Kai Wu, J. Andrew Zhang, and Y. Jay Guo

Fast and Accurate Linear Fitting for an Incompletely Sampled Gaussian Function With a Long Tail

Fitting experiment data onto a curve is a common signal processing technique to extract data features and establish the relationship between variables. Often, we expect the curve to comply with some analytical function and then turn data fitting into estimating the unknown parameters of a function. Among analytical functions for data fitting, the Gaussian function is the most widely used one due to its extensive applications in numerous science and engineering fields. To name just a few, the Gaussian function is highly popular in statistical signal processing and analysis, thanks to the central limit theorem [1], and the Gaussian function frequently appears in the quantum harmonic oscillator, quantum field theory, optics, lasers, and many other theories and models in physics [2]; moreover, the Gaussian function is widely applied in chemistry for depicting molecular orbitals, in computer science for imaging processing, and in artificial intelligence for defining neural networks.

Fitting a Gaussian function, or, simply, Gaussian fitting, is consistently of high interest to the signal processing community [3]–[6]. Since the Gaussian function is underlain by an exponential function, it is nonlinear and not easy to be fitted directly. One effective way of counteracting its exponential nature is to apply the natural logarithm, which has been applied in transferring the Gaussian fitting into a linear fitting [4]. However, the problem of the logarithmic transformation is that it makes the noise power vary over data samples, which can result in biased Gaussian fitting. The weighted least square (WLS) fitting is known to be effective in handling uneven noise backgrounds [7]. However, as unveiled in [5], the ideal weighting for linear Gaussian fitting is directly related to the unknown Gaussian function. To this end, an iterative WLS is developed in [5], starting with using the data samples (which are noisy values of a Gaussian function) as weights and then iteratively reconstructing the weights using the previously estimated function parameters.

For the iterative WLS, the number of iterations required for a satisfactory fitting performance can be large, particularly when an incompletely sampled Gaussian function with a long tail is given [see Figure 1(a) for such a case]. Establishing a good initialization is a common strategy for improving the convergence speed and performance of an iterative algorithm. Noticing the unavailability of a proper initialization for the iterative WLS, we aim to fill the blank by developing a high-quality one in this article. To do so, we introduce a few signal processing tricks to develop high-performance initial estimators for the three parameters of a Gaussian function. When our initial fitting results are applied, not only is the efficiency of the iterative WLS substantially improved, but its accuracy is also greatly enhanced, particularly for noisy and incompletely sampled Gaussian functions with long tails. These will be demonstrated by simulation results.

Digital Object Identifier 10.1109/MSP.2022.3194692
Date of current version: 27 October 2022

Prior art and motivation

Let us start by elaborating on the signal model. A Gaussian function can be written as

$$f(x) = Ae^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{1}$$

where x is the function variable, and A, μ, and σ are the parameters to be estimated. They represent the height, location, and width of the function, respectively. Directly fitting $f(x)$ can be cumbersome due to the exponential function. A well-known opponent of the exponential is the natural logarithm. Indeed, by taking the natural logarithm of both sides of (1), we can obtain the following polynomial after some basic rearrangements:

$$\ln(f(x)) = a + bx + cx^2, \tag{2}$$

where the coefficients a, b, and c are related to the Gaussian function parameters μ, σ, and A. Based on (1) and (2), it is easy to obtain

$$\mu = \frac{-b}{2c}; \quad \sigma = \sqrt{\frac{-1}{2c}}; \quad A = e^{a-b^2/(4c)}. \tag{3}$$

We see that the estimations of μ, σ, and A can be done through estimating a, b,

and c. Since a, b, and c are coefficients of a polynomial, they can be readily estimated by employing linear fitting based on, e.g., the least square (LS) criterion [7].

In modern signal processing, we generally deal with noisy digital signals. Thus, instead of $f(x)$ given in (1), the following signal is more likely to be dealt with:

$$y[n] = f[n] + \xi[n], \quad \text{s.t. } f[n] = f(n\delta_x), \quad n = 0, \ldots, N-1, \tag{4}$$

where n is the sample index, δ_x is the sampling interval of x, and $\xi[n]$ is an additive white Gaussian noise. If we take the natural logarithm of $y[n]$, we then have

$$\begin{aligned} \ln(y[n]) &= \ln(f[n] + \xi[n]) \\ &= \ln\left(f[n]\left(1 + \frac{\xi[n]}{f[n]}\right)\right), \\ &\approx \ln(f[n]) + \frac{\xi[n]}{f[n]} \\ &= a + b\delta_x n + c\delta_x^2 n^2 + \frac{\xi[n]}{f[n]}, \end{aligned} \tag{5}$$

where the first-order Taylor series $\ln(1+x) \approx x$ is applied to get the approximation, and $\ln(f[n])$ is written into a polynomial form based on (2). Next, we review several linear fitting methods through which the motivation of this work will be highlighted.

For the sake of conciseness, we employ vector/matrix forms for the sequential illustrations. In particular, let us define the following three vectors:

$$\begin{aligned} \mathbf{y} &= [\ln(y[0]), \ln(y[1]), \ldots, \ln(y[N-1])]^{\mathrm{T}}, \\ \mathbf{x} &= [0, \delta_x, \ldots, (N-1)\delta_x]^{\mathrm{T}}, \\ \boldsymbol{\theta} &= [a, b, c]^{\mathrm{T}}. \end{aligned} \tag{6}$$

Then, based on (5), the linear Gaussian fitting problem can be conveniently written as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\xi}, \quad \text{s.t.} \quad \mathbf{X} = [\mathbf{1}, \mathbf{x}, \mathbf{x} \odot \mathbf{x}], \tag{7}$$

where $\boldsymbol{\xi}$ denotes a column vector stacking the noise terms $\xi[n]/f[n]$ $(n = 0, 1, \ldots, N-1)$ in (5), and $\odot$ denotes the pointwise product. Due to the use of these vector/matrix forms, the estimators reviewed in the following sections look different from their descriptions in the original work. However, regardless of the forms, they are the same in essence.

