero, and a discontinuity is introduced. Furthermore, it was shown that these artifacts are not associated with just piecewise linear interpolation, but rather with any representation where the first derivative with respect to depth exhibits discontinuities. This important result provided the motivation to the community to identify better representations that are still easy to work with. A piecewise quadratic interpolant with continuous first derivatives and associated analytical solutions of the ray equations was presented by Stewart[2].

Another propagation artifact was identified by Pedersen and Gordon[3], where it was shown that ray-based methods will produce discontinuities in the gradient of the transmission loss whenever the second derivative with respect to depth has discontinuities. Moreover, it was argued that there is not much to be gained by using representations with a continuous third or higher order derivatives. A number of representations with a continuous second derivative were already in use at that time, such as the Epstein profile (where all of its derivatives are continuous), which was used in the context of approximating, rather than interpolating the profile.

It should be noted that the artifacts mentioned above are *not* strictly confined to ray-based methods of propagation modeling. While we are not aware of any rigorous arguments to that effect in the literature, one can readily confirm this assertion by numerical experiments. This should not come as a surprise because different interpolants inherently correspond to different profiles.

To demonstrate the above assertion, we chose the notional Munk profile for our numerical experiment. Because the profile is given by an analytic expression that can be computed at any arbitrary depth, we can utilize that expression directly in any given propagation model. We can then compare those predictions with ones generated using an interpolant of tabulated values of the same profile at discrete depths. We also know the exact speed gradients of the profile, an advantage one does not have when working with measurements.

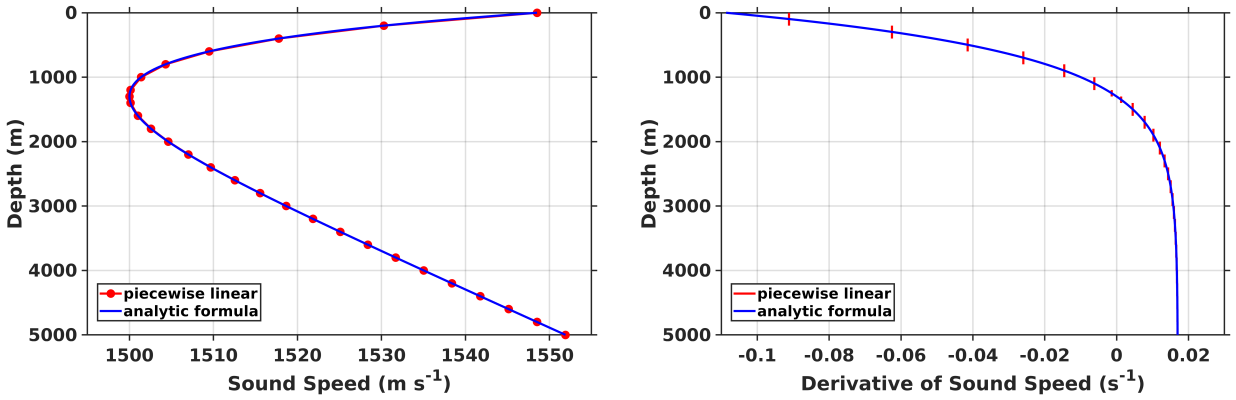
The Munk profile is given by the expression

$$c(z) = 1500\Big(1 + \epsilon(\bar{z} - 1 + \mathrm{e}^{-\bar{z}})\Big), \tag{3a}$$

$$\bar{z} = \frac{2(z - 1300)}{1300}, \tag{3b}$$

$$\epsilon = 0.00737, \tag{3c}$$

We tabulated this profile at multiples of 200 meters of depth, with an additional data point at 1300 meters, corresponding to the axis of the associated sound channel. The piecewise linear interpolant of these tabulated values is shown in Fig. 1, as well as the associated speed gradient.

