Adapting the Poisson-Influenced *K*-Means Algorithm for a Larger User Base

Brian P. M. Morris

Montgomery Blair High School

Zachary H. Levine, Ph.D.

National Institute of Standards and Technology

Abstract

A superconducting transition edge sensor (TES) can be a useful tool for counting the number of photons in a highly attenuated pulse of light, but it requires calibration for its outputs to be interpretable as photon numbers. The Poisson-Influenced *K*-Means Algorithm (PIKA) was created to calibrate a TES given an amount of information fundamentally limited by the nature of the device. We present a new implementation of PIKA in a Mathematica notebook, with integrated documentation and explanation, a new system for handling input and output, and corrections to bugs and errors that were present in the original version. This version of the algorithm is designed to be more transparent, accessible, and adaptable than its predecessor. The goal is enhanced ease of use and modification for researchers needing the original functionality of TES calibration, for those working on related applications, and for those working on unrelated applications to which the new implementation of PIKA can be adapted.

Introduction and Algorithm Explanation

Levine et al. [1] introduced the Poisson-Influenced K-Means Algorithm (PIKA), a variant of the unsupervised machine learning K-means clustering algorithm that uses the knowledge that cluster sizes should follow a Poisson distribution, as a means of calibrating a certain type of photon counter. The goal of this project was to adapt the existing implementation of the algorithm into a Mathematica notebook [5] for more widespread utility in calibration, and for adaptability to other situations that cluster data based on probability models other than the Poisson distribution. The notebook created has been accepted for publication in *The Mathematica Journal*, with an expected publication date in late 2015 or early 2016 [2].

Transition Edge Sensor Behavior

A transition edge sensor (TES) is a superconducting few-photon detector capable of counting the number of photons in a very weak pulse of light. The device is kept just below its superconducting phase transition temperature, above which it will enter the normal regime and lose its superconducting properties. Photons incident on the sensor heat it beyond that boundary, causing its resistance to rise sharply and then gradually fall to superconducting levels as the heat dissipates. A current is run through the TES, and the change in resistance is captured by the voltage signal of a superconducting quantum interference device (SQUID) inductively coupled to the TES circuit.

Figure 1 shows several groups of TES signal waveforms (each graph shows the set of signals elicited by an ensemble of laser pulses with an average number of photons per pulse given by $\bar{N}$). For $\bar{N} = 4$ one can clearly distinguish the different photon numbers and their relative frequencies; for higher numbers this is more difficult. Higher photon numbers create higher signal spikes, but at a certain point the TES saturates in the normal regime and additional photons change the signal maximum very little.

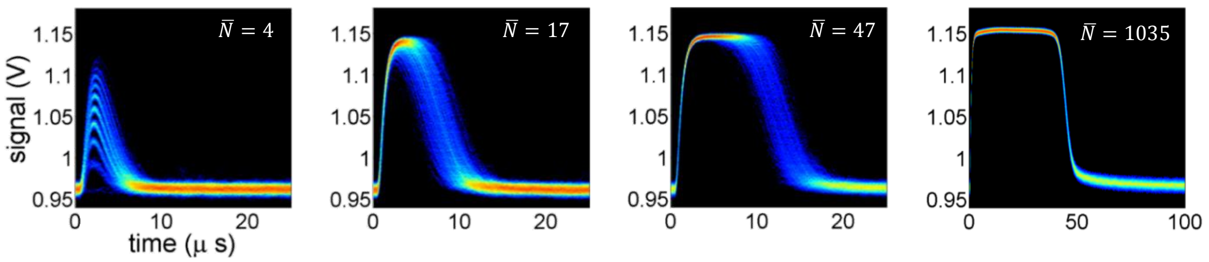

Figure 1: Sets of TES response waveforms elicited by ensembles of pulses with average photon numbers given by $\bar{N}$. Image from [3].

PIKA's goal is to characterize TES waveforms by photon numbers of the pulses that cause them. The photon numbers of individual pulses cannot directly be observed; we can only estimate the average photon number of all of the pulses, based on the nominal laser and attenuator parameters of the light source. The principal application of this algorithm is the calibration of TES detectors, but it has also been used for other purposes such as calibrating variable attenuators [4].