LS fitting

The first fitting method [4] reviewed here applies LS on (7) to estimate the three unknown coefficients in $\boldsymbol{\theta}$. The solution is classical and can be written as [7]

$$\hat{\boldsymbol{\theta}} = \mathbf{X}^{\dagger}\mathbf{y} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}, \tag{8}$$

where $\mathbf{X}^{\dagger}$ denotes the pseudoinverse of $\mathbf{X}$. The LS fitting is simple but not without problems. As can be seen from (5), the additive white Gaussian noise $\xi[n]$ is divided by $f[n]$. The division can severely increase the noise power at ns with $f[n] \approx 0$, causing the noise enhancement problem. As a consequence of the problem, LS can suffer from poor

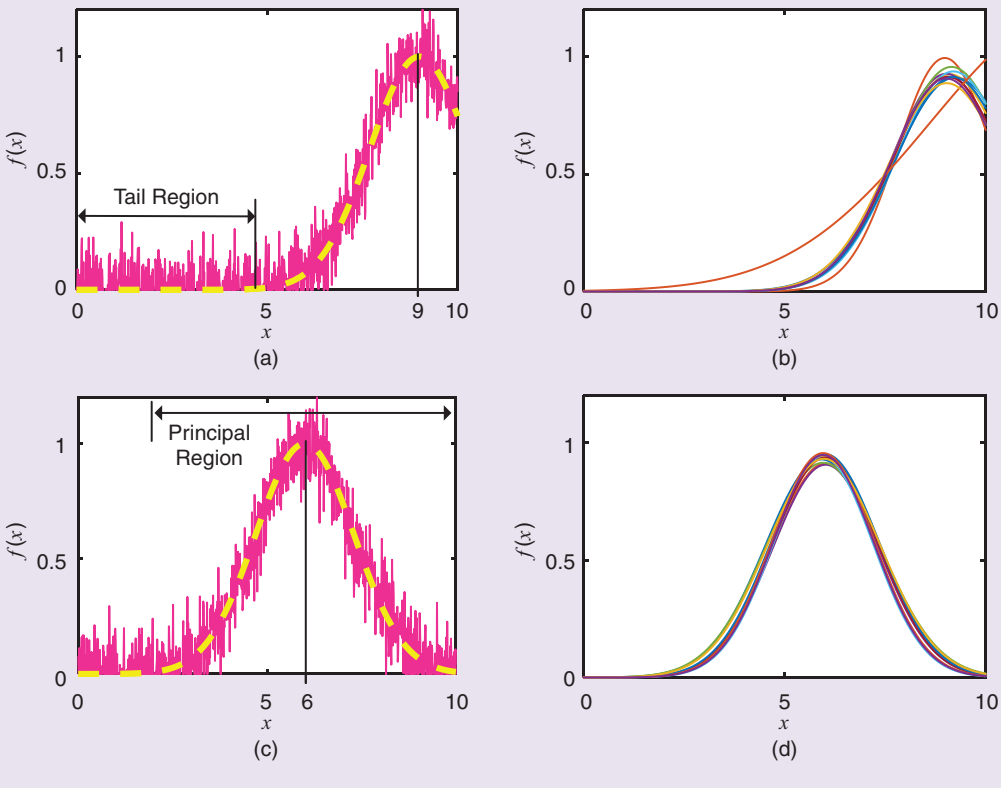

FIGURE 1. (a) The noisy samples of the Gaussian function with $A = 1$, $\mu = 9$, and $\sigma = 1.3$ are plotted, where the dashed curve is the function without noise. (b) The iterative WLS, as illustrated in (9), is run 10 times, each time with 12 iterations and independently generated noise, leading to the fitting results shown. (c) and (d) Other than resetting $\mu = 6$, the same results are plotted as in (a) and (b), respectively. As shown, the range of x is [0, 10], where the sampling interval is set as $\delta_x = 0.01$.

fitting performance, particularly when the majority of samples are from the tail region of a Gaussian function, such as the one plotted in Figure 1(a).

Iterative WLS fitting

To solve the noise enhancement issue, the authors of [5] propose replacing LS with WLS. In contrast to LS, which treats each sample equally, WLS applies different weights over samples. The purpose of weighting is to counterbalance noise variations. For this purpose, $f[n]$ is the ideal weight, as seen from (5). However, $f[n]$ is the digital sample of an unknown Gaussian function.

To solve the problem, an iterative WLS is developed in [5], where $y[n]$ (the noisy version of $f[n]$) is used as the initial weight. From the second iteration, the weight is constructed using the previous estimates of a, b, and c based on the relation depicted in (2). Let $\mathbf{w}_i$ denote the weighting vector at the ith iteration, collecting the weights over $n = 0, 1, \ldots, N-1$. Then, the iterative WLS can be executed as

$$\begin{aligned} &\hat{\boldsymbol{\theta}}_i = (\mathbf{X}_i^{\mathrm{T}}\mathbf{X}_i)^{-1}\mathbf{X}_i^{\mathrm{T}}\mathbf{y}_i, \quad i = 0, 1, \ldots \\ \text{s.t.} \quad &\mathbf{y}_i = \mathbf{w}_i \odot \mathbf{y}; \quad \mathbf{X}_i = [\mathbf{w}_i, \mathbf{w}_i \odot \mathbf{x}, \\ &\qquad \mathbf{w}_i \odot \mathbf{x} \odot \mathbf{x}] \\ &\mathbf{w}_i = e^{\mathbf{y}} \quad \text{if } i = 0; \quad \text{otherwise}, \\ &\mathbf{w}_i = e^{\mathbf{X}\hat{\boldsymbol{\theta}}_{i-1}}, \end{aligned} \tag{9}$$

where $\mathbf{x}$ and $\mathbf{y}$ are given in (6) and $\mathbf{X}$ in (7). The exponential is calculated pointwise in the last row.

Provided a sufficiently large number of iterations, iterative WLS can achieve a high-performance Gaussian fitting, generally better than LS. However, the more iterations, the more time-consuming iterative WLS would be. Moreover, the required number of iterations to achieve the same fitting performance changes with the proportion of the tail region.

Take the Gaussian function in Figure 1(a) for an illustration. The tail region is about half the whole sampled region. Perform 12 numbers of iterations based on (9) for 10 trials, each adding independently generated noise onto the same Gaussian function with $A = 1$, $\mu = 9$, and $\sigma = 1.3$. The fitting results are given in Figure 1(b). We see that some fitting results still substantially differ from the true function, even after 12 iterations. In contrast, perform the same fitting as described but change μ to six (equivalently reducing the tail region). The fitting results of 10 independent trials are plotted in Figure 1(d). Obviously, the results look much better than those in Figure 1(b).

A common solution to reducing the number of iterations required by an iterative algorithm is a good initialization. In our case, the quality of the initial weight vector, i.e., $\mathbf{w}_0$ in (9), can affect the overall number of iterations required by the iterative WLS to converge. Moreover, if the way of initializing $\mathbf{w}_0$ can be immune to the proportion of the tail region, the convergence performance of the iterative WLS can then be less dependent on the proportion of the tail region. Our work is mainly aimed at designing a way of initializing the weight vector of the iterative WLS so as to reduce the number of overall iterations and relieve the dependence of fitting performance on the proportion of the tail region.