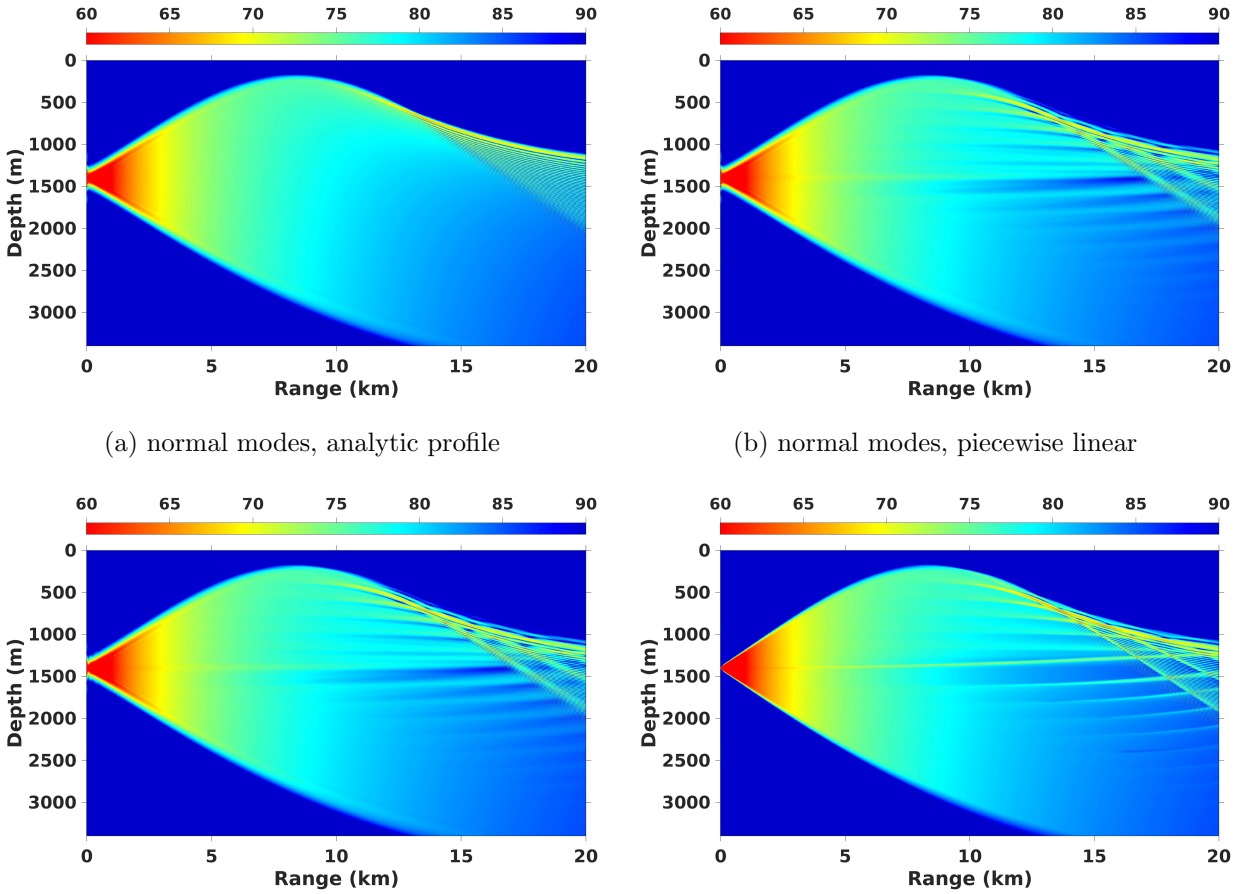


(a) sound speed

(b) sound speed gradient

FIG. 1: A comparison of the analytic Munk profile and a piecewise linear interpolant of discrete values of it. The piecewise linear interpolant is shown in red, the analytic Munk profile in blue. The sound speed is shown in panel (a), and the sound speed gradient in panel (b).

We then generated transmission loss predictions using normal modes, wavenumber integration, and ray-based methods using the piecewise linear interpolant shown in Fig. 1. Those predictions and a numerically exact prediction using normal modes, and the analytic formula for the Munk profile are shown in Fig. 2. An inspection of these results clearly shows that other propagation modeling approaches exhibit artifacts that are qualitatively similar to those predicted and observed in ray-based methods. An example that demonstrates the artifacts associated with interpolants that have a continuous first derivative, but a second derivative that is not continuous will be shown later in section V.



(a) normal modes, analytic profile

(b) normal modes, piecewise linear

(c) wavenumber integration, piecewise linear

(d) ray methods, piecewise linear

FIG. 2: Examples of artifacts as observed in transmission loss predictions using different modeling approaches for the Munk profile and a directed 500 Hz source located 100 meters below the sound channel axis at a depth of 1400 meters.

As the existence of artifacts became more widely known in the propagation modeling community, more suitable interpolants were identified. Akima polynomials [4], were subsequently used by some in the community. They have the advantage of being easy to compute, and the first derivative is always continuous. However, they are not an ideal solution as they do not have a continuous second derivative at the data points.

Cubic splines were first suggested as an interpolant for numerically oriented propagation modeling by Moler and Soloman[5], and for analytical solutions of the ray equations by Mezzino[6]. Cubic splines have continuous first and second derivatives, so they avoid the modeling artifacts described above. They became quite popular in a wide range of other application domains. As a result, user-friendly software for constructing cubic splines became

readily available.

While cubic splines might appear to be the ideal interpolant for propagation modeling, they sometimes suffer from the problem of unexpected oscillations or "wiggles" between the data points. While additional measurements would be needed to speak precisely about how an interpolant should behave between existing data points, the mean value theorem can provide some guidance. It tells us that between any two data points, the speed gradient will be exactly equal to the average speed gradient (the slope of the secant line connecting the data points), at *some* depth in the interval. An interpolant that produces speed gradients that differ substantially from the average is quite likely not representing the *intended* profile. These oscillations can sometimes be large enough to introduce new maxima or minima between the data points as we will demonstrate later. This issue was acknowledged, and some general guidelines for dealing with it were presented in the appendix of Mezzino[6]. We agree with the authors that these issues can almost always be mitigated with suitable manual intervention. However, the process can be frustrating for users that have no previous experience with this issue. Even for experienced users, the process can become rather time consuming for profiles with many data points. Although Akima splines fail to have a continuous second derivative, they remained popular because in general, they suffered from unwanted oscillations less often, and with less severity.

The issue of unwanted oscillations between the data points has more serious ramifications in other application domains, (e.g. sheet metal forming). This motivated the development of other interpolants specifically designed to address this issue, such as "splines under tension" as described by Cline[7]. The idea behind the interpolant comes from a mechanical analogue. Consider an elastic cord that has been threaded through eyelets located at the data points, and then stretched by pulling on each of the free ends. This interpolant has a continuous first and second derivative and has been used in propagation modeling codes. Its one disadvantage is that it has a free parameter, the tension factor, which corresponds to how hard the elastic cord is pulled in the mechanical analogue. Arriving at acceptable results generally involves experimenting with different values of the tension factor.

Another higher order interpolant that was specifically designed to avoid the oscillations of the cubic spline is the monotone piecewise cubic Hermite interpolating polynomial. Our interpolant borrows heavily from it, so we present an overview of the algorithm in the following section. Those readers who are not interested in the specific details of our algorithm can skip

to section V where we present some examples and identify some practical considerations.

III. THE MONOTONE PIECEWISE CUBIC HERMITE INTERPOLATING POLYNOMIAL

An interpolant that avoids some of the frustration associated with cubic splines is the monotone piecewise cubic Hermite interpolating polynomial, or monotone PCHIP. It *automatically* respects the monotonicity of the underlying data: on any interval where the data is monotone (increasing or decreasing), the interpolant will also be monotone. The monotonicity property prevents the introduction of new local maxima or minima between the data points, which can happen with the cubic spline.