K-*Means Clustering and the Poisson Distribution*

Traditional *K*-means clustering consists of taking some number of data and organizing them into clusters that minimize their members' squared distance from the cluster mean. Essentially this is a minimization of an objective function, the sum over each datum of its squared deviation from its cluster mean, where deviation is measured by some relevant definition of distance. We can use a similar approach by considering each signal as a high-dimensional vector and deviation as Euclidean distance, so the objective function becomes

$$O_K = \sum_{n=n_0}^{n_0+K-1} \sum_{i \in C_n} \frac{1}{N_t} \sum_t [V_i(t) - \bar{V}_n(t)]^2,$$

where V_i is the signal vector for observation i, $\bar{V}_n$ is the mean of cluster n, and N_t is the number of time points. One can do better, though, with the knowledge that photon numbers follow a Poisson distribution. We introduce another term $O_{PC} = -\ln \mathcal{L}$, where $\mathcal{L}$ is the likelihood, according to the Poisson distribution, of a group of clusters associated with a group of photon numbers having the particular sizes that a given clustering asserts that they do, given the mean photon number of the ensemble of pulses. The objective function is now

$$O_{KPC} = \frac{1}{2\sigma^2} O_K + O_{PC},$$

where σ is a constant relating the two terms, which we can estimate from the data because O_K is itself the negative log-likelihood of a normal distribution, and K is determined by which photon numbers we expect to see at least once from the Poisson distribution. PIKA minimizes the objective function by moving waveforms to neighboring clusters. Once the clusters are optimized, each waveform is assigned an effective photon number by a linear interpolation between the two closest cluster means.

Initial Clustering and Geometric Interpretation

PIKA needs an initial clustering upon which to improve. Random cluster assignment is an option, but a better alternative is to give the observations a rough order by photon number, so that the initial guess is actually a meaningful estimate. This is done via the dot product method: each observation is assigned an initial effective photon number

$$n_{eff} = \bar{N}\,\frac{\bar{V} \cdot V_i}{||\bar{V}||^2},$$

where $\bar{V}$ is the entire ensemble's mean, not a cluster mean $\bar{V}_n$. The initial clusters are sized to fit each observation and conform to the Poisson distribution, and the observations are placed in the clusters by order of effective photon number.

The geometric interpretation of PIKA and the dot product method is a curve and a line, respectively, evolving through hyperspace, shown in Figure 2. The dot product method projects each observation onto the mean waveform vector (a line) and then assumes that photon number scales linearly with distance along the mean vector (which is not actually true but suffices for a first guess) to convert distance relative to the mean to photon number relative to the mean. PIKA, in contrast, finds a piecewise linear approximation of a curve that passes through the cluster means, and projects each observation onto that. Both essentially measure photon number by progress along a one-dimensional path through high-dimensional space.

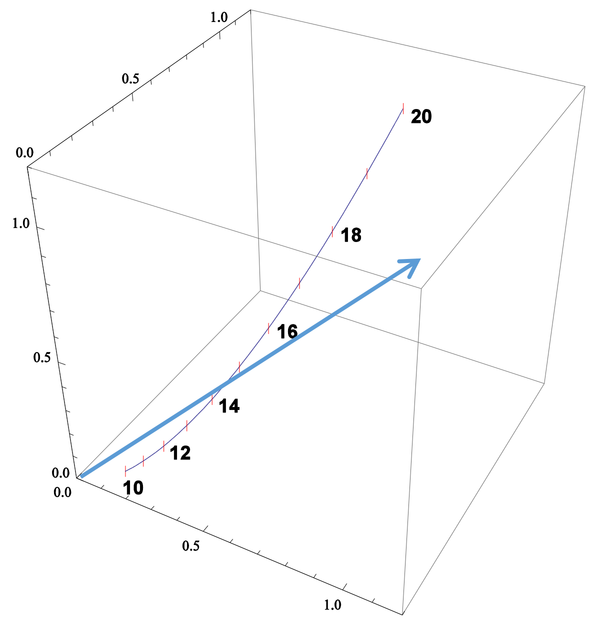

Figure 2: The geometric interpretation of PIKA and the dot product method: a line and a curve evolving through high-dimensional space. Image from [2].

Performance of the Original Implementation

Several papers [1], [4] have applied the Poisson-Influenced K-Means Algorithm (PIKA) to real data sets and demonstrated its level of utility. The results and data in this section come from Ref. [1], in which the algorithm was used to calibrate a transition edge sensor (TES). The raw data are voltage waveforms from the TES elicited by pulses of photons. PIKA is applied to the data to cluster the waveforms by photon number.