An interesting but not-good-enough initialization

An initialization for iterative WLS-based linear Gaussian fitting is developed in [6], which was originally motivated by separately fitting the parameters of a Gaussian function. In particular, exploiting the following relation

$$\int_{-\infty}^{\infty} f(x)\,\mathrm{d}x = A\sqrt{2\pi}\,\sigma, \tag{10}$$

the work proposes a simple estimation of σ, as given by

$$\hat{\sigma} = \frac{1}{\hat{A}} \sum_{n=0}^{N-1} y[n]\,\delta_x, \tag{11}$$

where $y[n]$ is the noisy sample of the Gaussian function to be estimated, δ_x is the sampling interval of x, and the summation approximates the integral of $f(x)$ described earlier. Moreover, $\hat{A}$ is estimated by

$$[\hat{A}, \hat{n}] = \max_n\, y[n]. \tag{12}$$

Similar to how $\max(\cdot)$ works in MATLAB [8], $\hat{n}$ is the index where the maximization is achieved. According to the middle relation in (3), c can be estimated as $\hat{c} = -1/2\hat{\sigma}^2$. Thus, the work in [6] removes c from the parameter vector $\boldsymbol{\theta}$ given in (6) and employs an iterative WLS, similar to (9) but with reduced dimension, to estimate a and b.

A major error source of $\hat{\sigma}$ obtained in (11) is the approximation error from using the summation in (11) to approximate the integral given in (10). Even if δ_x is fine enough, the approximation can still be problematic, depending on the proportion of the tail region in the sampled Gaussian function. For example, if the sampled function has a shape similar to the one given in Figure 1(c), we know that the summation can well approximate the integral given a fine δ_x. However, if the sampled function has a shape like the curve in Figure 1(a), the approximation error will be large regardless of δ_x. A condition is given in [6] stating when the summation in (11) can well approximate the integral in (10). Nevertheless, the question "What shall we do when the condition is not satisfied?"—which can be inevitable in practice—is yet to be answered.

Proposed Gaussian fitting

Looking at Figure 1(a), we know the summation in (11) cannot approximate the integral in (10). Now, instead of using the summation to approximate something unachievable, how about looking into a different question: "What can be approximated using the summation given in (11)?" We answer this question by performing the following computations (which look complex but are easily understandable):

$$\begin{aligned} \sum_{n=0}^{N-1} y[n]\,\delta_x &\overset{(a)}{\approx} \sum_{n=0}^{N-1} \delta_x f[n] \overset{(b)}{\approx} \int_0^{N\delta_x} f(x)\,\mathrm{d}x \\ &\overset{(c)}{=} \int_0^{\mu} f(x)\,\mathrm{d}x + \int_{\mu}^{N\delta_x} f(x)\,\mathrm{d}x \\ &\overset{(d)}{=} \frac{1}{2}\int_0^{2\mu} f(x)\,\mathrm{d}x \\ &\quad + \frac{1}{2}\int_{\mu-(N\delta_x-\mu)}^{N\delta_x} f(x)\,\mathrm{d}x \\ &\overset{(e)}{=} \frac{\sqrt{2\pi}A\sigma}{2}\,\mathrm{erf}\Big(\frac{\mu}{\sigma\sqrt{2}}\Big) \\ &\quad + \frac{\sqrt{2\pi}A\sigma}{2}\,\mathrm{erf}\Big(\frac{N\delta_x-\mu}{\sigma\sqrt{2}}\Big), \end{aligned} \tag{13}$$

where $\mathrm{erf}(\cdot)$ is the so-called error function. It can be defined as [1]

$$\mathrm{erf}(z)=\frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}\mathrm{d}t. \tag{14}$$

How each step in (13) is obtained is detailed as follows:

- $\overset{(a)}{\approx}$: This step replaces $y[n]$ with $f[n]$ by omitting the noise term $\xi[n]$, as given in (4).
- $\overset{(b)}{\approx}$: This is how the integral is often introduced in a math textbook. While the left side of $\overset{(b)}{\approx}$ approximately calculates the area below $f(x)$, the right side does so exactly.
- $\overset{(c)}{=}$: The integrating interval is split into contiguous halves.
- $\overset{(d)}{=}$: The integrating interval of each integral in $\overset{(c)}{=}$ is doubled in such a way that it becomes symmetric against $x=\mu$. Since $f(x)$ is also symmetric against $x=\mu$, see (1): the scaling coefficient $1/2$ can counterbalance the extension of the integrating interval.
- $\overset{(e)}{=}$: It is based on a known fact [1]:

$$\int_{\mu-\epsilon}^{\mu+\epsilon} f(x)\,\mathrm{d}x=\sqrt{2\pi}A\sigma\,\mathrm{erf}\Big(\frac{\epsilon}{\sigma\sqrt{2}}\Big).$$

Similar to $\overset{(c)}{=}$ in (13), we can also split the summation $\Sigma_{n=0}^{N-1}y[n]\delta_x$. Doing so, the two integrals on the right-hand side of $\overset{(c)}{=}$ in (13) can be, respectively, approximated by

$$\mathsf{S}_\beta \triangleq \sum_{n=0}^{\hat{n}-1} y[n]\delta_x \quad \text{and}$$
$$\mathsf{S}_\alpha \triangleq \sum_{n=\hat{n}}^{N-1} y[n]\delta_x, \tag{15}$$

where $\hat{n}$ is obtained in (12). (Note that $\hat{n}\delta_x$ is an estimate of μ.) Moreover, tracking the computations in (13), we can easily attain

$$\mathsf{S}_\beta \approx \frac{\sqrt{2\pi}\hat{A}\sigma}{2}\,\mathrm{erf}\Big(\frac{\hat{n}\delta_x}{\sigma\sqrt{2}}\Big);$$
$$\mathsf{S}_\alpha \approx \frac{\sqrt{2\pi}\hat{A}\sigma}{2}\,\mathrm{erf}\Big(\frac{N\delta_x-\hat{n}\delta_x}{\sigma\sqrt{2}}\Big), \tag{16}$$

where A and μ have been replaced by their estimates given in (12). The two equations in (16) provide possibilities for estimating σ, which is the only unknown left. However, due to the presence of the nonelementary function $\mathrm{erf}(\cdot)$, analytically solving σ from the equations is nontrivial. Moreover, we have two equations but one unknown. How to constructively exploit the information provided by both equations is also a critical problem. Here, we first develop an efficient method to estimate σ from either equation in (16), resulting in two estimates of σ; we then derive an asymptotically optimal combination of the two estimates.

Efficient estimation of σ

The two equations in (16) have the same structure. Therefore, let us focus on the top one for now. While solving σ analytically is difficult, numerical means can be resorted to. As is commonly done, we can select a large region of σ, discretize the region into fine grids, evaluate the values of the right-hand side in (16) on the grids, and identify the grid that leads to the closest result to S_β.

These steps are regular but not practically efficient. This is because evaluating the right-hand side of the equation in (16) needs the calculation of $\mathrm{erf}(\cdot)$ for each σ grid. From (14), we see that $\mathrm{erf}(\cdot)$ itself is an integral result. If we calculate $\mathrm{erf}(\cdot)$ onboard, it would be approximated by a summation over sufficiently fine grids of the integrating variable. This can be highly time-consuming, particular given that $\mathrm{erf}(\cdot)$ needs to be calculated for each entry in a large set of σ grids. Alternatively, we may choose to store a lookup table of $\mathrm{erf}(\cdot)$ onboard. This is doable but can also be troublesome, for the reason that the parameter of $\mathrm{erf}(\cdot)$, as dependent on the parameters of the Gaussian function to be fitted, can span over a large range in different applications. The trouble, however, can be relieved through a simple variable substitution.