It is constructed using the cubic Hermite polynomial basis, (not be confused with the infinite set of Hermite orthogonal polynomials), as they have several convenient properties. On the notional interval $[0, 1]$, the standard cubic Hermite polynomials are defined by

$$H_1(z) = (1 + 2z)(1 - z)^2, \tag{4a}$$

$$H_2(z) = z(1 - z)^2, \tag{4b}$$

$$H_3(z) = z^2(3 - 2z), \tag{4c}$$

$$H_4(z) = z^2(z - 1), \tag{4d}$$

Each of the polynomials and their first derivatives take on unit values or they vanish at the endpoints of the interval. The simplification that this provides will become clear in the discussion that follows. Here is a summary of the values of the cubic Hermite polynomials and their first derivatives at the endpoints.

$$H_1(0) = 1,\ H_1(1) = 0, \quad H_1'(0) = 0,\ H_1'(1) = 0, \tag{5a}$$

$$H_2(0) = 0,\ H_2(1) = 0, \quad H_2'(0) = 1,\ H_2'(1) = 0, \tag{5b}$$

$$H_3(0) = 0,\ H_3(1) = 1, \quad H_3'(0) = 0,\ H_3'(1) = 0, \tag{5c}$$

$$H_4(0) = 0,\ H_4(1) = 0, \quad H_4'(0) = 0,\ H_4'(1) = 1, \tag{5d}$$

The standard form of the cubic Hermite polynomials can be readily extended to any arbitrary interval $[z_i, z_{i+1}]$ by a change of variables. The transformed polynomials are given by the

expressions

$$\hat{H}_1(z) = \quad H_1\big(\,(z - z_i)/h\,\big), \tag{6a}$$

$$\hat{H}_2(z) = h\; H_2\big(\,(z - z_i)/h\,\big), \tag{6b}$$

$$\hat{H}_3(z) = \quad H_3\big(\,(z - z_i)/h\,\big), \tag{6c}$$

$$\hat{H}_4(z) = h\; H_4\big(\,(z - z_i)/h\,\big), \tag{6d}$$

where $h = z_{i+1} - z_i$ corresponds to the width of the given interval. The second and fourth polynomials need to be scaled by h so that their first derivatives take on the desired unit values at the left and right endpoints respectively.

The convenience now becomes apparent when we represent the piecewise polynomial $p_i(x)$ from Eq. 1 as a linear combination of the transformed cubic Hermite polynomials, or more specifically

$$p_i(x) = a_1\hat{H}_1(x) + a_2\hat{H}_2(x) + a_3\hat{H}_3(x) + a_4\hat{H}_4(x), \tag{7}$$

It is easy to verify that this is an interpolant of $c(z)$, and its derivative is an interpolant of $c'(z)$ on the given interval $[z_i, z_{i+1}]$ if we use the following specific values for the coefficients.

$$a_1 = c_i, \qquad a_2 = c'_i, \qquad a_3 = c_{i+1}, \qquad a_4 = c'_{i+1}, \tag{8}$$

The obvious practical difficulty with this is that experimental measurements can readily provide values of the sound speed, but *not* the derivative with respect to depth. (This is a difficulty that is shared with many other application domains where monotone PCHIP is popular). We will return to this issue shortly, after first addressing monotonicity.

For the piecewise polynomial $p_i(z)$ associated with the interval $[z_i, z_{i+1}]$ as given by Eqs. 7 and 8, consider the following definitions

$$\alpha_i = \frac{c'_i}{\Delta_i}, \qquad \beta_i = \frac{c'_{i+1}}{\Delta_i}, \tag{9}$$

where Δ_i represents the slope of the secant line connecting the endpoints as defined before in Eq. 2. In de Boor and Swartz[8], it was shown that if $0 \le \alpha_i \le 3$ and $0 \le \beta_i \le 3$ are satisfied for each interval, the interpolant will respect monotonicity. This criteria, sometimes referred to as the de Boor-Swartz criteria, was independently extended by Ferguson and Miller[9] and Fritsch and Carlson[10]. Their work showed that a sufficient condition for monotonicity is that the points (α_i, β_i) be contained within the shaded region shown in Fig. 3, which is

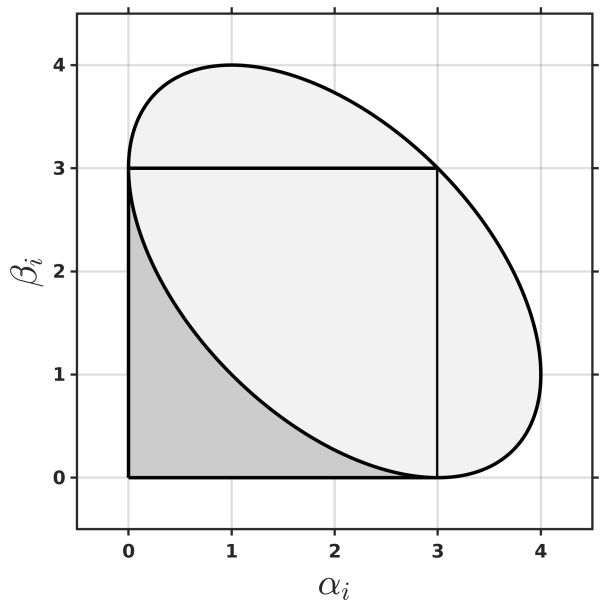


FIG. 3: A necessary and sufficient condition for the monotonicity of the interpolant is that for each interval i, the points (α_i, β_i) must be contained in the union of the two shaded regions shown.

the union of the de Boor-Swartz rectangle, and an ellipse. It was also shown that this is both a necessary and sufficient condition for monotonicity of the interpolant.

Despite the complicated shape of the monotonicity region shown in Fig. 3, several useful conclusions can be readily made.

- Neither α_i or β_i can have negative values. This is equivalent to saying that the sign of the derivative of the interpolant at any data point can never differ from the sign of the slopes of the secant lines on either side of it.