Figure 3 shows the optimized cluster means of two PIKA runs ($\overline{N} = 22.6$, solid red, and $\overline{N} = 31.6$, dotted blue). Waveforms in the raw data have shapes similar to these optimized means.

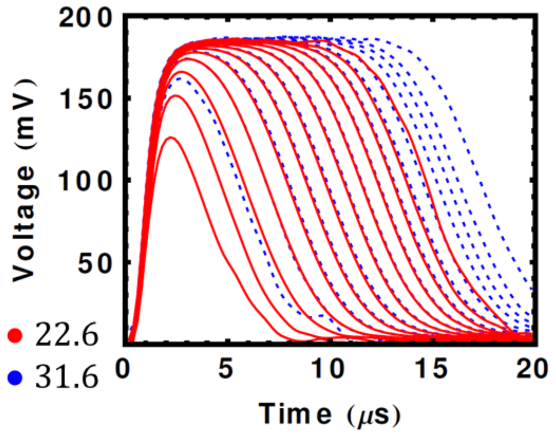

Figure 3: Optimized cluster mean waveforms from two ensembles with different average photon numbers (22.6, solid red, and 31.6, dotted blue). Image from [1].

The mean waveforms generated appear more or less independent of $\overline{N}$. This is consistent with physical reality, as the shape of an individual waveform should depend only on the photon number of the pulse that elicited it, not on the average photon number of the ensemble to which the pulse belongs. PIKA gives roughly the same shape for each particular photon number regardless of the mean photon number with which it is supplied.

The effective photon numbers (n) calculated by PIKA follow a Poisson distribution with a comb-like structure of Gaussians centered on the integers (Figure 4, with $\overline{N} = 2.00$).

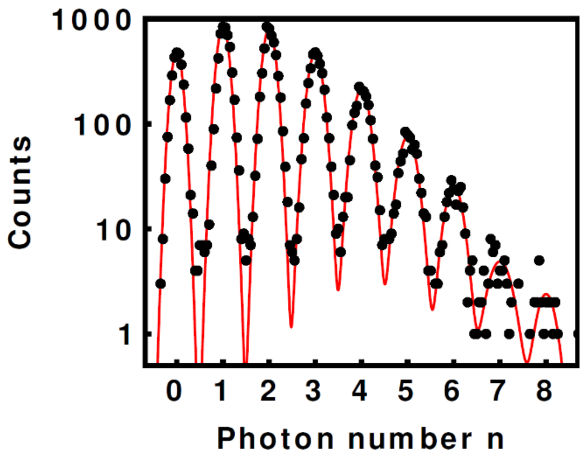

Figure 4: The effective photon numbers resulting from PIKA follow a Poisson distribution, with Gaussian spread around each integer effective photon number. This data set has an average photon number of 2.00. Image from Ref. [1].

As n increases, the teeth of the comb become less defined – that is, the peak visibility $(max - min)/(max + min)$ falls, and with it the photon-resolving capability. Figure 5 shows the decline in peak visibility for effective photon numbers given by the dot product method (blue diamonds) and by PIKA (red circles).

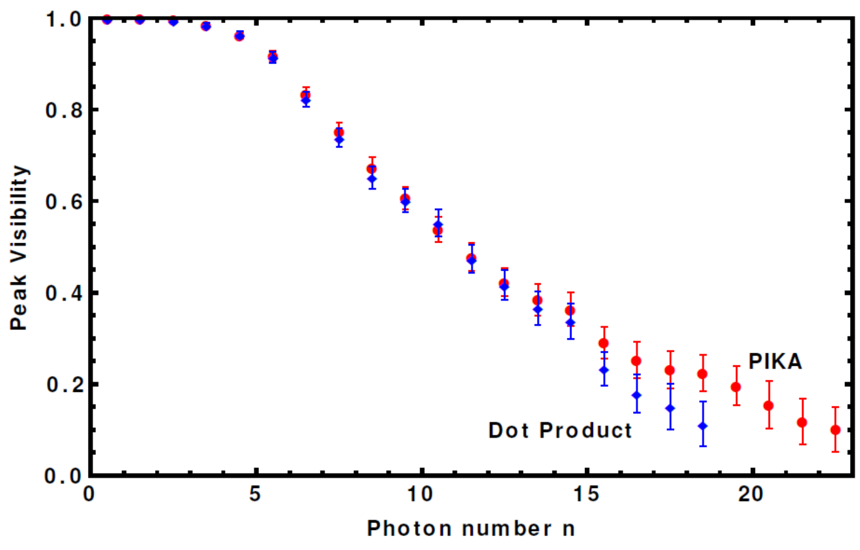

Figure 5: Peak visibility, with uncertainty, for effective photon numbers derived from the dot product method (blue diamonds) and PIKA (red circles). Image from Ref. [1].