Making the substitution of $\hat{n}\delta_x=\mathsf{k}\sigma$ in (16), we obtain

$$\mathsf{S}_\beta \approx \frac{\sqrt{2\pi}\hat{A}\hat{n}\delta_x}{2\mathsf{k}}\,\mathrm{erf}\Big(\frac{\mathsf{k}}{\sqrt{2}}\Big). \tag{17}$$

Clearly, the dependence of the $\mathrm{erf}(\cdot)$ function on μ (represented by $\hat{n}\delta_x$ and σ, as shown in (16), is now removed, making the $\mathrm{erf}(\cdot)$ function solely related to the coefficient k. Therefore, a significance of the variable substitution is that one lookup table of $\mathrm{erf}(\cdot)$ can be applied to a variety of applications with different Gaussian function parameters. Assuming $k=\mathsf{k}^\star$ makes the right-hand side of (17) closest to S_β, σ can then be estimated as $\hat{n}\delta_x/\mathsf{k}^\star$. Similarly, we can make the substitution for the bottom equation in (16) and obtain another estimate of σ. In summary, the two estimators can be established as in (18) shown at the bottom of the page. The two estimates would have different qualities, depending on how many samples are used for each. This further suggests that combining them is not as trivial as simply averaging them. Next, we develop a constructive way of combining them.

Asymptotically optimal σ estimation

The two estimates obtained in (18) are mainly differentiated by how many samples are used in their estimations. Therefore, identifying the impact of the employed samples on the estimation performance is helpful in determining a way to combine the estimates. One of the most common performance metrics for an estimator is the Cramér–Rao lower bound (CRLB) [7]. This points the direction of our next move.

Let us check the CRLB of $\hat{\sigma}_\beta$ first. It is estimated using the samples $y[n]$ for $n=0,1,\ldots,\hat{n}-1$. Referring to (4), $y[0],\ y[1],\ldots,y[\hat{n}-1]$ are jointly normally distributed with different means

$$\hat{\sigma}_\alpha=\frac{(N-\hat{n})\delta_x}{\mathsf{k}^\star};\quad \text{s.t.}\quad \mathsf{k}^\star:\operatorname{argmin}_{\mathsf{k}}\Big(\mathsf{S}_\alpha-\frac{\sqrt{2\pi}\hat{A}(N-\hat{n})\delta_x}{2\mathsf{k}}\,\mathrm{erf}\Big(\frac{\mathsf{k}}{\sqrt{2}}\Big)\Big)^2;$$
$$\hat{\sigma}_\beta=\frac{\hat{n}\delta_x}{k^\star};\quad \text{s.t.}\quad \mathsf{k}^\star:\operatorname{argmin}_{k}\Big(\mathsf{S}_\beta-\frac{\sqrt{2\pi}\hat{A}\hat{n}\delta_x}{2k}\,\mathrm{erf}\Big(\frac{k}{\sqrt{2}}\Big)\Big)^2. \tag{18}$$

but the same variance, as given by σ_ξ^2. Recall that σ_ξ^2 is the power of the noise term $\xi[n]$ $(\forall n)$ given in (4). Since we focus here on investigating the estimation performance of σ, we assume that A and μ are known. Then, the CRLB of $\hat{\sigma}_\beta$ estimation, as obtained based on $y[0], y[1], \ldots, y[\hat{n}-1]$, can be computed by

$$\begin{aligned}\mathrm{CRLB}\{\hat{\sigma}_\beta\} &= \frac{\sigma_\xi^2}{\sum_{n=0}^{\hat{n}-1}\left(\frac{\partial f[n]}{\partial \sigma}\right)^2} \\ &= \frac{\sigma_\xi^2}{\frac{\sum_{n=0}^{\hat{n}-1} f[n]^2(\mu-\delta_x n)^4}{\sigma^6}},\end{aligned} \tag{19}$$

where the middle result is a simple application of [7, Eq. (3.14)], and the first partial derivative of $f[n]$ can be readily derived based on its expression given in (4). With reference to (19), we can directly write the CRLB of $\hat{\sigma}_\alpha$, as given by

$$\mathrm{CRLB}\{\hat{\sigma}_\alpha\} = \frac{\sigma_\xi^2}{\frac{\sum_{\hat{n}}^{N-1} f[n]^2(\mu-\delta_x n)^4}{\sigma^6}}, \tag{20}$$

where the sole difference compared with (19) is the set of ns for the summation.

Jointly inspecting the two CRLBs, we see that they only differ by a linear coefficient; namely,

$$\frac{\mathrm{CRLB}\{\hat{\sigma}_\beta\}}{\mathrm{CRLB}\{\hat{\sigma}_\alpha\}} = \frac{\sum_{n=\hat{n}}^{N-1} f[n]^2(\mu-\delta_x n)^4}{\sum_{n=0}^{\hat{n}-1} f[n]^2(\mu-\delta_x n)^4}. \tag{21}$$

An insight from this result is that we only need a linear combination of the two σ estimates obtained in (18) to achieve an asymptotically optimal combined estimation. To further illustrate this, let us consider the following linear combination:

$$\hat{\sigma} = \rho\hat{\sigma}_\alpha + (1-\rho)\hat{\sigma}_\beta, \quad \rho \in (0,1). \tag{22}$$

The mean square error (MSE) of the combined estimate can be computed as

$$\begin{aligned}\mathbb{E}\{(\hat{\sigma}-\sigma)^2\} &= \mathbb{E}\{(\rho\hat{\sigma}_\alpha + (1-\rho)\hat{\sigma}_\beta \\ &\quad - \rho\sigma - (1-\rho)\sigma)^2\} \\ &= \rho^2\mathbb{E}\{(\hat{\sigma}_\alpha-\sigma)^2\} \\ &\quad + (1-\rho)^2\mathbb{E}\{(\hat{\sigma}_\beta-\sigma)^2\} \\ &\sim \rho^2\mathrm{CRLB}\{\hat{\sigma}_\alpha\} \\ &\quad + (1-\rho)^2\mathrm{CRLB}\{\hat{\sigma}_\beta\},\end{aligned} \tag{23}$$

where "$a \sim b$" denotes that a asymptotically approaches b or a constant linear scaling of b.