- A simple corollary to this observation is that whenever the slopes of the secant lines change sign (at local extrema in the data), or one or both vanish, the only value for the derivative of the interpolant at the intervening point that preserves monotonicity is zero.

- The derivatives that preserve monotonicity are *not* unique, which has given rise to several versions of monotone PCHIP.

A number of variants of the monotone PCHIP interpolant were developed over the years. The earliest version was developed by Butland[11], which set the derivative c_i' equal to the

harmonic mean, (not to be confused with the arithmetic or geometric means), of the adjacent secant slopes Δ_{i-1} and Δ_i whenever they have the same arithmetic sign, and zero otherwise. It was shown that the α_i and β_i associated with this choice for the derivative would always satisfy the monotonicity requirements without any further action.

Subsequent versions of monotone PCHIP followed this same general approach: compute the derivative c_i' using some function of the adjacent secant slopes that was cleverly constructed so that the monotonicity constraints are satisfied without any additional effort. Emphasis was placed on functions that produced interpolants that were visually pleasing in some sense, such as those documented by Fritsch and Butland[12, 13].

The de Boor-Swartz rectangle was often used as a sufficient condition for constructing monotone interpolants because of some of the simplifications it introduces. Determining if the point (α_i, β_i) is contained within the full monotonicity region described above requires the knowledge of both α_i and β_i due to the irregular, elliptical boundary of the region.

If the de Boor-Swartz rectangle is used instead of the full monotonicity region, we can address each data point in isolation, without any knowledge of the nearby data points. At interior data point i, we only need verify that β_{i-1}, associated with the segment to the left, satisfies $0 \leq \beta_{i-1} \leq 3$, and that α_i associated with the segment to the right satisfies $0 \leq \alpha_i \leq 3$. We use the de Boor-Swartz rectangle as a sufficient condition for monotonicity in our variant of monotone PCHIP to take advantage of this simplification.

While the existing implementations of monotone PCHIP are very popular in other application domains, they are not an ideal interpolant for use in propagation modeling. The existing algorithms have a continuous first derivative everywhere. However, the second derivative is generally *not* continuous at the data points. An example of the propagation modeling artifacts associated with the existing monotone PCHIP algorithms will be shown later in section V.

IV. A SMOOTHER VARIANT OF MONOTONE PCHIP

In this section, we present our variant of monotone PCHIP, one that is more suitable for use in propagation modeling. As noted above, the main deficiency of the existing variants, is that they do not have a continuous second derivative.

The essence of our idea is to create a new variant of monotone PCHIP that has a contin-

uous second derivative everywhere, with the possible exceptions of those data points where that goal conflicts with the conditions for monotonicity. The resulting interpolant will be more robust than the cubic spline in the sense that it will always respect the monotonicity of the underlying data. The defect of the resulting algorithm is that the second derivative can fail to be continuous at some of the data points.

The changes relative to the existing variants are straight forward. First, compute estimates of the derivatives of the sound speed at the data points using the cubic spline algorithm. Then adjust them accordingly at the data points where they fail to satisfy the monotonicity requirements. The adjustments used to enforce monotonicity at the interior points will now be described.

As noted earlier, we elected to use the de Boor-Swartz rectangle as a sufficient condition for monotonicity. This allows us to address monotonicity at each data point in isolation. At each interior data point i, we perform these checks and possible adjustments to the sound speed derivative c'_i computed using the cubic spline algorithm.

- If the slopes of the adjacent secant lines do not have the same sign, or more specifically if $\Delta_{i-1}\Delta_i \leq 0$, then reset the derivative c'_i to zero, and continue to the next data point.

- Compute the value of β_{i-1} using the current value of the derivative. If the inequality $\beta_{i-1} \leq 3$ is not satisfied, replace the derivative with $c'_i = 3\ \Delta_{i-1}$.

- Compute the value of α_i using the current value of the derivative. If the inequality $\alpha_i \leq 3$ is not satisfied, replace the derivative with $c'_i = 3\ \Delta_i$.

There are number of other lower level details of the algorithm that we will now discuss. Those readers that are not interested in implementing the algorithm themselves can safely skip to section V where will present some example applications.

- It should be noted that there is more than one version of the cubic spline algorithm. They differ primarily in what additional constraints are used to arrive at a set of coupled equations for the unknown derivatives that has a unique solution. We will mention the three most popular versions, but will not present any implementation details in the interests of brevity. Interested readers are referred to de Boor[14] for additional detail.

"Natural" cubic splines impose the additional constraint that the second derivative of the interpolant vanishes at the endpoints. Such splines degenerate into a straight line segment in the limit as one moves toward either endpoint. The "not-a-knot" cubic spline imposes the additional constraint that the third derivative of the interpolant is continuous at the points immediately *adjacent* to the endpoints. The "clamped" cubic spline uses values of the first derivative at the endpoints that are specified by the user.

We elected to use "clamped" cubic splines in our algorithm for two reasons. It generally produced slightly better results than the other two in our experiments, and the existing versions of monotone PCHIP use an estimate of the first derivative at the endpoints that history has shown to work well.

- To produce estimates of c'_i at the first and last data point, we used the non-centered three-point difference formula as described by Fritsch and Butland[12] which we summarize here for completeness.

  At the left endpoint, let $h_1 = (z_2 - z_1)$ and $h_2 = (z_3 - z_2)$, then compute the derivative using the expression

  $$c'_1 = \frac{(2h_1 + h_2)\Delta_1 - h_1\Delta_2}{h_1 + h_2}, \tag{10}$$

  The monotonicity requirements are addressed here, before the derivative is presented to the "clamped" cubic spline algorithm. The value of c'_1 computed in Eq. 10 automatically satisfies some of the monotonicity requirements. In the case where $\Delta_1\Delta_2 \geq 0$, it can be shown that it already satisfies the requirement that $\alpha_1 \leq 3$. However, if the sign of the computed derivative differs from that of Δ_1, it should be set to zero. In the case where $\Delta_1\Delta_2 < 0$, the requirement that $\alpha_i \leq 3$ must be checked, and adjustments must be made if it is not satisfied.