PIKA retains nonzero visibility (i.e. the uncertainty does not include 0) through $n = 23$, whereas the dot product method alone loses visibility after $n = 19$.

Materials and Methods

This project was conducted using the Wolfram Mathematica 10 “front end” in order to create a Mathematica notebook file (.nb) based on several Mathematica kernel files (.m). [5] The algorithm implementation is based on that described in Ref. [1], and the images in the notebook come from the public domain and from various researchers associated with the National Institute of Standards and Technology [1-3].

Transportation to Notebook:

The existing code had been created in an older version of Mathematica as a collection of Mathematica kernel files, which I combined into one document and updated to ensure compatibility with Mathematica 10 and the new integrated arrangement of code. The original implementation consisted of a main file that executed the algorithm on a given data set and called functions defined in ancillary files. In moving these files to the notebook, I organized the function definitions by type, removed unused functions, and further divided the main execution code into several newly defined functions that sequentially captured the main file’s functionality.

Error Correction:

There were two types of errors present in the early versions of the notebook file: those introduced in the transition from kernel files to the notebook, and those that existed in the original code. In resolving the first kind, the newly introduced errors, I identified and fixed syntactical and runtime errors by testing whether the code would execute without issue when run on various data sets, and I corrected logical errors by comparing the results of the updated code with those of the original code for various data sets. The preexisting errors were more pernicious, requiring a careful verification that each component of the updated code worked precisely as intended and that the mathematical reasoning underlying each step of the algorithm was sound and accurate.

Documentation:

The documentation given in the notebook file consisted of two varieties: a general explanation of the algorithm and a description of the functionality of individual elements of the code. The general explanation drew from and expanded upon the description given in the original paper [1], and gave more derivations justifying the proper function of the algorithm as well as additional information on the physical background of the code and its theoretical significance as a machine learning algorithm. The particular functionality descriptions elaborate on the documentation of the original implementation and provide explanatory information such as instructions for running the code with data and a terminology scheme for variable and function names.

Update of Input and Output Systems:

While the original implementation had strictly file-based input and output systems, I created new functions to handle input and output in a self-contained manner in the notebook. In the new version, the user has the option to give input using an external file or to use form-based input contained in the notebook document itself. Output is now centralized in a tab-based organizational object that can be viewed in the notebook or saved as a separate file.

Results

Structure of the Notebook

The notebook document [2] begins with an element that executes PIKA (Section 1). Activation of this element creates an input form that allows the user to specify options about the execution and to give a directory containing the raw data upon which PIKA should be run. This form, when submitted, runs the algorithm on the given data with the given options by calling the function **`runPIKA`**, defined in a later section.

Following the execution element is a section that describes the overall purpose of and theory behind the algorithm, as well as its physical motivation for TES calibration (Section 2). New to the notebook is a section clarifying the nomenclature of variables and functions in the implementation (Section 3).

This is followed by Section 4, which outlines a procedural skeleton of the implementation, including a definition for **`runPIKA`**, the primary runner function for the algorithm (Figure 6). The function consists of three

```
runPIKA[] := (

  initialSetup[];
  readAndFilterData[];
  optionallyRejectBackgroundTraces[];
  graphGlobalMeanTrace[];
  findDotProductEffectivePhotonNumbers[];

  Do[               (* Loop 1 *)
   createInitialClusters[];
   graphSampleMiddleClusters[];
   graphClusterMeans[];
   findInitialObjectiveFunction[];
   graphClusterDeviationsFromMean[];
   prepareOptimizationLoop[];

   Do[              (* Loop 2 *)
    prepareSubLoop[];

    Do[             (* Loop 3 *)
     getClusterAndNeighbors[];
     updateRelevantDeviations[];
     findChangeInObjectiveFunction[];
     decideWhetherToMove[];
     updateIfMoving[];
     ,
     {jObs, nObs}
    ];

    updateAnnealingTemperature[];
    ,
    {iCool, nCool}
   ];

   prepareAnalysisAndOutput[];
   graphNewSampleMiddleClusters[];
   graphNewClusterMeans[];
   graphProbabilityDistribution[];
   graphNewClusterToMeanDeviations[];
   graphNewMeanToMeanDeviations[];
   graphNewEffectivePhotonNumber[];

   ,
   {iNPhotonAvgList, Length[nPhotonAvgList]}
  ];

  showOutput[]

 )
```