Solving $\partial\mathbb{E}\{(\hat{\sigma}-\sigma)\}/\partial\rho = 0$ leads to the following optimal ρ:

$$\begin{aligned}\rho^\star &= \frac{\mathrm{CRLB}\{\hat{\sigma}_\beta\}}{\mathrm{CRLB}\{\hat{\sigma}_\beta\} + \mathrm{CRLB}\{\hat{\sigma}_\alpha\}} \\ &= \frac{\sum_{n=\hat{n}}^{N-1} f[n]^2(\mu-\delta_x n)^4}{\sum_{n=0}^{N-1} f[n]^2(\mu-\delta_x n)^4},\end{aligned}$$

where the CRLB expressions (19) and (20) have been applied. The optimality of $\rho^\star$ can be validated by plugging $\rho = \rho^\star$ into (23). Doing so yields

$$\mathbb{E}\{(\hat{\sigma}-\sigma)^2\} \sim \frac{\sigma_\xi^2}{\frac{\sum_{n=0}^{N-1} f[n]^2(\mu-\delta_x n)^4}{\sigma^6}}. \tag{24}$$

From the index ranges of the summations in (19), (20), and (24), we can see that the right-hand side of (24) becomes the CRLB of the σ estimation that is obtained based on all samples at hand—the best estimation performance for any unbiased estimator of σ based on $y[0], y[1], \ldots, y[N-1]$. That is, taking $\rho = \rho^\star$ in (22) leads to an asymptotically optimal unbiased σ estimation.

Note that $f[n]$ used for calculating $\rho^\star$ is unavailable. Thus, we replace $f[n]$ with its noisy version $y[n]$, attaining the following practically usable coefficient:

$$\rho^\star \approxeq \frac{\sum_{n=\hat{n}}^{N-1} y[n]^2(\mu-\delta_x n)^4}{\sum_{n=0}^{N-1} y[n]^2(\mu-\delta_x n)^4}. \tag{25}$$

If we define A^2/σ_ξ^2 as the estimation signal-to-noise ratio (SNR), where σ_ξ^2 is the power of the noise term $\xi[n]$ in (4), then the equality in (25) can be approached as the estimation SNR increases.

Improving the estimation performance of A and μ

So far, we have focused on introducing the novel estimation of σ. From (18), we can see that the proposed σ estimation requires the estimations of A and μ, i.e., $\hat{A}$ and $\hat{n}\delta_x$ therein. To ensure a clear logic flow, we used the naive way of estimating these two parameters, as described in (12). Here, we illustrate some simple yet more accurate methods for estimating A and μ.

For μ estimation, we introduce a local averaging to reduce the impact of noise. Define a rectangular window function as $W[n] = (1/L)$ for $n = 0, 1, \ldots, L-1$ and $W[n] = 0$ for other ns. The local averaging can be performed by using the window function to filter the sampled Gaussian function, i.e., $y[n]$ given in (4). An improved μ estimation can be achieved by searching for the peak of the filtered Gaussian function and then constructing using the peak index. This is expressed as

$$\hat{\mu} = \left(\hat{n} + \left\lfloor \frac{L}{2} \right\rfloor\right)\delta_x, \text{ s.t. } \hat{n} : \operatorname*{argmax}_n \sum_{l=0}^{L-1} W[l]\, y[n+l]. \tag{26}$$

The offset $\lfloor L/2 \rfloor$ is added because, in theory, if the sum of continuous L samples is maximum, those samples would be centered around the peak of a Gaussian function. Based on (26), we also obtain an estimate of A, i.e.,

$$\hat{A} = y\left[\hat{n} + \left\lfloor \frac{L}{2} \right\rfloor\right]. \tag{27}$$

Use these two estimates obtained in the proposed σ estimators, as given in (18). Then, combine the two estimates, as done in (22), with the optimal combining coefficient given in (25). This results in the final σ estimate. Unlike $\hat{\mu}$ and $\hat{\sigma}$ obtained using multiple samples, $\hat{A}$ given in (27) is based on a single sample and, hence, can suffer from a large estimation error. Noticing this, we suggest another refinement of $\hat{A}$ through minimizing the MSE, which is calculated as

$$\frac{1}{N}\sum_{n=0}^{N-1}\left(xe^{-\frac{(n\delta_x-\hat{\mu})^2}{2\hat{\sigma}^2}}-y[n]\right)^2,$$

with respect to x. The solution to the minimization is an improved A estimate, as given by

$$\hat{A}=\frac{\left(\sum_{n=0}^{N-1}e^{-\frac{(n\delta_x-\hat{\mu})^2}{2\hat{\sigma}^2}}y[n]\right)}{\left(\sum_{n=0}^{N-1}\left(e^{-\frac{(n\delta_x-\hat{\mu})^2}{2\hat{\sigma}^2}}\right)^2\right)}. \tag{28}$$

We summarize the proposed Gaussian fitting method in Table 1 under the code name *M3*. As mentioned at the end the "Iterative WLS Fitting" section, we locate our method as an initial-stage fitting. For the second stage, we perform the iterative WLS with the initial weighting vector, i.e., $\mathbf{w}_0$ in (9), constructed using our initial fitting results. This two-stage fitting is named *M4* in Table 1. We underline that, due to the high quality of the proposed initialization, M4 can converge much faster than the original iterative WLS, named *M5* in Table 1. This will be validated shortly by simulation results. Also provided in the table is the σ estimation method reviewed in the "An Interesting but Not-Good-Enough Initialization" section, named as *M1*. Moreover, the combination of M1 and the iterative WLS is referred to as *M2* in Table 1.

Simulation results

Simulation results are presented next to illustrate the performance of the five Gaussian fitting methods summarized in Table 1. The MATLAB simulation codes for generating Figures 2 and 3 can be downloaded from https://www.icloud.com/iclouddrive/005qLeEe1YOghnz4Om24ct76w\#publish_v2. Unless otherwise specified, the parameters summarized in Table 2 are primarily used in our simulations. For the original iterative WLS, we perform 12 iterations so that it can achieve a similar asymptotic performance as M4 in the high-SNR region. (This will be seen shortly in Figure 2.) In contrast, when the initialization from either M1 or (the proposed) M3 is employed, only two iterations are performed for the iterative WLS algorithm. Note that the running time for each method is provided in Table 1, where each time result is averaged over 10^5 independent trials.

Figure 2 plots the MSEs of the five estimators listed in Table 1 against the estimation SNR, as given by A^2/σ_ξ^2. Note that μ and σ are randomly generated for the 10^5 independent trials, conforming to the uniform distributions given in Table 2. As given in Table 2, $x\in[0,10]$ is set in the simulation. Thus, the settings of μ and σ make the Gaussian function to be fitted in each trial incompletely sampled with a long tail; a noisy version of the function is plotted in Figure 1(a).

From Figure 2(b), we see that M3, which is based on the proposed local averaging in (26), achieves an obviously better μ estimation performance than M1, which is based on the naive method given in (12). From Figure 2(c), we see that M3 substantially outperforms M1. This validates the competency of the proposed σ estimation scheme for the cases with the Gaussian function (to be fitted) incompletely sampled. From Figure 2(a), we see a significant improvement in M3, as compared with M1. This validates the advantage of using all samples for A estimation, as developed in (28).