  Similarly, for the case of the right endpoint, let $h_{n-2} = (z_{n-1} - z_{n-2})$ and $h_{n-1} = (z_n - z_{n-1})$, and compute the derivative using the expression

  $$c'_n = \frac{-h_{n-1}\Delta_{n-2} + (h_{n-2} + 2h_{n-1})\Delta_{n-1}}{h_{n-2} + h_{n-1}}, \tag{11}$$

  The monotonicity requirements are addressed here in a fashion similar to that used for the left endpoint above. The value of c'_n computed in Eq. 11 automatically satisfies some of the monotonicity requirements. In the case where $\Delta_{n-2}\Delta_{n-1} \geq 0$, it can be shown that it already satisfies the requirement that $\beta_{n-1} \leq 3$. However, if the sign

of the computed derivative differs from that of Δ_{n-1}, it should be set to zero. In the case where $\Delta_{n-2}\Delta_{n-1} < 0$, the requirement that $\beta_{n-1} \leq 3$ must be checked, and adjustments must be made if it is not satisfied.

- While it is possible to directly evaluate equations 7, 8, a more efficient approach to evaluating $p_i(z)$ on the interval $[z_i, z_{i+1}]$ is now described. If we denote the width of the interval by $h_i = z_{i+1} - z_i$, and the distance into the interval by $\Delta z = z - z_i$, it can be shown that the value of the interpolant is given by the standard polynomial $a_0 + a_1\Delta z + a_2\Delta z^2 + a_3\Delta z^3$ with the following coefficients.

$$a_0 = c_i, \tag{12a}$$

$$a_1 = c'_i, \tag{12b}$$

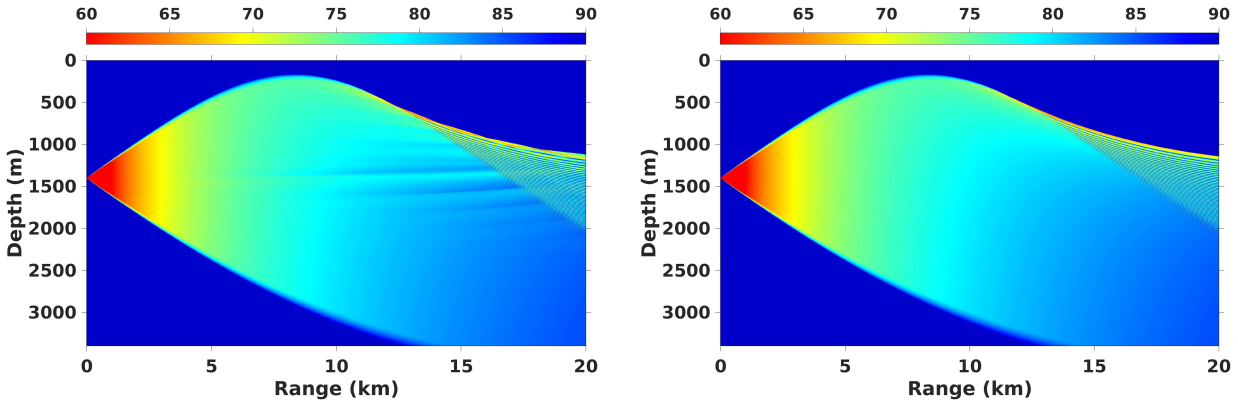
$$a_2 = \frac{3\,(c_{i+1} - c_i)}{{h_i}^2} - \frac{(2c'_i + c'_{i+1})}{h_i}, \tag{12c}$$

$$a_3 = \frac{(c'_i + c'_{i+1})}{{h_i}^2} - \frac{2\,(c_{i+1} - c_i)}{{h_i}^3}, \tag{12d}$$

V. EXAMPLES AND PRACTICAL CONSIDERATIONS

In this section we present some examples that demonstrate the performance of our new algorithm, as well as some guidelines for using it in practical situations.

Our first example was chosen with two goals in mind. The first is to present an example of the modeling artifacts described by Pedersen and Gordon[3] that are associated with interpolants that fail to have a continuous second derivative. Its other purpose is to demonstrate that in many cases, our PCHIP ACS interpolant will be identical to the cubic spline, and consequently have continuous first and second derivatives everywhere. We applied an existing variant of monotone PCHIP, (the one provided in Matlab to be specific), and our PCHIP ACS algorithm to the same discrete tabulated values of the Munk profile as shown in Fig. 1. The transmission loss predictions found using ray-based methods for the two interpolants are shown in Fig. 4. The afore mentioned artifacts can be readily seen in panel (a) that shows the prediction for the existing variant of monotone PCHIP. The discrete values of the Munk profile are sufficiently "well behaved" that the PCHIP ACS interpolant of them is identical to the cubic spline. The results shown in panel (b) of the figure are visually identical to the numerically exact results shown before in panel (a) of Fig. 2.