Figure 6: The definition for the **`runPIKA`** function.

nested loops, the outermost iterating over a list of mean photon numbers to test for the data, the middle iterating a preset number of times to optimize the clustering, and the innermost iterating over each waveform in the data set and deciding whether to move it to an adjacent cluster. On each level, **`runPIKA`** calls functions defined in the next section. This organizational scheme is new to the notebook: in the prior version, neither **`runPIKA`** nor the functions it calls were defined, and the *implementations* of those sub-functions comprised the main, three-layer body of the algorithm.

Section 5 defines the functions called by **`runPIKA`** in the order in which they are called, and Sections 6 and 7 define functions called by those functions. Section 6 focuses specifically on the formulas that efficiently compute the change in the objective function caused by a transfer of a waveform from one cluster to another, giving derivations to justify their validity in addition to defining functions for them. These formulas avoid the computationally expensive recalculation of the objective function from scratch that a naïve implementation of the algorithm would use. Section 8 contains general-purpose functions organized by type, including functions that handle form-based input and tabular graphical output, both of which are new to the notebook.

Correction of an Objective Function Update Formula

In the original paper, one of the formulas to update the *K*-means component of the objective function was incorrect (Equation (A4) in Ref. [1]). The notebook corrects the formula and gives a new derivation ensuring its validity, explained below.

We can decompose the *K*-means term of the objective function as

$$O_K = \sum_{n=n_0}^{n_0+K-1} J_n,$$

where

$$J_n = \sum_{i \in C_n} \frac{1}{N_t} \sum_t [V_i(t) - \bar{V}_n(t)]^2.$$

N_t is the number of time points, $V_i(t)$ is the waveform i evaluated at time t, and $\bar{V}_n(t)$ is the mean waveform of cluster C_n (associated with photon number n) evaluated at time t. If we transfer a waveform j from the cluster C_b to the cluster C_a, the new clusters formed are $C_a^+ = C_a \cup \{j\}$ and $C_b^- = C_b - \{j\}$. In addition, we can prove that

$$J_a^+ = J_a + \left(\frac{m_a}{m_a + 1}\right) \frac{1}{N_t} \sum_t \left[V_j(t) - \bar{V}_a(t)\right]^2$$

and

$$J_b^- = J_b - \left(\frac{m_b}{m_b - 1}\right) \frac{1}{N_t} \sum_t \left[V_j(t) - \bar{V}_b(t)\right]^2,$$

where m_n is the size of cluster C_n before the transfer. These formulas allow us to compute the change in the *K*-means component of the objective function without recalculating it in an inefficient manner.

The proof of these formulas requires two lemmas. Throughout the proof, we will treat the signal waveforms as vectors, so that $V_i(t)$ is understood to be the t^{th} component of the vector $\boldsymbol{V}_i$, and likewise with the mean waveforms $\bar{\boldsymbol{V}}_n$.

Lemma 1:

$$J_n = \sum_{i \in C_n} \frac{|\boldsymbol{V}_i - \bar{\boldsymbol{V}}_n|^2}{N_t} = \sum_{i,j \in C_n} \frac{\left|\boldsymbol{V}_i - \boldsymbol{V}_j\right|^2}{2 m_n N_t}$$

<u>Proof:</u>

The first expression for J_n is true by definition; now we show that the second is equivalent.

$$\sum_{i,j\in C_n} \frac{\left|\boldsymbol{V}_i - \boldsymbol{V}_j\right|^2}{2m_n N_t}$$

$$= \frac{1}{2m_n N_t}\left[m_n \sum_{i\in C_n} |\boldsymbol{V}_i|^2 - \sum_{i,j\in C_n} 2\,\boldsymbol{V}_i \cdot \boldsymbol{V}_j + m_n \sum_{j\in C_n} \left|\boldsymbol{V}_j\right|^2\right]$$

$$= \frac{1}{N_t}\sum_{i\in C_n} (|\boldsymbol{V}_i|^2 - \boldsymbol{V}_i \cdot \overline{\boldsymbol{V}}_n)$$

$$= \frac{1}{N_t}\sum_{i\in C_n} \boldsymbol{V}_i \cdot (\boldsymbol{V}_i - \overline{\boldsymbol{V}}_n)$$

$$= \frac{1}{N_t}\sum_{i\in C_n} (\boldsymbol{V}_i - \overline{\boldsymbol{V}}_n) \cdot (\boldsymbol{V}_i - \overline{\boldsymbol{V}}_n),$$

which is equivalent to the first expression. The final step is valid because $\sum_{i\in C_n}(\boldsymbol{V}_i - \overline{\boldsymbol{V}}_n) = 0$.