We remark that, as a price paid for performance improvement, the proposed initial fitting requires slightly more computational time than M1; see the last column of Table 1 for comparison. However, it is noteworthy that, thanks to the signal processing tricks introduced in the "Efficient Estimation of σ" section, the proposed σ estimator, as established in (18), only involves simple floating point arithmetic that can be readily handled by modern digital signal processors or field programming gate arrays.

From Figure 2, we further see that the proposed initial fitting results enable the iterative WLS to achieve much better performance for all three parameters than other initializations. The improvement is particularly noticeable in low-SNR regions. Moreover, we underline that M4 based on the proposed initialization only runs two iterations, while the original iterative WLS, i.e., M5, runs 12 iterations. This, on the one hand, illustrates the critical importance of improving the initialization for the iterative WLS, a main motivation of this work. On the other hand, this validates our success in developing a high-quality initialization for the iterative WLS.

In spite of the random changing of μ and σ over 10^5 independent trials, Figure 2 shows that our proposed initial fitting enables WLS to achieve

Table 1. A summary of the simulated methods, where the running time of each method is averaged over 10^5 independent trials.

Code Name	Method	Fitting Steps	Time (μs)
M1	[6]	Estimate $\hat{A}$ and $\hat{n}$ as done in (12), where $\hat{n}$ leads to $\hat{\mu}=\hat{n}\delta_x$; estimate $\hat{\sigma}$ using (11).	76.79
M2	[6] and [5]	Stage 1: Run M1 first, getting initial $\hat{A}$, $\hat{\mu}$, and $\hat{\sigma}$. Stage 2: Perform the iterative WLS based on (9), where $\mathbf{w}_0=\left[\hat{A}e^{-\frac{(n\delta_x-\hat{n}\delta_x)^2}{2\hat{\sigma}^2}}\Big\vert_{n=0,1,\ldots,N-1}\right]^{\mathrm{T}}$.	275
M3	New	Estimate $\hat{\mu}$ based on (26); estimate $\hat{A}$ using (27); perform the estimators in (18), attaining σ_α and σ_β; combine the two estimates in the linear manner depicted in (22), where the optimal ρ, as approximately calculated in (25), is used as the combination coefficient; and refine $\hat{A}$ as done in (28).	304.61
M4	New and [5]	Stage 1: Run M3 first, getting initial $\hat{A}$, $\hat{\mu}$, and $\hat{\sigma}$. Stage 2: This is the same as in M2.	502.81
M5	[5]	Perform the iterative WLS based on (9).	1,009.72

The simulations are run in MATLAB R2021a installed on a computing platform equipped with the Intel Xeon Gold 6238 R 2.2-GHz 38.5-MB L3 Cache (maximum turbo frequency: 4 GHz; minimum: 3 GHz).

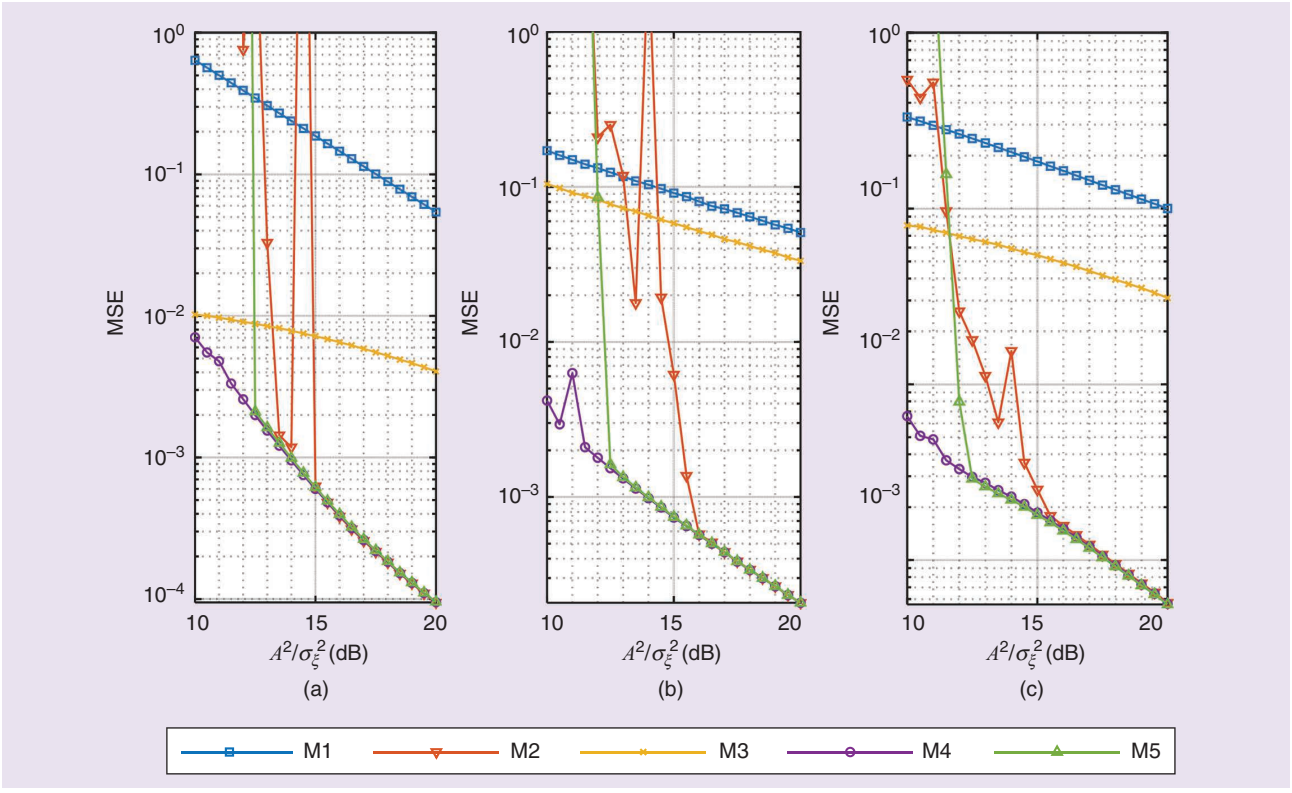

FIGURE 2. The MSEs of fitting results versus the SNR in the sampled Gaussian function with $A = 1$, where μ and σ are randomly generated based on the uniform distributions given in Table 2, for (a) $\hat{A}$, (b) $\hat{\mu}$, and (c) $\hat{\sigma}$. Note that σ_ξ^2 denotes the power of the noise term $\xi[n]$ given in (4). The MSE is calculated over 10^5 trials, each with independently generated and normally distributed noise.