(a) existing monotone PCHIP

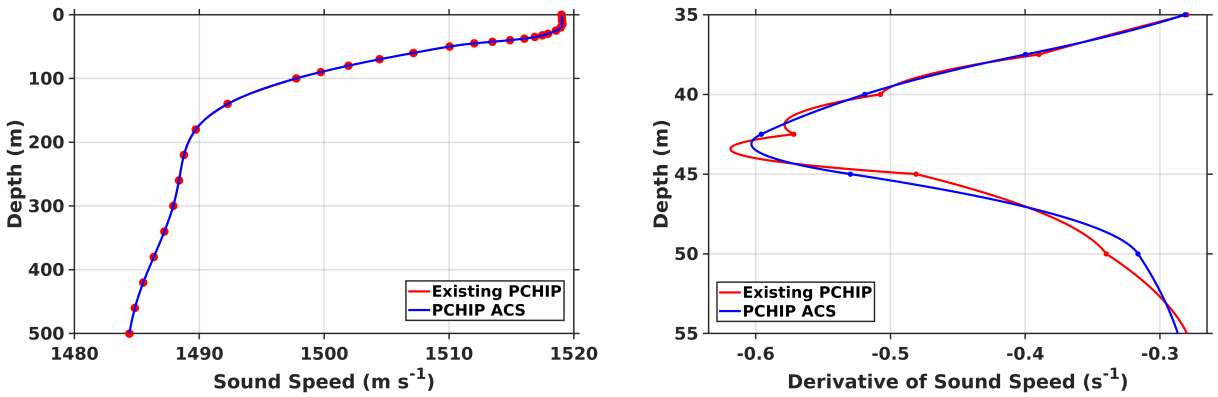
(b) monotone PCHIP ACS

FIG. 4: Transmission loss predictions using ray tracing and interpolants of discrete values of the Munk profile. The 500 Hz directed source is located 100 meters below the channel axis at a depth of 1400 meters. The predictions using the version of monotone PCHIP algorithm from Matlab are shown in (a) and for our monotone PCHIP ACS algorithm in (b).

To further demonstrate the additional smoothness offered by our interpolant, we applied an existing version of monotone PCHIP and PCHIP ACS to a profile measured off the coast of San Diego, California. The interpolated sound speeds are shown in panel (a) of Fig. 5, and are difficult to distinguish visually. The differences between the two interpolants become much more obvious when the associated speed gradients are examined, which are shown in panel (b) of the same figure. The existing version of monotone PCHIP exhibits "kinks" in the first derivative at some of the data points, which correspond to discontinuities in the second derivative, while our interpolant does not.

We had two goals in mind when we selected our last example. First, we wanted to present an example of the unwanted oscillations between the data points of the cubic spline, primarily for the benefit of readers who might not be familiar with the issue. Second, we wanted to demonstrate that PCHIP ACS generally behaves much better in such situations.

For our last example, we chose to apply several different interpolants to a notional profile with a surface duct that extends to a depth of 100 meters. One advantage of this choice, (with respect to the points that we wish to make), is that in the duct portion of the profile, we know precisely how the interpolants should behave between the data points to faithfully



(a) interpolants of sound speed

(b) interpolants of sound speed gradient

FIG. 5: A comparison of an existing version of monotone PCHIP (red line) and PCHIP ACS (blue) as applied to a measured profile. The interpolants of the sound speeds are shown in (a), and the associated speed gradients in (b).

represent the *intended* profile. Surface ducts form as a result of thorough mixing of the upper water column, producing nearly uniform temperature and salinity. In the duct, the pressure will vary linearly with depth, producing a profile where the sound speed also varies linearly with depth, with a constant speed gradient.

To further put this example into perspective, we state here what is probably the best bit of advice for using *any* interpolant with order higher than linear. That is to avoid sampling the profile in a manner where the spacing of the data points in depth departs *too much* from uniform spacing. We elected to violate that bit of advice in this example, to demonstrate just how serious the problem of unwanted oscillations can become. We will provide more detailed guidelines for using PCHIP ACS in practice later in this section.

We sampled our notional profile in a very non-uniform fashion. The duct portion has just two samples, one at the surface, and one at the bottom of the duct at 100 meters, while the remainder of the profile has much finer sampling. The piecewise linear, cubic spline and monotone PCHIP ACS interpolants of these tabulated values are shown in Fig. 6.

An inspection of Fig. 6 shows that the piecewise linear interpolant represents the duct portion of the profile exactly, in terms of sound speeds and speed gradients. However, the cubic spline fails to faithfully represent the duct portion of the profile in a rather spectacular fashion. The duct associated with it is only about 50 meters deep and exhibits speed gradients that are wildly in error. The transmission loss predictions for a source located at a

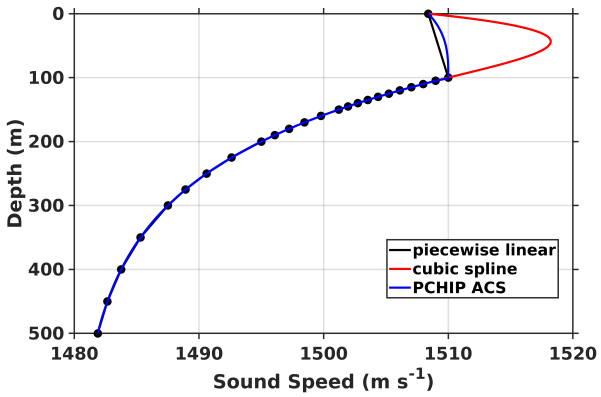
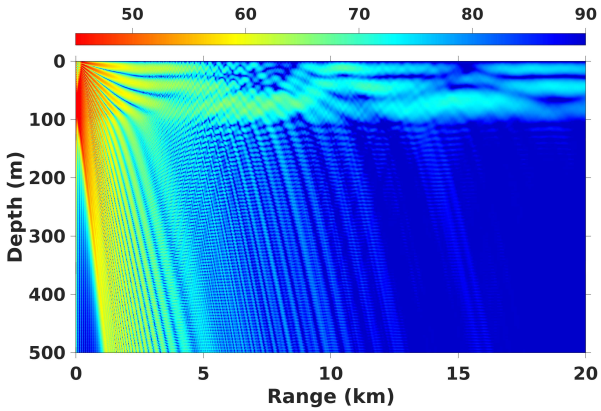


FIG. 6: A comparison of different interpolation algorithms applied to a notional surface duct profile. Piecewise linear interpolation (black), the cubic spline (red), and our monotone PCHIP ACS algorithm (blue).