■

Lemma 2:

$$|\boldsymbol{V}_k - \overline{\boldsymbol{V}}_n|^2 + \frac{N_t J_n}{m_n} = |\boldsymbol{V}_k|^2 - 2\,\boldsymbol{V}_k \cdot \overline{\boldsymbol{V}}_n + \frac{1}{m_n}\sum_{i\in C_n} |\boldsymbol{V}_i|^2$$

Proof:

$$|\boldsymbol{V}_k - \overline{\boldsymbol{V}}_n|^2 = |\boldsymbol{V}_k|^2 - 2\,\boldsymbol{V}_k \cdot \overline{\boldsymbol{V}}_n + \frac{1}{m_n^2}\sum_{i,j\in C_n} \boldsymbol{V}_i \cdot \boldsymbol{V}_j.$$

From the proof of the previous lemma,

$$J_n = \frac{1}{N_t}\left(\sum_{i\in C_n} |\boldsymbol{V}_i|^2 - \frac{1}{m_n}\sum_{i,j\in C_n} \boldsymbol{V}_i \cdot \boldsymbol{V}_j\right),$$

so,

$$\frac{1}{m_n}\sum_{i,j\in C_n} \boldsymbol{V}_i \cdot \boldsymbol{V}_j = \sum_{i\in C_n} |\boldsymbol{V}_i|^2 - J_n N_t.$$

Thus,

$$|\boldsymbol{V}_k - \overline{\boldsymbol{V}}_n|^2 = |\boldsymbol{V}_k|^2 - 2\,\boldsymbol{V}_k \cdot \overline{\boldsymbol{V}}_n + \frac{1}{m_n}\left(\sum_{i\in C_n} |\boldsymbol{V}_i|^2 - N_t J_n\right)$$

and the lemma follows.

■

Theorem 1:

Suppose we add or remove a waveform $\boldsymbol{V}_p$ to or from a cluster C_n, forming a cluster C_n'.

Then,

$$J_n' = J_n \pm \left(\frac{m_n}{m_n'}\right)\frac{\left|\boldsymbol{V}_p - \overline{\boldsymbol{V}}_n\right|^2}{N_t},$$

where $m_n' = m_n \pm 1$ is the size of the newly formed cluster.

Proof:

From Lemma 1,

$$J_n' = \sum_{i,j\in C_n'} \frac{\left|\boldsymbol{V}_i - \boldsymbol{V}_j\right|^2}{2m_n' N_t}.$$

(Note the summation over C_n', not C_n.) Since $\boldsymbol{V}_p$ is the waveform to add to or remove from C_n, we can write

$$J_n' = \frac{1}{2m_nN_t}\left(\sum_{i,j\in C_n}\left|\boldsymbol{V}_i - \boldsymbol{V}_j\right|^2 \pm \sum_{j\in C_n}\left|\boldsymbol{V}_p - \boldsymbol{V}_j\right|^2 \pm \sum_{i\in C_n}\left|\boldsymbol{V}_{\boldsymbol{i}} - \boldsymbol{V}_p\right|^2 \pm \left|\boldsymbol{V}_p - \boldsymbol{V}_p\right|^2\right)$$

$$= \frac{m_n}{m_n'}J_n \pm \frac{1}{m_n'N_t}\sum_{j\in C_n}\left|\boldsymbol{V}_p - \boldsymbol{V}_j\right|^2$$

$$= \frac{m_n}{m_n'}J_n \pm \frac{m_n}{m_n'N_t}\left(\left|\boldsymbol{V}_p\right|^2 - 2\,\boldsymbol{V}_p\cdot\overline{\boldsymbol{V}}_n + \frac{1}{m_n}\sum_{j\in C_n}\left|\boldsymbol{V}_j\right|^2\right).$$

From Lemma 2,

$$J_n' = \frac{m_n}{m_n'}J_n \pm \frac{m_n}{m_n'N_t}\left(\left|\boldsymbol{V}_p - \overline{\boldsymbol{V}}_n\right|^2 + \frac{N_t\,J_n}{m_n}\right)$$

$$= \left(\frac{m_n}{m_n'} \pm \frac{1}{m_n'}\right)J_n \pm \left(\frac{m_n}{m_n'}\right)\frac{\left|\boldsymbol{V}_p - \overline{\boldsymbol{V}}_n\right|^2}{N_t},$$

and the theorem follows.