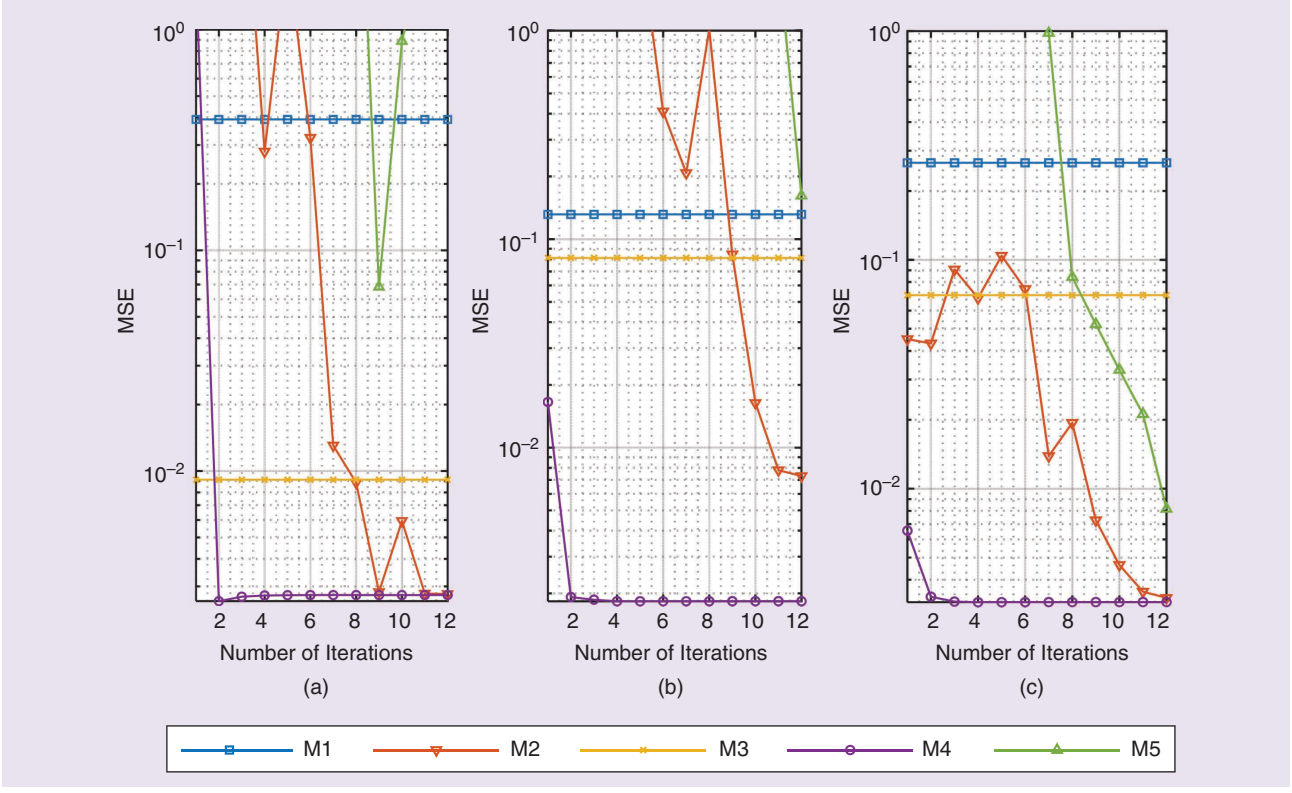

FIGURE 3. The MSEs of fitting results versus the number of iterations used in M5 and the second stage of M2/M4 for (a) $\hat{A}$, (b) $\hat{\mu}$, and (c) $\hat{\sigma}$, where $A = 1$, A^2/σ_ξ^2 is 12 dB, and μ and σ are randomly generated based on the uniform distributions given in Table 2. The MSE is calculated over 10^5 trials, each with independently generated and normally distributed noise.

consistently better and more stable performance than prior arts. This suggests that we have successfully relieved the dependence of the iterative WLS on the proportion of the tail region of a Gaussian function. In contrast, as illustrated in the "Iterative WLS Fitting" and "An Interesting but Not-Good-Enough Initialization" sections, the performance of M5 and M1 can be subject to how complete a Gaussian function is sampled. M2, which is based on M1, also has the dependence.

Figure 3 shows the MSEs of the five estimators listed in Table 1 against the number of iterations of the iterative WLS, performed in the second stage of M2/M4 and M5. For all three parameters, we can see that the proposed initialization (M3) nontrivially outperforms the state-of-the-art M1. We also see that M4 approximately converges after two iterations, while M2 and M5 present much slower convergence with the MSE performance, inferior to M4 even after 12 iterations. These observations highlight the critical importance of a good initialization to the iterative WLS, particularly when the sampled Gaussian function is noisy and incomplete with a long tail. They again validate the effectiveness of the proposed techniques, as enabled by the unveiled signal processing tricks, in these challenging scenarios.

Table 2. The simulation parameters.

Variable	Description	Value
A	A parameter of the Gaussian function determining its maximum amplitude [see (1)]	1
μ	A parameter of the Gaussian function indicating its peak location [see (1)]	$\mathcal{U}$ [8,9]
σ	A parameter of the Gaussian function indicating its width of the principal region [see Figure 1(c)]	$\mathcal{U}$ [1,1.3]
x	Function variable	[0,10]
δ_x	Sampling interval of x [see (4)]	0.01
k	Intermediate variable used in the proposed estimators given in (18)	0.1 : 0.01 : 10
L	Windows size for estimating μ as done in (26)	3
	Number of iterations for M5	12
	Number of iterations in stage 2 of M2/M4	2
σ_ξ^2	Power of the additive noise $\xi[n]$ given in (4)	−10 : 0.5 : 20 dB

$\mathcal{U}[a,b]$ denotes the uniform distribution in $[a,b]$.

Conclusions

In this article, we develop a high-quality initialization method for the iterative WLS-based linear Gaussian fitting algorithm. This is achieved by a few signal processing tricks, as summarized here:

- We introduce a simple local averaging technique that reduces the noise impact on estimating the peak location of a Gaussian function, i.e., μ.
- We provide a more precise integral result that is approximated by the summation of the Gaussian function samples, which results in two estimates of σ.
- We unveil the linear relation between the asymptotic performance of the two estimates and then design an asymptotically optimal combination of these estimates.
- We also improve the estimation of the peak amplitude of a Gaussian function by minimizing the MSE of the initial Gaussian fitting.

Corroborated by simulation results, the proposed initialization can substantially improve the accuracy of the iterative WLS-based linear Gaussian fitting, even in challenging scenarios with strong noises and the incompletely sampled Gaussian function with a long tail. Notably, the performance improvement is also accompanied by improved fitting efficiency.

Acknowledgment

We thank Prof. Rodrigo Capobianco Guido, *IEEE Signal Processing Magazine*'s area editor for columns and forum, and Dr. Wei Hu, associate editor for columns and forum, for managing the review of our article. We also thank the editors and the anonymous reviewers for providing constructive suggestions to improve our work. We further acknowledge the support of the Australian Research Council under the Discovery Project grant DP210101411.