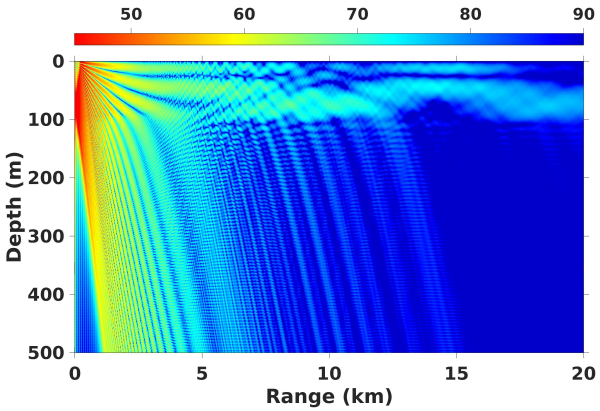
depth of 75 meters using wavenumber integration and these different interpolation algorithms are shown in Fig. 7. For the cubic spline interpolant, the expected propagation of sound in the duct is completely lost. Our PCHIP ACS also exhibits an unwanted oscillation in the duct, but it is not nearly as severe. It is worth pointing out that the existing versions of monotone PCHIP produces sound speeds that are qualitatively similar to those shown for PCHIP ACS in this case.

As noted above, we know precisely how the interpolants should behave in the surface duct between the data points. Using only two data points to define the duct demonstrates how the interpolant can misbehave when we have discarded *too much* meaningful information and give the interpolant too much "freedom" to misbehave. In panel (a) of Fig. 8, it can be seen that the introduction of three new data points in the duct produces good results for monotone PCHIP ACS. The results for uniform sampling in the duct, using the spacing that matches that used below the duct are shown in panel (b) of the same figure. All of the interpolants perform adequately in that case, although the cubic spline still exhibits some small oscillations.

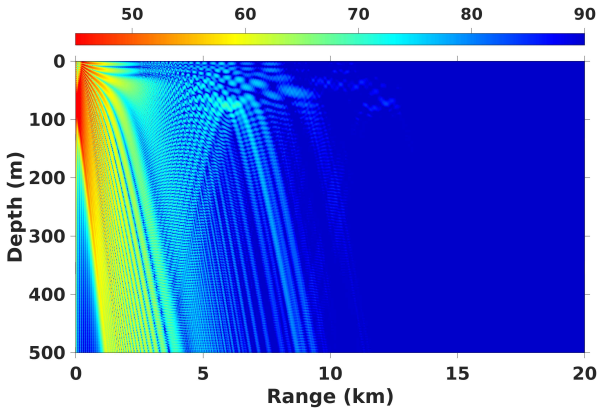
One final comment is in order regarding our surface duct example. As noted above in section IV, there are several versions of the cubic spline interpolant. They differ in what additional constraints are imposed above and beyond the basic constraint that the second derivatives must be continuous at the data points. The cubic spline shown in Fig. 6 is the "natural" variant that imposes the constraint that the second derivative of the interpolant



(a) piecewise linear



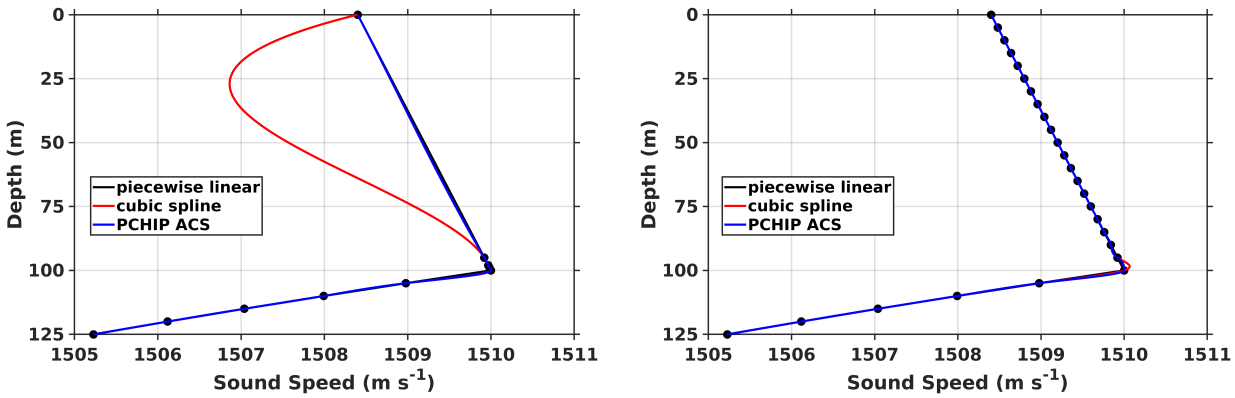
(b) monotone PCHIP ACS



(c) cubic spline

FIG. 7: A comparison of transmission loss predictions using wavenumber integration and different interpolation algorithms applied to the notional surface duct profile shown in Fig. 6. The 750 Hz omnidirectional source is located in the duct at a depth of 75 meters.

vanishes at the endpoints. An inspection of the slope of the cubic spline at the surface, (the first data point), reveals that the associated speed gradient there is spectacularly erroneous. This should not be surprising, as it is not otherwise constrained in that variant. This is an example of precisely the sort of behavior that motivated us to use the "clamped" variant of the cubic spline, together with a non-centered three-point difference formula in monotone PCHIP ACS. In this example, one could also make a very good case for directly providing a value of the speed gradient at the first data point rather than using the value computed from the difference formula. However, this is often not possible for sound speed profiles in general.



(a) (b)

FIG. 8: Examples of improved sampling to address the existence of unwanted oscillations. Piecewise linear interpolation (black), the cubic spline (red), and our monotone PCHIP ACS (blue). The addition of three new data points (a), and the addition of new data points with uniform spacing of five meters (b).

In the remainder of this section, we present some guidance for using our monotone PCHIP ACS algorithm in practical situations, and should prove to be helpful when using any interpolant with order higher than linear. As noted above, our best advice is to avoid sampling the profile in a manner that departs *too much* from uniform spacing of the data points in depth. Whenever there are (relatively) large gaps in depth between the samples, the interpolant will be "free" to potentially do something that the user does not expect.