■

Equation (A4) in Ref. [1] erroneously asserted that the formula for removal of a waveform from a cluster was $J_b^- = J_b - \left(\frac{m_b-1}{m_b}\right)\frac{1}{N_t}\sum_t\left[V_j(t) - \bar{V}_b(t)\right]^2$ instead of $J_b^- = J_b - \left(\frac{m_b}{m_b-1}\right)\frac{1}{N_t}\sum_t\left[V_j(t) - \bar{V}_b(t)\right]^2$.

Discussion

The notebook document [2] was designed to improve the original Poisson-Influenced *K*-Means Algorithm (PIKA) implementation in terms of accessibility, adaptability, and transparency, with the goal being to streamline the end-user experience for scientific researchers and metrology labs using and modifying the algorithm for their own purposes. We foresaw two primary categories of uses for the notebook. The first included transition edge sensor (TES) calibration and similar applications [1], [4], for which ease of use would be essential. The second included applications to scenarios governed by probability distributions other than the Poisson distribution and scenarios without a metric for ordering signal waveforms (the effective photon number of a waveform supplies such a metric in TES calibration), for which ease of modification would also be important.

Motivation for Transition from the Original Implementation

PIKA was originally implemented [1] as a Mathematica kernel package, as opposed to a Mathematica notebook, consisting of a main execution file that called functions defined in separate, ancillary files. Input and output were strictly file-based: a separate user-defined input file would set constants and options and give a directory containing the data to process, and graphical and numerical outputs as well as the log file would be saved to the directory containing the input file. The code and documentation were separated from the overarching algorithm description (found in the original paper. The objective of the transition to the notebook was to give a more cohesive presentation of the implementation's functionality in order to expedite the end-user's comprehension.

Advantages of the New Version

The notebook is a single multi-sectioned document that integrates functional implementation, specific documentation, and general explanation. The documentation is more extensive than in the original version, with inline comments supplemented by explanatory paragraphs in a contrasting text style to facilitate visual differentiation between code and documentation. The lengthy main execution file from the original implementation has been split into isolated, individually digestible pieces that make clear the nested-loop structure of the algorithm, its overall functionality, and the contributions of particular sections. Unused functions have been removed and obsolete and unclear names, for example variable names whose meaning drifted during the development of the algorithm, have been changed. Mathematica allows for cosmetic formatting of certain elements of active code (such as, for instance, using a full-size fraction bar in place of a slash for the division function), which we have adopted in the notebook as appropriate to visually clarify the scope and nature of the operators used.

The input/output system is almost entirely new to the notebook. The notebook retains the option to use a separate, static file for user-defined options and constants, but by default, input is handled through an interactive form contained in the document itself. Output is now centralized in an interactive tab-based object (Figure 7) containing graphical, textual, and numerical output, organized and labeled by the type of output and the number of the run from which it came. The new systems for input and output make the new implementation far more self-contained than the old, although both systems allow one to work instead with separate files.