Authors

Kai Wu (kai.wu@uts.edu.au) received his Ph.D. degree from Xidian University, China, in 2019 and his Ph.D. degree from the University of Technology Sydney (UTS), Australia, in 2020. He is a research fellow at the Global Big Data Technologies Centre, UTS, Sydney, NSW 2007, Australia. His Xidian Ph.D. won the Excellent Ph.D. Thesis Award 2019 from the Chinese Institute of Electronic Engineering. His UTS Ph.D. was awarded The Chancellor's List 2020. His research interests include array signal processing and its applications in radar and communications. He is a Member of IEEE.

J. Andrew Zhang (andrew.zhang@uts.edu.au) received his Ph.D. degree from the Australian National University in 2004. He is an associate professor in the School of Electrical and Data Engineering, University of Technology Sydney, Sydney, NSW 2007, Australia. He was a researcher with Data61, Commonwealth Scientific and Industrial Research Organization (CSIRO), Australia, from 2010 to 2016; Networked Systems, National ICT Australia, Australia, from 2004 to 2010; and ZTE Corp., Nanjing, China, from 1999 to 2001. He has published more than 200 papers in leading international journals and conference proceedings and has won five best paper awards. He received the CSIRO Chairman's Medal and the Australian Engineering Innovation Award in 2012 for exceptional research achievements in multigigabit wireless communications. His research interests include the area of signal processing for wireless communications and sensing. He is a Senior Member of IEEE.

Y. Jay Guo (jay.guo@uts.edu.au) received his Ph.D. degree from Xian

Jiaotong University, Xi'an, China, in 1987. He is a distinguished professor with and the director of the Global Big Data Technologies Centre at the University of Technology Sydney, Sydney, NSW 2007, Australia, and the technical director of the New South Wales Connectivity Innovation Network Australia. He has won a number of prestigious awards, including Australian Engineering Excellence Awards (2007 and 2012) and the CSIRO Chairman's Medal (2007 and 2012). He was named one of the most influential engineers in Australia in 2014 and 2015 and one of the top researchers in Australia in 2020 and 2021. His research interests include antennas, millimeter-wave and terahertz communications and sensing systems, and big data technologies. He is a Fellow of IEEE.

References

[1] A. D. Poularikas, *Handbook of Formulas and Tables for Signal Processing*. Boca Raton, FL, USA: CRC Press, 2018.

[2] "What is a Gaussian function?" LogicPlum. https://logicplum.com/knowledge-base/gaussian-function/ (Accessed: Mar. 11, 2022).

[3] E. Kheirati Roonizi, "A new algorithm for fitting a Gaussian function riding on the polynomial background," *IEEE Signal Process. Lett.*, vol. 20, no. 11, pp. 1062–1065, 2013, doi: 10.1109/LSP.2013.2280577.

[4] R. A. Caruana, R. B. Searle, T. Heller, and S. I. Shupack, "Fast algorithm for the resolution of spectra," *Anal. Chem.*, vol. 58, no. 6, pp. 1162–1167, 1986, doi: 10.1021/ac00297a041.

[5] H. Guo, "A simple algorithm for fitting a Gaussian function [DSP Tips and Tricks]," *IEEE Signal Process. Mag.*, vol. 28, no. 5, pp. 134–137, 2011, doi: 10.1109/MSP.2011.941846.

[6] I. Al-Nahhal, O. A. Dobre, E. Basar, C. Moloney, and S. Ikki, "A fast, accurate, and separable method for fitting a Gaussian function [Tips & Tricks]," *IEEE Signal Process. Mag.*, vol. 36, no. 6, pp. 157–163, 2019, doi: 10.1109/MSP.2019.2927685.

[7] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1993.

[8] "Help center." MathWorks. https://au.mathworks.com/help/matlab/ (Accessed: Mar. 11, 2022).

FROM THE EDITOR

(continued from page 3)

at the Epicenter of Ground-Shaking Research," reports research results of three projects focused on various issues related to earthquakes, from forecasting to localization of victims.

Among these articles, many are using, at least partly, machine learning methods. Although some are considering sparsity priors, I believe that usefulness, pros and cons, cost of these methods (which are very greedy in power, computation, and memory), could be discussed more thoroughly. Benchmarks with more classical methods should use metrics that are able to take into account at least these parameters and not only a simple performance index.

If you have any publication ideas for *SPM*, I encourage you to contact the area editors or me (see the editorial board in the *SPM* web pages) to discuss your idea. Remember that the articles in *SPM* are not suited for the publishing of either new results or surveys. They are tutorial-like articles that must be comprehensive for a wide audience and accompanied by a relevant selection of both figures and references as Robert Heath, the previous *SPM* editor-in-chief explained very clearly in [4]. Concerning the "Lecture Notes" and Tips & Tricks" columns, I also think that sharing data and codes would be an actual added value to these articles, and I encourage authors to use Code Ocean facilities (https://innovate.ieee.org/ieee-code-ocean/) for this purpose.

References

[1] R. Couillet, D. Trystram, and T. Menissier, "The submerged part of the AI-Ceberg [Perspectives]," *IEEE Signal Process. Mag.*, vol. 39, no. 5, pp. 10–17, Sep. 2022, doi: 10.1109/MSP.2022.3182938.

[2] "IEEE Panel of Editors 2022." IEEE.tv. Accessed: Apr. 29, 2022. [Online]. Available: https://ieeetv.ieee.org/channels/communities/welcome-from-panel-of-editors-chair-poe-2022-0

[3] "IEEE Publication Services and Products Board Operations Manual 2021," IEEE Publications, Piscataway, NJ, USA, 2022. [Online]. Available: https://pspb.ieee.org/images/files/files/opsmanual.pdf

[4] R. W. Heath, "Reflections on tutorials and surveys [From the Editor]," *IEEE Signal Process. Mag.*, vol. 37, no. 5, pp. 3–4, Sep. 2020, doi: 10.1109/MSP.2020.3006648.

SP FORUM

(continued from page 75)

across a number of academic publishing companies. He joined IEEE initially as an editor for *IEEE Press* before transitioning to the Intellectual Property Rights (IPR) Office as an IPR specialist.

Christian Jutten (christian.jutten@grenoble-inp.fr) received his master's and Ph.D. degrees in electrical engineering at Institut National Polytechnique de Grenoble, France. He has been a professor since 1989 and an emeritus professor since 2019 at Université Grenoble Alpes, Grenoble 38402 France. Since 1979, his research has focused on statistical signal processing and machine learning with applications in biomedical engineering, speech processing, hyperspectral imaging, and chemical engineering. Since 2019, he has been a scientific advisor for scientific integrity for the French National Center of Scientific Research and, since January 2021, editor-in-chief of *IEEE Signal Processing Magazine*. He is a Fellow of IEEE.