As we noted in the introduction, piecewise linear interpolation has the advantage that it requires no special skills to visualize how it will behave between the data points. This is decidedly not the case with higher order interpolants, as they can sometimes exhibit behavior that comes as a surprise to even the most skilled users. The second item on our list of suggestions is to *always* construct a high resolution plot, (with samples between the data points), of higher order interpolants before using them to compute propagation predictions. If you are concerned about propagation artifacts, it is also a good idea to construct a high resolution plot of the speed gradient associated with the interpolant such as was shown in panel (b) of Fig. 5.

The problem of unwanted oscillations between the data points will often, but not always, occur near a local extrema in the data itself, where the slope of the adjacent secant lines have different arithmetic signs. The surface duct and Beaufort lens are examples of profiles

that exhibit local extrema that are physically real. If your tabulated profile uses nearly uniform spacing, but you are still experiencing difficulties, a quick solution is to add a few new data points in a fashion similar to what was done in panel (a) of Fig. 8.

A more rigorous solution, and one that often produces an interpolant with much smaller or sometimes even no unwanted oscillations is to employ "corner cutting". The idea is analogous to the introduction of fillets in wood or metal working, namely the replacement of a sharp corner with a rounded one, (usually motivated by safety considerations). A simple variant of corner cutting was documented in the appendix of of Mezzino[6]. This topic has been researched extensively in the mathematical literature, such as in de Boor[15].

As we mentioned in the introduction, interpolation algorithms are not designed to mitigate measurement noise or unwanted fine scale structure from sound speed profiles. It is the responsibility of the user to address this issue first, using an algorithm that is consistent with the intended purpose of the associated propagation prediction. We shall however, speak to the practical considerations associated with two extremes of noise mitigation.

There are certainly situations where the micro-structure of the profile is deemed to be an important component of the associated propagation prediction. In such situations, there may be very little or perhaps even no noise mitigation. Situations like this, where there can be a very large number of data points, are a good example of where cubic splines can be frustrating and time consuming to deal with. In this case, there is no additional effort required, just apply the PCHIP ACS interpolant, and enjoy the additional robustness that the algorithm offers.

At the opposite extreme is the case where extensive noise mitigation has been used. A propagation prediction that is representative of some spatial or temporal mean profile is sometimes the desired end result. In this situation, there needs to be a compromise between meaningful averaging and interpolation considerations. Tabulated profiles that are too sparsely sampled are more likely to exhibit unwanted oscillations. When the original raw data is unavailable, corrective steps such as the addition of extra points, or corner cutting may be required when using higher order interpolants.

To summarize, it is quite possible that there are exceptional cases, but we have found that our monotone PCHIP ACS interpolant required user intervention much less often that did cubic splines. When issues did arise with our interpolant, it required much less effort to rectify the problem.

VI. CONCLUSION

The monotone PCHIP algorithm has seen popular use in a wide variety of application domains, largely because it automatically honors the monotonicity of the underlying data. This property avoids the unwanted oscillations between data points which can be a source of frustration when working with higher order interpolating polynomials. We have introduced a variant of monotone PCHIP that shares this important property. It also avoids, (to the extent possible), the artifacts of propagation predictions associated with interpolants that do not have continuous first and second derivatives, such as piecewise linear interpolation. The monotone PCHIP ACS algorithm provides those with an interest in accurate acoustic propagation modeling with another interpolant that we hope will be a welcome addition to those currently in use.

Acknowledgments

This research was supported by the United States Office of Naval Research under contract number N68335-17-C-0553.

[1] M. A. Pedersen, The Journal of the Acoustical Society of America **33**, 465 (1961), https://doi.org/10.1121/1.1908693, URL https://doi.org/10.1121/1.1908693.

[2] K. R. Stewart, The Journal of the Acoustical Society of America **38**, 339 (1965), https://doi.org/10.1121/1.1909671, URL https://doi.org/10.1121/1.1909671.

[3] M. A. Pedersen and D. F. Gordon, The Journal of the Acoustical Society of America **41**, 419 (1967), https://doi.org/10.1121/1.1910353, URL https://doi.org/10.1121/1.1910353.

[4] H. Akima, Journal of the Association for Computing Machinery **17**, 589 (1970), URL https://doi.org/10.1145/321607.321609.

[5] C. B. Moler and L. P. Solomon, The Journal of the Acoustical Society of America **48**, 739 (1970), https://doi.org/10.1121/1.1912197, URL https://doi.org/10.1121/1.1912197.

[6] M. J. Mezzino, The Journal of the Acoustical Society of America **53**, 581 (1973), https://doi.org/10.1121/1.1913361, URL https://doi.org/10.1121/1.1913361.

[7] A. K. Cline, Communications of the ACM **17**, 218 (1974), URL http://doi.acm.org/10.1145/360924.360971.

[8] C. de Boor and B. Swartz, Journal of Approximation Theory **21**, 411 (1977), URL https://doi.org/10.1016/0021-9045(77)90011-9.

[9] J. Ferguson and K. Miller, Tech. Rep. 5322-3-5, TRW Inc. (1969).

[10] F. N. Fritsch and R. Carlson, Siam Journal on Numerical Analysis **17**, 238 (1980), URL https://doi.org/10.1137/0717021.

[11] J. Butland, Proc. Computer Graphics **80**, 409 (1980).

[12] F. N. Fritsch and J. Butland, Tech. Rep. UCRL-85104, Lawrence Livermore National Laboratory (1980).

[13] F. N. Fritsch and J. Butland, SIAM Journal on Scientific and Statistical Computing **5**, 300 (1984), https://doi.org/10.1137/0905021, URL https://doi.org/10.1137/0905021.

[14] C. de Boor, *A Practical Guide to Splines* (Springer-Verlag, New York, 1978).

[15] C. De Boor, Computer Aided Geometric Design **7**, 389 (1990).