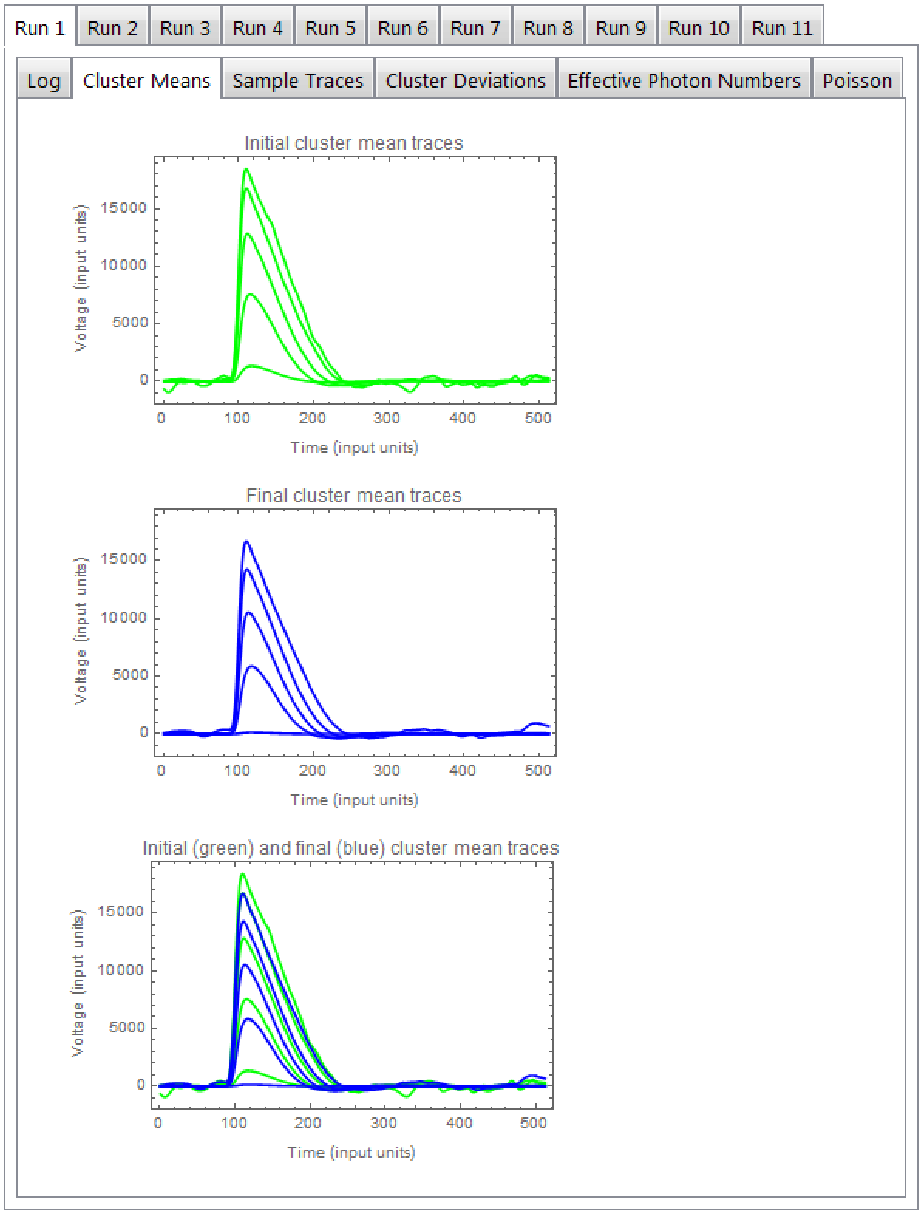

Figure 7: The interactive object that centralized output in the notebook. Graphs of initial and final cluster mean waveforms are shown here.

Finally, the notebook makes some efficiency improvements over the original implementation, especially with the system for reading data, and corrects some bugs and errors present in the original version. A particularly important correction was to the error in one of the efficient objective function update formulas mentioned in the previous section. The error was not of great practical importance, since the quantity erroneously reciprocated was very close to unity and so the change was negligible, but the correction puts the algorithm on a more firm theoretical basis. In addition, the notebook supplies more rigorous justifications for the update formulas.

Conclusion

We have presented a new implementation of the successful and useful Poisson-Influenced *K*-Means Algorithm, intended for easier use and adaptation by other researchers. We anticipate its utility not only for its original purpose in transition edge sensor calibration and related applications, but also for other applications involving non-Poisson statistics or signals without a natural ordering.

References

[1] Z. Levine et al., “Algorithm for finding clusters with a known distribution and its application to photon-number resolution using a superconducting transition-edge sensor,” *J. Opt. Soc. Am. B* 29, 2066-2073 (2012).

[2] B. Morris and Z. Levine, “The Poisson-Influenced K-Means Algorithm, a Maximum-Likelihood Procedure for Clusters with a Known Probability Distribution” *The Mathematica Journal*, accepted.

[3] T. Gerrits et al., “Extending single-photon optimized superconducting transition edge sensors beyond the single-photon counting regime,” *Opt. Express* 20, 23798-23810 (2012).

[4] Z. Levine et al., “Absolute calibration of a variable attenuator using few-photon pulses,” *Opt. Express* 23, 16372-16382 (2015).

[5] Mention of commercial products does not imply endorsement by the authors’ institutions.